

MSP432E4 SimpleLink™ Microcontrollers

Technical Reference Manual



Literature Number: SLAU723A
October 2017–Revised October 2018

Preface	77
1 Cortex®-M4F Processor	79
1.1 Introduction	80
1.2 Block Diagram	81
1.3 Overview	82
1.3.1 System-Level Interface	82
1.3.2 Integrated Configurable Debug	82
1.3.3 Trace Port Interface Unit (TPIU)	82
1.3.4 Cortex-M4F System Component Details	83
1.4 Programming Model	83
1.4.1 Processor Mode and Privilege Levels for Software Execution	83
1.4.2 Stacks	84
1.4.3 Exceptions and Interrupts	97
1.4.4 Data Types	97
1.5 Memory Model	97
1.5.1 Memory Regions, Types, and Attributes	100
1.5.2 Memory System Ordering of Memory Accesses	100
1.5.3 Behavior of Memory Accesses	100
1.5.4 Software Ordering of Memory Accesses	101
1.5.5 Bit-Banding	102
1.5.6 Data Storage	104
1.5.7 Synchronization Primitives	104
1.6 Exception Model	105
1.6.1 Exception States	106
1.6.2 Exception Types	106
1.6.3 Exception Handlers	108
1.6.4 Vector Table	108
1.6.5 Exception Priorities	109
1.6.6 Interrupt Priority Grouping	110
1.6.7 Exception Entry and Return	110
1.7 Fault Handling	113
1.7.1 Fault Types	113
1.7.2 Fault Escalation and Hard Faults	113
1.7.3 Fault Status Registers and Fault Address Registers	114
1.7.4 Lockup	114
1.8 Power Management	114
1.8.1 Entering Sleep Modes	115
1.8.2 Wake Up From Sleep Mode	115
1.9 Instruction Set Summary	115
2 Cortex-M4 Peripherals	121
2.1 Introduction	122
2.2 Functional Description	122
2.2.1 System Timer (SysTick)	122
2.2.2 Nested Vectored Interrupt Controller (NVIC)	123
2.2.3 System Control Block (SCB)	124

2.2.4	Memory Protection Unit (MPU)	124
2.2.5	Floating-Point Unit (FPU)	128
2.3	SysTick Registers.....	133
2.3.1	STCTRL Register (Offset = 0x10) [reset = 0x0]	134
2.3.2	STRELOAD Register (Offset = 0x14) [reset = 0x0]	135
2.3.3	STCURRENT Register (Offset = 0x18) [reset = 0x0]	136
2.4	NVIC Registers.....	137
2.4.1	EN0 to EN3 Registers.....	139
2.4.2	DIS0 to DIS3 Registers	140
2.4.3	PEND0 to PEND 3 Registers	141
2.4.4	UNPEND0 to UNPEND3 Registers	142
2.4.5	ACTIVE0 to ACTIVE3 Registers.....	143
2.4.6	PRI0 to PRI28 Registers.....	144
2.4.7	SWTRIG Register (Offset = 0xF00) [reset = 0x0]	146
2.5	SCB Registers.....	147
2.5.1	ACTLR Register (Offset = 0x8) [reset = 0x0]	148
2.5.2	CPUID Register (Offset = 0xD00) [reset = 0x410FC241]	149
2.5.3	INTCTRL Register (Offset = 0xD04) [reset = 0x0]	150
2.5.4	VTABLE Register (Offset = 0xD08) [reset = 0x0]	152
2.5.5	APINT Register (Offset = 0xD0C) [reset = 0xFA050000]	153
2.5.6	SYSCTRL Register (Offset = 0xD10) [reset = 0x0]	155
2.5.7	CFGCTRL Register (Offset = 0xD14) [reset = 0x200]	156
2.5.8	SYSPRI1 Register (Offset = 0xD18) [reset = 0x0]	157
2.5.9	SYSPRI2 Register (Offset = 0xD1C) [reset = 0x0]	158
2.5.10	SYSPRI3 Register (Offset = 0xD20) [reset = 0x0]	159
2.5.11	SYSHNDCTRL Register (Offset = 0xD24) [reset = 0x0]	160
2.5.12	FAULTSTAT Register (Offset = 0xD28) [reset = 0x0]	162
2.5.13	HFAULTSTAT Register (Offset = 0xD2C) [reset = 0x0]	165
2.5.14	MMADDR Register (Offset = 0xD34) [reset = X]	166
2.5.15	FAULTADDR Register (Offset = 0xD38) [reset = X]	167
2.6	MPU Registers	168
2.6.1	MPUTYPE Register (Offset = 0xD90) [reset = 0x800].....	169
2.6.2	MPUCTRL Register (Offset = 0xD94) [reset = 0x0].....	170
2.6.3	MPUNUMBER Register (Offset = 0xD98) [reset = 0x0]	172
2.6.4	MPUBASEn Registers	173
2.6.5	MPUATTRn Registers.....	175
2.7	FPU Registers.....	177
2.7.1	CPAC Register (Offset = 0xD88) [reset = 0x0]	178
2.7.2	FPCC Register (Offset = 0xF34) [reset = 0xC0000000].....	179
2.7.3	FPCA Register (Offset = 0xF38) [reset = X]	181
2.7.4	FPDSC Register (Offset = 0xF3C) [reset = X]	182
3	JTAG Interface	183
3.1	Introduction	184
3.2	Block Diagram.....	184
3.3	Functional Description.....	185
3.3.1	JTAG Interface Pins	185
3.3.2	JTAG TAP Controller	187
3.3.3	Shift Registers.....	187
3.3.4	Operational Considerations.....	187
3.4	Initialization and Configuration	190
3.5	Register Descriptions.....	190
3.5.1	Instruction Register (IR).....	190
3.5.2	Data Registers	192

4	System Control.....	194
4.1	Functional Description.....	195
4.1.1	Device Identification	195
4.1.2	Reset Control.....	195
4.1.3	Nonmaskable Interrupt.....	201
4.1.4	Power Control	202
4.1.5	Clock Control	203
4.1.6	System Control.....	210
4.2	System Control Registers.....	217
4.2.1	DID0 Register (Offset = 0x0) [reset = X].....	222
4.2.2	DID1 Register (Offset = 0x4) [reset = X].....	224
4.2.3	PTBOCTL Register (Offset = 0x38) [reset = 0x3]	226
4.2.4	RIS Register (Offset = 0x50) [reset = 0x0]	227
4.2.5	IMC Register (Offset = 0x54) [reset = 0x0].....	229
4.2.6	MISC Register (Offset = 0x58) [reset = 0x0]	230
4.2.7	RESC Register (Offset = 0x5C) [reset = X]	232
4.2.8	PWRTC Register (Offset = 0x60) [reset = 0x0]	234
4.2.9	NMIC Register (Offset = 0x64) [reset = 0x0]	235
4.2.10	MOSCCTL Register (Offset = 0x7C) [reset = 0xC].....	237
4.2.11	RSCLKCFG Register (Offset = 0xB0) [reset = 0x0].....	239
4.2.12	MENTIM0 Register (Offset = 0xC0) [reset = 0x00200030]	241
4.2.13	ALTCLKCFG Register (Offset = 0x138) [reset = 0x0]	244
4.2.14	DSCLKCFG Register (Offset = 0x144) [reset = 0x0]	245
4.2.15	DIVSCLK Register (Offset = 0x148) [reset = 0x0].....	247
4.2.16	SYSPROP Register (Offset = 0x14C) [reset = 0x00031F31]	248
4.2.17	PIOSCCAL Register (Offset = 0x150) [reset = 0x0]	250
4.2.18	PIOSCCSTAT Register (Offset = 0x154) [reset = 0x00400040]	251
4.2.19	PLLREQ0 Register (Offset = 0x160) [reset = 0x0]	252
4.2.20	PLLREQ1 Register (Offset = 0x164) [reset = 0x0]	253
4.2.21	PLLSTAT Register (Offset = 0x168) [reset = 0x0]	254
4.2.22	SLPPWRCFG Register (Offset = 0x188) [reset = 0x0]	255
4.2.23	DSLPPWRCFG Register (Offset = 0x18C) [reset = 0x0].....	256
4.2.24	NVMSTAT Register (Offset = 0x1A0) [reset = 0x1]	258
4.2.25	LDOSPCTL Register (Offset = 0x1B4) [reset = 0x18]	259
4.2.26	LDOSPCAL Register (Offset = 0x1B8) [reset = 0x1818]	260
4.2.27	LDODPCTL Register (Offset = 0x1BC) [reset = 0x12].....	261
4.2.28	LDODPCAL Register (Offset = 0x1C0) [reset = 0x1212].....	262
4.2.29	SDPMST Register (Offset = 0x1CC) [reset = 0x0]	263
4.2.30	RESBEHAVCTL Register (Offset = 0x1D8) [reset = 0x00FFFFFF].....	265
4.2.31	HSSR Register (Offset = 0x1F4) [reset = 0x0]	266
4.2.32	USBPDS Register (Offset = 0x280) [reset = 0x3F]	267
4.2.33	USBMPC Register (Offset = 0x284) [reset = 0x3].....	268
4.2.34	EMACPDS Register (Offset = 0x288) [reset = 0x3F]	269
4.2.35	EMACMPC Register (Offset = 0x28C) [reset = 0x3]	270
4.2.36	LCDPDS Register (Offset = 0x290) [reset = 0x3F].....	271
4.2.37	LCDMPC Register (Offset = 0x294) [reset = 0x3].....	272
4.2.38	CAN0PDS Register (Offset = 0x298) [reset = 0x3F].....	273
4.2.39	CAN0MPC Register (Offset = 0x29C) [reset = 0x3].....	274
4.2.40	CAN1PDS Register (Offset = 0x2A0) [reset = 0x3F]	275
4.2.41	CAN1MPC Register (Offset = 0x2A4) [reset = 0x3].....	276
4.2.42	PPWD Register (Offset = 0x300) [reset = 0x3]	277
4.2.43	PPTIMER Register (Offset = 0x304) [reset = 0xFF]	278
4.2.44	PPGPIO Register (Offset = 0x308) [reset = 0x3FFFFF]	279

4.2.45	PPDMA Register (Offset = 0x30C) [reset = 0x1]	281
4.2.46	PPEPI Register (Offset = 0x310) [reset = 0x1]	282
4.2.47	PPHIB Register (Offset = 0x314) [reset = 0x01]	283
4.2.48	PPUART Register (Offset = 0x318) [reset = 0xFF]	284
4.2.49	PPSSI Register (Offset = 0x31C) [reset = 0xF]	285
4.2.50	PPI2C Register (Offset = 0x320) [reset = 0x3FF]	286
4.2.51	PPUSB Register (Offset = 0x328) [reset = 0x1]	287
4.2.52	PPEPHY Register (Offset = 0x330) [reset = 0x1]	288
4.2.53	PPCAN Register (Offset = 0x334) [reset = 0x3]	289
4.2.54	PPADC Register (Offset = 0x338) [reset = 0x3]	290
4.2.55	PPACMP Register (Offset = 0x33C) [reset = 0x1]	291
4.2.56	PPPWM Register (Offset = 0x340) [reset = 0x1]	292
4.2.57	PPQEI Register (Offset = 0x344) [reset = 0x1]	293
4.2.58	PPEEPROM Register (Offset = 0x358) [reset = 0x01]	294
4.2.59	PPCCM Register (Offset = 0x374) [reset = 0x01]	295
4.2.60	PPLCD Register (Offset = 0x390) [reset = 0x01]	296
4.2.61	PPOWIRE Register (Offset = 0x398) [reset = 0x01]	297
4.2.62	PPEMAC Register (Offset = 0x39C) [reset = 0x01]	298
4.2.63	PPPRB Register (Offset = 0x3A0) [reset = 0x00]	299
4.2.64	SRWD Register (Offset = 0x500) [reset = 0x00]	300
4.2.65	SRTIMER Register (Offset = 0x504) [reset = 0x00]	301
4.2.66	SRGPIO Register (Offset = 0x508) [reset = 0x00]	302
4.2.67	SRDMA Register (Offset = 0x50C) [reset = 0x00]	304
4.2.68	SREPI Register (Offset = 0x510) [reset = 0x00]	305
4.2.69	SRHIB Register (Offset = 0x514) [reset = 0x00]	306
4.2.70	SRUART Register (Offset = 0x518) [reset = 0x00]	307
4.2.71	SRSSI Register (Offset = 0x51C) [reset = 0x0]	309
4.2.72	SRI2C Register (Offset = 0x520) [reset = 0x0]	310
4.2.73	SRUSB Register (Offset = 0x528) [reset = 0x0]	312
4.2.74	SREPHY Register (Offset = 0x530) [reset = 0x0]	313
4.2.75	SRCAN Register (Offset = 0x534) [reset = 0x0]	314
4.2.76	SRADC Register (Offset = 0x538) [reset = 0x0]	315
4.2.77	SRACMP Register (Offset = 0x53C) [reset = 0x0]	316
4.2.78	SRPWM Register (Offset = 0x540) [reset = 0x0]	317
4.2.79	SRQEI Register (Offset = 0x544) [reset = 0x0]	318
4.2.80	SREEEPROM Register (Offset = 0x558) [reset = 0x0]	319
4.2.81	SRCCM Register (Offset = 0x574) [reset = 0x0]	320
4.2.82	SRLCD Register (Offset = 0x590) [reset = 0x0]	321
4.2.83	SROWIRE Register (Offset = 0x598) [reset = 0x0]	322
4.2.84	SREMAC Register (Offset = 0x59C) [reset = 0x0]	323
4.2.85	RCGCWD Register (Offset = 0x600) [reset = 0x0]	324
4.2.86	RCGCTIMER Register (Offset = 0x604) [reset = 0x00]	325
4.2.87	RCGCGPIO Register (Offset = 0x608) [reset = 0x00]	326
4.2.88	RCGCDMA Register (Offset = 0x60C) [reset = 0x0]	328
4.2.89	RCGCEPI Register (Offset = 0x610) [reset = 0x0]	329
4.2.90	RCGCHIB Register (Offset = 0x614) [reset = 0x1]	330
4.2.91	RCGCUART Register (Offset = 0x618) [reset = 0x00]	331
4.2.92	RCGCSSI Register (Offset = 0x61C) [reset = 0x0]	332
4.2.93	RCGCI2C Register (Offset = 0x620) [reset = 0x00]	333
4.2.94	RCGUSB Register (Offset = 0x628) [reset = 0x0]	335
4.2.95	RCGCEPHY Register (Offset = 0x630) [reset = 0x0]	336
4.2.96	RCGCCAN Register (Offset = 0x634) [reset = 0x0]	337
4.2.97	RCGCADC Register (Offset = 0x638) [reset = 0x0]	338

4.2.98	RCGCACMP Register (Offset = 0x63C) [reset = 0x0]	339
4.2.99	RCGCPWM Register (Offset = 0x640) [reset = 0x0]	340
4.2.100	RCGCQEI Register (Offset = 0x644) [reset = 0x0]	341
4.2.101	RCGCEEPROM Register (Offset = 0x658) [reset = 0x0]	342
4.2.102	RCGCCCCM Register (Offset = 0x674) [reset = 0x0]	343
4.2.103	RCGCLCD Register (Offset = 0x690) [reset = 0x0]	344
4.2.104	RCGCOWIRE Register (Offset = 0x698) [reset = 0x0]	345
4.2.105	RCGCEMAC Register (Offset = 0x69C) [reset = 0x0]	346
4.2.106	SCGCWD Register (Offset = 0x700) [reset = 0x0]	347
4.2.107	SCGCTIMER Register (Offset = 0x704) [reset = 0x00]	348
4.2.108	SCGCGPIO Register (Offset = 0x708) [reset = 0x00]	350
4.2.109	SCGCDMA Register (Offset = 0x70C) [reset = 0x0]	352
4.2.110	SCGCEPI Register (Offset = 0x710) [reset = 0x0]	353
4.2.111	SCGCHIB Register (Offset = 0x714) [reset = 0x1]	354
4.2.112	SCGCUART Register (Offset = 0x718) [reset = 0x00]	355
4.2.113	SCGCSSI Register (Offset = 0x71C) [reset = 0x0]	356
4.2.114	SCGCI2C Register (Offset = 0x720) [reset = 0x00]	357
4.2.115	SCGCUSB Register (Offset = 0x728) [reset = 0x0]	359
4.2.116	SCGCEPHY Register (Offset = 0x730) [reset = 0x0]	360
4.2.117	SCGCCAN Register (Offset = 0x734) [reset = 0x0]	361
4.2.118	SCGCADC Register (Offset = 0x738) [reset = 0x0]	362
4.2.119	SCGCACMP Register (Offset = 0x73C) [reset = 0x0]	363
4.2.120	SCGCPWM Register (Offset = 0x740) [reset = 0x0]	364
4.2.121	SCGCQEI Register (Offset = 0x744) [reset = 0x0]	365
4.2.122	SCGCEEPROM Register (Offset = 0x758) [reset = 0x0]	366
4.2.123	SCGCCCCM Register (Offset = 0x774) [reset = 0x0]	367
4.2.124	SCGCLCD Register (Offset = 0x790) [reset = 0x0]	368
4.2.125	SCGCOWIRE Register (Offset = 0x798) [reset = 0x0]	369
4.2.126	SCGCEMAC Register (Offset = 0x79C) [reset = 0x0]	370
4.2.127	DCGCWD Register (Offset = 0x800) [reset = 0x0]	371
4.2.128	DCGCTIMER Register (Offset = 0x804) [reset = 0x00]	372
4.2.129	DCGCGPIO Register (Offset = 0x808) [reset = 0x00]	374
4.2.130	DCGCDMA Register (Offset = 0x80C) [reset = 0x0]	376
4.2.131	DCGCEPI Register (Offset = 0x810) [reset = 0x0]	377
4.2.132	DCGCHIB Register (Offset = 0x814) [reset = 0x1]	378
4.2.133	DCGCUART Register (Offset = 0x818) [reset = 0x00]	379
4.2.134	DCGCSSI Register (Offset = 0x81C) [reset = 0x0]	381
4.2.135	DCGCI2C Register (Offset = 0x820) [reset = 0x00]	382
4.2.136	DCGCUSB Register (Offset = 0x828) [reset = 0x0]	384
4.2.137	DCGCEPHY Register (Offset = 0x830) [reset = 0x0]	385
4.2.138	DCGCCAN Register (Offset = 0x834) [reset = 0x0]	386
4.2.139	DCGCADC Register (Offset = 0x838) [reset = 0x0]	387
4.2.140	DCGCACMP Register (Offset = 0x83C) [reset = 0x0]	388
4.2.141	DCGCPWM Register (Offset = 0x840) [reset = 0x0]	389
4.2.142	DCGCQEI Register (Offset = 0x844) [reset = 0x0]	390
4.2.143	DCGCEEPROM Register (Offset = 0x858) [reset = 0x0]	391
4.2.144	DCGCCCCM Register (Offset = 0x874) [reset = 0x0]	392
4.2.145	DCGCLCD Register (Offset = 0x890) [reset = 0x0]	393
4.2.146	DCGCOWIRE Register (Offset = 0x898) [reset = 0x0]	394
4.2.147	DCGCEMAC Register (Offset = 0x89C) [reset = 0x0]	395
4.2.148	PCWD Register (Offset = 0x900) [reset = 0x3]	396
4.2.149	PCTIMER Register (Offset = 0x904) [reset = 0xFF]	398
4.2.150	PCGPIO Register (Offset = 0x908) [reset = 0x3FFFFF]	401

4.2.151	PCDMA Register (Offset = 0x90C) [reset = 0x1]	405
4.2.152	PCEPI Register (Offset = 0x910) [reset = 0x1]	407
4.2.153	PCHIB Register (Offset = 0x914) [reset = 0x1]	409
4.2.154	PCUART Register (Offset = 0x918) [reset = 0xFF]	411
4.2.155	PCSSI Register (Offset = 0x91C) [reset = 0xF]	414
4.2.156	PCI2C Register (Offset = 0x920) [reset = 0x3FF]	416
4.2.157	PCUSB Register (Offset = 0x928) [reset = 0x1]	419
4.2.158	PCEPHY Register (Offset = 0x930) [reset = 0x0]	420
4.2.159	PCCAN Register (Offset = 0x934) [reset = 0x3]	422
4.2.160	PCADC Register (Offset = 0x938) [reset = 0x3]	424
4.2.161	PCACMP Register (Offset = 0x93C) [reset = 0x1]	426
4.2.162	PCPWM Register (Offset = 0x940) [reset = 0x1]	428
4.2.163	PCQEI Register (Offset = 0x944) [reset = 0x1]	430
4.2.164	PCEEPROM Register (Offset = 0x958) [reset = 0x1]	432
4.2.165	PCCCM Register (Offset = 0x974) [reset = 0x1]	434
4.2.166	PCLCD Register (Offset = 0x990) [reset = 0x1]	436
4.2.167	PCOWIRE Register (Offset = 0x998) [reset = 0x1]	437
4.2.168	PEMAC Register (Offset = 0x99C) [reset = 0x1]	439
4.2.169	PRWD Register (Offset = 0xA00) [reset = 0x0]	440
4.2.170	PRTIMER Register (Offset = 0xA04) [reset = 0x00]	441
4.2.171	PRGPIO Register (Offset = 0xA08) [reset = 0x00]	443
4.2.172	PRDMA Register (Offset = 0xA0C) [reset = 0x0]	445
4.2.173	PREPI Register (Offset = 0xA10) [reset = 0x0]	446
4.2.174	PRHIB Register (Offset = 0xA14) [reset = 0x1]	447
4.2.175	PRUART Register (Offset = 0xA18) [reset = 0x00]	448
4.2.176	PRSSI Register (Offset = 0xA1C) [reset = 0x0]	450
4.2.177	PRI2C Register (Offset = 0xA20) [reset = 0x00]	451
4.2.178	PRUSB Register (Offset = 0xA28) [reset = 0x0]	453
4.2.179	PREPHY Register (Offset = 0xA30) [reset = 0x0]	454
4.2.180	PRCAN Register (Offset = 0xA34) [reset = 0x0]	455
4.2.181	PRADC Register (Offset = 0xA38) [reset = 0x0]	456
4.2.182	PRACMP Register (Offset = 0xA3C) [reset = 0x0]	457
4.2.183	PRPWM Register (Offset = 0xA40) [reset = 0x0]	458
4.2.184	PRQEI Register (Offset = 0xA44) [reset = 0x0]	459
4.2.185	PREEPROM Register (Offset = 0xA58) [reset = 0x0]	460
4.2.186	PRCCM Register (Offset = 0xA74) [reset = 0x0]	461
4.2.187	PRLCD Register (Offset = 0xA90) [reset = 0x0]	462
4.2.188	PROWIRE Register (Offset = 0xA98) [reset = 0x0]	463
4.2.189	PREMAC Register (Offset = 0xA9C) [reset = 0x0]	464
4.2.190	UNIQUEID0 to UNIQUEID3 Register (Offset = 0xF20 to 0xF2C) [reset = X]	465
4.3	Cryptographic System Control (CCM) Registers	466
4.3.1	CCMCGREQ Register (Offset = 0x204) [reset = 0x0]	467
5	Processor Support and Exception Module	468
5.1	Functional Description	469
5.2	System Exception Registers	470
5.2.1	SYSEXCRIS Register (Offset = 0x0) [reset = 0x0]	471
5.2.2	SYSEXCIM Register (Offset = 0x4) [reset = 0x0]	472
5.2.3	SYSEXCMIIS Register (Offset = 0x8) [reset = 0x0]	473
5.2.4	SYSEXCIC Register (Offset = 0xC) [reset = 0x0]	474
6	Hibernation Module	475
6.1	Introduction	476
6.2	Block Diagram	477
6.3	Functional Description	477

6.3.1	Register Access Timing	478
6.3.2	Hibernation Clock Source.....	478
6.3.3	System Implementation	480
6.3.4	Battery Management	481
6.3.5	Real-Time Clock	482
6.3.6	Tamper	484
6.3.7	Battery-Backed Memory	488
6.3.8	Power Control Using HIB	488
6.3.9	Power Control Using VDD3ON Mode	488
6.3.10	Initiating Hibernate.....	488
6.3.11	Waking from Hibernate	488
6.3.12	Arbitrary Power Removal	489
6.3.13	Interrupts and Status	490
6.4	Initialization and Configuration	490
6.4.1	Initialization	490
6.4.2	RTC Match Functionality (No Hibernation)	491
6.4.3	RTC Match and Wake From Hibernation	491
6.4.4	External Wake From Hibernation	491
6.4.5	RTC or External Wake From Hibernation	492
6.4.6	Tamper Initialization	492
6.5	HIB Registers.....	493
6.5.1	HIBRTCC Register (Offset = 0x0) [reset = 0x0]	495
6.5.2	HIBRTCM0 Register (Offset = 0x4) [reset = 0xFFFFFFFF].....	496
6.5.3	HIBRTCLD Register (Offset = 0xC) [reset = 0x0]	497
6.5.4	HIBCTL Register (Offset = 0x10) [reset = 0x80002000].....	498
6.5.5	HIBIM Register (Offset = 0x14) [reset = 0x0]	502
6.5.6	HIBRIS Register (Offset = 0x18) [reset = 0x0].....	504
6.5.7	HIBMIS Register (Offset = 0x1C) [reset = 0x0]	506
6.5.8	HIBIC Register (Offset = 0x20) [reset = 0x0].....	508
6.5.9	HIBRTCT Register (Offset = 0x24) [reset = 0x7FFF]	510
6.5.10	HIBRTCSS Register (Offset = 0x28) [reset = 0x0]	511
6.5.11	HIBIO Register (Offset = 0x2C) [reset = 0x80000000]	512
6.5.12	HIBDATA Register (Offset = 0x030 to 0x06F) [reset = X].....	513
6.5.13	HIBCALCTL Register (Offset = 0x300) [reset = 0x0]	514
6.5.14	HIBCAL0 Register (Offset = 0x310) [reset = 0x0]	515
6.5.15	HIBCAL1 Register (Offset = 0x314) [reset = 0x0]	516
6.5.16	HIBCALD0 Register (Offset = 0x320) [reset = 0x0]	517
6.5.17	HIBCALD1 Register (Offset = 0x324) [reset = 0x0]	518
6.5.18	HIBCALM0 Register (Offset = 0x330) [reset = 0x0].....	519
6.5.19	HIBCALM1 Register (Offset = 0x334) [reset = 0x0].....	520
6.5.20	HIBLOCK Register (Offset = 0x360) [reset = 0x0]	521
6.5.21	HIBTPCTL Register (Offset = 0x400) [reset = 0x0]	522
6.5.22	HIBTPSTAT Register (Offset = 0x404) [reset = 0x0]	524
6.5.23	HIBTPIO Register (Offset = 0x410) [reset = 0x0]	525
6.5.24	HIBTPLOG0, HIBTPLOG2, HIBTPLOG4, HIBTPLOG6 Registers (Offset = 0x4E0 to 0x4F8) [reset = 0x0]	527
6.5.25	HIBTPLOG1, HIBTPLOG3, HIBTPLOG5, HIBTPLOG7 Registers (Offset = 0x4E4 to 0x4FC) [reset = 0x0]	528
6.5.26	HIBPP Register (Offset = 0xFC0) [reset = 0x2]	529
6.5.27	HIBCC Register (Offset = 0xFC8) [reset = 0x0]	530
7	Internal Memory.....	531
7.1	Block Diagram.....	532
7.2	Functional Description.....	532
7.2.1	SRAM	533

7.2.2	ROM	533
7.2.3	Flash Memory	535
7.2.4	EEPROM.....	544
7.2.5	Bus Matrix Memory Accesses.....	549
7.3	Flash Registers	550
7.3.1	FMA Register (Offset = 0x0) [reset = 0x0].....	551
7.3.2	FMD Register (Offset = 0x4) [reset = 0x0]	552
7.3.3	FMC Register (Offset = 0x8) [reset = 0x0]	553
7.3.4	FCRIS Register (Offset = 0xC) [reset = 0x0].....	555
7.3.5	FCIM Register (Offset = 0x10) [reset = 0x0]	557
7.3.6	FCMISC Register (Offset = 0x14) [reset = 0x0]	559
7.3.7	FMC2 Register (Offset = 0x20) [reset = 0x0]	561
7.3.8	FWBVAL Register (Offset = 0x30) [reset = 0x0].....	562
7.3.9	FLPEKEY Register (Offset = 0x3C) [reset = 0xFFFF]	563
7.3.10	FWB0 to FWB31 Registers (Offset = 0x100 to 0x17C) [reset = 0x0].....	564
7.3.11	FLASHPP Register (Offset = 0xFC0) [reset = 0x701401FF]	565
7.3.12	SSIZE Register (Offset = 0xFC4) [reset = 0x3FF]	566
7.3.13	FLASHCONF Register (Offset = 0xFC8) [reset = 0x0]	567
7.3.14	ROMSWMAP Register (Offset = 0xFCC) [reset = 0x0].....	568
7.3.15	FLASHDMASZ Register (Offset = 0xFD0) [reset = 0x0]	569
7.3.16	FLASHDMAST Register (Offset = 0xFD4) [reset = 0x0]	570
7.4	EEPROM Registers	571
7.4.1	EESIZE Register (Offset = 0x0) [reset = 0x00600600]	572
7.4.2	EEBLOCK Register (Offset = 0x4) [reset = 0x0]	573
7.4.3	EEOFFSET Register (Offset = 0x8) [reset = 0x0].....	574
7.4.4	EERDWR Register (Offset = 0x10) [reset = X]	575
7.4.5	EERDWRINC Register (Offset = 0x14) [reset = X].....	576
7.4.6	EEDONE Register (Offset = 0x18) [reset = 0x0]	577
7.4.7	EESUPP Register (Offset = 0x1C) [reset = X].....	579
7.4.8	EEUNLOCK Register (Offset = 0x20) [reset = X]	580
7.4.9	EEPROT Register (Offset = 0x30) [reset = 0x0].....	581
7.4.10	EEPASS0 to EEPASS2 Registers (Offset = 0x34 to 0x3C) [reset = X]	582
7.4.11	EEINT Register (Offset = 0x40) [reset = 0x0].....	583
7.4.12	EEHIDE0 Register (Offset = 0x50) [reset = 0x0]	584
7.4.13	EEHIDE1 Register (Offset = 0x54) [reset = 0x0]	585
7.4.14	EEHIDE2 Register (Offset = 0x58) [reset = 0x0]	586
7.4.15	EEDBGME Register (Offset = 0x80) [reset = 0x0]	587
7.4.16	EEPROMPP Register (Offset = 0xFC0) [reset = 0x1FF].....	588
7.5	System Control Memory Registers	589
7.5.1	RVP Register (Offset = 0xD4) [reset = 0x0101FFF0].....	590
7.5.2	BOOTCFG Register (Offset = 0x1D0) [reset = 0xFFFFFFFF].....	591
7.5.3	USER_REG0 to USER_REG3 Registers (Offset = 0x1E0 to 0x1EC) [reset = 0xFFFFFFFF]	594
7.5.4	FMPRE0 to FMPRE15 Registers (Offset = 0x200 to 0x23C) [reset = 0xFFFFFFFF]	595
7.5.5	FMPPE0 to FMPPE15 Registers (Offset = 0x400 to 0x43C) [reset = 0xFFFFFFFF].....	597
8	Micro Direct Memory Access (μDMA)	599
8.1	Introduction	600
8.2	Block Diagram.....	600
8.3	Functional Description.....	601
8.3.1	Priority	602
8.3.2	Arbitration Size	602
8.3.3	Request Types	602
8.3.4	Channel Configuration	603
8.3.5	Transfer Modes	604

8.3.6	Transfer Size and Increment	612
8.3.7	Peripheral Interface.....	612
8.3.8	Software Request.....	613
8.3.9	Interrupts and Errors	613
8.4	Initialization and Configuration	613
8.4.1	Module Initialization.....	613
8.4.2	Configuring a Memory-to-Memory Transfer	614
8.4.3	Configuring a Peripheral for Simple Transmit.....	615
8.4.4	Configuring a Peripheral for Ping-Pong Receive	616
8.4.5	Configuring Channel Assignments	618
8.5	μDMA Channel Control Structure Registers	620
8.5.1	DMASRCENDP Register (Offset = 0x0) [reset = X].....	621
8.5.2	DMADSTENDP Register (Offset = 0x4) [reset = X]	622
8.5.3	DMACHCTL Register (Offset = 0x8) [reset = X].....	623
8.6	μDMA Registers.....	626
8.6.1	DMASTAT Register (Offset = 0x0) [reset = 0x001F0000].....	628
8.6.2	DMACFG Register (Offset = 0x4) [reset = X]	629
8.6.3	DMACTLBASE Register (Offset = 0x8) [reset = 0x0]	630
8.6.4	DMAALTBASE Register (Offset = 0xC) [reset = 0x200].....	631
8.6.5	DMAWAITSTAT Register (Offset = 0x10) [reset = 0x03C3CF00].....	632
8.6.6	DMASWREQ Register (Offset = 0x14) [reset = X]	633
8.6.7	DMAUSEBURSTSET Register (Offset = 0x18) [reset = 0x0]	634
8.6.8	DMAUSEBURSTCLR Register (Offset = 0x1C) [reset = X].....	635
8.6.9	DMAREQMASKSET Register (Offset = 0x20) [reset = 0x0]	636
8.6.10	DMAREQMASKCLR Register (Offset = 0x24) [reset = X]	637
8.6.11	DMAENASET Register (Offset = 0x28) [reset = 0x0]	638
8.6.12	DMAENACLAR Register (Offset = 0x2C) [reset = X]	639
8.6.13	DMAALTSET Register (Offset = 0x30) [reset = 0x0]	640
8.6.14	DMAALTCLR Register (Offset = 0x34) [reset = X]	641
8.6.15	DMAPRIOSET Register (Offset = 0x38) [reset = 0x0]	642
8.6.16	DMAPRIOCLR Register (Offset = 0x3C) [reset = X]	643
8.6.17	DMAERRCLR Register (Offset = 0x4C) [reset = 0x0]	644
8.6.18	DMACHMAP0 Register (Offset = 0x510) [reset = 0x0]	645
8.6.19	DMACHMAP1 Register (Offset = 0x514) [reset = 0x0]	646
8.6.20	DMACHMAP2 Register (Offset = 0x518) [reset = 0x0]	647
8.6.21	DMACHMAP3 Register (Offset = 0x51C) [reset = 0x0]	648
8.6.22	DMAPeriphID4 Register (Offset = 0xFD0) [reset = 0x4].....	649
8.6.23	DMAPeriphID0 Register (Offset = 0xFE0) [reset = 0x30]	650
8.6.24	DMAPeriphID1 Register (Offset = 0xFE4) [reset = 0xB2]	651
8.6.25	DMAPeriphID2 Register (Offset = 0xFE8) [reset = 0xB]	652
8.6.26	DMAPeriphID3 Register (Offset = 0xFEC) [reset = 0x0]	653
8.6.27	DMAPCellID0 Register (Offset = 0xFF0) [reset = 0xD].....	654
8.6.28	DMAPCellID1 Register (Offset = 0xFF4) [reset = 0xF0].....	655
8.6.29	DMAPCellID2 Register (Offset = 0xFF8) [reset = 0x5]	656
8.6.30	DMAPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]	657
9	Advance Encryption Standard Accelerator (AES)	658
9.1	AES Overview	659
9.2	AES Functional Description	659
9.2.1	AES Block Diagram	659
9.2.2	AES Algorithm.....	662
9.2.3	AES Operating Modes	663
9.2.4	AES Software Reset.....	671
9.2.5	Power Management	671

9.2.6	Hardware Requests	671
9.3	AES Performance Information	672
9.4	AES Module Programming Guide	674
9.4.1	AES Low - Level Programming Models.....	674
9.5	AES Registers.....	679
9.5.1	AES_KEYn_n Register (Offset = 0x000 to 0x03C) [reset = 0x0].....	681
9.5.2	AES_IV_IN_0 to AES_IV_IN_3 Registers (Offset = 0x40 to 0x4C) [reset = 0x0]	682
9.5.3	AES_CTRL Register (Offset = 0x50) [reset = 0x80000000]	683
9.5.4	AES_C_LENGTH_0 to AES_C_LENGTH_1 Registers (Offset = 0x54 to 0x58) [reset = 0x0]	686
9.5.5	AES_AUTH_LENGTH Register (Offset = 0x5C) [reset = 0x0].....	687
9.5.6	AES_DATA_IN_0 to AES_DATA_IN_3 Registers (Offset = 0x60 to 0x6C) [reset = 0x0].....	688
9.5.7	AES_TAG_OUT_0 to AES_TAG_OUT_3 Registers (Offset = 0x70 to 0x7C) [reset = 0x0]	689
9.5.8	AES_REVISION Register (Offset = 0x80) [reset = 0x41]	690
9.5.9	AES_SYSCONFIG Register (Offset = 0x84) [reset = 0x1]	691
9.5.10	AES_SYSSTATUS Register (Offset = 0x88) [reset = 0x1].....	693
9.5.11	AES_IRQSTATUS Register (Offset = 0x8C) [reset = 0x0]	694
9.5.12	AES_IRQENABLE Register (Offset = 0x90) [reset = 0x0]	695
9.5.13	AES_DIRTYBITS Register (Offset = 0x94) [reset = 0x0].....	696
9.6	AES μ DMA Registers	697
9.6.1	AES_DMAIM Register (Offset = 0x20) [reset = 0x0]	698
9.6.2	AES_DMARIS Register (Offset = 0x24) [reset = 0x0]	699
9.6.3	AES_DMAMIS Register (Offset = 0x28) [reset = 0x0]	700
9.6.4	AES_DMAIC Register (Offset = 0x2C) [reset = 0x0].....	701
10	Analog-to-Digital Converter (ADC)	702
10.1	Introduction	703
10.2	Block Diagram.....	703
10.3	Functional Description.....	704
10.3.1	Sample Sequencers	704
10.3.2	Module Control	705
10.3.3	Hardware Sample Averaging Circuit	709
10.3.4	Analog-to-Digital Converter.....	710
10.3.5	Differential Sampling	712
10.3.6	Internal Temperature Sensor.....	713
10.3.7	Digital Comparator Unit	714
10.4	Initialization and Configuration	717
10.4.1	Module Initialization	717
10.4.2	Sample Sequencer Configuration	717
10.5	ADC Registers	719
10.5.1	ADCACTSS Register (Offset = 0x0) [reset = 0x0]	721
10.5.2	ADCRIS Register (Offset = 0x4) [reset = 0x0]	723
10.5.3	ADCIM Register (Offset = 0x8) [reset = 0x0]	725
10.5.4	ADCISC Register (Offset = 0xC) [reset = 0x0].....	727
10.5.5	ADCOSTAT Register (Offset = 0x10) [reset = 0x0]	730
10.5.6	ADCEMUX Register (Offset = 0x14) [reset = 0x0]	731
10.5.7	ADCUSTAT Register (Offset = 0x18) [reset = 0x0]	734
10.5.8	ADCTSSEL Register (Offset = 0x1C) [reset = 0x0]	735
10.5.9	ADCSSPRI Register (Offset = 0x20) [reset = 0x3210]	737
10.5.10	ADCSPC Register (Offset = 0x24) [reset = 0x0]	738
10.5.11	ADCPSSI Register (Offset = 0x28) [reset = X]	740
10.5.12	ADCSAC Register (Offset = 0x30) [reset = 0x0]	742
10.5.13	ADCDCISC Register (Offset = 0x34) [reset = 0x0]	743
10.5.14	ADCCTL Register (Offset = 0x38) [reset = 0x0]	745
10.5.15	ADCSSMUX0 Register (Offset = 0x40) [reset = 0x0]	746

10.5.16	ADCSSCTL0 Register (Offset = 0x44) [reset = 0x0]	748
10.5.17	ADCSSFIFO0 to ADCSSFIFO3 Registers [reset = X]	752
10.5.18	ADCSSFSTAT0 to ADCSSFSTAT3 Registers [reset = 0x00000100].....	753
10.5.19	ADCSSOP0 Register (Offset = 0x50) [reset = 0x0]	754
10.5.20	ADCSSDC0 Register (Offset = 0x54) [reset = 0x0]	755
10.5.21	ADCSSMUX0 Register (Offset = 0x58) [reset = 0x0].....	756
10.5.22	ADCSSSTSH0 Register (Offset = 0x5C) [reset = 0x0].....	758
10.5.23	ADCSSMUX1 and ADCSSMUX2 Registers [reset = 0x0]	760
10.5.24	ADCSSCTL1 and ADCSSCTL2 Registers [reset = 0x0].....	761
10.5.25	ADCSSOP1 and ADCSSOP2 Registers [reset = 0x0]	764
10.5.26	ADCSSDC1 and ADCSSDC2 Registers [reset = 0x0]	765
10.5.27	ADCSSMUX1 and ADCSSMUX2 Registers [reset = 0x0]	766
10.5.28	ADCSSSTSH1 and ADCSSSTSH2 Registers [reset = 0x0]	768
10.5.29	ADCSSMUX3 Register (Offset = 0xA0) [reset = 0x0]	769
10.5.30	ADCSSCTL3 Register (Offset = 0xA4) [reset = 0x0]	770
10.5.31	ADCSSOP3 Register (Offset = 0xB0) [reset = 0x0]	771
10.5.32	ADCSSDC3 Register (Offset = 0xB4) [reset = 0x0]	772
10.5.33	ADCSSMUX3 Register (Offset = 0xB8) [reset = 0x0]	773
10.5.34	ADCSSSTSH3 Register (Offset = 0xBC) [reset = 0x0]	774
10.5.35	ADCDCRIC Register (Offset = 0xD00) [reset = 0x0]	775
10.5.36	ADCDCCTL0 to ADCDCCTL7 Registers [reset = 0x0]	779
10.5.37	ADCDCCMP0 to ADCDCCMP7 Registers [reset = 0x0].....	781
10.5.38	ADCPP Register (Offset = 0xFC0) [reset = 0x01B02187]	782
10.5.39	ADCPC Register (Offset = 0xFC4) [reset = 0x7].....	784
10.5.40	ADCCC Register (Offset = 0xFC8) [reset = 0x1]	785
11	Controller Area Network (CAN) Module	786
11.1	Introduction	787
11.2	Block Diagram.....	787
11.3	Functional Description.....	788
11.3.1	Initialization	789
11.3.2	Operation	789
11.3.3	Transmitting Message Objects.....	790
11.3.4	Configuring a Transmit Message Object	790
11.3.5	Updating a Transmit Message Object	791
11.3.6	Accepting Received Message Objects.....	792
11.3.7	Receiving a Data Frame	792
11.3.8	Receiving a Remote Frame	792
11.3.9	Receive and Transmit Priority.....	793
11.3.10	Configuring a Receive Message Object.....	793
11.3.11	Handling of Received Message Objects	794
11.3.12	Handling of Interrupts.....	795
11.3.13	Test Mode.....	796
11.3.14	Bit Timing Configuration Error Considerations	797
11.3.15	Bit Time and Bit Rate	797
11.3.16	Calculating the Bit Timing Parameters	799
11.4	CAN Registers	803
11.4.1	CANCTL Register (Offset = 0x0) [reset = 0x1]	805
11.4.2	CANSTS Register (Offset = 0x4) [reset = 0x0]	807
11.4.3	CANERR Register (Offset = 0x8) [reset = 0x0].....	809
11.4.4	CANBIT Register (Offset = 0xC) [reset = 0x2301]	810
11.4.5	CANINT Register (Offset = 0x10) [reset = 0x0].....	811
11.4.6	CANTST Register (Offset = 0x14) [reset = X]	812
11.4.7	CANBRPE Register (Offset = 0x18) [reset = X]	813

11.4.8	CANIFnCRQ Register (Offset = 0x20) [reset = 0x1]	814
11.4.9	CANIFnCMSK Register [reset = 0x0]	815
11.4.10	CANIFnMSK1 Register [reset = 0xFFFF]	817
11.4.11	CANIFnMSK2 Register [reset = 0xE0FF]	818
11.4.12	CANIFnARB1 Register [reset = 0x0]	819
11.4.13	CANIFnARB2 Register [reset = 0x0]	820
11.4.14	CANIFnMCTL Register [reset = 0x0]	821
11.4.15	CANIFnDnn Register [reset = 0x0]	823
11.4.16	CANTXRQn Register [reset = 0x0]	824
11.4.17	CANNWDAn Register [reset = 0x0]	825
11.4.18	CANMSGnINT Register [reset = 0x0]	826
11.4.19	CANMSGnVAL Register [reset = 0x0]	827
12	Analog Comparators	828
12.1	Introduction	829
12.2	Block Diagram	830
12.3	Functional Description	830
12.3.1	Internal Reference Programming	831
12.4	Initialization and Configuration	833
12.5	Comparator Registers	834
12.5.1	ACMIS Register (Offset = 0x0) [reset = 0x0]	835
12.5.2	ACRIS Register (Offset = 0x4) [reset = 0x0]	836
12.5.3	ACINTEN Register (Offset = 0x8) [reset = 0x0]	837
12.5.4	ACREFCTL Register (Offset = 0x10) [reset = 0x0]	838
12.5.5	ACSTATn Register [reset = 0x0]	839
12.5.6	ACCTLn Register [reset = 0x0]	840
12.5.7	ACMPPP Register (Offset = 0xFC0) [reset = 0x70007]	841
13	Cyclical Redundancy Check (CRC)	842
13.1	Introduction	843
13.2	Functional Description	843
13.2.1	CRC Support	843
13.3	Initialization and Configuration	844
13.3.1	CRC Initialization and Configuration	844
13.4	CRC Registers	846
13.4.1	CRCCTRL Register (Offset = 400h) [reset = 0h]	847
13.4.2	CRCSEED Register (Offset = 410h) [reset = 0h]	849
13.4.3	CRCDIN Register (Offset = 414h) [reset = 0h]	850
13.4.4	CRCRSLTPP Register (Offset = 418h) [reset = 0h]	851
14	Data Encryption Standard Accelerator (DES)	852
14.1	Introduction	853
14.2	DES Functional Description	853
14.3	DES Block Diagram	854
14.3.1	μDMA Control	854
14.3.2	Interrupt Control	854
14.3.3	Register Interface	855
14.3.4	DES Engine	855
14.4	Software Reset	856
14.5	DES Supported Modes of Operation	856
14.5.1	ECB Feedback Mode	856
14.6	DES Module Programming Guide – Low Level Programming Models	858
14.6.1	Surrounding Modules Global Initialization	858
14.6.2	Operational Modes Configuration	859
14.6.3	DES Events Servicing	861
14.7	DES Registers	863

14.7.1	DES_KEYn_n Registers (Offset = 0x00 to 0x14) [reset = 0x0].....	864
14.7.2	DES_IV_L Register (Offset = 0x18) [reset = 0x0]	865
14.7.3	DES_IV_H Register (Offset = 0x1C) [reset = 0x0]	866
14.7.4	DES_CTRL Register (Offset = 0x20) [reset = 0x80000000]	867
14.7.5	DES_LENGTH Register (Offset = 0x24) [reset = 0x0].....	868
14.7.6	DES_DATA_L Register (Offset = 0x28) [reset = 0x0]	869
14.7.7	DES_DATA_H Register (Offset = 0x2C) [reset = 0x0].....	870
14.7.8	DES_REVISION Register (Offset = 0x30) [reset = 0x21]	871
14.7.9	DES_SYSCONFIG Register (Offset = 0x34) [reset = 0x1].....	872
14.7.10	DES_SYSSTATUS Register (Offset = 0x38) [reset = 0x1]	873
14.7.11	DES_IRQSTATUS Register (Offset = 0x3C) [reset = 0x0]	874
14.7.12	DES_IRQENABLE Register (Offset = 0x40) [reset = 0x0].....	875
14.7.13	DES_DIRTYBITS Register (Offset = 0x44) [reset = 0x0]	876
14.8	DES μ DMA Registers	877
14.8.1	DES_DMAIM Register (Offset = 0x30) [reset = 0x0].....	878
14.8.2	DES_DMARIS Register (Offset = 0x34) [reset = 0x0]	879
14.8.3	DES_DMAMIS Register (Offset = 0x38) [reset = 0x0]	880
14.8.4	DES_DMAIC Register (Offset = 0x3C) [reset = 0x0]	881
15	Ethernet Controller	882
15.1	Introduction	883
15.2	Block Diagram.....	884
15.3	Functional Description.....	884
15.3.1	Ethernet Clock Control	884
15.3.2	MII and RMII Signals	887
15.3.3	DMA Controller.....	888
15.3.4	TX/RX Controller.....	909
15.3.5	MAC Operation.....	913
15.3.6	IEEE 1588 and Advanced Timestamp Function.....	915
15.3.7	Frame Filtering	922
15.3.8	Source Address, VLAN, and CRC Insertion, Replacement or Deletion	924
15.3.9	Checksum Offload Engine.....	925
15.3.10	MAC Management Counters	926
15.3.11	Power Management Module.....	926
15.3.12	Serial Management Interface	929
15.3.13	Reduced Media Independent Interface (RMII)	929
15.3.14	Interrupt Configuration.....	929
15.4	Ethernet PHY.....	929
15.4.1	Integrated PHY Block Diagram	930
15.4.2	Functional Description	930
15.4.3	Interface Configuration	935
15.5	Initialization and Configuration	935
15.5.1	Ethernet PHY Initialization.....	936
15.6	EMAC Registers	939
15.6.1	EMACCFG Register (Offset = 0x0) [reset = 0x8000]	942
15.6.2	EMACFRAMEFLTR Register (Offset = 0x4) [reset = 0x0]	946
15.6.3	EMACHASHTBLH Register (Offset = 0x8) [reset = 0x0]	949
15.6.4	EMACHASHTBLL Register (Offset = 0xC) [reset = 0x0]	950
15.6.5	EMACMIIADDR Register (Offset = 0x10) [reset = 0x0]	951
15.6.6	EMACMIIDATA Register (Offset = 0x14) [reset = 0x0]	953
15.6.7	EMACFLOWCTL Register (Offset = 0x18) [reset = 0x0]	954
15.6.8	EMACVLANTG Register (Offset = 0x1C) [reset = 0x0].....	956
15.6.9	EMACSTATUS Register (Offset = 0x24) [reset = 0x0]	958
15.6.10	EMACRWUFF Register (Offset = 0x28) [reset = 0x0].....	960

15.6.11	EMACPMCTLSTAT Register (Offset = 0x2C) [reset = 0x0]	961
15.6.12	EMACLPICLSTAT Register (Offset = 0x30) [reset = 0x0]	963
15.6.13	EMACLPITIMERCTRL Register (Offset = 0x34) [reset = 0x0]	965
15.6.14	EMACRIS Register (Offset = 0x38) [reset = 0x0]	966
15.6.15	EMACIM Register (Offset = 0x3C) [reset = 0x0]	968
15.6.16	EMACADDR0H Register (Offset = 0x40) [reset = 0x8000FFFF]	969
15.6.17	EMACADDR0L Register (Offset = 0x44) [reset = 0xFFFFFFFF]	970
15.6.18	EMACADDR1H Register (Offset = 0x48) [reset = 0xFFFF]	971
15.6.19	EMACADDR1L Register (Offset = 0x4C) [reset = 0xFFFFFFFF]	972
15.6.20	EMACADDR2H Register (Offset = 0x50) [reset = 0xFFFF]	973
15.6.21	EMACADDR2L Register (Offset = 0x54) [reset = 0xFFFFFFFF]	974
15.6.22	EMACADDR3H Register (Offset = 0x58) [reset = 0xFFFF]	975
15.6.23	EMACADDR3L Register (Offset = 0x5C) [reset = 0xFFFFFFFF]	976
15.6.24	EMACWDOGTO Register (Offset = 0xDC) [reset = 0x0]	977
15.6.25	EMACMMCTRL Register (Offset = 0x100) [reset = 0x0]	978
15.6.26	EMACMMCRXRIS Register (Offset = 0x104) [reset = 0x0]	980
15.6.27	EMACMMCTXRIS Register (Offset = 0x108) [reset = 0x0]	982
15.6.28	EMACMMCTXIM Register (Offset = 0x10C) [reset = 0x0]	984
15.6.29	EMACMMCTXIM Register (Offset = 0x110) [reset = 0x0]	985
15.6.30	EMACTXCNTGB Register (Offset = 0x118) [reset = 0x0]	986
15.6.31	EMACTXCNTSCOL Register (Offset = 0x14C) [reset = 0x0]	987
15.6.32	EMACTXCNTMCOL Register (Offset = 0x150) [reset = 0x0]	988
15.6.33	EMACTXOCTCNTG Register (Offset = 0x164) [reset = 0x0]	989
15.6.34	EMACRXCNTGB Register (Offset = 0x180) [reset = 0x0]	990
15.6.35	EMACRXCNTCRCERR Register (Offset = 0x194) [reset = 0x0]	991
15.6.36	EMACRXCNTALGNERR Register (Offset = 0x198) [reset = 0x0]	992
15.6.37	EMACRXCNTGUNI Register (Offset = 0x1C4) [reset = 0x0]	993
15.6.38	EMACVLNINCREP Register (Offset = 0x584) [reset = 0x0]	994
15.6.39	EMACVLNHASH Register (Offset = 0x588) [reset = 0x0]	995
15.6.40	EMACTIMSTCTRL Register (Offset = 0x700) [reset = 0x2000]	996
15.6.41	EMACSUBSECINC Register (Offset = 0x704) [reset = 0x0]	999
15.6.42	EMACTIMSEC Register (Offset = 0x708) [reset = 0x0]	1000
15.6.43	EMACTIMNANO Register (Offset = 0x70C) [reset = 0x0]	1001
15.6.44	EMACTIMSECU Register (Offset = 0x710) [reset = 0x0]	1002
15.6.45	EMACTIMNANOU Register (Offset = 0x714) [reset = 0x0]	1003
15.6.46	EMACTIMADD Register (Offset = 0x718) [reset = 0x0]	1004
15.6.47	EMACTARGSEC Register (Offset = 0x71C) [reset = 0x0]	1005
15.6.48	EMACTARGNANO Register (Offset = 0x720) [reset = 0x0]	1006
15.6.49	EMACHWORDSEC Register (Offset = 0x724) [reset = 0x0]	1007
15.6.50	EMACTIMSTAT Register (Offset = 0x728) [reset = 0x0]	1008
15.6.51	EMACPPSCTRL Register (Offset = 0x72C) [reset = 0x0]	1009
15.6.52	EMACPPS0INTVL Register (Offset = 0x760) [reset = 0x0]	1012
15.6.53	EMACPPS0WIDTH Register (Offset = 0x764) [reset = 0x0]	1013
15.6.54	EMACDMABUSMOD Register (Offset = 0xC00) [reset = 0x00020101]	1014
15.6.55	EMACTXPOLL Register (Offset = 0xC04) [reset = 0x0]	1017
15.6.56	EMACRXPOLL Register (Offset = 0xC08) [reset = 0x0]	1018
15.6.57	EMACRXDLADDR Register (Offset = 0xC0C) [reset = 0x0]	1019
15.6.58	EMACTXDLADDR Register (Offset = 0xC10) [reset = 0x0]	1020
15.6.59	EMACDMARIS Register (Offset = 0xC14) [reset = 0x0]	1021
15.6.60	EMACDMAOPMODE Register (Offset = 0xC18) [reset = 0x0]	1025
15.6.61	EMACDMAIM Register (Offset = 0xC1C) [reset = 0x0]	1029
15.6.62	EMACMFBOC Register (Offset = 0xC20) [reset = 0x0]	1031
15.6.63	EMACRXINTWDT Register (Offset = 0xC24) [reset = 0x0]	1032

15.6.64	EMACHOSTXDESC Register (Offset = 0xC48) [reset = 0x0]	1033
15.6.65	EMACHOSRXDESC Register (Offset = 0xC4C) [reset = 0x0]	1034
15.6.66	EMACHOSTXBA Register (Offset = 0xC50) [reset = 0x0]	1035
15.6.67	EMACHOSRXBA Register (Offset = 0xC54) [reset = 0x0]	1036
15.6.68	EMACPP Register (Offset = 0xFC0) [reset = 0x103]	1037
15.6.69	EMACPC Register (Offset = 0xFC4) [reset = 0x0080040E]	1038
15.6.70	EMACCC Register (Offset = 0xFC8) [reset = 0x0]	1041
15.6.71	EPHYRIS Register (Offset = 0xFD0) [reset = 0x0]	1042
15.6.72	EPHYIM Register (Offset = 0xFD4) [reset = 0x0]	1043
15.6.73	EPHYMISC Register (Offset = 0xFD8) [reset = 0x0]	1044
15.7	MII Management (EPHY) Registers	1045
15.7.1	EPHYBMCR Register (Address = 0x0) [reset = 0x3100]	1047
15.7.2	EPHYBMSR Register (Address = 0x1) [reset = 0x7849]	1049
15.7.3	EPHYID1 Register (Address = 0x2) [reset = 0x2000]	1051
15.7.4	EPHYID2 Register (Address = 0x3) [reset = 0xA221]	1052
15.7.5	EPHYANA Register (Address = 0x4) [reset = 0x01E1]	1053
15.7.6	EPHYANLPA Register (Address = 0x5) [reset = 0x0]	1055
15.7.7	EPHYANER Register (Address = 0x6) [reset = 0x4]	1056
15.7.8	EPHYANNPTR Register (Address = 0x7) [reset = 0x2001]	1057
15.7.9	EPHYANLNPTR Register (Address = 0x8) [reset = 0x0]	1058
15.7.10	EPHYCFG1 Register (Address = 0x9) [reset = 0x0]	1059
15.7.11	EPHYCFG2 Register (Address = 0xA) [reset = 0x4]	1061
15.7.12	EPHYCFG3 Register (Address = 0xB) [reset = 0x0]	1062
15.7.13	EPHYREGCTL Register (Address = 0xD) [reset = 0x0]	1063
15.7.14	EPHYADDAR Register (Address = 0xE) [reset = 0x0]	1064
15.7.15	EPHYSTS Register (Address = 0x10) [reset = 0x2]	1065
15.7.16	EPHYSCR Register (Address = 0x11) [reset = 0x103]	1067
15.7.17	EPHYMISR1 Register (Address = 0x12) [reset = 0x0]	1069
15.7.18	EPHYMISR2 Register (Address = 0x13) [reset = 0x0]	1071
15.7.19	EPHYFCSCR Register (Address = 0x14) [reset = 0x0]	1073
15.7.20	EPHYRXERCNT Register (Address = 0x15) [reset = 0x0]	1074
15.7.21	EPHYBISTCR Register (Address = 0x16) [reset = 0x100]	1075
15.7.22	EPHYLEDCR Register (Address = 0x18) [reset = 0x400]	1077
15.7.23	EPHYCTL Register (Address = 0x19) [reset = 0x8000]	1078
15.7.24	EPHY10BTSC Register (Address = 0x1A) [reset = 0x0]	1079
15.7.25	EPHYBICSR1 Register (Address = 0x1B) [reset = 0x7D]	1080
15.7.26	EPHYBICSR2 Register (Address = 0x1C) [reset = 0x5EE]	1081
15.7.27	EPHYCDCR Register (Address = 0x1E) [reset = 0x102]	1082
15.7.28	EPHYRCR Register (Address = 0x1F) [reset = 0x0]	1083
15.7.29	EPHYLEDCFG Register (Address = 0x25) [reset = 0x510]	1084
16	External Peripheral Interface (EPI)	1086
16.1	Introduction	1087
16.2	EPI Block Diagram	1088
16.3	Functional Description	1088
16.3.1	Master Access to EPI	1089
16.3.2	Nonblocking Reads	1089
16.3.3	DMA Operation	1090
16.4	Initialization and Configuration	1090
16.4.1	EPI Interface Options	1091
16.4.2	SDRAM Mode	1091
16.4.3	Host Bus Mode	1095
16.4.4	General-Purpose Mode	1115
16.5	EPI Registers	1122

16.5.1	EPICFG Register (Offset = 0x0) [reset = 0x0]	1124
16.5.2	EPIBAUD Register (Offset = 0x4) [reset = 0x0]	1125
16.5.3	EPIBAUD2 Register (Offset = 0x8) [reset = 0x0]	1126
16.5.4	EPISDRAMCFG Register (Offset = 0x10) [reset = 0x82EE0000]	1127
16.5.5	EPIHB8CFG Register (Offset = 0x10) [reset = 0x0008FF00]	1129
16.5.6	EPIHB16CFG Register (Offset = 0x10) [reset = 0x0008FF00]	1132
16.5.7	EPIGPCFG Register (Offset = 0x10) [reset = 0x0]	1136
16.5.8	EPIHB8CFG2 Register (Offset = 0x14) [reset = 0x00080000]	1138
16.5.9	EPIHB16CFG2 Register (Offset = 0x14) [reset = 0x00080000]	1142
16.5.10	EPIADDRMAP Register (Offset = 0x1C) [reset = 0x0]	1147
16.5.11	EPIRSIZE0 and EPIRSIZE1 Registers [reset = 0x3]	1149
16.5.12	EPIRADDR0 and EPIRADDR1 Registers [reset = 0x0]	1150
16.5.13	EPIRPSTD0 and EPIRPSTD1 Registers [reset = 0x0]	1151
16.5.14	EPISTAT Register (Offset = 0x60) [reset = 0x0]	1152
16.5.15	EPIRFIFOCNT Register (Offset = 0x6C) [reset = X]	1154
16.5.16	EPIREADFIFO0 to EPIREADFIFO7 Registers (Offset = 0x70 to 0x8C) [reset = X]	1155
16.5.17	EPIFIFOLVL Register (Offset = 0x200) [reset = 0x33]	1156
16.5.18	EPIWFIFOCNT Register (Offset = 0x204) [reset = 0x4]	1158
16.5.19	EPIDMATXCNT Register (Offset = 0x208) [reset = 0x0]	1159
16.5.20	EPIIM Register (Offset = 0x210) [reset = 0x0]	1160
16.5.21	EPIRIS Register (Offset = 0x214) [reset = 0x4]	1161
16.5.22	EPIMIS Register (Offset = 0x218) [reset = 0x0]	1163
16.5.23	EPIEISC Register (Offset = 0x21C) [reset = 0x0]	1165
16.5.24	EPIHB8CFG3 Register (Offset = 0x308) [reset = 0x00080000]	1166
16.5.25	EPIHB16CFG3 Register (Offset = 0x308) [reset = 0x00080000]	1168
16.5.26	EPIHB8CFG4 Register (Offset = 0x30C) [reset = 0x00080000]	1170
16.5.27	EPIHB16CFG4 Register (Offset = 0x30C) [reset = 0x00080000]	1172
16.5.28	EPIHB8TIME Register (Offset = 0x310) [reset = 0x00022000]	1174
16.5.29	EPIHB16TIME Register (Offset = 0x310) [reset = 0x00022000]	1176
16.5.30	EPIHB8TIME2 Register (Offset = 0x314) [reset = 0xA000]	1178
16.5.31	EPIHB16TIME2 Register (Offset = 0x314) [reset = 0x00022000]	1180
16.5.32	EPIHB8TIME3 Register (Offset = 0x318) [reset = 0xA000]	1182
16.5.33	EPIHB16TIME3 Register (Offset = 0x318) [reset = 0x00022000]	1184
16.5.34	EPIHB8TIME4 Register (Offset = 0x31C) [reset = 0xA000]	1186
16.5.35	EPIHB16TIME4 Register (Offset = 0x31C) [reset = 0x00022000]	1188
16.5.36	EPIHBPSRAM Register (Offset = 0x360) [reset = 0x0]	1190
17	General-Purpose Input/Outputs (GPIOs)	1191
17.1	Introduction	1192
17.2	Pad Capabilities	1192
17.3	Functional Description	1193
17.3.1	Data Control	1194
17.3.2	Interrupt Control	1195
17.3.3	Mode Control	1197
17.3.4	Commit Control	1197
17.3.5	Pad Control	1197
17.3.6	Identification	1198
17.4	Initialization and Configuration	1198
17.5	GPIO Registers	1201
17.5.1	GPIODATA Register (Offset = 0x0) [reset = 0x0]	1204
17.5.2	GPIODIR Register (Offset = 0x400) [reset = 0x0]	1205
17.5.3	GPIOIS Register (Offset = 0x404) [reset = 0x0]	1206
17.5.4	GPIOIBE Register (Offset = 0x408) [reset = 0x0]	1207
17.5.5	GPIOIEV Register (Offset = 0x40C) [reset = 0x0]	1208

17.5.6	GPIOIM Register (Offset = 0x410) [reset = 0x0]	1209
17.5.7	GPIORIS Register (Offset = 0x414) [reset = 0x0]	1210
17.5.8	GPIONIS Register (Offset = 0x418) [reset = 0x0]	1211
17.5.9	GPIOICR Register (Offset = 0x41C) [reset = 0x0]	1212
17.5.10	GPIOAFSEL Register (Offset = 0x420) [reset = X]	1213
17.5.11	GPIODR2R Register (Offset = 0x500) [reset = 0xFF]	1215
17.5.12	GPIODR4R Register (Offset = 0x504) [reset = 0x0]	1216
17.5.13	GPIODR8R Register (Offset = 0x508) [reset = 0x0]	1217
17.5.14	GPIOODR Register (Offset = 0x50C) [reset = 0x0]	1218
17.5.15	GPIOPUR Register (Offset = 0x510) [reset = X]	1219
17.5.16	GPIOPDR Register (Offset = 0x514) [reset = 0x0]	1221
17.5.17	GPIOSLR Register (Offset = 0x518) [reset = 0x0]	1223
17.5.18	GPIDEN Register (Offset = 0x51C) [reset = X]	1224
17.5.19	GPIOLOCK Register (Offset = 0x520) [reset = 0x1]	1226
17.5.20	GPIOCR Register (Offset = 0x524) [reset = X]	1227
17.5.21	GPIOAMSEL Register (Offset = 0x528) [reset = 0x0]	1228
17.5.22	GPIOCTL Register (Offset = 0x52C) [reset = X]	1229
17.5.23	GPIOADCTL Register (Offset = 0x530) [reset = 0x0]	1231
17.5.24	GPIDMACTL Register (Offset = 0x534) [reset = 0x0]	1232
17.5.25	GPIONSI Register (Offset = 0x538) [reset = 0x0]	1233
17.5.26	GPIDR12R Register (Offset = 0x53C) [reset = 0x0]	1234
17.5.27	GPIOWAKEPEN Register (Offset = 0x540) [reset = 0x0]	1235
17.5.28	GPIOWAKELVL Register (Offset = 0x544) [reset = 0x0]	1236
17.5.29	GPIOWAKESTAT Register (Offset = 0x548) [reset = 0x0]	1237
17.5.30	GPIOPP Register (Offset = 0xFC0) [reset = 0x1]	1238
17.5.31	GPIOPC Register (Offset = 0xFC4) [reset = 0x0]	1239
17.5.32	GPIOPeriphID4 Register (Offset = 0xFD0) [reset = 0x0]	1241
17.5.33	GPIOPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]	1242
17.5.34	GPIOPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]	1243
17.5.35	GPIOPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]	1244
17.5.36	GPIOPeriphID0 Register (Offset = 0xFE0) [reset = 0x61]	1245
17.5.37	GPIOPeriphID1 Register (Offset = 0xFE4) [reset = 0x0]	1246
17.5.38	GPIOPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]	1247
17.5.39	GPIOPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]	1248
17.5.40	GPIOCellID0 Register (Offset = 0xFF0) [reset = 0xD]	1249
17.5.41	GPIOCellID1 Register (Offset = 0xFF4) [reset = 0xF0]	1250
17.5.42	GPIOCellID2 Register (Offset = 0xFF8) [reset = 0x5]	1251
17.5.43	GPIOCellID3 Register (Offset = 0xFFC) [reset = 0xB1]	1252
18	General-Purpose Timers	1253
18.1	Introduction	1254
18.2	Block Diagram	1254
18.3	Functional Description	1256
18.3.1	GPTM Reset Conditions	1256
18.3.2	Timer Clock Source	1257
18.3.3	Timer Modes	1257
18.3.4	Wait-for-Trigger Mode	1265
18.3.5	Synchronizing GP Timer Blocks	1266
18.3.6	DMA Operation	1267
18.3.7	ADC Operation	1267
18.3.8	Accessing Concatenated 16- or 32-Bit GPTM Register Values	1267
18.4	Initialization and Configuration	1268
18.4.1	One-Shot and Periodic Timer Mode	1268
18.4.2	Real-Time Clock (RTC) Mode	1268

18.4.3	Input Edge-Count Mode	1269
18.4.4	Input Edge Time Mode	1269
18.4.5	PWM Mode.....	1269
18.5	GPTM Registers	1271
18.5.1	GPTMCFG Register (Offset = 0x0) [reset = X].....	1273
18.5.2	GPTMTAMR Register (Offset = 0x4) [reset = 0x0]	1274
18.5.3	GPTMTBMR Register (Offset = 0x8) [reset = 0x0]	1277
18.5.4	GPTMCTL Register (Offset = 0xC) [reset = 0x0]	1280
18.5.5	GPTMSYNC Register (Offset = 0x10) [reset = 0x0].....	1282
18.5.6	GPTMIMR Register (Offset = 0x18) [reset = 0x0]	1284
18.5.7	GPTMRIS Register (Offset = 0x1C) [reset = 0x0]	1286
18.5.8	GPTMMIS Register (Offset = 0x20) [reset = X]	1288
18.5.9	GPTMICR Register (Offset = 0x24) [reset = X]	1290
18.5.10	GPTMTAILR Register (Offset = 0x28) [reset = 0xFFFFFFFF].....	1292
18.5.11	GPTMTBILR Register (Offset = 0x2C) [reset = 0xFFFF].....	1293
18.5.12	GPTMTAMATCHR Register (Offset = 0x30) [reset = 0xFFFFFFFF]	1294
18.5.13	GPTMTBMATCHR Register (Offset = 0x34) [reset = 0xFFFF].....	1295
18.5.14	GPTMTAPR Register (Offset = 0x38) [reset = 0x0].....	1296
18.5.15	GPTMTBPR Register (Offset = 0x3C) [reset = 0x0]	1297
18.5.16	GPTMTAPMR Register (Offset = 0x40) [reset = 0x0]	1298
18.5.17	GPTMTBPMR Register (Offset = 0x44) [reset = 0x0]	1299
18.5.18	GPTMTAR Register (Offset = 0x48) [reset = 0xFFFFFFFF].....	1300
18.5.19	GPTMTBR Register (Offset = 0x4C) [reset = 0xFFFF]	1301
18.5.20	GPTMTAV Register (Offset = 0x50) [reset = 0xFFFFFFFF].....	1302
18.5.21	GPTMTBV Register (Offset = 0x54) [reset = 0xFFFF].....	1303
18.5.22	GPTMRTCPD Register (Offset = 0x58) [reset = 0x7FFF].....	1304
18.5.23	GPTMTAPS Register (Offset = 0x5C) [reset = 0x0]	1305
18.5.24	GPTMTBPS Register (Offset = 0x60) [reset = 0x0].....	1306
18.5.25	GPTMDMAEV Register (Offset = 0x6C) [reset = 0x0].....	1307
18.5.26	GPTMADCEV Register (Offset = 0x70) [reset = 0x0].....	1309
18.5.27	GPTMPP Register (Offset = 0xFC0) [reset = 0x70].....	1311
18.5.28	GPTMCC Register (Offset = 0xFC8) [reset = 0x0].....	1312
19	Inter-Integrated Circuit (I²C) Interface	1313
19.1	Introduction.....	1314
19.2	Block Diagram	1315
19.3	Functional Description	1315
19.3.1	I ² C Bus Functional Overview	1316
19.3.2	Available Speed Modes	1320
19.3.3	Interrupts	1323
19.3.4	Loopback Operation	1323
19.3.5	FIFO and μ DMA Operation	1324
19.3.6	Command Sequence Flow Charts	1326
19.4	Initialization and Configuration.....	1333
19.4.1	Configure the I ² C Module to Transmit a Single Byte as a Master	1333
19.4.2	Configure the I ² C Master to High Speed Mode	1334
19.5	I2C Registers	1335
19.5.1	I2CMSA Register (Offset = 0x0) [reset = 0x0].....	1336
19.5.2	I2CMCS Register (Offset = 0x4) [reset = 0x20]	1337
19.5.3	I2CMDR Register (Offset = 0x8) [reset = 0x0]	1343
19.5.4	I2CMTPR Register (Offset = 0xC) [reset = 0x1]	1344
19.5.5	I2CMIMR Register (Offset = 0x10) [reset = 0x0].....	1345
19.5.6	I2CMRIS Register (Offset = 0x14) [reset = 0x0]	1347
19.5.7	I2CMMIS Register (Offset = 0x18) [reset = 0x0].....	1349

19.5.8	I2CMICR Register (Offset = 0x1C) [reset = 0x0]	1351
19.5.9	I2CMCR Register (Offset = 0x20) [reset = 0x0]	1353
19.5.10	I2CMCLKOCNT Register (Offset = 0x24) [reset = 0x0]	1354
19.5.11	I2CMBMON Register (Offset = 0x2C) [reset = 0x3]	1355
19.5.12	I2CMLEN Register (Offset = 0x30) [reset = 0x0]	1356
19.5.13	I2CMBCNT Register (Offset = 0x34) [reset = 0x0]	1357
19.5.14	I2CSOAR Register (Offset = 0x800) [reset = 0x0]	1358
19.5.15	I2CSCSR Register (Offset = 0x804) [reset = 0x0]	1359
19.5.16	I2CSDR Register (Offset = 0x808) [reset = 0x0]	1361
19.5.17	I2CSIMR Register (Offset = 0x80C) [reset = 0x0]	1362
19.5.18	I2CSRIS Register (Offset = 0x810) [reset = 0x0]	1364
19.5.19	I2CSMIS Register (Offset = 0x814) [reset = 0x0]	1366
19.5.20	I2CSICR Register (Offset = 0x818) [reset = 0x0]	1368
19.5.21	I2CSOAR2 Register (Offset = 0x81C) [reset = 0x0]	1369
19.5.22	I2CSACKCTL Register (Offset = 0x820) [reset = 0x0]	1370
19.5.23	I2CFIFODATA Register (Offset = 0xF00) [reset = 0x0]	1371
19.5.24	I2CFIFOCTL Register (Offset = 0xF04) [reset = 0x00040004]	1372
19.5.25	I2CFIFOSTATUS Register (Offset = 0xF08) [reset = 0x00010005]	1374
19.5.26	I2CPP Register (Offset = 0xFC0) [reset = 0x1]	1375
19.5.27	I2CPC Register (Offset = 0xFC4) [reset = 0x1]	1376
20	LCD Controller	1377
20.1	Introduction	1378
20.2	Block Diagram	1378
20.3	Functional Description	1379
20.3.1	Clocking	1379
20.3.2	LCD DMA Engine	1380
20.3.3	LIDD Bus Operation	1382
20.3.4	Raster Control	1383
20.3.5	LCD Frame Buffer	1385
20.3.6	Palette RAM	1386
20.3.7	Palette	1391
20.3.8	Grayscale and Serializer – Passive (STN) Mode	1391
20.3.9	Grayscale and Serializer – Active (TFT) Mode	1391
20.3.10	Color and Grayscale Intensities and Modulation Rates	1391
20.3.11	Summary of Color Depth	1391
20.3.12	Output Format	1392
20.3.13	Subpicture Feature	1393
20.4	Interrupts	1394
20.5	Bus Transaction Modes	1395
20.6	Initialization and Configuration	1395
20.7	LCD Registers	1397
20.7.1	LCDPID Register (Offset = 0x0) [reset = X]	1398
20.7.2	LCDCTL Register (Offset = 0x4) [reset = 0x0]	1399
20.7.3	LCDLIDDCTL Register (Offset = 0xC) [reset = 0x0]	1401
20.7.4	LIDDCS0CFG Register (Offset = 0x10) [reset = 0x00440044]	1403
20.7.5	LIDDCS0ADDR Register (Offset = 0x14) [reset = 0x0]	1404
20.7.6	LIDDCS0DATA Register (Offset = 0x18) [reset = 0x0]	1405
20.7.7	LIDDCS1CFG Register (Offset = 0x1C) [reset = 0x00440044]	1406
20.7.8	LIDDCS1ADDR Register (Offset = 0x20) [reset = 0x0]	1407
20.7.9	LIDDCS1DATA Register (Offset = 0x24) [reset = 0x0]	1408
20.7.10	LCDRASTRCTL Register (Offset = 0x28) [reset = 0x0]	1409
20.7.11	LCDRASTRTIM0 Register (Offset = 0x2C) [reset = 0x0]	1412
20.7.12	LCDRASTRTIM1 Register (Offset = 0x30) [reset = 0x0]	1413

20.7.13	LCDRASTRIM2 Register (Offset = 0x34) [reset = 0x0]	1414
20.7.14	LCDRASTRSUBP1 Register (Offset = 0x38) [reset = 0x0]	1416
20.7.15	LCDRASTRSUBP2 Register (Offset = 0x3C) [reset = 0x0]	1417
20.7.16	LCDDMACTL Register (Offset = 0x40) [reset = 0x0]	1418
20.7.17	LCDDMABAFB0 Register (Offset = 0x44) [reset = 0x0]	1420
20.7.18	LCDDMACAFB0 Register (Offset = 0x48) [reset = 0x0]	1421
20.7.19	LCDDMABAFB1 Register (Offset = 0x4C) [reset = 0x0]	1422
20.7.20	LCDDMACAFB1 Register (Offset = 0x50) [reset = 0x0]	1423
20.7.21	LCDSYSCFG Register (Offset = 0x54) [reset = 0x28]	1424
20.7.22	LCDRISET Register (Offset = 0x58) [reset = 0x0]	1425
20.7.23	LCDMISCLR Register (Offset = 0x5C) [reset = 0x0]	1427
20.7.24	LCDIM Register (Offset = 0x60) [reset = 0x0]	1429
20.7.25	LCDIENC Register (Offset = 0x64) [reset = 0x0]	1431
20.7.26	LCDCLKEN Register (Offset = 0x6C) [reset = 0x0]	1433
20.7.27	LCDCLKRESET Register (Offset = 0x70) [reset = 0x0]	1434
21	Pulse Width Modulator (PWM)	1435
21.1	Introduction	1436
21.2	Block Diagram	1437
21.3	Functional Description	1438
21.3.1	Clock Configuration	1438
21.3.2	PWM Timer	1438
21.3.3	PWM Comparators	1439
21.3.4	PWM Signal Generator	1440
21.3.5	Dead-Band Generator	1441
21.3.6	Interrupt or ADC-Trigger Selector	1441
21.3.7	Synchronization Methods	1441
21.3.8	Fault Conditions	1442
21.3.9	Output Control Block	1443
21.4	Initialization and Configuration	1443
21.5	PWM Registers	1445
21.5.1	PWMCTL Register (Offset = 0x0) [reset = 0x0]	1448
21.5.2	PWMSYNC Register (Offset = 0x4) [reset = 0x0]	1449
21.5.3	PWMENABLE Register (Offset = 0x8) [reset = 0x0]	1450
21.5.4	PWMINVERT Register (Offset = 0xC) [reset = 0x0]	1452
21.5.5	PWMFAULT Register (Offset = 0x10) [reset = 0x0]	1454
21.5.6	PWMINTEN Register (Offset = 0x14) [reset = 0x0]	1456
21.5.7	PWMRIS Register (Offset = 0x18) [reset = 0x0]	1458
21.5.8	PWMISC Register (Offset = 0x1C) [reset = 0x0]	1460
21.5.9	PWMSTATUS Register (Offset = 0x20) [reset = 0x0]	1462
21.5.10	PWMFAULTVAL Register (Offset = 0x24) [reset = 0x0]	1463
21.5.11	PWMENUPD Register (Offset = 0x28) [reset = 0x0]	1465
21.5.12	PWMnCTL Register [reset = 0x0]	1467
21.5.13	PWMnINTEN Register [reset = 0x0]	1470
21.5.14	PWMnRIS Register [reset = 0x0]	1472
21.5.15	PWMnISC Register [reset = 0x0]	1474
21.5.16	PWMnLOAD Register [reset = 0x0]	1476
21.5.17	PWMnCOUNT Register [reset = 0x0]	1477
21.5.18	PWMnCMPA Register [reset = 0x0]	1478
21.5.19	PWMnCMPB Register [reset = 0x0]	1479
21.5.20	PWMnGENA Register [reset = 0x0]	1480
21.5.21	PWMnGENB Register [reset = 0x0]	1482
21.5.22	PWMnDBCTL Register [reset = 0x0]	1484
21.5.23	PWMnDBRISE Register [reset = 0x0]	1485

21.5.24	PWMnDBFALL Register [reset = 0x0]	1486
21.5.25	PWMnFLTSRC0 Register [reset = 0x0]	1487
21.5.26	PWMnFLTSRC1 Register [reset = 0x0]	1489
21.5.27	PWMnMINFLTPER Register [reset = 0x0]	1491
21.5.28	PWMnFLTSEN Register [reset = 0x0]	1492
21.5.29	PWMnFLTSTAT0 Register [reset = 0x0]	1493
21.5.30	PWMnFLTSTAT1 Register [reset = 0x0]	1495
21.5.31	PWMPP Register (Offset = 0xFC0) [reset = 0x344]	1498
21.5.32	PWMCC Register (Offset = 0xFC8) [reset = 0x5]	1499
22	1-Wire Master Module	1500
22.1	Introduction	1501
22.2	Block Diagram	1501
22.3	Functional Description	1502
22.3.1	1-Wire Protocol	1502
22.3.2	Transport Protocol	1504
22.3.3	Overdrive	1505
22.3.4	Timing Override	1505
22.3.5	Command Protocol.....	1506
22.3.6	Search (Enumeration) and Sub-Byte	1506
22.3.7	Interrupts	1507
22.3.8	DMA.....	1507
22.3.9	1-Wire Timing	1508
22.4	Initialization and Configuration.....	1508
22.5	One-Wire Master Registers	1510
22.5.1	ONEWIRECS Register (Offset = 0x0) [reset = 0x0]	1511
22.5.2	ONEWIRETIM Register (Offset = 0x4) [reset = 0x0]	1513
22.5.3	ONEWIREDATW Register (Offset = 0x8) [reset = 0x0]	1514
22.5.4	ONEWIREDATR Register (Offset = 0xC) [reset = 0x0]	1515
22.5.5	ONEWIREIM Register (Offset = 0x100) [reset = 0x0]	1516
22.5.6	ONEWIRERIS Register (Offset = 0x104) [reset = 0x0]	1517
22.5.7	ONEWIREMIS Register (Offset = 0x108) [reset = 0x0]	1518
22.5.8	ONEWIREICR Register (Offset = 0x10C) [reset = 0x0]	1519
22.5.9	ONEWIREDMA Register (Offset = 0x120) [reset = 0x0]	1520
22.5.10	ONEWIREPP Register (Offset = 0xFC0) [reset = 0x11].....	1521
23	Quad Synchronous Serial Interface (QSSI)	1522
23.1	Introduction	1523
23.2	Block Diagram	1523
23.3	Functional Description	1524
23.3.1	Bit Rate Generation	1525
23.3.2	FIFO Operation	1525
23.3.3	Advanced, Bi- and Quad- SSI Function	1526
23.3.4	SSInFSS Function	1527
23.3.5	High Speed Clock Operation	1527
23.3.6	Interrupts	1528
23.3.7	Frame Formats	1528
23.3.8	DMA Operation	1534
23.4	Initialization and Configuration.....	1534
23.4.1	Enhanced Mode Configuration	1535
23.5	QSSI Registers	1537
23.5.1	SSICR0 Register (Offset = 0x0) [reset = 0x0]	1539
23.5.2	SSICR1 Register (Offset = 0x4) [reset = 0x0]	1541
23.5.3	SSIDR Register (Offset = 0x8) [reset = 0x0]	1543
23.5.4	SSISR Register (Offset = 0xC) [reset = 0x3]	1544

23.5.5	SSICPSR Register (Offset = 0x10) [reset = 0x0]	1545
23.5.6	SSIIM Register (Offset = 0x14) [reset = 0x0]	1546
23.5.7	SSIRIS Register (Offset = 0x18) [reset = 0x8]	1547
23.5.8	SSIMIS Register (Offset = 0x1C) [reset = 0x0]	1549
23.5.9	SSIICR Register (Offset = 0x20) [reset = 0x0]	1551
23.5.10	SSIDMACTL Register (Offset = 0x24) [reset = 0x0]	1552
23.5.11	SSIPP Register (Offset = 0xFC0) [reset = 0xD]	1553
23.5.12	SSICC Register (Offset = 0xFC8) [reset = 0x0]	1554
23.5.13	SSIPeriphID4 Register (Offset = 0xFD0) [reset = 0x0]	1555
23.5.14	SSIPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]	1556
23.5.15	SSIPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]	1557
23.5.16	SSIPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]	1558
23.5.17	SSIPeriphID0 Register (Offset = 0xFE0) [reset = 0x22]	1559
23.5.18	SSIPeriphID1 Register (Offset = 0xFE4) [reset = 0x0]	1560
23.5.19	SSIPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]	1561
23.5.20	SSIPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]	1562
23.5.21	SSIPCellID0 Register (Offset = 0xFF0) [reset = 0xD]	1563
23.5.22	SSIPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]	1564
23.5.23	SSIPCellID2 Register (Offset = 0xFF8) [reset = 0x5]	1565
23.5.24	SSIPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]	1566
24	Quadrature Encoder Interface (QEI)	1567
24.1	Introduction	1568
24.2	Block Diagram	1568
24.3	Functional Description	1570
24.4	Initialization and Configuration	1572
24.5	QEI Registers	1573
24.5.1	QEICTL Register (Offset = 0x0) [reset = 0x0]	1574
24.5.2	QEISTAT Register (Offset = 0x4) [reset = 0x0]	1576
24.5.3	QEIPPOS Register (Offset = 0x8) [reset = 0x0]	1577
24.5.4	QEIMAXPOS Register (Offset = 0xC) [reset = 0x0]	1578
24.5.5	QEILOAD Register (Offset = 0x10) [reset = 0x0]	1579
24.5.6	QEITIME Register (Offset = 0x14) [reset = 0x0]	1580
24.5.7	QEICOUNT Register (Offset = 0x18) [reset = 0x0]	1581
24.5.8	QEISPEED Register (Offset = 0x1C) [reset = 0x0]	1582
24.5.9	QEINTEN Register (Offset = 0x20) [reset = 0x0]	1583
24.5.10	QEIRIS Register (Offset = 0x24) [reset = 0x0]	1584
24.5.11	QEISC Register (Offset = 0x28) [reset = 0x0]	1585
25	SHA/MD5 Accelerator	1586
25.1	SHA/MD5 Functional Description	1587
25.1.1	SHA/MD5 Block Diagram	1587
25.1.2	Power Management	1588
25.1.3	Reset Management	1588
25.1.4	μDMA and Interrupt Requests	1588
25.1.5	Operation Description	1589
25.1.6	SHA/MD5 Performance Information	1594
25.1.7	SHA/MD5 Programming Guide	1595
25.2	SHA/MD5 Registers	1600
25.2.1	SHA_DIGEST_n and SHA_IDIGEST_n Registers (Offset = 0x000 to 0x03C) [reset = 0x0]	1603
25.2.2	SHA_DIGEST_COUNT Register (Offset = 0x40) [reset = 0x0]	1604
25.2.3	SHA_MODE Register (Offset = 0x44) [reset = 0x0]	1605
25.2.4	SHA_LENGTH Register (Offset = 0x48) [reset = 0x0]	1607
25.2.5	SHA_DATA_n_IN Registers (Offset = 0x080 to 0x0BC) [reset = 0x0]	1608
25.2.6	SHA_REVISION Register (Offset = 0x100) [reset = 0x40000C03]	1609

25.2.7	SHA_SYSCONFIG Register (Offset = 0x110) [reset = 0x1]	1610
25.2.8	SHA_SYSSTATUS Register (Offset = 0x114) [reset = 0x1]	1612
25.2.9	SHA_IRQSTATUS Register (Offset = 0x118) [reset = 0x8]	1613
25.2.10	SHA_IRQENABLE Register (Offset = 0x11C) [reset = 0x3]	1614
25.3	SHA/MD5 μ DMA Registers	1615
25.3.1	SHA_DMAIM Register (Offset = 0x10) [reset = 0x0]	1616
25.3.2	SHA_DMARIS Register (Offset = 0x14) [reset = 0x0]	1617
25.3.3	SHA_DMAMIS Register (Offset = 0x18) [reset = 0x0]	1618
25.3.4	SHA_DMAIC Register (Offset = 0x1C) [reset = 0x0]	1619
26	Universal Asynchronous Receiver/Transmitter (UART)	1620
26.1	Introduction	1621
26.2	Block Diagram	1622
26.3	Functional Description	1622
26.3.1	Transmit and Receive Logic	1622
26.3.2	Baud-Rate Generation	1623
26.3.3	Data Transmission	1624
26.3.4	Serial IR (SIR)	1624
26.3.5	ISO 7816 Support	1625
26.3.6	Modem Handshake Support	1625
26.3.7	9-Bit UART Mode	1627
26.3.8	FIFO Operation	1627
26.3.9	Interrupts	1627
26.3.10	Loopback Operation	1628
26.3.11	DMA Operation	1628
26.4	Initialization and Configuration	1629
26.5	UART Registers	1631
26.5.1	UARTDR Register (Offset = 0x0) [reset = 0x0]	1633
26.5.2	UARTSR/UARTECR Register (Offset = 0x4) [reset = 0x0]	1634
26.5.3	UARTFR Register (Offset = 0x18) [reset = 0x90]	1635
26.5.4	UARTILPR Register (Offset = 0x20) [reset = 0x0]	1637
26.5.5	UARTIBRD Register (Offset = 0x24) [reset = 0x0]	1638
26.5.6	UARTFBRD Register (Offset = 0x28) [reset = 0x0]	1639
26.5.7	UARTLCRH Register (Offset = 0x2C) [reset = 0x0]	1640
26.5.8	UARTCTL Register (Offset = 0x30) [reset = 0x300]	1642
26.5.9	UARTIFLS Register (Offset = 0x34) [reset = 0x12]	1645
26.5.10	UARTIM Register (Offset = 0x38) [reset = 0x0]	1646
26.5.11	UARTRIS Register (Offset = 0x3C) [reset = 0x0]	1649
26.5.12	UARTMIS Register (Offset = 0x40) [reset = 0x0]	1651
26.5.13	UARTICR Register (Offset = 0x44) [reset = 0x0]	1653
26.5.14	UARTDMACTL Register (Offset = 0x48) [reset = 0x0]	1655
26.5.15	UART9BITADDR Register (Offset = 0xA4) [reset = 0x0]	1656
26.5.16	UART9BITAMASK Register (Offset = 0xA8) [reset = 0xFF]	1657
26.5.17	UARTPP Register (Offset = 0xFC0) [reset = 0xF]	1658
26.5.18	UARTCC Register (Offset = 0xFC8) [reset = 0x0]	1659
26.5.19	UARTPeriphID4 Register (Offset = 0xFD0) [reset = 0x60]	1660
26.5.20	UARTPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]	1661
26.5.21	UARTPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]	1662
26.5.22	UARTPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]	1663
26.5.23	UARTPeriphID0 Register (Offset = 0xFE0) [reset = 0x11]	1664
26.5.24	UARTPeriphID1 Register (Offset = 0xFE4) [reset = 0x0]	1665
26.5.25	UARTPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]	1666
26.5.26	UARTPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]	1667
26.5.27	UARTPCellID0 Register (Offset = 0xFF0) [reset = 0xD]	1668

26.5.28	UARTPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]	1669
26.5.29	UARTPCellID2 Register (Offset = 0xFF8) [reset = 0x5]	1670
26.5.30	UARTPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]	1671
27	Universal Serial Bus (USB) Controller	1672
27.1	Introduction	1673
27.2	Block Diagram	1673
27.3	Functional Description	1674
27.3.1	Operation as a Device	1674
27.3.2	Operation as a Host	1679
27.3.3	OTG Mode	1682
27.3.4	ULPI Interface	1684
27.3.5	Link Power Management (LPM)	1684
27.3.6	USB DMA Controller	1686
27.3.7	USB Clock Structure	1692
27.4	Initialization and Configuration	1692
27.4.1	Pin Configuration	1693
27.4.2	Endpoint Configuration	1693
27.5	USB Registers	1694
27.5.1	USBFADDR Register (Offset = 0x0) [reset = 0x0]	1700
27.5.2	USBPOWER Register (Offset = 0x1) [reset = 0xA0]	1701
27.5.3	USBTXIS Register (Offset = 0x2) [reset = 0x0]	1703
27.5.4	USBRXIS Register (Offset = 0x4) [reset = 0x0]	1704
27.5.5	USBTXIE Register (Offset = 0x6) [reset = 0xFF]	1705
27.5.6	USBRXIE Register (Offset = 0x8) [reset = 0xFE]	1706
27.5.7	USBIS Register (Offset = 0xA) [reset = 0x0]	1707
27.5.8	USBIE Register (Offset = 0xB) [reset = 0x6]	1709
27.5.9	USBFRAME Register (Offset = 0xC) [reset = 0x0]	1711
27.5.10	USBEPIDX Register (Offset = 0xE) [reset = 0x0]	1712
27.5.11	USBTEST Register (Offset = 0xF) [reset = 0x0]	1713
27.5.12	USBFIFOn Register [reset = 0x0]	1715
27.5.13	USBDEVCTL Register (Offset = 0x60) [reset = 0x80]	1716
27.5.14	USBCCONF Register (Offset = 0x61) [reset = 0x0]	1717
27.5.15	USBnXFIFOSZ Register [reset = 0x0]	1718
27.5.16	USBnXFIFOADD Register [reset = 0x0]	1719
27.5.17	ULPIVBUSCTL Register (Offset = 0x70) [reset = 0x0]	1720
27.5.18	ULPIREGDATA Register (Offset = 0x74) [reset = 0x0]	1721
27.5.19	ULPIREGADDR Register (Offset = 0x75) [reset = 0x0]	1722
27.5.20	ULPIREGCTL Register (Offset = 0x76) [reset = 0x0]	1723
27.5.21	USBEPINFO Register (Offset = 0x78) [reset = 0x77]	1724
27.5.22	USBRAMINFO Register (Offset = 0x79) [reset = 0x8A]	1725
27.5.23	USBCONTIM Register (Offset = 0x7A) [reset = 0x5C]	1726
27.5.24	USBVPLEN Register (Offset = 0x7B) [reset = 0x3C]	1727
27.5.25	USBHSEOF Register (Offset = 0x7C) [reset = 0x80]	1728
27.5.26	USBFSEOF Register (Offset = 0x7D) [reset = 0x77]	1729
27.5.27	USBLSEOF Register (Offset = 0x7E) [reset = 0x72]	1730
27.5.28	USBTXFUNCADDRn Register [reset = 0x0]	1731
27.5.29	USBTXHUBADDRn Register [reset = 0x0]	1732
27.5.30	USBTXHUBPORTn Register [reset = 0x0]	1733
27.5.31	USBRXFUNCADDRn Register [reset = 0x0]	1734
27.5.32	USBRXHUBADDRn Register [reset = 0x0]	1735
27.5.33	USBRXHUBPORTn Register [reset = 0x0]	1736
27.5.34	USBCSRL0 Register (Offset = 0x102) [reset = 0x0]	1737
27.5.35	USBCSRH0 Register (Offset = 0x103) [reset = 0x0]	1740

27.5.36	USBCOUNT0 Register (Offset = 0x108) [reset = 0x0]	1742
27.5.37	USBTYPE0 Register (Offset = 0x10A) [reset = 0x0]	1743
27.5.38	USBNAKLMT Register (Offset = 0x10B) [reset = 0x0]	1744
27.5.39	USBTXMAXPn Register [reset = 0x0]	1745
27.5.40	USBTXCSSLn Register [reset = 0x0]	1746
27.5.41	USBTXCSSLn Register [reset = 0x0]	1749
27.5.42	USBRXMAXPn Register [reset = 0x0]	1751
27.5.43	USBRXCSSLn Register [reset = 0x0]	1752
27.5.44	USBRXCSSLn Register [reset = 0x0]	1755
27.5.45	USBRXCOUNTh Register [reset = 0x0]	1758
27.5.46	USBTXTYPEh Register [reset = 0x0]	1759
27.5.47	USBTXINTERVALh Register [reset = 0x0]	1760
27.5.48	USBRXTYPEh Register [reset = 0x0]	1761
27.5.49	USBRXINTERVALh Register [reset = 0x0]	1762
27.5.50	USBDMAINTR Register (Offset = 0x200) [reset = 0x0]	1763
27.5.51	USBDMACTLn Register [reset = 0x0]	1764
27.5.52	USBDMAADDRh Register [reset = 0x0]	1766
27.5.53	USBDMACOUNTh Register [reset = 0x0]	1767
27.5.54	USBRQPKTCOUNTh Register [reset = 0x0]	1768
27.5.55	USBRXDPKTBUFDIS Register (Offset = 0x340) [reset = 0x0]	1769
27.5.56	USBTXDPKTBUFDIS Register (Offset = 0x342) [reset = 0x0]	1770
27.5.57	USBCTO Register (Offset = 0x344) [reset = 0x0]	1771
27.5.58	USBHHSRTN Register (Offset = 0x346) [reset = 0x0]	1772
27.5.59	USBHSBT Register (Offset = 0x348) [reset = 0x0]	1773
27.5.60	USBLPMATTR Register (Offset = 0x360) [reset = 0x0]	1774
27.5.61	USBLPMCCTRL Register (Offset = 0x362) [reset = 0x0]	1775
27.5.62	USBLPMIM Register (Offset = 0x363) [reset = 0x0]	1777
27.5.63	USBLPMRIS Register (Offset = 0x364) [reset = 0x0]	1778
27.5.64	USBLPMFADDR Register (Offset = 0x365) [reset = 0x0]	1780
27.5.65	USBEPIC Register (Offset = 0x400) [reset = 0x0]	1781
27.5.66	USBEPICRIS Register (Offset = 0x404) [reset = 0x0]	1783
27.5.67	USBEPICM Register (Offset = 0x408) [reset = 0x0]	1784
27.5.68	USBEPICISC Register (Offset = 0x40C) [reset = 0x0]	1785
27.5.69	USBDRRIS Register (Offset = 0x410) [reset = 0x0]	1786
27.5.70	USBDRLM Register (Offset = 0x414) [reset = 0x0]	1787
27.5.71	USBDRLISC Register (Offset = 0x418) [reset = 0x0]	1788
27.5.72	USBGPCS Register (Offset = 0x41C) [reset = 0x0]	1789
27.5.73	USBVDC Register (Offset = 0x430) [reset = 0x0]	1790
27.5.74	USBVDCRIS Register (Offset = 0x434) [reset = 0x0]	1791
27.5.75	USBVDCIM Register (Offset = 0x438) [reset = 0x0]	1792
27.5.76	USBVDCISC Register (Offset = 0x43C) [reset = 0x0]	1793
27.5.77	USBPP Register (Offset = 0xFC0) [reset = 0x8F1]	1794
27.5.78	USBPC Register (Offset = 0xFC4) [reset = 0x0]	1795
27.5.79	USBCC Register (Offset = 0xFC8) [reset = 0x0]	1796
28	Watchdog Timers	1797
28.1	Introduction	1798
28.2	Block Diagram	1798
28.3	Functional Description	1799
28.3.1	Register Access Timing	1800
28.4	Initialization and Configuration	1800
28.5	WDT Registers	1801
28.5.1	WDTLOAD Register (Offset = 0x0) [reset = 0xFFFFFFFF]	1802
28.5.2	WDTVALUE Register (Offset = 0x4) [reset = 0xFFFFFFFF]	1802

28.5.3	WDTCTL Register (Offset = 0x8)	1803
28.5.4	WDTICR Register (Offset = 0xC) [reset = X]	1805
28.5.5	WDTRIS Register (Offset = 0x10) [reset = 0x0]	1806
28.5.6	WDTMIS Register (Offset = 0x14) [reset = 0x0]	1807
28.5.7	WDTTEST Register (Offset = 0x418) [reset = 0x0]	1808
28.5.8	WDTLOCK Register (Offset = 0xC00) [reset = 0x0]	1809
28.5.9	WDTPeriphID4 Register (Offset = 0xFD0) [reset = 0x0]	1810
28.5.10	WDTPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]	1810
28.5.11	WDTPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]	1811
28.5.12	WDTPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]	1811
28.5.13	WDTPeriphID0 Register (Offset = 0xFE0) [reset = 0x5]	1812
28.5.14	WDTPeriphID1 Register (Offset = 0xFE4) [reset = 0x18]	1812
28.5.15	WDTPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]	1813
28.5.16	WDTPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]	1813
28.5.17	WDTPCellID0 Register (Offset = 0xFF0) [reset = 0xD]	1814
28.5.18	WDTPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]	1814
28.5.19	WDTPCellID2 Register (Offset = 0xFF8) [reset = 0x6]	1815
28.5.20	WDTPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]	1815
Revision History		1816

List of Figures

1-1.	CPU Block Diagram.....	81
1-2.	TPIU Block Diagram	82
1-3.	Cortex-M4F Register Set.....	85
1-4.	R_0 to R_12 Register.....	86
1-5.	SP Register	86
1-6.	LR Register	87
1-7.	PC Register	87
1-8.	PSR Register	89
1-9.	PRIMASK Register.....	91
1-10.	FAULTMASK Register.....	92
1-11.	BASEPRI Register	93
1-12.	CONTROL Register.....	94
1-13.	FPSC Register.....	95
1-14.	Bit-Band Mapping.....	103
1-15.	Data Storage	104
1-16.	Vector Table	109
1-17.	Exception Stack Frame	111
2-1.	SRD Use Example.....	127
2-2.	FPU Register Bank	129
2-3.	STCTRL Register	134
2-4.	STRELOAD Register	135
2-5.	STCURRENT Register.....	136
2-6.	EN0 to EN3 Registers.....	139
2-7.	DISn Register	140
2-8.	PENDn Register	141
2-9.	UNPENDn Register.....	142
2-10.	ACTIVEn Register	143
2-11.	PRIn Register	145
2-12.	SWTRIG Register.....	146
2-13.	ACTLR Register.....	148
2-14.	CPUID Register	149
2-15.	INTCTRL Register	150
2-16.	VTABLE Register	152
2-17.	APINT Register	153
2-18.	SYSCTRL Register	155
2-19.	CFGCTRL Register.....	156
2-20.	SYSPRI1 Register	157
2-21.	SYSPRI2 Register	158
2-22.	SYSPRI3 Register	159
2-23.	SYSHNDCTRL Register	160
2-24.	FAULTSTAT Register	162
2-25.	HFAULTSTAT Register	165
2-26.	MMADDR Register	166
2-27.	FAULTADDR Register	167
2-28.	MPUTYPE Register	169
2-29.	MPUCTRL Register	170
2-30.	MPUNUMBER Register.....	172

2-31.	MPUBASEn Register	173
2-32.	MPUATTRn Register	175
2-33.	CPAC Register	178
2-34.	FPCC Register	179
2-35.	FPCA Register	181
2-36.	FPDSC Register	182
3-1.	JTAG Module Block Diagram	184
3-2.	Test Access Port State Machine.....	187
3-3.	IDCODE Register Format.....	192
3-4.	BYPASS Register Format	192
3-5.	Boundary Scan Register Format.....	193
4-1.	Basic RST Configuration.....	197
4-2.	External Circuitry to Extend POR.....	198
4-3.	Reset Circuit Controlled by Switch	198
4-4.	Power Architecture	202
4-5.	Main Clock Tree.....	205
4-6.	Module Clock Selection	213
4-7.	DID0 Register	222
4-8.	DID1 Register	224
4-9.	PTBOCTL Register	226
4-10.	RIS Register	227
4-11.	IMC Register	229
4-12.	MISC Register.....	230
4-13.	RESC Register	232
4-14.	PWRTC Register.....	234
4-15.	NMIC Register.....	235
4-16.	MOSCCTL Register	237
4-17.	RSCLKCFG Register.....	239
4-18.	MEMTIM0 Register	241
4-19.	ALTCLKCFG Register.....	244
4-20.	DSCLKCFG Register.....	245
4-21.	DIVSCLK Register	247
4-22.	SYSPROP Register	248
4-23.	PIOCCAL Register.....	250
4-24.	PIOSCSTAT Register	251
4-25.	PLLFREQ0 Register.....	252
4-26.	PLLFREQ1 Register.....	253
4-27.	PLLSTAT Register.....	254
4-28.	SLPPWRCFG Register	255
4-29.	DSLPPWRCFG Register	256
4-30.	NVMSTAT Register.....	258
4-31.	LDOSPCTL Register	259
4-32.	LDOSPCAL Register	260
4-33.	LDODPCTL Register	261
4-34.	LDODPCAL Register	262
4-35.	SDPMST Register	263
4-36.	RESBEHAVCTL Register.....	265
4-37.	HSSR Register	266
4-38.	USBPDS Register	267

4-39.	USBMPC Register	268
4-40.	EMACPDS Register	269
4-41.	EMACMPC Register	270
4-42.	LCDPDS Register	271
4-43.	LCDMPC Register	272
4-44.	CAN0PDS Register	273
4-45.	CAN0MPC Register	274
4-46.	CAN1PDS Register	275
4-47.	CAN1MPC Register	276
4-48.	PPWD Register	277
4-49.	PPTIMER Register	278
4-50.	PPGPIO Register	279
4-51.	PPDMA Register	281
4-52.	PPEPI Register	282
4-53.	PPHIB Register	283
4-54.	PPUART Register	284
4-55.	PPSSI Register	285
4-56.	PPI2C Register	286
4-57.	PPUSB Register	287
4-58.	PPEPHY Register	288
4-59.	PPCAN Register	289
4-60.	PPADC Register	290
4-61.	PPACMP Register	291
4-62.	PPPWM Register	292
4-63.	PPQEI Register	293
4-64.	PPEEPROM Register	294
4-65.	PPCCM Register	295
4-66.	PPLCD Register	296
4-67.	PPOWIRE Register	297
4-68.	PPEMAC Register	298
4-69.	PPPRB Register	299
4-70.	SRWD Register	300
4-71.	SRTIMER Register	301
4-72.	SRGPIO Register	302
4-73.	SRDMA Register	304
4-74.	SREPI Register	305
4-75.	SRHIB Register	306
4-76.	SRUART Register	307
4-77.	SRSSI Register	309
4-78.	SRI2C Register	310
4-79.	SRUSB Register	312
4-80.	SREPHY Register	313
4-81.	SRCAN Register	314
4-82.	SRADC Register	315
4-83.	SRACMP Register	316
4-84.	SRPWM Register	317
4-85.	SRQEI Register	318
4-86.	SREEEPROM Register	319
4-87.	SRCCM Register	320

4-88. SRLCD Register	321
4-89. SROWIRE Register.....	322
4-90. SREMAC Register	323
4-91. RCGCWD Register	324
4-92. RCGCTIMER Register	325
4-93. RCGCGPIO Register	326
4-94. RCGCDMA Register	328
4-95. RCGCEPI Register	329
4-96. RCGCHIB Register	330
4-97. RCGCUART Register	331
4-98. RCGCSSI Register	332
4-99. RCGCI2C Register	333
4-100. RCGCUSB Register	335
4-101. RCGCEPHY Register	336
4-102. RCGCCAN Register.....	337
4-103. RCGCADC Register.....	338
4-104. RCGCACMP Register.....	339
4-105. RCGCPWM Register	340
4-106. RCGCQEI Register	341
4-107. RCGCEEPROM Register.....	342
4-108. RCGCCCM Register	343
4-109. RCGLCD Register	344
4-110. RGCOWIRE Register	345
4-111. RGCEMAC Register.....	346
4-112. SCGCWD Register	347
4-113. SCGCTIMER Register	348
4-114. SCGCGPIO Register	350
4-115. SCGCDMA Register.....	352
4-116. SCGCEPI Register	353
4-117. SCGCHIB Register	354
4-118. SCGCUART Register	355
4-119. SCGCSSI Register	356
4-120. SCGCI2C Register	357
4-121. SCGCUSB Register	359
4-122. SCGCEPHY Register	360
4-123. SCGCCAN Register	361
4-124. SCGCADC Register	362
4-125. SCGCACMP Register.....	363
4-126. SCGCPWM Register	364
4-127. SCGCQEI Register	365
4-128. SCGCEEPROM Register.....	366
4-129. SCGCCCM Register	367
4-130. SCGLCD Register	368
4-131. SCGCOWIRE Register	369
4-132. SCGCEMAC Register.....	370
4-133. DCGCWD Register	371
4-134. DCGCTIMER Register	372
4-135. DCGCGPIO Register.....	374
4-136. DCGCDMA Register	376

4-137. DCGCEPI Register	377
4-138. DCGCHIB Register	378
4-139. DCGCUART Register	379
4-140. DCGCSSI Register	381
4-141. DCGCI2C Register	382
4-142. DCGCUSB Register	384
4-143. DCGCEPHY Register	385
4-144. DCGCCAN Register	386
4-145. DCGCADC Register	387
4-146. DCGCACMP Register	388
4-147. DCGCPWM Register	389
4-148. DCGCQEI Register	390
4-149. DCGCEEPROM Register	391
4-150. DCGCCCM Register	392
4-151. DCGCLCD Register	393
4-152. DCGCOWIRE Register	394
4-153. DCGCEMAC Register	395
4-154. PCWD Register	396
4-155. PCTIMER Register	398
4-156. PCGPIO Register	401
4-157. PCDMA Register	405
4-158. PCEPI Register	407
4-159. PCHIB Register	409
4-160. PCUART Register	411
4-161. PCSSI Register	414
4-162. PCI2C Register	416
4-163. PCUSB Register	419
4-164. PCEPHY Register	420
4-165. PCCAN Register	422
4-166. PCADC Register	424
4-167. PCACMP Register	426
4-168. PCPWM Register	428
4-169. PCQEI Register	430
4-170. PCEEPROM Register	432
4-171. PCCCM Register	434
4-172. PCLCD Register	436
4-173. PCOWIRE Register	437
4-174. PCEMAC Register	439
4-175. PRWD Register	440
4-176. PRTIMER Register	441
4-177. PRGPIO Register	443
4-178. PRDMA Register	445
4-179. PREPI Register	446
4-180. PRHIB Register	447
4-181. PRUART Register	448
4-182. PRSSI Register	450
4-183. PRI2C Register	451
4-184. PRUSB Register	453
4-185. PREPHY Register	454

4-186. PRCAN Register	455
4-187. PRADC Register	456
4-188. PRACMP Register	457
4-189. PRPWM Register	458
4-190. PRQEI Register	459
4-191. PREEPROM Register	460
4-192. PRCCM Register.....	461
4-193. PRLCD Register	462
4-194. PROWIRE Register.....	463
4-195. PREMAC Register.....	464
4-196. UNIQUEIDn Register	465
4-197. CCMCGREQ Register	467
5-1. SYSEXCRIS Register	471
5-2. SYSEXCIM Register	472
5-3. SYSEXCMIIS Register.....	473
5-4. SYSEXCIC Register.....	474
6-1. Hibernation Module Block Diagram	477
6-2. Using a Crystal as the Hibernation Clock Source with a Single Battery Source	479
6-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode	480
6-4. Using a Regulator for Both V_{DD} and V_{BAT}	481
6-5. Counter Behavior with a TRIM Value of 0x8002	484
6-6. Counter Behavior with a TRIM Value of 0x7FFC.....	484
6-7. Tamper Block Diagram.....	485
6-8. Tamper Pad With Glitch Filtering	486
6-9. HIBRTCC Register	495
6-10. HIBRTCM0 Register.....	496
6-11. HIBRTCLD Register	497
6-12. HIBCTL Register.....	498
6-13. HIBIM Register	502
6-14. HIBRIS Register	504
6-15. HIBMIS Register	506
6-16. HIBIC Register	508
6-17. HIBRTCT Register.....	510
6-18. HIBRTCSS Register.....	511
6-19. HIBIO Register	512
6-20. HIBDATA Register.....	513
6-21. HIBCALCTL Register.....	514
6-22. HIBCAL0 Register	515
6-23. HIBCAL1 Register	516
6-24. HIBCALLD0 Register.....	517
6-25. HIBCALLD1 Register.....	518
6-26. HIBCALM0 Register	519
6-27. HIBCALM1 Register	520
6-28. HIBLOCK Register.....	521
6-29. HIBTPCTL Register	522
6-30. HIBTPSTAT Register.....	524
6-31. HIBTPIO Register.....	525
6-32. HIBTPLOG0 Register	527
6-33. HIBTPLOG1 Register	528

6-34.	HIBPP Register	529
6-35.	HIBCC Register	530
7-1.	Internal Memory Block Diagram	532
7-2.	Boot Configuration Flow	534
7-3.	Flash Memory Configuration	535
7-4.	Single 256-Bit Prefetch Buffer Set.....	536
7-5.	Four 256-Bit Prefetch Buffer Configuration.....	537
7-6.	Single Cycle Access, 0 Wait States	537
7-7.	Prefetch Fills From Flash	538
7-8.	Mirror Mode Function.....	539
7-9.	FMA Register.....	551
7-10.	FMD Register.....	552
7-11.	FMC Register.....	553
7-12.	FCRIS Register	555
7-13.	FCIM Register	557
7-14.	FCMISC Register	559
7-15.	FMC2 Register	561
7-16.	FWBVAL Register	562
7-17.	FLPEKEY Register	563
7-18.	FWBn Register	564
7-19.	FLASHPP Register	565
7-20.	SSIZE Register.....	566
7-21.	FLASHCONF Register	567
7-22.	ROMSWMAP Register	568
7-23.	FLASHDMASZ Register	569
7-24.	FLASHDMAST Register	570
7-25.	EESIZE Register.....	572
7-26.	EEBLOCK Register.....	573
7-27.	EEOFFSET Register	574
7-28.	EERDWR Register	575
7-29.	EERDWRINC Register.....	576
7-30.	EEDONE Register	577
7-31.	EESUPP Register	579
7-32.	EEUNLOCK Register.....	580
7-33.	EEPROT Register	581
7-34.	EEPASS0 to EEPASS2 Registers.....	582
7-35.	EEINT Register	583
7-36.	EEHIDE0 Register.....	584
7-37.	EEHIDE1 Register.....	585
7-38.	EEHIDE2 Register.....	586
7-39.	EEDBGME Register	587
7-40.	EEPROMPP Register	588
7-41.	RVP Register	590
7-42.	Boot Configuration Flow	591
7-43.	BOOTCFG Register	592
7-44.	USER_REGn Register	594
7-45.	FMPREn Register	596
7-46.	FMPPEn Register.....	598
8-1.	μDMA Block Diagram	601

8-2.	Example of Ping-Pong μ DMA Transaction	606
8-3.	Memory Scatter-Gather, Setup and Configuration	608
8-4.	Memory Scatter-Gather, μ DMA Copy Sequence	609
8-5.	Peripheral Scatter-Gather, Setup and Configuration	610
8-6.	Peripheral Scatter-Gather, μ DMA Copy Sequence	611
8-7.	DMASRCENDP Register	621
8-8.	DMADSTENDP Register	622
8-9.	DMACHCTL Register	623
8-10.	DMASTAT Register	628
8-11.	DMACFG Register	629
8-12.	DMACTLBASE Register	630
8-13.	DMAALTBASE Register	631
8-14.	DMAWAITSTAT Register	632
8-15.	DMASWREQ Register	633
8-16.	DMAUSEBURSTSET Register	634
8-17.	DMAUSEBURSTCLR Register	635
8-18.	DMAREQMASKSET Register	636
8-19.	DMAREQMASKCLR Register	637
8-20.	DMAENASET Register	638
8-21.	DMAENACL R Register	639
8-22.	DMAALTSET Register	640
8-23.	DMAALTCLR Register	641
8-24.	DMAPRIOSET Register	642
8-25.	DMAPRIOCLR Register	643
8-26.	DMAERRCLR Register	644
8-27.	DMACHMAP0 Register	645
8-28.	DMACHMAP1 Register	646
8-29.	DMACHMAP2 Register	647
8-30.	DMACHMAP3 Register	648
8-31.	DMAPeriphID4 Register	649
8-32.	DMAPeriphID0 Register	650
8-33.	DMAPeriphID1 Register	651
8-34.	DMAPeriphID2 Register	652
8-35.	DMAPeriphID3 Register	653
8-36.	DMAPCellID0 Register	654
8-37.	DMAPCellID1 Register	655
8-38.	DMAPCellID2 Register	656
8-39.	DMAPCellID3 Register	657
9-1.	AES Block Diagram	660
9-2.	AES – ECB Feedback Mode	664
9-3.	AES – CBC Feedback Mode	664
9-4.	AES Encryption With CTR/ICM Mode	665
9-5.	AES – CFB Feedback Mode	665
9-6.	AES – F8 Mode	666
9-7.	AES – XTS Operation	667
9-8.	AES – F9 Operation	667
9-9.	AES – CBC-MAC Authentication Mode	668
9-10.	AES – GCM Operation	669
9-11.	AES – CCM Operation	670

9-12.	AES Polling Mode	676
9-13.	AES Interrupt Service	678
9-14.	AES_KEYn_n Register.....	681
9-15.	AES_IV_IN_n Register.....	682
9-16.	AES_CTRL Register	683
9-17.	AES_C_LENGTH_n Register	686
9-18.	AES_AUTH_LENGTH Register	687
9-19.	AES_DATA_IN_n Register	688
9-20.	AES_TAG_OUT_n Register.....	689
9-21.	AES_REVISION Register.....	690
9-22.	AES_SYSCONFIG Register.....	691
9-23.	AES_SYSSTATUS Register	693
9-24.	AES_IRQSTATUS Register	694
9-25.	AES_IRQENABLE Register	695
9-26.	AES_DIRTYBITS Register	696
9-27.	AES_DMAIM Register.....	698
9-28.	AES_DMARIS Register	699
9-29.	AES_DMAMIS Register.....	700
9-30.	AES_DMAIC Register.....	701
10-1.	Implementation of Two ADC Blocks.....	703
10-2.	ADC Module Block Diagram.....	704
10-3.	ADC Sample Phases	708
10-4.	Doubling the ADC Sample Rate	708
10-5.	Skewed Sampling.....	709
10-6.	Sample Averaging Example	710
10-7.	ADC Input Equivalency	710
10-8.	ADC Voltage Reference	711
10-9.	ADC Conversion Result.....	711
10-10.	Differential Voltage Representation	713
10-11.	Internal Temperature Sensor Characteristic	713
10-12.	Low-Band Operation (CIC = 0x0 or CTC = 0x0).....	716
10-13.	Mid-Band Operation (CIC = 0x1 or CTC = 0x1)	716
10-14.	High-Band Operation (CIC = 0x3 or CTC = 0x3)	717
10-15.	ADCACTSS Register.....	721
10-16.	ADCRIS Register	723
10-17.	ADCIM Register.....	725
10-18.	ADCISC Register	727
10-19.	ADCOSTAT Register.....	730
10-20.	ADCEMUX Register	731
10-21.	ADCUSTAT Register	734
10-22.	ADCTSSEL Register	735
10-23.	ADCSSPRI Register.....	737
10-24.	ADCSPC Register	738
10-25.	ADCPSSI Register.....	740
10-26.	ADCSAC Register	742
10-27.	ADCDCISC Register	743
10-28.	ADCCTL Register.....	745
10-29.	ADCSSMUX0 Register.....	746
10-30.	ADCSSCTL0 Register.....	748

10-31. ADCSSFIFOn Register	752
10-32. ADCSSFSTATn Register	753
10-33. ADCSSOP0 Register	754
10-34. ADCSSDC0 Register	755
10-35. ADCSSEMUX0 Register	756
10-36. ADCSSTSH0 Register	758
10-37. ADCSSMUXn Register	760
10-38. ADCSSCTLn Register	761
10-39. ADCSSOP1 Register	764
10-40. ADCSSDCn Register	765
10-41. ADCSSEMUXn Register	766
10-42. ADCSSTSHn Register	768
10-43. ADCSSMUX3 Register	769
10-44. ADCSSCTL3 Register	770
10-45. ADCSSOP3 Register	771
10-46. ADCSSDC3 Register	772
10-47. ADCSSEMUX3 Register	773
10-48. ADCSSTSH3 Register	774
10-49. ADCDCRIC Register	775
10-50. ADCDCCTLn Register	779
10-51. ADCDCCMPn Register	781
10-52. ADCPP Register	782
10-53. ADCPC Register	784
10-54. ADCCC Register	785
11-1. CAN Controller Block Diagram	787
11-2. CAN Data Frame or Remote Frame	788
11-3. Message Objects in a FIFO Buffer	795
11-4. CAN Bit Time	798
11-5. CANCTL Register	805
11-6. CANSTS Register	807
11-7. CANERR Register	809
11-8. CANBIT Register	810
11-9. CANINT Register	811
11-10. CANTST Register	812
11-11. CANBRPE Register	813
11-12. CANIFnCRQ Register	814
11-13. CANIFnCMSK Register	815
11-14. CANIFnMSK1 Register	817
11-15. CANIFnMSK2 Register	818
11-16. CANIFnARB1 Register	819
11-17. CANIFnARB2 Register	820
11-18. CANIFnMCTL Register	821
11-19. CANIFnDnn Register	823
11-20. CANTXRQn Register	824
11-21. CANNWDAn Register	825
11-22. CANMSGnINT Register	826
11-23. CANMSGnVAL Register	827
12-1. Analog Comparator Module Block Diagram	830
12-2. Structure of Comparator Unit	831

12-3. Comparator Internal Reference Structure	831
12-4. ACMIS Register	835
12-5. ACRIS Register	836
12-6. ACINTEN Register.....	837
12-7. ACREFTL Register	838
12-8. ACSTATn Register	839
12-9. ACCTLn Register	840
12-10. ACMPPP Register	841
13-1. CRCCTRL Register.....	847
13-2. CRCSEED Register	849
13-3. CRCDIN Register	850
13-4. CRCRSLTPP Register	851
14-1. DES Block Diagram	854
14-2. DES – ECB Feedback Mode	856
14-3. DES3DES – CBC Feedback Mode.....	857
14-4. DES3DES – CFB Feedback Mode	857
14-5. DES Polling Mode	860
14-6. DES Interrupt Service	861
14-7. DES Context Input Event Service	862
14-8. DES_KEYn_n Register	864
14-9. DES_IV_L Register	865
14-10. DES_IV_H Register	866
14-11. DES_CTRL Register	867
14-12. DES_LENGTH Register	868
14-13. DES_DATA_L Register	869
14-14. DES_DATA_H Register	870
14-15. DES_REVISION Register	871
14-16. DES_SYSCONFIG Register	872
14-17. DES_SYSSTATUS Register	873
14-18. DES_IRQSTATUS Register	874
14-19. DES_IRQENABLE Register	875
14-20. DES_DIRTYBITS Register	876
14-21. DES_DMAIM Register	878
14-22. DES_DMARIS Register	879
14-23. DES_DMAMIS Register.....	880
14-24. DES_DMAIC Register.....	881
15-1. Ethernet MAC With Integrated PHY Interface	884
15-2. Ethernet MAC and PHY Clock Structure	885
15-3. MII Clock Structure	886
15-4. RMII Clock Structure	887
15-5. Enhanced Transmit Descriptor Structure	891
15-6. Enhanced Receive Descriptor Structure.....	896
15-7. TX DMA Default Operation Using Descriptors	902
15-8. TX DMA OSF Mode Operation Using Descriptors	904
15-9. RX DMA Operation Flow.....	907
15-10. Networked Time Synchronization.....	916
15-11. System Time Update Using Fine Correction Method	917
15-12. Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction	920
15-13. Wake-Up Frame Filter Register Bank.....	927

15-14. Integrated PHY Diagram	930
15-15. Interface to Ethernet Jack	935
15-16. EMACCFG Register	942
15-17. EMACFRAMEFLTR Register	946
15-18. EMACHASHTBLH Register	949
15-19. EMACHASHTBLL Register	950
15-20. EMACMIIADDR Register	951
15-21. EMACMIIDATA Register	953
15-22. EMACFLOWCTL Register	954
15-23. EMACVLANTG Register	956
15-24. EMACSTATUS Register	958
15-25. EMACRWUFF Register	960
15-26. EMACPMCTCTLSTAT Register	961
15-27. EMACLPICTLSTAT Register	963
15-28. EMACLPITIMERCTRL Register	965
15-29. EMACRIS Register	966
15-30. EMACIM Register	968
15-31. EMACADDR0H Register	969
15-32. EMACADDR0L Register	970
15-33. EMACADDR1H Register	971
15-34. EMACADDR1L Register	972
15-35. EMACADDR2H Register	973
15-36. EMACADDR2L Register	974
15-37. EMACADDR3H Register	975
15-38. EMACADDR3L Register	976
15-39. EMACWDOGTO Register	977
15-40. EMACMMCCTRL Register	978
15-41. EMACMMCRXRIS Register	980
15-42. EMACMMCTXRIS Register	982
15-43. EMACMMCRXIM Register	984
15-44. EMACMMCTXIM Register	985
15-45. EMACTXCNTGB Register	986
15-46. EMACTXCNTSCOL Register	987
15-47. EMACTXCNTMCOL Register	988
15-48. EMACTXOCTCNTG Register	989
15-49. EMACRXCNTGB Register	990
15-50. EMACRXCNTCRCERR Register	991
15-51. EMACRXCNTALGNERR Register	992
15-52. EMACRXCNTGUNI Register	993
15-53. EMACVLNINCREP Register	994
15-54. EMACVLNHASH Register	995
15-55. EMACTIMSTCTRL Register	996
15-56. EMACSUBSECINC Register	999
15-57. EMACTIMSEC Register	1000
15-58. EMACTIMNANO Register	1001
15-59. EMACTIMSECU Register	1002
15-60. EMACTIMNANOU Register	1003
15-61. EMACTIMADD Register	1004
15-62. EMACTARGSEC Register	1005

15-63. EMACTARGNANO Register	1006
15-64. EMACHWORDSEC Register	1007
15-65. EMACTIMSTAT Register	1008
15-66. EMACPPSCTRL Register	1009
15-67. EMACPPS0INTVL Register	1012
15-68. EMACPPS0WIDTH Register	1013
15-69. EMACDMABUSMOD Register	1014
15-70. EMACTXPOLLD Register	1017
15-71. EMACRXPOLLD Register	1018
15-72. EMACRXDLADDR Register	1019
15-73. EMACTXDLADDR Register	1020
15-74. EMACDMARIS Register	1021
15-75. EMACDMAOPMODE Register	1025
15-76. EMACDMAIM Register	1029
15-77. EMACMFOBC Register	1031
15-78. EMACRXINTWDT Register	1032
15-79. EMACHOSTXDESC Register	1033
15-80. EMACHOSRXDESC Register	1034
15-81. EMACHOSTXBA Register	1035
15-82. EMACHOSRXBA Register	1036
15-83. EMACPP Register	1037
15-84. EMACPC Register	1038
15-85. EMACCC Register	1041
15-86. EPHYRIS Register	1042
15-87. EPHYIM Register	1043
15-88. EPHYMISC Register	1044
15-89. EPHYBMCR Register	1047
15-90. EPHYBMSR Register	1049
15-91. EPHYID1 Register	1051
15-92. EPHYID2 Register	1052
15-93. EPHYANA Register	1053
15-94. EPHYANLPA Register	1055
15-95. EPHYANER Register	1056
15-96. EPHYANNPTR Register	1057
15-97. EPHYANLNPTR Register	1058
15-98. EPHYCFG1 Register	1059
15-99. EPHYCFG2 Register	1061
15-100. EPHYCFG3 Register	1062
15-101. EPHYREGCTL Register	1063
15-102. EPHYADDAR Register	1064
15-103. EPHYSTS Register	1065
15-104. EPHYSCR Register	1067
15-105. EPHYMISR1 Register	1069
15-106. EPHYMISR2 Register	1071
15-107. EPHYFCSCR Register	1073
15-108. EPHYRXERCNT Register	1074
15-109. EPHYBISTCR Register	1075
15-110. EPHYLEDCR Register	1077
15-111. EPHYCTL Register	1078

15-112. EPHY10BTSC Register	1079
15-113. EPHYBICSR1 Register	1080
15-114. EPHYBICSR2 Register	1081
15-115. EPHYCDCR Register	1082
15-116. EPHYRCR Register	1083
15-117. EPHYLEDCFG Register	1084
16-1. EPI Block Diagram	1088
16-2. SDRAM Nonblocking Read Cycle.....	1093
16-3. SDRAM Normal Read Cycle.....	1094
16-4. SDRAM Write Cycle	1095
16-5. iRDY Access Stalls, IRDYDLY = 01, 10, 11	1105
16-6. iRDY Signal Connection.....	1105
16-7. PSRAM Burst Read.....	1107
16-8. PSRAM Burst Write	1108
16-9. Read Delay During Refresh Event	1108
16-10. Write Delay During Refresh Event	1109
16-11. Example Schematic for Muxed Host-Bus 16 Mode.....	1110
16-12. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0	1113
16-13. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0	1113
16-14. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0	1113
16-15. Host-Bus Write Cycle with Multiplexed Address and Data and ALE With Dual or Quad CSn	1114
16-16. Continuous Read Mode Accesses	1114
16-17. Write Followed by Read to External FIFO	1115
16-18. Two-Entry FIFO.....	1115
16-19. Single-Cycle Single Write Access, FRM50 = 0, FRMCNT = 0, WR2CYC = 0	1118
16-20. Two-Cycle Read, Write Accesses, FRM50 = 0, FRMCNT = 0, WR2CYC = 1	1118
16-21. Read Accesses, FRM50 = 0, FRMCNT = 0	1119
16-22. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 0	1119
16-23. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 1	1119
16-24. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 2	1120
16-25. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 0	1120
16-26. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 1	1120
16-27. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 2	1120
16-28. EPI Clock Operation, CLKGATE = 1, WR2CYC = 0	1121
16-29. EPI Clock Operation, CLKGATE = 1, WR2CYC = 1	1121
16-30. EPICFG Register	1124
16-31. EPIBAUD Register	1125
16-32. EPIBAUD2 Register.....	1126
16-33. EPISDRAMCFG Register	1127
16-34. EPIHB8CFG Register.....	1129
16-35. EPIHB16CFG Register	1132
16-36. EPIGPCFG Register	1136
16-37. EPIHB8CFG2 Register	1138
16-38. EPIHB16CFG2 Register.....	1142
16-39. EPIADDRMAP Register	1147
16-40. EPIRSIZE _n Register	1149
16-41. EPIRADDR _n Register	1150
16-42. EPIRPSTD _n Register	1151
16-43. EPISTAT Register.....	1152

16-44. EPIRFIFOCNT Register	1154
16-45. EPIREADFIFO Register	1155
16-46. EPIFIFOLVL Register.....	1156
16-47. EPIWFIFOCNT Register	1158
16-48. EPIDMATXCNT Register.....	1159
16-49. EPIIM Register.....	1160
16-50. EPIRIS Register	1161
16-51. EPIMIS Register	1163
16-52. EPIEISC Register	1165
16-53. EPIHB8CFG3 Register	1166
16-54. EPIHB16CFG3 Register.....	1168
16-55. EPIHB8CFG4 Register	1170
16-56. EPIHB16CFG4 Register.....	1172
16-57. EPIHB8TIME Register	1174
16-58. EPIHB16TIME Register	1176
16-59. EPIHB8TIME2 Register	1178
16-60. EPIHB16TIME2 Register	1180
16-61. EPIHB8TIME3 Register	1182
16-62. EPIHB16TIME3 Register	1184
16-63. EPIHB8TIME4 Register	1186
16-64. EPIHB16TIME4 Register	1188
16-65. EPIHBPSRAM Register	1190
17-1. Digital I/O Pads	1193
17-2. Analog and Digital I/O Pads	1194
17-3. GPIODATA Write Example	1195
17-4. GPIODATA Read Example	1195
17-5. GPIODATA Register	1204
17-6. GPIODIR Register.....	1205
17-7. GPIOIS Register.....	1206
17-8. GPIOIBE Register	1207
17-9. GPIOIEV Register	1208
17-10. GPIOIM Register	1209
17-11. GPIORIS Register.....	1210
17-12. GPIOMIS Register	1211
17-13. GPIOICR Register.....	1212
17-14. GPIOAFSEL Register.....	1214
17-15. GPIODR2R Register	1215
17-16. GPIODR4R Register	1216
17-17. GPIODR8R Register	1217
17-18. GPIOODR Register	1218
17-19. GPIOPUR Register	1220
17-20. GPIOPDR Register	1222
17-21. GPIOSLR Register	1223
17-22. GPIODEN Register	1225
17-23. GPIOLOCK Register	1226
17-24. GPIOCR Register	1227
17-25. GPIOAMSEL Register	1228
17-26. GPIOPCTL Register	1230
17-27. GPIOADCCTL Register	1231

17-28. GPIODMACTL Register	1232
17-29. GPIOI Register.....	1233
17-30. GPIODR12R Register	1234
17-31. GPIOWAKEPEN Register.....	1235
17-32. GPIOWAKELVL Register	1236
17-33. GPIOWAKESTAT Register	1237
17-34. GPIOPP Register.....	1238
17-35. GPIOPC Register	1240
17-36. GPIOPeriphID4 Register	1241
17-37. GPIOPeriphID5 Register	1242
17-38. GPIOPeriphID6 Register	1243
17-39. GPIOPeriphID7 Register	1244
17-40. GPIOPeriphID0 Register	1245
17-41. GPIOPeriphID1 Register	1246
17-42. GPIOPeriphID2 Register	1247
17-43. GPIOPeriphID3 Register	1248
17-44. GPIOPCellID0 Register.....	1249
17-45. GPIOPCellID1 Register.....	1250
17-46. GPIOPCellID2 Register.....	1251
17-47. GPIOPCellID3 Register.....	1252
18-1. GPTM Module Block Diagram	1255
18-2. Input Edge-Count Mode Example, Counting Down	1261
18-3. 16-Bit Input Edge-Time Mode Example	1262
18-4. 16-Bit PWM Mode Example	1263
18-5. CCP Output, GPTMTnMATCHR > GPTMTnILR	1264
18-6. CCP Output, GPTMTnMATCHR = GPTMTnILR	1264
18-7. CCP Output, GPTMTnILR > GPTMTnMATCHR	1265
18-8. Timer Daisy Chain	1265
18-9. GPTMCFG Register	1273
18-10. GPTMTAMR Register	1274
18-11. GPTMTBMR Register	1277
18-12. GPTMCTL Register	1280
18-13. GPTMSYNC Register.....	1282
18-14. GPTMIMR Register	1284
18-15. GPTMRIS Register.....	1286
18-16. GPTMMIS Register	1288
18-17. GPTMICR Register	1290
18-18. GPTMTAILR Register	1292
18-19. GPTMTBILR Register	1293
18-20. GPTMTAMATCHR Register	1294
18-21. GPTMTBMATCHR Register	1295
18-22. GPTMTAPR Register.....	1296
18-23. GPTMTBPR Register	1297
18-24. GPTMTAPMR Register.....	1298
18-25. GPTMTBPMR Register.....	1299
18-26. GPTMTAR Register.....	1300
18-27. GPTMTBR Register.....	1301
18-28. GPTMTAV Register	1302
18-29. GPTMTBV Register	1303

18-30. GPTMRTCPD Register	1304
18-31. GPTMTAPS Register	1305
18-32. GPTMTBPS Register	1306
18-33. GPTMDMAEV Register.....	1307
18-34. GPTMADCEV Register.....	1309
18-35. GPTMPP Register.....	1311
18-36. GPTMCC Register	1312
19-1. I ² C Block Diagram	1315
19-2. I ² C Bus Configuration	1316
19-3. START and STOP Conditions	1316
19-4. Complete Data Transfer With a 7-Bit Address	1317
19-5. R/S Bit in First Byte	1317
19-6. Data Validity During Bit Transfer on the I ² C Bus	1317
19-7. High-Speed Data Format.....	1322
19-8. Master Single Transmit.....	1326
19-9. Master Single Receive.....	1327
19-10. Master Transmit of Multiple Data Bytes	1328
19-11. Master Receive of Multiple Data Bytes.....	1329
19-12. Master Receive With Repeated START After Master Transmit.....	1330
19-13. Master Transmit With Repeated START After Master Receive.....	1330
19-14. Standard High-Speed Mode Master Transmit	1331
19-15. Slave Command Sequence	1332
19-16. I2CMSA Register	1336
19-17. I2CMCS Register — Read-Only Status Register	1337
19-18. I2CMCS Register — Write-Only Control Register	1338
19-19. I2CMDR Register.....	1343
19-20. I2CMTPR Register	1344
19-21. I2CMIMR Register.....	1345
19-22. I2CMRIS Register	1347
19-23. I2CMMIS Register.....	1349
19-24. I2CMICR Register	1351
19-25. I2CMCR Register.....	1353
19-26. I2CMCLKOCNT Register.....	1354
19-27. I2CMBMON Register	1355
19-28. I2CMBLEN Register	1356
19-29. I2CMBCNT Register	1357
19-30. I2CSOAR Register	1358
19-31. I2CSCSR Register — Read-Only Status Register.....	1359
19-32. I2CSCSR Register — Write-Only Control Register	1360
19-33. I2CSDR Register	1361
19-34. I2CSIMR Register	1362
19-35. I2CSRIS Register	1364
19-36. I2CSMIS Register	1366
19-37. I2CSICR Register	1368
19-38. I2CSOAR2 Register.....	1369
19-39. I2CSACKCTL Register	1370
19-40. I2CFIFODATA Register	1371
19-41. I2CFIFOCTL Register	1372
19-42. I2CFIFOSTATUS Register	1374

19-43. I2CPP Register	1375
19-44. I2CPC Register	1376
20-1. LCD Block Diagram	1379
20-2. Input and Output Clocks.....	1379
20-3. LCD Raster Data Path.....	1385
20-4. Palette RAM Structure for 1, 2, and 4 Bits Per Pixel	1386
20-5. 24-bpp Packed Data Format.....	1388
20-6. 24-bpp Unpacked Data Format.....	1388
20-7. 24-bpp Color RGB Remapping on LCDDATA[23:0]	1389
20-8. 16-bpp Data Format.....	1389
20-9. 16-bpp Color Component Ordering	1389
20-10. 12-bpp Data Format.....	1390
20-11. 12-bpp Color Component Ordering	1390
20-12. 1-, 2-, 4-, and 8-bpp Dither Output	1390
20-13. Monochrome and Color Output.....	1393
20-14. Example of Subpicture.....	1394
20-15. Subpicture Output With HOLS Bit Variations	1394
20-16. LCDPID Register	1398
20-17. LCDCTL Register	1399
20-18. LCDLIDCTL Register	1401
20-19. LIDDCS0CFG Register.....	1403
20-20. LIDDCS0ADDR Register	1404
20-21. LIDDCS0DATA Register	1405
20-22. LIDDCS1CFG Register.....	1406
20-23. LIDDCS1ADDR Register	1407
20-24. LIDDCS1DATA Register	1408
20-25. LCDRASTRCTL Register	1409
20-26. LCDRASTRTIM0 Register	1412
20-27. LCDRASTRTIM1 Register	1413
20-28. LCDRASTRTIM2 Register	1414
20-29. LCDRASTRSUBP1 Register.....	1416
20-30. LCDRASTRSUBP2 Register.....	1417
20-31. LCDDMACTL Register	1418
20-32. LCDDMABAFB0 Register	1420
20-33. LCDDMACAFB0 Register	1421
20-34. LCDDMABAFB1 Register	1422
20-35. LCDDMACAFB1 Register.....	1423
20-36. LCDSYSCFG Register	1424
20-37. LCDRISSET Register.....	1425
20-38. LCDMISCLR Register	1427
20-39. LCDIM Register.....	1429
20-40. LCDIENC Register	1431
20-41. LCDCLKEN Register.....	1433
20-42. LCDCLKRESET Register	1434
21-1. PWM Module Diagram.....	1437
21-2. PWM Generator Block Diagram	1438
21-3. PWM Count Down Mode	1439
21-4. PWM Count Up/Down Mode.....	1440
21-5. PWM Generation Example In Count-Up/Down Mode	1440

21-6. PWM Dead-Band Generator	1441
21-7. PWMCTL Register	1448
21-8. PWMSYNC Register	1449
21-9. PWMENABLE Register	1450
21-10. PWMINVERT Register	1452
21-11. PWMFAULT Register	1454
21-12. PWMINTEN Register	1456
21-13. PWMRIS Register	1458
21-14. PWMISC Register	1460
21-15. PWMSTATUS Register	1462
21-16. PWMFAULTVAL Register	1463
21-17. PWMENUPD Register	1465
21-18. PWMnCTL Register	1467
21-19. PWMnINTEN Register	1470
21-20. PWMnRIS Register	1472
21-21. PWMnISC Register	1474
21-22. PWMnLOAD Register	1476
21-23. PWMnCOUNT Register	1477
21-24. PWMnCMPA Register	1478
21-25. PWMnCMPB Register	1479
21-26. PWMnGENA Register	1480
21-27. PWMnGENB Register	1482
21-28. PWMnDBCTL Register	1484
21-29. PWMnDBRISE Register	1485
21-30. PWMnDBFALL Register	1486
21-31. PWMnFLTSRC0 Register	1487
21-32. PWMnFLTSRC1 Register	1489
21-33. PWMnMINFLTPER Register	1491
21-34. PWMnFLTSEN Register	1492
21-35. PWMnFLTSTAT0 Register	1493
21-36. PWMnFLTSTAT1 Register	1495
21-37. PWMPP Register	1498
21-38. PWMCC Register	1499
22-1. 1-Wire Block Diagram	1501
22-2. 1-Wire Reset Protocol	1502
22-3. 1-Wire Master Transmitting a 1	1503
22-4. 1-Wire Master Transmitting a 0'	1503
22-5. 1-Wire Master Receiving a 1 Signal from a Slave	1504
22-6. 1-Wire Master Receiving a 0 Signal from a Slave	1504
22-7. ONEWIRECS Register	1511
22-8. ONEWIRETIM Register	1513
22-9. ONEWIREDATW Register	1514
22-10. ONEWIREDATR Register	1515
22-11. ONEWIREIM Register	1516
22-12. ONEWIRERIS Register	1517
22-13. ONEWIREMIS Register	1518
22-14. ONEWIREICR Register	1519
22-15. ONEWIREDMA Register	1520
22-16. ONEWIREPP Register	1521

23-1. QSSI module with Advanced, Bi-SSI and Quad-SSI Support	1524
23-2. TI Synchronous Serial Frame Format (Single Transfer)	1529
23-3. TI Synchronous Serial Frame Format (Continuous Transfer)	1530
23-4. Freescale SPI Format (Single Transfer) with SPO = 0 and SPH = 0	1530
23-5. Freescale SPI Format (Continuous Transfer) with SPO = 0 and SPH = 0	1531
23-6. Freescale SPI Frame Format with SPO = 0 and SPH = 1	1531
23-7. Freescale SPI Frame Format (Single Transfer) With SPO = 1 and SPH = 0	1532
23-8. Freescale SPI Frame Format (Continuous Transfer) With SPO = 1 and SPH = 0	1532
23-9. Freescale SPI Frame Format With SPO = 1 and SPH = 1	1533
23-10. SSICR0 Register	1539
23-11. SSICR1 Register	1541
23-12. SSIDR Register	1543
23-13. SSISR Register	1544
23-14. SSICPSR Register	1545
23-15. SSIIM Register	1546
23-16. SSIRIS Register	1547
23-17. SSIMIS Register	1549
23-18. SSIICR Register	1551
23-19. SSIDMACTL Register	1552
23-20. SSIPP Register	1553
23-21. SSICC Register	1554
23-22. SSIPeriphID4 Register	1555
23-23. SSIPeriphID5 Register	1556
23-24. SSIPeriphID6 Register	1557
23-25. SSIPeriphID7 Register	1558
23-26. SSIPeriphID0 Register	1559
23-27. SSIPeriphID1 Register	1560
23-28. SSIPeriphID2 Register	1561
23-29. SSIPeriphID3 Register	1562
23-30. SSIPCellID0 Register	1563
23-31. SSIPCellID1 Register	1564
23-32. SSIPCellID2 Register	1565
23-33. SSIPCellID3 Register	1566
24-1. QEI Block Diagram	1569
24-2. QEI Input Signal Logic	1569
24-3. Quadrature Encoder and Velocity Predivider Operation	1571
24-4. QEICTL Register	1574
24-5. QEISTAT Register	1576
24-6. QEIPOS Register	1577
24-7. QEIMAXPOS Register	1578
24-8. QEILOAD Register	1579
24-9. QEITIME Register	1580
24-10. QEICOUNT Register	1581
24-11. QEISPEED Register	1582
24-12. QEIINTEN Register	1583
24-13. QEIRIS Register	1584
24-14. QEIISC Register	1585
25-1. SHA/MD5 Module Block Diagram	1587
25-2. SHA/MD5 Polling Mode	1597

25-3. SHA/MD5 Interrupt Subroutine	1599
25-4. SHA_ODIGEST_n and SHA_IDIGEST_n Registers	1603
25-5. SHA_DIGEST_COUNT Register	1604
25-6. SHA_MODE Register	1605
25-7. SHA_LENGTH Register	1607
25-8. SHA_DATA_n_IN Register	1608
25-9. SHA_REVISION Register	1609
25-10. SHA_SYSCONFIG Register	1610
25-11. SHA_SYSSTATUS Register	1612
25-12. SHA_IRQSTATUS Register	1613
25-13. SHA_IRQENABLE Register	1614
25-14. SHA_DMAIM Register	1616
25-15. SHA_DMARIS Register	1617
25-16. SHA_DMAMIS Register	1618
25-17. SHA_DMAIC Register	1619
26-1. UART Module Block Diagram	1622
26-2. UART Character Frame	1623
26-3. IrDA Data Modulation	1625
26-4. UARTDR Register	1633
26-5. UARTRSR/UARTECR Register	1634
26-6. UARTFR Register	1635
26-7. UARTILPR Register	1637
26-8. UARTIBRD Register	1638
26-9. UARTFBRD Register	1639
26-10. UARTLCRH Register	1640
26-11. UARTCTL Register	1642
26-12. UARTIFLS Register	1645
26-13. UARTIM Register	1646
26-14. UARTRIS Register	1649
26-15. UARTMIS Register	1651
26-16. UARTICR Register	1653
26-17. UARTDMACTL Register	1655
26-18. UART9BITADDR Register	1656
26-19. UART9BITAMASK Register	1657
26-20. UARTRIS Register	1658
26-21. UARTCC Register	1659
26-22. UARTPeriphID4 Register	1660
26-23. UARTPeriphID5 Register	1661
26-24. UARTPeriphID6 Register	1662
26-25. UARTPeriphID7 Register	1663
26-26. UARTPeriphID0 Register	1664
26-27. UARTPeriphID1 Register	1665
26-28. UARTPeriphID2 Register	1666
26-29. UARTPeriphID3 Register	1667
26-30. UARTPCelIID0 Register	1668
26-31. UARTPCelIID1 Register	1669
26-32. UARTPCelIID2 Register	1670
26-33. UARTPCelIID3 Register	1671
27-1. USB Module Block Diagram	1673

27-2. USBFADDR Register	1700
27-3. USBPOWER Register (OTG A / Host)	1701
27-4. USBPOWER Register (OTG B / Device)	1702
27-5. USBTXIS Register	1703
27-6. USBRXIS Register	1704
27-7. USBTXIE Register	1705
27-8. USBRXIE Register	1706
27-9. USBIS Register (OTG A / Host)	1707
27-10. USBIS Register (OTG B / Device)	1708
27-11. USBIE Register (OTG A / Host)	1709
27-12. USBIE Register (OTG B / Device)	1710
27-13. USBFRAME Register	1711
27-14. USBEPIDX Register	1712
27-15. USBTEST Register (OTG A / Host)	1713
27-16. USBTEST Register (OTG B / Device)	1714
27-17. USBFIFOn Register	1715
27-18. USBDEVCTL Register	1716
27-19. USBCCONF Register	1717
27-20. USBnXFIFOSZ Register	1718
27-21. USBnXFIFOADD Register	1719
27-22. ULPIVBUSCTL Register	1720
27-23. ULPIREGDATA Register	1721
27-24. ULPIREGADDR Register	1722
27-25. ULPIREGCTL Register	1723
27-26. USBEPINFO Register	1724
27-27. USBRAMINFO Register	1725
27-28. USBCONTIM Register	1726
27-29. USBVPLEN Register	1727
27-30. USBHSEOF Register	1728
27-31. USBFSEOF Register	1729
27-32. USBLSEOF Register	1730
27-33. USBTXFUNCADDRn Register	1731
27-34. USBTXHUBADDRn Register	1732
27-35. USBTXHUBPORTn Register	1733
27-36. USBRXFUNCADDRn Register	1734
27-37. USBRXHUBADDRn Register	1735
27-38. USBRXHUBPORTn Register	1736
27-39. USBCSRL0 Register (OTG A / Host)	1737
27-40. USBCSRL0 Register (OTG B / Device)	1738
27-41. USBCSRH0 Register (OTG A / Host)	1740
27-42. USBCSRH0 Register (OTG B / Device)	1741
27-43. USBCOUNT0 Register	1742
27-44. USBTYPE0 Register	1743
27-45. USBNAKLMT Register	1744
27-46. USBTXMAXPn Register	1745
27-47. USBTXCSRLn Register (OTG A / Host)	1746
27-48. USBTXCSRLn Register (OTG B / Device)	1747
27-49. USBTXCSRHn Register (OTG A / Host)	1749
27-50. USBTXCSRHn Register (OTG B / Device)	1750

27-51. USBRXMAXPn Register	1751
27-52. USBRXCSRLn Register (OTG A / Host)	1752
27-53. USBRXCSRLn Register (OTG B / Device)	1753
27-54. USBRXCSRHn Register (OTG A / Host)	1755
27-55. USBRXCSRHn Register (OTG B / Device)	1756
27-56. USBRXCOUNTn Register.....	1758
27-57. USBTXTYPEn Register	1759
27-58. USBTXINTERVALn Register	1760
27-59. USBRXTYPEn Register	1761
27-60. USBRXINTERVALn Register	1762
27-61. USBDMAINTR Register	1763
27-62. USBDMACTLn Register	1764
27-63. USBDMAADDRn Register	1766
27-64. USBDMACOUNTn Register	1767
27-65. USBRQPKTCOUNTn Register	1768
27-66. USBRXDPKTBUFDIS Register.....	1769
27-67. USBTXDPKTBUFDIS Register.....	1770
27-68. USBCTO Register.....	1771
27-69. USBHHSRTN Register	1772
27-70. USBHSBT Register	1773
27-71. USBLPMATTR Register	1774
27-72. USBLPMCNTL Register (OTG A / Host)	1775
27-73. USBLPMCNTL Register (OTG B / Device)	1775
27-74. USBLPMIM Register	1777
27-75. USBLPMRIS Register (OTG A / Host)	1778
27-76. USBLPMRIS Register (OTG B / Device).....	1779
27-77. USBLPMFADDR Register.....	1780
27-78. USBEPC Register	1781
27-79. USBEPCRIS Register	1783
27-80. USBEPCIM Register	1784
27-81. USBEPCISC Register	1785
27-82. USBDRRIS Register	1786
27-83. USBDRIM Register	1787
27-84. USBDRISC Register	1788
27-85. USBGPCS Register.....	1789
27-86. USBVDC Register.....	1790
27-87. USBVDCRIS Register	1791
27-88. USBVDCIM Register.....	1792
27-89. USBVDCISC Register	1793
27-90. USBPP Register	1794
27-91. USBPC Register.....	1795
27-92. USBCC Register	1796
28-1. WDT Module Block Diagram.....	1799
28-2. WDTLOAD Register	1802
28-3. WDTVALUE Register	1802
28-4. WDTCTL Register.....	1803
28-5. WDTICR Register	1805
28-6. WDTRIS Register	1806
28-7. WDTMIS Register	1807

28-8. WDTTEST Register	1808
28-9. WDTLOCK Register	1809
28-10. WDTPeriphID4 Register	1810
28-11. WDTPeriphID5 Register	1810
28-12. WDTPeriphID6 Register	1811
28-13. WDTPeriphID7 Register	1811
28-14. WDTPeriphID0 Register	1812
28-15. WDTPeriphID1 Register	1812
28-16. WDTPeriphID2 Register	1813
28-17. WDTPeriphID3 Register	1813
28-18. WDTPCellID0 Register	1814
28-19. WDTPCellID1 Register	1814
28-20. WDTPCellID2 Register	1815
28-21. WDTPCellID3 Register	1815

List of Tables

1-1.	Summary of Processor Mode, Privilege Level, and Stack Use	84
1-2.	Cortex-M4F Registers	85
1-3.	Cortex-M4F Access Type Codes.....	86
1-4.	R_0 to R_12 Register Field Descriptions	86
1-5.	SP Register Field Descriptions	86
1-6.	LR Register Field Descriptions	87
1-7.	PC Register Field Descriptions	87
1-8.	PSR Register Combinations	88
1-9.	PSR Register Field Descriptions	89
1-10.	PRIMASK Register Field Descriptions	91
1-11.	FAULTMASK Register Field Descriptions	92
1-12.	BASEPRI Register Field Descriptions	93
1-13.	CONTROL Register Field Descriptions	94
1-14.	FPSC Register Field Descriptions	95
1-15.	Memory Map	97
1-16.	Memory Access Behavior.....	101
1-17.	SRAM Memory Bit-Banding Regions	102
1-18.	Peripheral Memory Bit-Banding Regions	102
1-19.	Exception Types	107
1-20.	Exception Return Behavior	112
1-21.	Faults	113
1-22.	Fault Status and Fault Address Registers.....	114
1-23.	Cortex-M4F Instruction Summary	116
2-1.	Core Peripheral Register Regions.....	122
2-2.	Memory Attributes Summary	125
2-3.	TEX, S, C, and B Bit Field Encoding	127
2-4.	Cache Policy for Memory Attribute Encoding	128
2-5.	AP Bit Field Encoding	128
2-6.	Memory Region Attributes for MSP432E4 Microcontrollers.....	128
2-7.	NaN Handling	131
2-8.	QNaN and SNaN Handling	131
2-9.	SYSTICK Registers.....	133
2-10.	SYSTICK Access Type Codes.....	133
2-11.	STCTRL Register Field Descriptions.....	134
2-12.	STRELOAD Register Field Descriptions.....	135
2-13.	STCURRENT Register Field Descriptions	136
2-14.	NVIC Registers.....	137
2-15.	NVIC Access Type Codes	138
2-16.	EN0 to EN3 Registers Field Descriptions	139
2-17.	DISn Register Field Descriptions	140
2-18.	PENDn Register Field Descriptions	141
2-19.	UNPENDn Register Field Descriptions	142
2-20.	ACTIVEn Register Field Descriptions	143
2-21.	PRIn Register Field Descriptions	145
2-22.	SWTRIG Register Field Descriptions	146
2-23.	SCB Registers.....	147
2-24.	SCB Access Type Codes	147

2-25.	ACTLR Register Field Descriptions	148
2-26.	CPUID Register Field Descriptions	149
2-27.	INTCTRL Register Field Descriptions	150
2-28.	VTABLE Register Field Descriptions	152
2-29.	Interrupt Priority Levels	153
2-30.	APINT Register Field Descriptions	153
2-31.	SYSCTRL Register Field Descriptions	155
2-32.	CFGCTRL Register Field Descriptions	156
2-33.	SYSPRI1 Register Field Descriptions	157
2-34.	SYSPRI2 Register Field Descriptions	158
2-35.	SYSPRI3 Register Field Descriptions	159
2-36.	SYSHNDCTRL Register Field Descriptions	160
2-37.	FAULTSTAT Register Field Descriptions	163
2-38.	HFAULTSTAT Register Field Descriptions	165
2-39.	MMADDR Register Field Descriptions	166
2-40.	FAULTADDR Register Field Descriptions	167
2-41.	MPU Registers	168
2-42.	MPU Access Type Codes	168
2-43.	MPUTYPE Register Field Descriptions	169
2-44.	MPUCTRL Register Field Descriptions	171
2-45.	MPUNUMBER Register Field Descriptions	172
2-46.	MPUBASEn Register Field Descriptions	173
2-47.	Example SIZE Field Values	175
2-48.	MPUATTRn Register Field Descriptions	176
2-49.	FPU Registers	177
2-50.	FPU Access Type Codes	177
2-51.	CPAC Register Field Descriptions	178
2-52.	FPCC Register Field Descriptions	179
2-53.	FPCA Register Field Descriptions	181
2-54.	FPDSC Register Field Descriptions	182
3-1.	JTAG Port Pins State After POR or RST Assertion	185
3-2.	JTAG Instruction Register Commands	190
4-1.	Reset Sources	195
4-2.	Clock Source Options	204
4-3.	Clock Source State Following POR	204
4-4.	System Clock Frequency	208
4-5.	Examples of System Clock Frequencies	209
4-6.	Actual PLL Frequency	210
4-7.	Peripheral Memory Power Control	214
4-8.	Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage	215
4-9.	System Control Registers	217
4-10.	System Control Access Type Codes	221
4-11.	DID0 Register Field Descriptions	222
4-12.	DID1 Register Field Descriptions	224
4-13.	PTBOCTL Register Field Descriptions	226
4-14.	RIS Register Field Descriptions	227
4-15.	IMC Register Field Descriptions	229
4-16.	MISC Register Field Descriptions	230
4-17.	RESC Register Field Descriptions	232

4-18.	PWRTC Register Field Descriptions	234
4-19.	NMIC Register Field Descriptions	235
4-20.	MOSCCTL Register Field Descriptions	237
4-21.	RSCLKCFG Register Field Descriptions	239
4-22.	MEMTIM0 Register Configuration versus Frequency	241
4-23.	MEMTIM0 Register Field Descriptions.....	242
4-24.	ALTCLKCFG Register Field Descriptions	244
4-25.	MOSC Configurations	245
4-26.	DSCLKCFG Register Field Descriptions	246
4-27.	DIVSCLK Register Field Descriptions.....	247
4-28.	SYSPROP Register Field Descriptions	248
4-29.	PIOCCAL Register Field Descriptions	250
4-30.	PIOSTAT Register Field Descriptions.....	251
4-31.	PLLREQ0 Register Field Descriptions	252
4-32.	PLLREQ1 Register Field Descriptions	253
4-33.	PLLSTAT Register Field Descriptions	254
4-34.	SLPPWRCFG Register Field Descriptions	255
4-35.	DSLPPWRCFG Register Field Descriptions	256
4-36.	NVMSTAT Register Field Descriptions	258
4-37.	Maximum System Clock and PIOC Frequency with Respect to LDO Voltage	259
4-38.	LDOSPCTL Register Field Descriptions.....	259
4-39.	LDOPCAL Register Field Descriptions.....	260
4-40.	LDODPCTL Register Field Descriptions.....	261
4-41.	LDODPCAL Register Field Descriptions.....	262
4-42.	SDPMST Register Field Descriptions	263
4-43.	RESBEHAVCTL Register Field Descriptions	265
4-44.	HSSR Register Field Descriptions.....	266
4-45.	USBPDS Register Field Descriptions	267
4-46.	USBMPC Register Field Descriptions.....	268
4-47.	EMACPDS Register Field Descriptions	269
4-48.	EMACMPC Register Field Descriptions	270
4-49.	LCDPDS Register Field Descriptions	271
4-50.	LCDMPC Register Field Descriptions	272
4-51.	CAN0PDS Register Field Descriptions	273
4-52.	CAN0MPC Register Field Descriptions	274
4-53.	CAN1PDS Register Field Descriptions	275
4-54.	CAN1MPC Register Field Descriptions	276
4-55.	PPWD Register Field Descriptions	277
4-56.	PPTIMER Register Field Descriptions	278
4-57.	PPGPIO Register Field Descriptions.....	279
4-58.	PPDMA Register Field Descriptions.....	281
4-59.	PPEPI Register Field Descriptions	282
4-60.	PPHIB Register Field Descriptions	283
4-61.	PPUART Register Field Descriptions	284
4-62.	PPSSI Register Field Descriptions	285
4-63.	PPI2C Register Field Descriptions	286
4-64.	PPUSB Register Field Descriptions	287
4-65.	PPEPHY Register Field Descriptions	288
4-66.	PPCAN Register Field Descriptions.....	289

4-67.	PPADC Register Field Descriptions	290
4-68.	PPACMP Register Field Descriptions	291
4-69.	PPPWM Register Field Descriptions	292
4-70.	PPQEI Register Field Descriptions	293
4-71.	PPEEPROM Register Field Descriptions	294
4-72.	PPCCM Register Field Descriptions	295
4-73.	PPLCD Register Field Descriptions	296
4-74.	PPOWIRE Register Field Descriptions	297
4-75.	PPEMAC Register Field Descriptions	298
4-76.	PPPRB Register Field Descriptions	299
4-77.	SRWD Register Field Descriptions	300
4-78.	SRTIMER Register Field Descriptions	301
4-79.	SRGPIO Register Field Descriptions	302
4-80.	SRDMA Register Field Descriptions	304
4-81.	SREPI Register Field Descriptions	305
4-82.	SRHIB Register Field Descriptions	306
4-83.	SRUART Register Field Descriptions	307
4-84.	SRSSI Register Field Descriptions	309
4-85.	SRI2C Register Field Descriptions	310
4-86.	SRUSB Register Field Descriptions	312
4-87.	SREPHY Register Field Descriptions	313
4-88.	SRCAN Register Field Descriptions	314
4-89.	SRADC Register Field Descriptions	315
4-90.	SRACMP Register Field Descriptions	316
4-91.	SRPWM Register Field Descriptions	317
4-92.	SRQEI Register Field Descriptions	318
4-93.	SREEEPROM Register Field Descriptions	319
4-94.	SRCCM Register Field Descriptions	320
4-95.	SRLCD Register Field Descriptions	321
4-96.	SROWIRE Register Field Descriptions	322
4-97.	SREMAC Register Field Descriptions	323
4-98.	RCGCWD Register Field Descriptions	324
4-99.	RCGCTIMER Register Field Descriptions	325
4-100.	RCGCGPIO Register Field Descriptions	326
4-101.	RCGCDMA Register Field Descriptions	328
4-102.	RCGCEPI Register Field Descriptions	329
4-103.	RCGCHIB Register Field Descriptions	330
4-104.	RCGCUART Register Field Descriptions	331
4-105.	RCGCSSI Register Field Descriptions	332
4-106.	RCGCI2C Register Field Descriptions	333
4-107.	RCGCUSB Register Field Descriptions	335
4-108.	RCGCEPHY Register Field Descriptions	336
4-109.	RCGCCAN Register Field Descriptions	337
4-110.	RCGCADC Register Field Descriptions	338
4-111.	RCGCACMP Register Field Descriptions	339
4-112.	RCGCPWM Register Field Descriptions	340
4-113.	RCGCQEI Register Field Descriptions	341
4-114.	RCGC EEPROM Register Field Descriptions	342
4-115.	RCGCCCM Register Field Descriptions	343

4-116. RCGCLCD Register Field Descriptions	344
4-117. RCGCOWIRE Register Field Descriptions	345
4-118. RCGCEMAC Register Field Descriptions	346
4-119. SCGCWD Register Field Descriptions	347
4-120. SCGCTIMER Register Field Descriptions	348
4-121. SCGCGPIO Register Field Descriptions	350
4-122. SCGCDMA Register Field Descriptions	352
4-123. SCGCEPI Register Field Descriptions	353
4-124. SCGCHIB Register Field Descriptions	354
4-125. SCGCUART Register Field Descriptions	355
4-126. SCGCSSI Register Field Descriptions	356
4-127. SCGCI2C Register Field Descriptions	357
4-128. SCGUSB Register Field Descriptions	359
4-129. SCGCEPHY Register Field Descriptions	360
4-130. SCGCCAN Register Field Descriptions	361
4-131. SCGCADC Register Field Descriptions	362
4-132. SCGCACMP Register Field Descriptions	363
4-133. SCGCPWM Register Field Descriptions	364
4-134. SCGCQEI Register Field Descriptions	365
4-135. SCGCEEPROM Register Field Descriptions	366
4-136. SCGCCCM Register Field Descriptions	367
4-137. SCGCLCD Register Field Descriptions	368
4-138. SCGCOWIRE Register Field Descriptions	369
4-139. SCGCEMAC Register Field Descriptions	370
4-140. DCGCWD Register Field Descriptions	371
4-141. DCGCTIMER Register Field Descriptions	372
4-142. DCGCGPIO Register Field Descriptions	374
4-143. DCGCDMA Register Field Descriptions	376
4-144. DCGCEPI Register Field Descriptions	377
4-145. DCGCHIB Register Field Descriptions	378
4-146. DCGCUART Register Field Descriptions	379
4-147. DCGCSSI Register Field Descriptions	381
4-148. DCGCI2C Register Field Descriptions	382
4-149. DCGUSB Register Field Descriptions	384
4-150. DCGCEPHY Register Field Descriptions	385
4-151. DCGCCAN Register Field Descriptions	386
4-152. DCGCADC Register Field Descriptions	387
4-153. DCGCACMP Register Field Descriptions	388
4-154. DCGCPWM Register Field Descriptions	389
4-155. DCGCQEI Register Field Descriptions	390
4-156. DCGCEEPROM Register Field Descriptions	391
4-157. DCGCCCM Register Field Descriptions	392
4-158. DCGCLCD Register Field Descriptions	393
4-159. DGCOWIRE Register Field Descriptions	394
4-160. DCGCEMAC Register Field Descriptions	395
4-161. Module Power Control	396
4-162. PCWD Register Field Descriptions	397
4-163. Module Power Control	398
4-164. PCTIMER Register Field Descriptions	399

4-165. Module Power Control	401
4-166. PCGPIO Register Field Descriptions.....	402
4-167. Module Power Control	405
4-168. PCDMA Register Field Descriptions	406
4-169. Module Power Control	407
4-170. PCEPI Register Field Descriptions	408
4-171. Module Power Control	409
4-172. PCHIB Register Field Descriptions	410
4-173. Module Power Control	411
4-174. PCUART Register Field Descriptions	412
4-175. Module Power Control	414
4-176. PCSSI Register Field Descriptions	415
4-177. Module Power Control	416
4-178. PCI2C Register Field Descriptions	417
4-179. Module Power Control	419
4-180. PCUSB Register Field Descriptions.....	419
4-181. Module Power Control	420
4-182. PCEPHY Register Field Descriptions	421
4-183. Module Power Control	422
4-184. PCCAN Register Field Descriptions.....	422
4-185. Module Power Control	424
4-186. PCADC Register Field Descriptions.....	425
4-187. Module Power Control	426
4-188. PCACMP Register Field Descriptions.....	427
4-189. Module Power Control	428
4-190. PCPWM Register Field Descriptions.....	429
4-191. Module Power Control	430
4-192. PCQEI Register Field Descriptions.....	431
4-193. Module Power Control	432
4-194. PCEEPROM Register Field Descriptions.....	433
4-195. Module Power Control	434
4-196. PCCCM Register Field Descriptions	435
4-197. Module Power Control	436
4-198. PCLCD Register Field Descriptions	436
4-199. Module Power Control	437
4-200. PCOWIRE Register Field Descriptions	438
4-201. Module Power Control	439
4-202. PCEMAC Register Field Descriptions.....	439
4-203. PRWD Register Field Descriptions	440
4-204. PRTIMER Register Field Descriptions	441
4-205. PRGPIO Register Field Descriptions.....	443
4-206. PRDMA Register Field Descriptions	445
4-207. PREPI Register Field Descriptions	446
4-208. PRHIB Register Field Descriptions	447
4-209. PRUART Register Field Descriptions	448
4-210. PRSSI Register Field Descriptions	450
4-211. PRI2C Register Field Descriptions	451
4-212. PRUSB Register Field Descriptions.....	453
4-213. PREPHY Register Field Descriptions	454

4-214. PRCAN Register Field Descriptions.....	455
4-215. PRADC Register Field Descriptions.....	456
4-216. PRACMP Register Field Descriptions.....	457
4-217. PRPWM Register Field Descriptions.....	458
4-218. PRQEI Register Field Descriptions.....	459
4-219. PREEPROM Register Field Descriptions.....	460
4-220. PRCCM Register Field Descriptions	461
4-221. PRLCD Register Field Descriptions	462
4-222. PROWIRE Register Field Descriptions	463
4-223. PREMAC Register Field Descriptions.....	464
4-224. UNIQUEIDn Register Field Descriptions	465
4-225. CCM Registers	466
4-226. CCM Access Type Codes	466
4-227. CCMCGREQ Register Field Descriptions	467
5-1. System Exception Registers	470
5-2. System Exception Access Type Codes.....	470
5-3. SYSEXCRIIS Register Field Descriptions.....	471
5-4. SYSEXCIM Register Field Descriptions	472
5-5. SYSEXCMIIS Register Field Descriptions	473
5-6. SYSEXCIC Register Field Descriptions	474
6-1. HIB Clock Source Configurations.....	478
6-2. HIB Registers.....	493
6-3. HIB Access Type Codes	494
6-4. HIBRTCC Register Field Descriptions	495
6-5. HIBRTCM0 Register Field Descriptions	496
6-6. HIBRTCLD Register Field Descriptions.....	497
6-7. HIBCTL Register Field Descriptions.....	499
6-8. HIBIM Register Field Descriptions.....	502
6-9. HIBRIS Register Field Descriptions	504
6-10. HIBMIS Register Field Descriptions	506
6-11. HIBIC Register Field Descriptions	508
6-12. HIBRTCT Register Field Descriptions	510
6-13. HIBRTCSS Register Field Descriptions	511
6-14. HIBIO Register Field Descriptions.....	512
6-15. HIBDATA Register Field Descriptions	513
6-16. HIBCALCTL Register Field Descriptions	514
6-17. HIBCAL0 Register Field Descriptions	515
6-18. HIBCAL1 Register Field Descriptions	516
6-19. HIBCALLD0 Register Field Descriptions	517
6-20. HIBCALLD1 Register Field Descriptions	518
6-21. HIBCALM0 Register Field Descriptions.....	519
6-22. HIBCALM1 Register Field Descriptions.....	520
6-23. HIBLOCK Register Field Descriptions	521
6-24. HIBTPCTL Register Field Descriptions	522
6-25. HIBTPSTAT Register Field Descriptions	524
6-26. HIBTPIO Register Field Descriptions	525
6-27. HIBTPLOG0 Register Field Descriptions	527
6-28. HIBTPLOG1 Register Field Descriptions	528
6-29. HIBPP Register Field Descriptions	529

6-30.	HIBCC Register Field Descriptions	530
7-1.	MEMTIM0 Register Configuration and Frequency	536
7-2.	Flash Memory Protection Policy Combinations	540
7-3.	User-Programmable Flash Memory Resident Registers	544
7-4.	MEMTIM0 Register Configuration and Frequency	546
7-5.	Master Memory Access Availability	549
7-6.	Flash Registers	550
7-7.	FLASH Access Type Codes	550
7-8.	FMA Register Field Descriptions	551
7-9.	FMD Register Field Descriptions	552
7-10.	FMC Register Field Descriptions	553
7-11.	FCRIS Register Field Descriptions	555
7-12.	FCIM Register Field Descriptions	557
7-13.	FCMISC Register Field Descriptions	559
7-14.	FMC2 Register Field Descriptions	561
7-15.	FWBVAL Register Field Descriptions	562
7-16.	FLPEKEY Register Field Descriptions	563
7-17.	FWBn Register Field Descriptions	564
7-18.	FLASHPP Register Field Descriptions	565
7-19.	SSIZE Register Field Descriptions	566
7-20.	FLASHCONF Register Field Descriptions	567
7-21.	ROMSWMAP Register Field Descriptions	568
7-22.	FLASHDMASZ Register Field Descriptions	569
7-23.	FLASHDMAST Register Field Descriptions	570
7-24.	EEPROM Registers	571
7-25.	EEPROM Access Type Codes	571
7-26.	EESIZE Register Field Descriptions	572
7-27.	EEBLOCK Register Field Descriptions	573
7-28.	EEOFFSET Register Field Descriptions	574
7-29.	EERDWR Register Field Descriptions	575
7-30.	EERDWRINC Register Field Descriptions	576
7-31.	EEDONE Register Field Descriptions	577
7-32.	EESUPP Register Field Descriptions	579
7-33.	EEUNLOCK Register Field Descriptions	580
7-34.	EEPROT Register Field Descriptions	581
7-35.	EEPASS0 to EEPASS2 Registers Field Descriptions	582
7-36.	EEINT Register Field Descriptions	583
7-37.	EEHIDE0 Register Field Descriptions	584
7-38.	EEHIDE1 Register Field Descriptions	585
7-39.	EEHIDE2 Register Field Descriptions	586
7-40.	EEDBGME Register Field Descriptions	587
7-41.	EEPROMPP Register Field Descriptions	588
7-42.	System Control Memory Registers	589
7-43.	SYSTEM_CONTROL_MEMORY Access Type Codes	589
7-44.	RVP Register Field Descriptions	590
7-45.	BOOTCFG Register Field Descriptions	592
7-46.	USER_REGn Register Field Descriptions	594
7-47.	FMPREn Register Field Descriptions	596
7-48.	FMPPEn Register Field Descriptions	598

8-1.	Request Type Support	602
8-2.	Control Structure Memory Map	604
8-3.	Channel Control Structure	604
8-4.	μDMA Read Example: 8-Bit Peripheral	612
8-5.	μDMA Interrupt Assignments.....	613
8-6.	Channel Control Structure Offsets for Channel 30	614
8-7.	Channel Control Word Configuration for Memory Transfer Example	614
8-8.	Channel Control Structure Offsets for Channel 7	615
8-9.	Channel Control Word Configuration for Peripheral Transmit Example	616
8-10.	Primary and Alternate Channel Control Structure Offsets for Channel 8	617
8-11.	Channel Control Word Configuration for Peripheral Ping-Pong Receive Example	617
8-12.	μDMA Channel Control Structure Registers	620
8-13.	μDMA Channel Control Structure Access Type Codes.....	620
8-14.	DMASRCENDP Register Field Descriptions	621
8-15.	DMADSTENDP Register Field Descriptions	622
8-16.	DMACHCTL Register Field Descriptions	623
8-17.	XFERMODE Bit Field Values	625
8-18.	μDMA Registers.....	626
8-19.	UDMA Access Type Codes	626
8-20.	DMASTAT Register Field Descriptions	628
8-21.	DMACFG Register Field Descriptions	629
8-22.	DMACTLBASE Register Field Descriptions	630
8-23.	DMAALTBASE Register Field Descriptions	631
8-24.	DMAWAITSTAT Register Field Descriptions	632
8-25.	DMASWREQ Register Field Descriptions	633
8-26.	DMAUSEBURSTSET Register Field Descriptions	634
8-27.	DMAUSEBURSTCLR Register Field Descriptions	635
8-28.	DMAREQMASKSET Register Field Descriptions	636
8-29.	DMAREQMASKCLR Register Field Descriptions	637
8-30.	DMAENASET Register Field Descriptions	638
8-31.	DMAENACLRL Register Field Descriptions	639
8-32.	DMAALTSET Register Field Descriptions	640
8-33.	DMAALTCLR Register Field Descriptions	641
8-34.	DMAPIOSET Register Field Descriptions	642
8-35.	DMAPIOCLR Register Field Descriptions	643
8-36.	DMAERRCLR Register Field Descriptions	644
8-37.	DMACHMAP0 Register Field Descriptions	645
8-38.	DMACHMAP1 Register Field Descriptions	646
8-39.	DMACHMAP2 Register Field Descriptions	647
8-40.	DMACHMAP3 Register Field Descriptions	648
8-41.	DMAPeriphID4 Register Field Descriptions	649
8-42.	DMAPeriphID0 Register Field Descriptions	650
8-43.	DMAPeriphID1 Register Field Descriptions	651
8-44.	DMAPeriphID2 Register Field Descriptions	652
8-45.	DMAPeriphID3 Register Field Descriptions	653
8-46.	DMAPCellID0 Register Field Descriptions	654
8-47.	DMAPCellID1 Register Field Descriptions	655
8-48.	DMAPCellID2 Register Field Descriptions	656
8-49.	DMAPCellID3 Register Field Descriptions	657

9-1.	Key-Block Round Combinations	663
9-2.	Interrupts and Events.....	671
9-3.	AES Module Performance (Input/Output Block Size = 128)	672
9-4.	AES Module Packet Mode Switch Overhead	673
9-5.	AES Registers.....	679
9-6.	AES Access Type Codes	679
9-7.	AES Key Register Descriptions	681
9-8.	AES_KEYn_n Register Field Descriptions	681
9-9.	AES_IV_IN_n Register Field Descriptions	682
9-10.	AES_CTRL Register Field Descriptions	683
9-11.	AES_C_LENGTH_n Register Field Descriptions	686
9-12.	AES_AUTH_LENGTH Register Field Descriptions	687
9-13.	AES_DATA_IN_n Register Field Descriptions	688
9-14.	AES_TAG_OUT_n Register Field Descriptions	689
9-15.	AES_REVISION Register Field Descriptions	690
9-16.	AES_SYSCONFIG Register Field Descriptions	691
9-17.	AES_SYSSTATUS Register Field Descriptions	693
9-18.	AES_IRQSTATUS Register Field Descriptions	694
9-19.	AES_IRQENABLE Register Field Descriptions	695
9-20.	AES_DIRTYBITS Register Field Descriptions	696
9-21.	AES μ DMA Registers	697
9-22.	AES μ DMA Access Type Codes.....	697
9-23.	AES_DMAIM Register Field Descriptions	698
9-24.	AES_DMARIS Register Field Descriptions.....	699
9-25.	AES_DMAMIS Register Field Descriptions	700
9-26.	AES_DMAIC Register Field Descriptions.....	701
10-1.	Samples and FIFO Depth of Sequencers	704
10-2.	Sample and Hold Width in ADC Clocks	706
10-3.	R_S and f_{CONV} Values with Varying N_{SH} Values and $f_{ADC} = 16$ MHz	707
10-4.	R_S and f_{CONV} Values with Varying N_{SH} Values and $f_{ADC} = 32$ MHz	707
10-5.	Differential Sampling Pairs	712
10-6.	ADC Registers	719
10-7.	ADC Access Type Codes.....	720
10-8.	ADCACTSS Register Field Descriptions	721
10-9.	ADCRIS Register Field Descriptions	723
10-10.	ADCIM Register Field Descriptions	725
10-11.	ADCISC Register Field Descriptions	727
10-12.	ADCOSTAT Register Field Descriptions	730
10-13.	ADCEMUX Register Field Descriptions.....	731
10-14.	ADCUSTAT Register Field Descriptions.....	734
10-15.	ADCTSSEL Register Field Descriptions	735
10-16.	ADCSPRI Register Field Descriptions	737
10-17.	ADCSPC Register Field Descriptions	739
10-18.	ADCPSSI Register Field Descriptions	740
10-19.	ADCSAC Register Field Descriptions	742
10-20.	ADCDISC Register Field Descriptions	743
10-21.	ADCCTL Register Field Descriptions	745
10-22.	ADCSSMUX0 Register Field Descriptions	746
10-23.	ADCSSCTL0 Register Field Descriptions	748

10-24. ADCSSFIFOn Register Field Descriptions	752
10-25. ADCSSFSTATn Register Field Descriptions	753
10-26. ADCSSOP0 Register Field Descriptions	754
10-27. ADCSSDC0 Register Field Descriptions	755
10-28. ADCSSEMUX0 Register Field Descriptions	756
10-29. Sample and Hold Width in ADC Clocks	758
10-30. ADCSSTSH0 Register Field Descriptions	758
10-31. ADCSSMUXn Register Field Descriptions	760
10-32. ADCSSCTLn Register Field Descriptions	761
10-33. ADCSSOP1 Register Field Descriptions	764
10-34. ADCSSDCn Register Field Descriptions	765
10-35. ADCSSEMUXn Register Field Descriptions	766
10-36. Sample and Hold Width in ADC Clocks	768
10-37. ADCSSTSHn Register Field Descriptions	768
10-38. ADCSSMUX3 Register Field Descriptions	769
10-39. ADCSSCTL3 Register Field Descriptions	770
10-40. ADCSSOP3 Register Field Descriptions	771
10-41. ADCSSDC3 Register Field Descriptions	772
10-42. ADCSSEMUX3 Register Field Descriptions	773
10-43. Sample and Hold Width in ADC Clocks	774
10-44. ADCSSTSH3 Register Field Descriptions	774
10-45. ADCDCRIC Register Field Descriptions	775
10-46. ADCDCCTLn Register Field Descriptions	779
10-47. ADCDCCMPn Register Field Descriptions	781
10-48. ADCPP Register Field Descriptions	782
10-49. ADCPC Register Field Descriptions	784
10-50. ADCCC Register Field Descriptions	785
11-1. Message Object Configurations	792
11-2. CAN Protocol Ranges	798
11-3. CANBIT Register Values	799
11-4. CANBIT Register Values for High Baud Rate Example	801
11-5. CANBIT Register Values for Low Baud Rate Example	802
11-6. CAN Registers	803
11-7. CAN Access Type Codes	804
11-8. CANCTL Register Field Descriptions	805
11-9. CANSTS Register Field Descriptions	807
11-10. CANERR Register Field Descriptions	809
11-11. CANBIT Register Field Descriptions	810
11-12. CANINT Register Field Descriptions	811
11-13. CANTST Register Field Descriptions	812
11-14. CANBRPE Register Field Descriptions	813
11-15. CANIFnCRQ Register Field Descriptions	814
11-16. CANIFnCMSK Register Field Descriptions	815
11-17. CANIFnMSK1 Register Field Descriptions	817
11-18. CANIFnMSK2 Register Field Descriptions	818
11-19. CANIFnARB1 Register Field Descriptions	819
11-20. CANIFnARB2 Register Field Descriptions	820
11-21. CANIFnMCTL Register Field Descriptions	821
11-22. CANIFnDnn Register Field Descriptions	823

11-23. CANTXRQn Register Field Descriptions	824
11-24. CANNWDAn Register Field Descriptions.....	825
11-25. CANMSGnINT Register Field Descriptions	826
11-26. CANMSGnVAL Register Field Descriptions.....	827
12-1. Internal Reference Voltage and ACREFTL Field Values	832
12-2. Analog Comparator Voltage Reference Characteristics ($V_{DDA} = 3.3\text{ V}$, EN = 1, and RNG = 0)	832
12-3. Analog Comparator Voltage Reference Characteristics ($V_{DDA} = 3.3\text{ V}$, EN = 1, and RNG = 1)	833
12-4. Comparator Registers	834
12-5. ACMP Access Type Codes.....	834
12-6. ACMIS Register Field Descriptions.....	835
12-7. ACRIS Register Field Descriptions	836
12-8. ACINTEN Register Field Descriptions	837
12-9. ACREFTL Register Field Descriptions	838
12-10. ACSTATn Register Field Descriptions	839
12-11. ACCTLn Register Field Descriptions.....	840
12-12. ACMPPP Register Field Descriptions.....	841
13-1. Endian Configuration	844
13-2. Endian Configuration With Bit Reversal	844
13-3. CRC Registers	846
13-4. CRC Access Type Codes.....	846
13-5. CRCCTRL Register Field Descriptions	847
13-6. CRCSEED Register Field Descriptions	849
13-7. CRCDIN Register Field Descriptions.....	850
13-8. CRCRSLTPP Register Field Descriptions.....	851
14-1. Key Repartition.....	853
14-2. DES Reset Description.....	856
14-3. DES Global Initialization	858
14-4. DES Algorithm Type Configuration	858
14-5. 3DES Algorithm Type Configuration	859
14-6. DES Interrupt Mode	860
14-7. DES DMA Mode	861
14-8. DES Registers.....	863
14-9. DES Access Type Codes	863
14-10. DES_KEYn_n Register Field Descriptions	864
14-11. DES_IV_L Register Field Descriptions.....	865
14-12. DES_IV_H Register Field Descriptions	866
14-13. DES_CTRL Register Field Descriptions	867
14-14. DES_LENGTH Register Field Descriptions	868
14-15. DES_DATA_L Register Field Descriptions	869
14-16. DES_DATA_H Register Field Descriptions	870
14-17. DES_REVISION Register Field Descriptions	871
14-18. DES_SYSCONFIG Register Field Descriptions	872
14-19. DES_SYSSTATUS Register Field Descriptions	873
14-20. DES_IRQSTATUS Register Field Descriptions.....	874
14-21. DES_IRQENABLE Register Field Descriptions.....	875
14-22. DES_DIRTYBITS Register Field Descriptions	876
14-23. DES μ DMA Registers	877
14-24. DES μ DMA Access Type Codes	877
14-25. DES_DMAIM Register Field Descriptions	878

14-26. DES_DMARIS Register Field Descriptions.....	879
14-27. DES_DMAMIS Register Field Descriptions	880
14-28. DES_DMAIC Register Field Descriptions	881
15-1. MII and RMI Signals	887
15-2. Enhanced Transmit Descriptor 0 (TDES0)	892
15-3. Enhanced Transmit Descriptor 1 (TDES1)	894
15-4. Enhanced Transmit Descriptor 2 (TDES2)	894
15-5. Enhanced Transmit Descriptor 3 (TDES3)	894
15-6. Enhanced Transmit Descriptor 6 (TDES6)	895
15-7. Enhanced Transmit Descriptor 7 (TDES7)	895
15-8. Enhanced Receive Descriptor 0 (RDES0).....	896
15-9. RDES0 Checksum Offload Bits	898
15-10. Enhanced Receive Descriptor 1 (RDES1).....	898
15-11. Enhanced Receive Descriptor 2 (RDES2).....	899
15-12. Enhanced Receive Descriptor 3 (RDES3).....	899
15-13. Enhanced Received Descriptor 4 (RDES4)	899
15-14. Enhanced Receive Descriptor 6 (RDES6).....	900
15-15. Enhanced Receive Descriptor 7 (RDES7).....	901
15-16. TX MAC Flow Control	912
15-17. RX MAC Flow Control.....	912
15-18. VLAN Match Status.....	923
15-19. CRC Replacement Based on Bit 27 and Bit 24 of TDES0	925
15-20. Forced Mode Configurations	931
15-21. Advertised Mode Configurations	931
15-22. EMACPC to PHY Register Mapping	936
15-23. EMAC Registers	939
15-24. EMAC Access Type Codes.....	941
15-25. EMACCFG Register Field Descriptions.....	942
15-26. EMACFRAMEFLTR Register Field Descriptions	946
15-27. EMACHASHTBLH Register Field Descriptions	949
15-28. EMACHASHTBLL Register Field Descriptions	950
15-29. EMACMIIADDR Register Field Descriptions	951
15-30. EMACMIIDATA Register Field Descriptions	953
15-31. EMACFLOWCTL Register Field Descriptions	954
15-32. EMACVLANTG Register Field Descriptions.....	956
15-33. EMACSTATUS Register Field Descriptions.....	958
15-34. EMACRWUFF Register Field Descriptions.....	960
15-35. EMACPMCTLSTAT Register Field Descriptions.....	961
15-36. EMACLPICLSTAT Register Field Descriptions	963
15-37. EMACLPITIMERCTRL Register Field Descriptions	965
15-38. EMACRIS Register Field Descriptions	966
15-39. EMACIM Register Field Descriptions	968
15-40. EMACADDR0H Register Field Descriptions	969
15-41. EMACADDR0L Register Field Descriptions.....	970
15-42. EMACADDR1H Register Field Descriptions	971
15-43. EMACADDR1L Register Field Descriptions.....	972
15-44. EMACADDR2H Register Field Descriptions	973
15-45. EMACADDR2L Register Field Descriptions.....	974
15-46. EMACADDR3H Register Field Descriptions	975

15-47. EMACADDR3L Register Field Descriptions.....	976
15-48. EMACWDOGTO Register Field Descriptions	977
15-49. EMACMMCCTRL Register Field Descriptions.....	978
15-50. EMACMMCRXRIS Register Field Descriptions.....	980
15-51. EMACMMCTXRIS Register Field Descriptions	982
15-52. EMACMMCRXIM Register Field Descriptions	984
15-53. EMACMMCTXIM Register Field Descriptions	985
15-54. EMACTXCNTGB Register Field Descriptions	986
15-55. EMACTXCNTSCOL Register Field Descriptions	987
15-56. EMACTXCNTMCOL Register Field Descriptions.....	988
15-57. EMACTXOCTCNTG Register Field Descriptions.....	989
15-58. EMACRXCNTGB Register Field Descriptions	990
15-59. EMACRXCNTCRCERR Register Field Descriptions	991
15-60. EMACRXCNTALGNERR Register Field Descriptions	992
15-61. EMACRXCNTGUNI Register Field Descriptions	993
15-62. EMACVLNINCREP Register Field Descriptions	994
15-63. EMACVLANHASH Register Field Descriptions.....	995
15-64. EMACTIMSTCTRL Register Field Descriptions	996
15-65. EMACSUBSECINC Register Field Descriptions.....	999
15-66. EMACTIMSEC Register Field Descriptions.....	1000
15-67. EMACTIMNANO Register Field Descriptions	1001
15-68. EMACTIMSECU Register Field Descriptions.....	1002
15-69. EMACTIMNANOU Register Field Descriptions	1003
15-70. EMACTIMADD Register Field Descriptions	1004
15-71. EMACTARGSEC Register Field Descriptions	1005
15-72. EMACTARGNANO Register Field Descriptions.....	1006
15-73. EMACHWORDSEC Register Field Descriptions.....	1007
15-74. EMACTIMSTAT Register Field Descriptions	1008
15-75. EMACPPSCTRL Register Field Descriptions	1009
15-76. PPSCTRL Bit Field Values.....	1010
15-77. EMACPPS0INTVL Register Field Descriptions	1012
15-78. EMACPPS0WIDTH Register Field Descriptions	1013
15-79. EMACDMABUSMOD Register Field Descriptions	1014
15-80. EMACTXPOLLD Register Field Descriptions.....	1017
15-81. EMACRXPOLLD Register Field Descriptions	1018
15-82. EMACRXDLADDR Register Field Descriptions	1019
15-83. EMACTXDLADDR Register Field Descriptions	1020
15-84. EMACDMARIS Register Field Descriptions	1021
15-85. EMACDMAOPMODE Register Field Descriptions.....	1025
15-86. EMACDMAIM Register Field Descriptions	1029
15-87. EMACMFOC Register Field Descriptions	1031
15-88. EMACRXINTWDT Register Field Descriptions.....	1032
15-89. EMACHOSTXDESC Register Field Descriptions	1033
15-90. EMACHOSRXDESC Register Field Descriptions.....	1034
15-91. EMACHOSTXBA Register Field Descriptions	1035
15-92. EMACHOSRXBA Register Field Descriptions.....	1036
15-93. EMACPP Register Field Descriptions	1037
15-94. EMACPC Register Field Descriptions	1038
15-95. EMACCC Register Field Descriptions	1041

15-96. EPHYRIS Register Field Descriptions	1042
15-97. EPHYIM Register Field Descriptions	1043
15-98. EPHYMISC Register Field Descriptions.....	1044
15-99. MII Management (EPHY) Registers	1045
15-100. EPHY Access Type Codes	1045
15-101. EPHYBMCR Register Field Descriptions	1047
15-102. EPHYBMSR Register Field Descriptions	1049
15-103. EPHYID1 Register Field Descriptions.....	1051
15-104. EPHYID2 Register Field Descriptions.....	1052
15-105. EPHYANA Register Field Descriptions	1053
15-106. EPHYANLPA Register Field Descriptions	1055
15-107. EPHYANER Register Field Descriptions	1056
15-108. EPHYANNPTR Register Field Descriptions.....	1057
15-109. EPHYANLNPTR Register Field Descriptions	1058
15-110. EPHYCFG1 Register Field Descriptions.....	1059
15-111. EPHYCFG2 Register Field Descriptions.....	1061
15-112. EPHYCFG3 Register Field Descriptions.....	1062
15-113. EPHYREGCTL Register Field Descriptions	1063
15-114. EPHYADDAR Register Field Descriptions	1064
15-115. EPHYSTS Register Field Descriptions.....	1065
15-116. EPHYSCR Register Field Descriptions	1067
15-117. EPHYMISR1 Register Field Descriptions.....	1069
15-118. EPHYMISR2 Register Field Descriptions.....	1071
15-119. EPHYFCSCR Register Field Descriptions.....	1073
15-120. EPHYRXERCNT Register Field Descriptions.....	1074
15-121. EPHYBISTCR Register Field Descriptions	1075
15-122. EPHYLEDCR Register Field Descriptions.....	1077
15-123. EPHYCTL Register Field Descriptions.....	1078
15-124. EPHY10BTSC Register Field Descriptions.....	1079
15-125. EPHYBICSR1 Register Field Descriptions	1080
15-126. EPHYBICSR2 Register Field Descriptions	1081
15-127. EPHYCDCR Register Field Descriptions	1082
15-128. EPHYRCR Register Field Descriptions	1083
15-129. EPHYLEDCFG Register Field Descriptions	1084
16-1. EPI Interface Options	1091
16-2. EPI SDRAM x16 Signal Connections.....	1092
16-3. CSCFGEXT and CSCFG Encodings	1096
16-4. Dual- and Quad- Chip Select Address Mappings.....	1097
16-5. Chip Select Configuration Register Assignment	1098
16-6. Capabilities of Host Bus 8 and Host Bus 16 Modes	1098
16-7. EPI Host-Bus 8 Signal Connections	1100
16-8. EPI Host-Bus 16 Signal Connections.....	1102
16-9. PSRAM Fixed Latency Wait State Configuration	1107
16-10. Data Phase Wait State Programming	1111
16-11. EPI General-Purpose Signal Connections	1117
16-12. EPI Registers	1122
16-13. EPI Access Type Codes	1123
16-14. EPICFG Register Field Descriptions.....	1124
16-15. EPIBAUD Register Field Descriptions.....	1125

16-16. EPIBAUD2 Register Field Descriptions	1126
16-17. EPISDRAMCFG Register Field Descriptions	1127
16-18. EPIHB8CFG Register Field Descriptions	1130
16-19. EPIHB16CFG Register Field Descriptions	1133
16-20. EPIGPCFG Register Field Descriptions	1136
16-21. EPIHB8CFG2 Register Field Descriptions	1139
16-22. EPIHB16CFG2 Register Field Descriptions	1143
16-23. EPIADDRMAP Register Field Descriptions	1148
16-24. EPIRSIZE _n Register Field Descriptions	1149
16-25. EPIRADDR _n Register Field Descriptions	1150
16-26. EPIRPSTD _n Register Field Descriptions	1151
16-27. EPISTAT Register Field Descriptions	1152
16-28. EPIRFIFOCNT Register Field Descriptions	1154
16-29. EPIREADFIFO _n Register Field Descriptions	1155
16-30. EPIFIFOLVL Register Field Descriptions	1156
16-31. EPIWFIFOCNT Register Field Descriptions	1158
16-32. EPIDMATXCNT Register Field Descriptions	1159
16-33. EPIIM Register Field Descriptions	1160
16-34. EPIRIS Register Field Descriptions	1161
16-35. EPIMIS Register Field Descriptions	1163
16-36. EPIEISC Register Field Descriptions	1165
16-37. EPIHB8CFG3 Register Field Descriptions	1166
16-38. EPIHB16CFG3 Register Field Descriptions	1168
16-39. EPIHB8CFG4 Register Field Descriptions	1170
16-40. EPIHB16CFG4 Register Field Descriptions	1172
16-41. EPIHB8TIME Register Field Descriptions	1174
16-42. EPIHB16TIME Register Field Descriptions	1176
16-43. EPIHB8TIME2 Register Field Descriptions	1178
16-44. EPIHB16TIME2 Register Field Descriptions	1180
16-45. EPIHB8TIME3 Register Field Descriptions	1182
16-46. EPIHB16TIME3 Register Field Descriptions	1184
16-47. EPIHB8TIME4 Register Field Descriptions	1186
16-48. EPIHB16TIME4 Register Field Descriptions	1188
16-49. EPIHBPSRAM Register Field Descriptions	1190
17-1. GPIO Drive Strength Options	1198
17-2. GPIO Pad Configuration Examples	1199
17-3. GPIO Interrupt Configuration Example	1200
17-4. GPIO Pins With Special Considerations	1201
17-5. GPIO Registers	1202
17-6. GPIO Access Type Codes	1203
17-7. GPIODATA Register Field Descriptions	1204
17-8. GPIODIR Register Field Descriptions	1205
17-9. GPIOIS Register Field Descriptions	1206
17-10. GPIOIBE Register Field Descriptions	1207
17-11. GPIOIEV Register Field Descriptions	1208
17-12. GPIOIM Register Field Descriptions	1209
17-13. GPIORIS Register Field Descriptions	1210
17-14. GPIOMIS Register Field Descriptions	1211
17-15. GPIOICR Register Field Descriptions	1212

17-16. GPIO Pins With Special Considerations	1213
17-17. GPIOAFSEL Register Field Descriptions	1214
17-18. GPIODR2R Register Field Descriptions.....	1215
17-19. GPIODR4R Register Field Descriptions.....	1216
17-20. GPIODR8R Register Field Descriptions.....	1217
17-21. GPIOODR Register Field Descriptions	1218
17-22. GPIO Pins With Special Considerations	1219
17-23. GPIOPUR Register Field Descriptions	1220
17-24. GPIO Pins With Special Considerations	1221
17-25. GPIOPDR Register Field Descriptions	1222
17-26. GPIOSLR Register Field Descriptions.....	1223
17-27. GPIO Pins With Special Considerations	1224
17-28. GPIODEN Register Field Descriptions	1225
17-29. GPIOLOCK Register Field Descriptions.....	1226
17-30. GPIOCR Register Field Descriptions	1227
17-31. GPIOAMSEL Register Field Descriptions.....	1228
17-32. GPIO Pins With Special Considerations	1229
17-33. GPIOPCTL Register Field Descriptions	1230
17-34. GPIOADCCTL Register Field Descriptions	1231
17-35. GPIODMACTL Register Field Descriptions.....	1232
17-36. GPIOSI Register Field Descriptions	1233
17-37. GPIODR12R Register Field Descriptions	1234
17-38. GPIOWAKEPEN Register Field Descriptions	1235
17-39. GPIOWAKELVL Register Field Descriptions	1236
17-40. GPIOWAKESTAT Register Field Descriptions	1237
17-41. GPIOPP Register Field Descriptions	1238
17-42. GPIO Drive Strength Options	1239
17-43. GPIOPC Register Field Descriptions	1240
17-44. GPIOPeriphID4 Register Field Descriptions	1241
17-45. GPIOPeriphID5 Register Field Descriptions	1242
17-46. GPIOPeriphID6 Register Field Descriptions	1243
17-47. GPIOPeriphID7 Register Field Descriptions	1244
17-48. GPIOPeriphID0 Register Field Descriptions	1245
17-49. GPIOPeriphID1 Register Field Descriptions	1246
17-50. GPIOPeriphID2 Register Field Descriptions	1247
17-51. GPIOPeriphID3 Register Field Descriptions	1248
17-52. GPIOCellID0 Register Field Descriptions	1249
17-53. GPIOCellID1 Register Field Descriptions	1250
17-54. GPIOCellID2 Register Field Descriptions	1251
17-55. GPIOCellID3 Register Field Descriptions	1252
18-1. Available CCP Pins	1255
18-2. General-Purpose Timer Capabilities	1256
18-3. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes.....	1257
18-4. 16-Bit Timer With Prescaler Configurations	1258
18-5. Counter Values When the Timer is Enabled in RTC Mode.....	1259
18-6. Counter Values When the Timer is Enabled in Input Edge-Count Mode	1260
18-7. Counter Values When the Timer is Enabled in Input Event-Count Mode	1261
18-8. Counter Values When the Timer is Enabled in PWM Mode.....	1262
18-9. Time-out Actions for GPTM Modes	1266

18-10. GPTM Registers	1271
18-11. GPTM Access Type Codes	1272
18-12. GPTMCFG Register Field Descriptions	1273
18-13. GPTMTAMR Register Field Descriptions	1274
18-14. GPTMTBMR Register Field Descriptions	1277
18-15. GPTMCTL Register Field Descriptions	1280
18-16. GPTMSYNC Register Field Descriptions	1282
18-17. GPTMIMR Register Field Descriptions	1284
18-18. GPTMRIS Register Field Descriptions	1286
18-19. GPTMMIS Register Field Descriptions	1288
18-20. GPTMICR Register Field Descriptions	1290
18-21. GPTMTAILR Register Field Descriptions	1292
18-22. GPTMTBILR Register Field Descriptions	1293
18-23. GPTMTAMATCHR Register Field Descriptions	1294
18-24. GPTMTBMATCHR Register Field Descriptions	1295
18-25. GPTMTAPR Register Field Descriptions	1296
18-26. GPTMTBPR Register Field Descriptions	1297
18-27. GPTMTAPMR Register Field Descriptions	1298
18-28. GPTMTBPMR Register Field Descriptions	1299
18-29. GPTMTAR Register Field Descriptions	1300
18-30. GPTMTBR Register Field Descriptions	1301
18-31. GPTMTAV Register Field Descriptions	1302
18-32. GPTMTBV Register Field Descriptions	1303
18-33. GPTMRTCPD Register Field Descriptions	1304
18-34. GPTMTAPS Register Field Descriptions	1305
18-35. GPTMTBPS Register Field Descriptions	1306
18-36. GPTMDMAEV Register Field Descriptions	1307
18-37. GPTMADCEV Register Field Descriptions	1309
18-38. GPTMPP Register Field Descriptions	1311
18-39. GPTMCC Register Field Descriptions	1312
19-1. Examples of I ² C Master Timer Period Versus Speed Mode	1321
19-2. Examples of I ² C Master Timer Period in High-Speed Mode	1322
19-3. I2C Registers	1335
19-4. I2C Access Type Codes	1335
19-5. I2CMSA Register Field Descriptions	1336
19-6. I2CMCS Register Field Descriptions — Read-Only Status Register	1337
19-7. I2CMCS Register Field Descriptions — Write-Only Control Register	1339
19-8. Write Field Decoding for I2CMCS[6:0]	1340
19-9. I2CMDR Register Field Descriptions	1343
19-10. I2CMTPR Register Field Descriptions	1344
19-11. I2CMIMR Register Field Descriptions	1345
19-12. I2CMRIS Register Field Descriptions	1347
19-13. I2CMMIS Register Field Descriptions	1349
19-14. I2CMICR Register Field Descriptions	1351
19-15. I2CMCR Register Field Descriptions	1353
19-16. I2CMCLKOCNT Register Field Descriptions	1354
19-17. I2CMBMON Register Field Descriptions	1355
19-18. I2CMBLEN Register Field Descriptions	1356
19-19. I2CMBCNT Register Field Descriptions	1357

19-20. I2CSOAR Register Field Descriptions	1358
19-21. I2CSCSR Register Field Descriptions — Read-Only Status Register	1359
19-22. I2CSCSR Register Field Descriptions — Write-Only Control Register	1360
19-23. I2CSDR Register Field Descriptions	1361
19-24. I2CSIMR Register Field Descriptions	1362
19-25. I2CSRIS Register Field Descriptions	1364
19-26. I2CSMIS Register Field Descriptions	1366
19-27. I2CSICR Register Field Descriptions	1368
19-28. I2CSOAR2 Register Field Descriptions	1369
19-29. I2CSACKCTL Register Field Descriptions	1370
19-30. I2CFIFODATA Register Field Descriptions	1371
19-31. I2CFIFOCTL Register Field Descriptions	1372
19-32. I2CFIFOSTATUS Register Field Descriptions	1374
19-33. I2CPP Register Field Descriptions	1375
19-34. I2CPC Register Field Descriptions	1376
20-1. LIDD I/O Name Map	1382
20-2. Operation Modes Supported by Raster Controller	1383
20-3. Type Encoding in Palette Entry 0 Buffer	1387
20-4. Intensities and Modulation Rates	1391
20-5. Number of Colors and Shades of Gray Available on Screen	1392
20-6. LCD Registers	1397
20-7. LCD Access Type Codes	1397
20-8. LCDPID Register Field Descriptions	1398
20-9. LCDCTL Register Field Descriptions	1399
20-10. SYSCLK to Pixel Clock (LCDCP) Frequency Conversion Table	1400
20-11. LCDLIDDCTL Register Field Descriptions	1401
20-12. LCD Alternate Signal Functions in LIDD Mode	1402
20-13. LIDDCS0CFG Register Field Descriptions	1403
20-14. LIDDCS0ADDR Register Field Descriptions	1404
20-15. LIDDCS0DATA Register Field Descriptions	1405
20-16. LIDDCS1CFG Register Field Descriptions	1406
20-17. LIDDCS1ADDR Register Field Descriptions	1407
20-18. LIDDCS1DATA Register Field Descriptions	1408
20-19. LCDRASTRCTL Register Field Descriptions	1409
20-20. LCDRASTRTIM0 Register Field Descriptions	1412
20-21. LCDRASTRTIM1 Register Field Descriptions	1413
20-22. LCDRASTRTIM2 Register Field Descriptions	1414
20-23. LCDRASTRSUBP1 Register Field Descriptions	1416
20-24. LCDRASTRSUBP2 Register Field Descriptions	1417
20-25. LCDDMACTL Register Field Descriptions	1418
20-26. LCDDMABAFB0 Register Field Descriptions	1420
20-27. LCDDMACAFB0 Register Field Descriptions	1421
20-28. LCDDMABAFB1 Register Field Descriptions	1422
20-29. LCDDMACAFB1 Register Field Descriptions	1423
20-30. LCDSYSCFG Register Field Descriptions	1424
20-31. LCDRISSET Register Field Descriptions	1425
20-32. LCDMISCLR Register Field Descriptions	1427
20-33. LCDIM Register Field Descriptions	1429
20-34. LCDIENC Register Field Descriptions	1431

20-35. LCDCLKEN Register Field Descriptions	1433
20-36. LCDCLKRESET Register Field Descriptions	1434
21-1. PWM Registers	1445
21-2. PWM Access Type Codes	1447
21-3. PWMCTL Register Field Descriptions	1448
21-4. PWMSYNC Register Field Descriptions.....	1449
21-5. PWMENABLE Register Field Descriptions	1450
21-6. PWMINVERT Register Field Descriptions	1452
21-7. PWMFAULT Register Field Descriptions	1454
21-8. PWMINTEN Register Field Descriptions	1456
21-9. PWMRIS Register Field Descriptions.....	1458
21-10. PWMISC Register Field Descriptions.....	1460
21-11. PWMSTATUS Register Field Descriptions	1462
21-12. PWMFAULTVAL Register Field Descriptions	1463
21-13. PWMENUPD Register Field Descriptions.....	1465
21-14. PWMnCTL Register Field Descriptions	1467
21-15. PWMnINTEN Register Field Descriptions	1470
21-16. PWMnRIS Register Field Descriptions	1472
21-17. PWMnISC Register Field Descriptions	1474
21-18. PWMnLOAD Register Field Descriptions	1476
21-19. PWMnCOUNT Register Field Descriptions	1477
21-20. PWMnCMPA Register Field Descriptions.....	1478
21-21. PWMnCMPB Register Field Descriptions.....	1479
21-22. PWMnGENA Register Field Descriptions	1480
21-23. PWMnGENB Register Field Descriptions	1482
21-24. PWMnDBCTL Register Field Descriptions.....	1484
21-25. PWMnDBRISE Register Field Descriptions	1485
21-26. PWMnDBFALL Register Field Descriptions	1486
21-27. PWMnFLTSRC0 Register Field Descriptions.....	1487
21-28. PWMnFLTSRC1 Register Field Descriptions.....	1489
21-29. PWMnMINFLTPER Register Field Descriptions	1491
21-30. PWMnFLTSEN Register Field Descriptions	1492
21-31. PWMnFLTSTAT0 Register Field Descriptions	1493
21-32. PWMnFLTSTAT1 Register Field Descriptions	1495
21-33. PWMPP Register Field Descriptions	1498
21-34. PWMCC Register Field Descriptions	1499
22-1. Active Time Periods in Overdrive Mode.....	1505
22-2. Bit Field Definitions for 1-Wire Timing and Override (ONEWIRETIM) Register	1505
22-3. One-Wire Master Registers	1510
22-4. OWIRE Access Type Codes.....	1510
22-5. ONEWIRECS Register Field Descriptions	1511
22-6. ONEWIRETIM Register Field Descriptions	1513
22-7. ONEWIREDATW Register Field Descriptions.....	1514
22-8. ONEWIREDATR Register Field Descriptions	1515
22-9. ONEWIREIM Register Field Descriptions.....	1516
22-10. ONEWIRERIS Register Field Descriptions	1517
22-11. ONEWIREMIS Register Field Descriptions	1518
22-12. ONEWIREICR Register Field Descriptions	1519
22-13. ONEWIREDMA Register Field Descriptions.....	1520

22-14. ONEWIREPP Register Field Descriptions	1521
23-1. QSSI Transaction Encodings	1526
23-2. SSInFss Functionality.....	1527
23-3. Legacy Mode TI, Freescale SPI Frame Format Features	1529
23-4. QSSI Registers	1537
23-5. QSSI Access Type Codes	1537
23-6. SSICR0 Register Field Descriptions	1539
23-7. SSICR1 Register Field Descriptions	1541
23-8. SSIDR Register Field Descriptions	1543
23-9. SSISR Register Field Descriptions.....	1544
23-10. SSICPSR Register Field Descriptions	1545
23-11. SSIIIM Register Field Descriptions	1546
23-12. SSIRIS Register Field Descriptions.....	1547
23-13. SSIMIS Register Field Descriptions	1549
23-14. SSICR Register Field Descriptions.....	1551
23-15. SSIDMACTL Register Field Descriptions	1552
23-16. SSIPP Register Field Descriptions.....	1553
23-17. SSICC Register Field Descriptions	1554
23-18. SSIPeriphID4 Register Field Descriptions	1555
23-19. SSIPeriphID5 Register Field Descriptions	1556
23-20. SSIPeriphID6 Register Field Descriptions	1557
23-21. SSIPeriphID7 Register Field Descriptions	1558
23-22. SSIPeriphID0 Register Field Descriptions	1559
23-23. SSIPeriphID1 Register Field Descriptions	1560
23-24. SSIPeriphID2 Register Field Descriptions	1561
23-25. SSIPeriphID3 Register Field Descriptions	1562
23-26. SSIPCellID0 Register Field Descriptions.....	1563
23-27. SSIPCellID1 Register Field Descriptions.....	1564
23-28. SSIPCellID2 Register Field Descriptions.....	1565
23-29. SSIPCellID3 Register Field Descriptions.....	1566
24-1. QEI Registers	1573
24-2. QEI Access Type Codes	1573
24-3. QEICTL Register Field Descriptions	1574
24-4. QEISTAT Register Field Descriptions	1576
24-5. QEIPPOS Register Field Descriptions	1577
24-6. QEIMAXPOS Register Field Descriptions	1578
24-7. QEILOAD Register Field Descriptions.....	1579
24-8. QEITIME Register Field Descriptions.....	1580
24-9. QEICOUNT Register Field Descriptions	1581
24-10. QEISPEED Register Field Descriptions	1582
24-11. QEINTEN Register Field Descriptions	1583
24-12. QEIRIS Register Field Descriptions	1584
24-13. QEISC Register Field Descriptions	1585
25-1. Interrupts and Events	1589
25-2. SHA/MD5 Module Algorithm Selection	1589
25-3. Outer Digest Registers	1590
25-4. Inner Digest Registers	1590
25-5. SHA Digest Processed in Three Passes.....	1592
25-6. SHA Digest Processed in One Pass.....	1593

25-7. SHA/MD5 Performance.....	1594
25-8. Continuing a Prior HMAC	1596
25-9. SHA-1 Apply on the Key	1596
25-10. Interrupt Mode	1597
25-11. DMA Mode	1598
25-12. SHA/MD5 Inner/Outer Digest/HMAC Key Register Mapping.....	1600
25-13. SHA/MD5 Registers.....	1601
25-14. SHA/MD5 Access Type Codes	1602
25-15. SHA_ODIGEST_n and SHA_IDIGEST_n Registers Field Descriptions	1603
25-16. SHA_DIGEST_COUNT Register Field Descriptions	1604
25-17. SHA_MODE Register Field Descriptions	1605
25-18. SHA_LENGTH Register Field Descriptions.....	1607
25-19. SHA_DATA_n_IN Register Field Descriptions	1608
25-20. SHA_REVISION Register Field Descriptions.....	1609
25-21. SHA_SYSCONFIG Register Field Descriptions.....	1610
25-22. SHA_SYSSTATUS Register Field Descriptions.....	1612
25-23. SHA_IRQSTATUS Register Field Descriptions	1613
25-24. SHA_IRQENABLE Register Field Descriptions	1614
25-25. SHA_MD5_UDMA Registers	1615
25-26. SHA/MD5 μ DMA Access Type Codes	1615
25-27. SHA_DMAIM Register Field Descriptions.....	1616
25-28. SHA_DMARIS Register Field Descriptions	1617
25-29. SHA_DMAMIS Register Field Descriptions.....	1618
25-30. SHA_DMAIC Register Field Descriptions	1619
26-1. Flow Control Mode	1626
26-2. UART Registers	1631
26-3. UART Access Type Codes.....	1632
26-4. UARTDR Register Field Descriptions	1633
26-5. UARTRSR/UARTECR Register Field Descriptions.....	1634
26-6. UARTFR Register Field Descriptions.....	1635
26-7. UARTILPR Register Field Descriptions	1637
26-8. UARTIBRD Register Field Descriptions.....	1638
26-9. UARTFBRD Register Field Descriptions	1639
26-10. UARTLCRH Register Field Descriptions	1640
26-11. UARTCTL Register Field Descriptions	1642
26-12. UARTIFLS Register Field Descriptions.....	1645
26-13. UARTIM Register Field Descriptions	1646
26-14. UARTRIS Register Field Descriptions.....	1649
26-15. UARTMIS Register Field Descriptions	1651
26-16. UARTICR Register Field Descriptions.....	1653
26-17. UARTDMACTL Register Field Descriptions	1655
26-18. UART9BITADDR Register Field Descriptions	1656
26-19. UART9BITAMASK Register Field Descriptions	1657
26-20. UARTPP Register Field Descriptions.....	1658
26-21. UARTCC Register Field Descriptions	1659
26-22. UARTPeriphID4 Register Field Descriptions	1660
26-23. UARTPeriphID5 Register Field Descriptions	1661
26-24. UARTPeriphID6 Register Field Descriptions	1662
26-25. UARTPeriphID7 Register Field Descriptions	1663

26-26. UARTPeriphID0 Register Field Descriptions	1664
26-27. UARTPeriphID1 Register Field Descriptions	1665
26-28. UARTPeriphID2 Register Field Descriptions	1666
26-29. UARTPeriphID3 Register Field Descriptions	1667
26-30. UARTPCellID0 Register Field Descriptions.....	1668
26-31. UARTPCellID1 Register Field Descriptions.....	1669
26-32. UARTPCellID2 Register Field Descriptions.....	1670
26-33. UARTPCellID3 Register Field Descriptions.....	1671
27-1. USB Device LPM Transaction Response	1684
27-2. Endpoint Interrupt Associated With USBRXCSRLn.RXRDY = 1	1687
27-3. Endpoint Interrupt Associated With USBTXCSRLn.TXRDY = 1	1687
27-4. USB0CLK Direction During ULPI Mode	1692
27-5. USB Registers	1694
27-6. USB Access Type Codes	1699
27-7. USBFADDR Register Field Descriptions	1700
27-8. USBPOWER Register Field Descriptions (OTG A / Host)	1701
27-9. USBPOWER Register Field Descriptions (OTG B / Device)	1702
27-10. USBTXIS Register Field Descriptions	1703
27-11. USBRXIS Register Field Descriptions	1704
27-12. USBTXIE Register Field Descriptions	1705
27-13. USBRXIE Register Field Descriptions	1706
27-14. USBIS Register Field Descriptions (OTG A / Host).....	1707
27-15. USBIS Register Field Descriptions (OTG B / Device).....	1708
27-16. USBIE Register Field Descriptions (OTG A / Host).....	1709
27-17. USBIE Register Field Descriptions (OTG B / Device)	1710
27-18. USBFRAME Register Field Descriptions.....	1711
27-19. USBEPIDX Register Field Descriptions	1712
27-20. USBTEST Register Field Descriptions (OTG A / Host)	1713
27-21. USBTEST Register Field Descriptions (OTG B / Device)	1714
27-22. USBFIFOn Register Field Descriptions	1715
27-23. USBDEVCTL Register Field Descriptions	1716
27-24. USBCCONF Register Field Descriptions	1717
27-25. USBnXFIFOSZ Register Field Descriptions	1718
27-26. USBnXFIFOADD Register Field Descriptions	1719
27-27. ULPIVBUSCTL Register Field Descriptions	1720
27-28. ULPIREGDATA Register Field Descriptions.....	1721
27-29. ULPIREGADDR Register Field Descriptions	1722
27-30. ULPIREGCTL Register Field Descriptions.....	1723
27-31. USBEPINFO Register Field Descriptions	1724
27-32. USBRAMINFO Register Field Descriptions.....	1725
27-33. USBCONTIM Register Field Descriptions	1726
27-34. USBVPLEN Register Field Descriptions	1727
27-35. USBHSEOF Register Field Descriptions.....	1728
27-36. USBFSEOF Register Field Descriptions	1729
27-37. USBLSEOF Register Field Descriptions	1730
27-38. USBTXFUNCADDRn Register Field Descriptions	1731
27-39. USBTXHUBADDRn Register Field Descriptions.....	1732
27-40. USBTXHUBPORTn Register Field Descriptions	1733
27-41. USBRXFUNCADDRn Register Field Descriptions.....	1734

27-42. USBRXHUBADDRn Register Field Descriptions	1735
27-43. USBRXHUBPORTn Register Field Descriptions.....	1736
27-44. USBCSRL0 Register Field Descriptions (OTG A / Host)	1737
27-45. USBCSRL0 Register Field Descriptions (OTG B / Device).....	1738
27-46. USBCSRH0 Register Field Descriptions (OTG A / Host)	1740
27-47. USBCSRH0 Register Field Descriptions (OTG B / Device)	1741
27-48. USBCOUNT0 Register Field Descriptions	1742
27-49. USBTYPE0 Register Field Descriptions.....	1743
27-50. USBNAKLMT Register Field Descriptions	1744
27-51. USBTXMAXPn Register Field Descriptions	1745
27-52. USBTXCSRLn Register Field Descriptions (OTG A / Host)	1746
27-53. USBTXCSRLn Register Field Descriptions (OTG B / Device)	1747
27-54. USBTXCSRHn Register Field Descriptions (OTG A / Host)	1749
27-55. USBTXCSRHn Register Field Descriptions (OTG B / Device).....	1750
27-56. USBRXMAXPn Register Field Descriptions	1751
27-57. USBRXCSRLn Register Field Descriptions (OTG A / Host).....	1752
27-58. USBRXCSRLn Register Field Descriptions (OTG B / Device)	1753
27-59. USBRXCSRHn Register Field Descriptions (OTG A / Host)	1755
27-60. USBRXCSRHn Register Field Descriptions (OTG B / Device)	1756
27-61. USBRXCOUNTn Register Field Descriptions	1758
27-62. USBTXTYPEn Register Field Descriptions	1759
27-63. USBTXINTERVALn Register Values	1760
27-64. USBTXINTERVALn Register Field Descriptions	1760
27-65. USBRXTYPEn Register Field Descriptions.....	1761
27-66. USBRXINTERVALn Register Values.....	1762
27-67. USBRXINTERVALn Register Field Descriptions.....	1762
27-68. USBDMAINTR Register Field Descriptions.....	1763
27-69. USBDMACTLn Register Field Descriptions	1764
27-70. USBDMAADDRn Register Field Descriptions	1766
27-71. USBDMACOUNTn Register Field Descriptions	1767
27-72. USBRQPKTCOUNTn Register Field Descriptions.....	1768
27-73. USBRXDPKTBUFDIS Register Field Descriptions	1769
27-74. USBTXDPKTBUFDIS Register Field Descriptions.....	1770
27-75. USBCTO Register Field Descriptions	1771
27-76. USBHHSRTN Register Field Descriptions	1772
27-77. USBHSBT Register Field Descriptions	1773
27-78. USBLPMATTR Register Field Descriptions	1774
27-79. USBLPMCNTL Register Field Descriptions (OTG A / Host).....	1775
27-80. USBLPMCNTL Register Field Descriptions (OTG B / Device).....	1775
27-81. USBLPMIM Register Field Descriptions.....	1777
27-82. USBLPMRIS Register Field Descriptions (OTG A / Host)	1778
27-83. USBLPMRIS Register Field Descriptions (OTG B / Device)	1779
27-84. USBLPMFADDR Register Field Descriptions	1780
27-85. USBEPC Register Field Descriptions.....	1781
27-86. USBEPCRIS Register Field Descriptions	1783
27-87. USBEPCIM Register Field Descriptions.....	1784
27-88. USBEPCISC Register Field Descriptions	1785
27-89. USBDRRIS Register Field Descriptions.....	1786
27-90. USBDRIM Register Field Descriptions	1787

27-91. USBDRISC Register Field Descriptions	1788
27-92. USBGPCS Register Field Descriptions	1789
27-93. USBVDC Register Field Descriptions	1790
27-94. USBVDCRIS Register Field Descriptions	1791
27-95. USBVDCIM Register Field Descriptions	1792
27-96. USBVDCISC Register Field Descriptions	1793
27-97. USBPP Register Field Descriptions	1794
27-98. USBPC Register Field Descriptions	1795
27-99. USBCC Register Field Descriptions	1796
28-1. WDT Registers	1801
28-2. WDT Access Type Codes	1801
28-3. WDTLOAD Register Field Descriptions	1802
28-4. WDTVALUE Register Field Descriptions	1802
28-5. WDTCTL Register Field Descriptions	1803
28-6. WDTICR Register Field Descriptions	1805
28-7. WDTRIS Register Field Descriptions	1806
28-8. WDTMIS Register Field Descriptions	1807
28-9. WDTTEST Register Field Descriptions	1808
28-10. WDTLOCK Register Field Descriptions	1809
28-11. WDTPeriphID4 Register Field Descriptions	1810
28-12. WDTPeriphID5 Register Field Descriptions	1810
28-13. WDTPeriphID6 Register Field Descriptions	1811
28-14. WDTPeriphID7 Register Field Descriptions	1811
28-15. WDTPeriphID0 Register Field Descriptions	1812
28-16. WDTPeriphID1 Register Field Descriptions	1812
28-17. WDTPeriphID2 Register Field Descriptions	1813
28-18. WDTPeriphID3 Register Field Descriptions	1813
28-19. WDTPCellID0 Register Field Descriptions	1814
28-20. WDTPCellID1 Register Field Descriptions	1814
28-21. WDTPCellID2 Register Field Descriptions	1815
28-22. WDTPCellID3 Register Field Descriptions	1815

About This Document

This technical reference manual (TRM) describes the MSP432E4 family of microcontrollers, including the functional blocks of the system-on-chip (SoC) device designed around the Arm® Cortex®-M4F core.

Audience

This TRM is intended for system software developers, hardware designers, and application developers.

About This Manual

This document is organized into sections that correspond to each major feature.

Related Documents

The following related documents are available at www.ti.com/msp432. Visit the web site for additional documentation, including application notes and white papers.

- [MSP432E4 SimpleLink™ Microcontrollers Silicon Errata](#)
- [Bootloader for MSP432E4 SimpleLink™ Microcontrollers User's Guide](#)

The following related documents may also be useful:

- [Arm® Cortex-M4 Errata](#)
- [Arm® Cortex-M4 Technical Reference Manual](#)
- [Arm® Debug Interface V5 Architecture Specification](#)
- [Arm® Embedded Trace Macrocell Architecture Specification](#)
- [Cortex-M4 instruction set chapter in the Arm® Cortex-M4 Devices Generic User Guide](#)
- [IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture](#)

Documentation Conventions

The following table lists the conventions used in this document.

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, PBORCTL is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, SRCRn represents any (or all) of the three Software Reset Control registers: SRCR0, SRCR1, and SRCR2.
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0xnnn	A hexadecimal increment to a register address, relative to that module base address as specified in Table 1-15 .
RESERVED	Register bits marked RESERVED are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 to 31 in that register.
Register Bit Field Reset Value	
0	Bit cleared to 0 on chip reset
1	Bit set to 1 on chip reset

Notation	Meaning
X	Nondeterministic
Pin and Signal Notation	
[]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.
assert a signal	Change the value of the signal from the logically false state to the logically true state. For active-high signals, the asserted signal value is 1 (high); for active-low signals, the asserted signal value is 0 (low). The active polarity (high or low) is defined by the signal name (see SIGNAL and SIGNAL below).
deassert a signal	Change the value of the signal from the logically true state to the logically false state.
SIGNAL	Signal names are in uppercase. An overbar on a signal name indicates that it is active-low. To assert SIGNAL is to drive it low; to deassert SIGNAL is to drive it high.
SIGNAL	Signal names are in uppercase. An active-high signal has no overbar. To assert SIGNAL is to drive it high; to deassert SIGNAL is to drive it low.
Numbers	
X	An uppercase X indicates any of several values is allowed, where X can be any legal pattern. For example, a binary value of 0X00 can be either 0100 or 0000, a hex value of 0xX is 0x0 or 0x1, and so on.
0x	Hexadecimal numbers have a prefix of 0x. For example, 0x00FF is the hexadecimal number FF. All other numbers within register tables are assumed to be binary. Within conceptual information, binary numbers are indicated with a b suffix, for example, 1011b, and decimal numbers are written without a prefix or suffix.

Trademarks

SimpleLink is a trademark of Texas Instruments.
Arm7, CoreSight are trademarks of Arm Limited.
Arm, Cortex, Thumb, PrimeCell are registered trademarks of Arm Limited.
All other trademarks are the property of their respective owners.

Cortex[®]-M4F Processor

The Arm[®] Cortex[®]-M4F processor provides a high-performance low-cost platform that meets the system requirements of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

Topic	Page
1.1 Introduction	80
1.2 Block Diagram	81
1.3 Overview	82
1.4 Programming Model	83
1.5 Memory Model	97
1.6 Exception Model	105
1.7 Fault Handling	113
1.8 Power Management	114
1.9 Instruction Set Summary	115

1.1 Introduction

Features of the Arm Cortex-M4F processor include:

- 32-bit Cortex-M4F architecture optimized for small-footprint embedded applications
- 120-MHz operation, 150-DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb®-2 mixed 16- and 32-bit instruction set delivers the high performance expected of a 32-bit Arm core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications.
 - Single-cycle multiply instruction and hardware divide
 - Atomic bit manipulation (bit-banding), delivering maximum memory use and streamlined peripheral control
 - Unaligned data access, enabling data to be efficiently packed into memory
- IEEE 754-compliant single-precision Floating-Point Unit (FPU)
- 16-bit SIMD vector processing unit
- Fast code execution permits slower processor clock or increases sleep mode time.
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the Arm7™ processor family for better performance and power efficiency
- Optimized for single-cycle flash memory use up to specific frequencies; see [Chapter 7](#) for more information.
- Ultra-low power consumption with integrated sleep modes

The MSP432E4 microcontrollers build on this core to bring high-performance 32-bit computing to cost-conscious applications requiring significant control processing and connectivity capabilities such as:

- Low-power hand-held smart devices
- Gaming equipment
- Network appliances and switches
- Home and commercial site monitoring and control
- Electronic point-of-sale (POS) machines
- Motion control
- Medical instrumentation
- Remote connectivity and monitoring
- Test and measurement equipment
- Factory automation
- Fire and security
- Smart energy and smart grid solutions
- Intelligent lighting control
- Transportation

This chapter provides information on the MSP432E4 implementation of the Cortex-M4F processor, including the programming model, the memory model, the exception model, fault handling, and power management.

For technical details on the instruction set, see the Cortex-M4 instruction set chapter in the [Arm Cortex-M4 Devices Generic User Guide](#).

1.2 Block Diagram

The Cortex-M4F processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware; this hardware includes IEEE 754-compliant single-precision floating-point computation, a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic, and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4F processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M4F processor implements a version of the Thumb instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M4F instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M4F processor closely integrates a nested vectored interrupt controller (NVIC), to deliver industry-leading interrupt performance. The NVIC includes a nonmaskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing interrupt latency. The hardware stacking of registers and the ability to suspend load-multiple and store-multiple operations further reduce interrupt latency. Interrupt handlers do not require any assembler stubs, which removes code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, including deep-sleep mode, which enables rapid power down of the entire device.

Figure 1-1 shows the CPU block diagram.

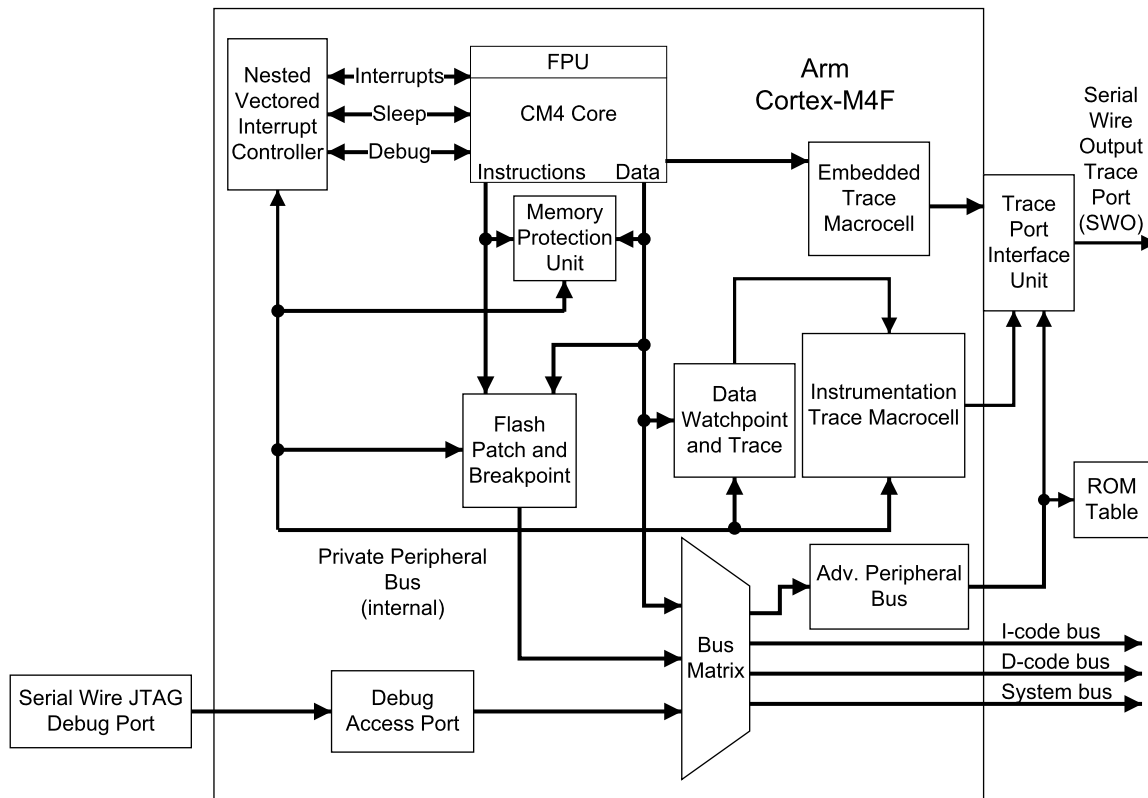


Figure 1-1. CPU Block Diagram

1.3 Overview

1.3.1 System-Level Interface

The Cortex-M4F processor provides multiple interfaces using AMBA technology to provide high-speed low-latency memory accesses. The core supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks, and thread-safe Boolean data handling.

The Cortex-M4F processor has an MPU that provides fine-grain memory control, enabling applications to implement security privilege levels and separate code, data and stack on a task-by-task basis.

1.3.2 Integrated Configurable Debug

The Cortex-M4F processor implements a complete hardware debug solution, providing high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices. The MSP432E4 implementation replaces the Arm SW-DP and JTAG-DP with the Arm CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module. See the Arm Debug Interface V5 Architecture Specification for details on SWJ-DP.

For system trace, the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system trace events, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through one pin.

The Embedded Trace Macrocell (ETM) delivers unrivaled instruction trace capture in an area smaller than traditional trace units, enabling full instruction trace. For more details on the Arm ETM, see the Arm Embedded Trace Macrocell Architecture Specification.

The Flash Patch and Breakpoint (FPB) unit provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions for up to eight words of program code in the code memory region. This FPB enables applications stored in a read-only area of Flash memory to be patched in another area of on-chip SRAM or flash memory. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration.

For more information on the Cortex-M4F debug capabilities, see the Arm Debug Interface V5 Architecture Specification.

1.3.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M4F trace data from the ITM, and an off-chip Trace Port Analyzer (see [Figure 1-2](#)).

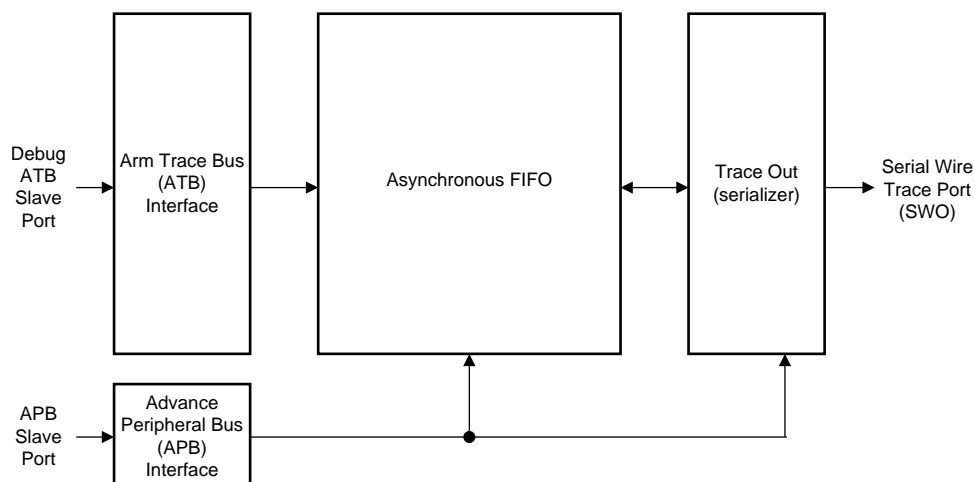


Figure 1-2. TPIU Block Diagram

1.3.4 Cortex-M4F System Component Details

The Cortex-M4F includes the following system components:

- SysTick
A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see [Section 2.2.1](#))
- NVIC
An embedded interrupt controller that supports low latency interrupt processing (see [Section 2.2.2](#))
- System Control Block (SCB)
The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see [Section 2.2.3](#)).
- MPU
Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see [Section 2.2.4](#)).
- FPU
Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square-root operations. The FPU also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions (see [Section 2.2.5](#)).

1.4 Programming Model

This section describes the Cortex-M4F programming model. In addition to the individual core register descriptions, information about the processor modes and privilege levels for software execution and stacks is included.

1.4.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M4F has two modes of operation:

- Thread mode
Used to execute application software. The processor enters thread mode when it comes out of reset.
- Handler mode
Used to handle exceptions. When the processor has finished exception processing, it returns to thread mode.

The Cortex-M4F also has two privilege levels:

- Unprivileged mode
In this mode, software has the following restrictions:
 - Limited access to the MSR and MRS instructions and no use of the CPS instruction
 - No access to the system timer, NVIC, or system control block
 - Possibly restricted access to memory or peripherals
- Privileged mode
In this mode, software can use all the instructions and has access to all resources.

In thread mode, the CONTROL register (see [Figure 1-12](#)) controls whether software execution is privileged or unprivileged. In handler mode, software execution is always privileged.

Only privileged software can write to the CONTROL register to change the privilege level for software execution in thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.

1.4.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks: the main stack and the process stack, with a pointer for each held in independent registers (see the SP register on [Figure 1-5](#)).

In thread mode, the CONTROL register (see [Figure 1-12](#)) controls whether the processor uses the main stack or the process stack. In handler mode, the processor always uses the main stack. [Table 1-1](#) lists the options for processor operations.

Table 1-1. Summary of Processor Mode, Privilege Level, and Stack Use

Processor Mode	Use	Privilege Level	Stack Used
Thread	Applications	Privileged or unprivileged ⁽¹⁾	Main stack or process stack ⁽¹⁾
Handler	Exception handlers	Always privileged	Main stack

⁽¹⁾ See CONTROL ([Figure 1-12](#)).

1.4.2.1 Cortex-M4F Registers

Figure 1-3 shows the Cortex-M4F register set. Table 1-2 lists the Core registers. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.

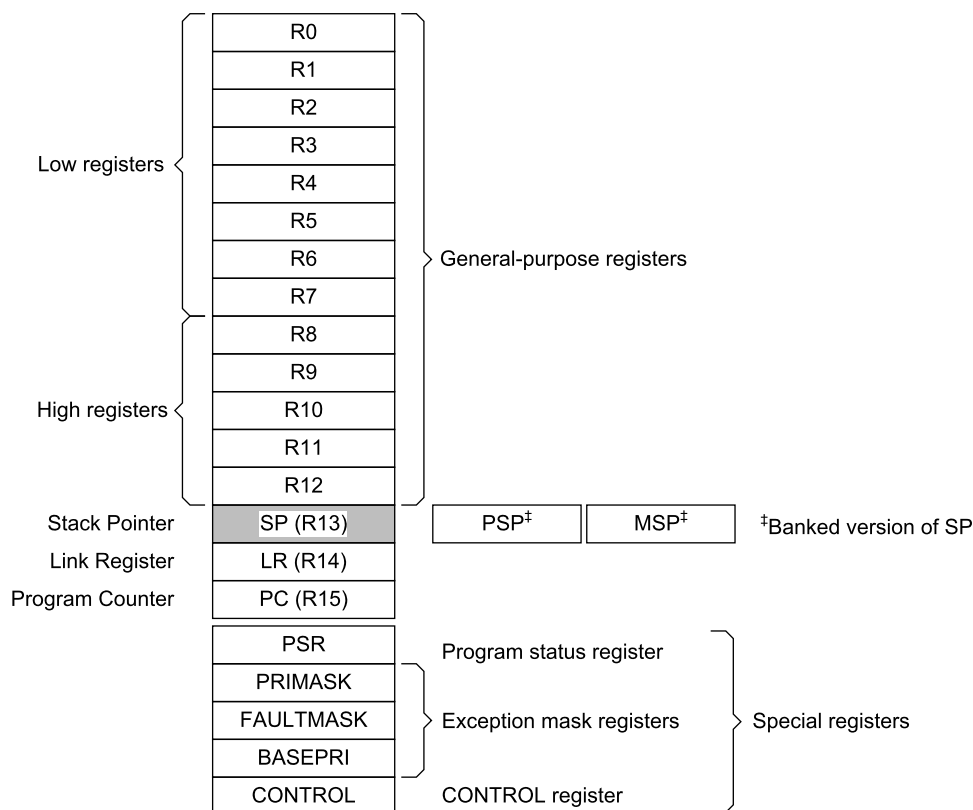


Figure 1-3. Cortex-M4F Register Set

Table 1-2. Cortex-M4F Registers

Acronym	Register Name	Section
R_0 to R_12	Cortex General-Purpose Register 0 to Cortex General-Purpose Register 12	Section 1.4.2.1.1
SP	Stack Pointer	Section 1.4.2.1.2
LR	Link Register	Section 1.4.2.1.3
PC	Program Counter	Section 1.4.2.1.4
PSR	Program Status Register	Section 1.4.2.1.5
PRIMASK	Priority Mask Register	Section 1.4.2.1.6
FAULTMASK	Fault Mask Register	Section 1.4.2.1.7
BASEPRI	Base Priority Mask Register	Section 1.4.2.1.8
CONTROL	Control Register	Section 1.4.2.1.9
FPSC	Floating-Point Status Control	Section 1.4.2.1.10

NOTE: The register type shown in the register descriptions refers to type during program execution in thread mode and handler mode. Debug access can differ.

Complex bit access types are encoded to fit into small table cells. Table 1-3 lists the codes that are used for access types in this section.

Table 1-3. Cortex-M4F Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

1.4.2.1.1 Cortex General-Purpose Register 0 (R0) to Cortex General-Purpose Register 12 (R12)

R_0 to R_12 is shown in [Figure 1-4](#) and described in [Table 1-4](#).

Return to [Summary Table](#).

The Rn registers are 32-bit general-purpose registers for data operations and can be accessed from either privileged or unprivileged mode.

Figure 1-4. R_0 to R_12 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0h																															

Table 1-4. R_0 to R_12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31:0	DATA	R/W	0h	Register data

1.4.2.1.2 Stack Pointer (SP)

SP is shown in [Figure 1-5](#) and described in [Table 1-5](#).

Return to [Summary Table](#).

The Stack Pointer (SP) is register R13. In thread mode, the function of this register changes depending on the ASP bit in the Control Register (CONTROL) register. When the ASP bit is clear, this register is the Main Stack Pointer (MSP). When the ASP bit is set, this register is the Process Stack Pointer (PSP). On reset, the ASP bit is clear, and the processor loads the MSP with the value from address 0x0000.0000. The MSP can only be accessed in privileged mode; the PSP can be accessed in either privileged or unprivileged mode.

Figure 1-5. SP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP																															
R/W-X																															

Table 1-5. SP Register Field Descriptions

Bit	Field	Type	Reset	Description
31:0	SP	R/W	X	This field is the address of the stack pointer.

1.4.2.1.3 Link Register (LR)

LR is shown in [Figure 1-6](#) and described in [Table 1-6](#).

Return to [Summary Table](#).

The Link Register (LR) is register R14, and it stores the return information for subroutines, function calls, and exceptions. The Link Register can be accessed from either privileged or unprivileged mode.

EXC_RETURN is loaded into the LR on exception entry. See [Table 1-20](#) for the values and description.

Figure 1-6. LR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINK																															
R/W-FFFFFFFFh																															

Table 1-6. LR Register Field Descriptions

Bit	Field	Type	Reset	Description
31:0	LINK	R/W	FFFFFFFFh	This field is the return address.

1.4.2.1.4 Program Counter (PC)

PC is shown in [Figure 1-7](#) and described in [Table 1-7](#).

Return to [Summary Table](#).

The Program Counter (PC) is register R15, and it contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x0000.0004. Bit 0 of the reset vector is loaded into the THUMB bit of the EPSR at reset and must be 1. The PC register can be accessed in either privileged or unprivileged mode.

Figure 1-7. PC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC																															
R/W-X																															

Table 1-7. PC Register Field Descriptions

Bit	Field	Type	Reset	Description
31:0	PC	R/W	X	This field is the current program address.

1.4.2.1.5 Program Status Register (PSR)

PSR is shown in [Figure 1-8](#) and described in [Table 1-9](#).

Return to [Summary Table](#).

NOTE: This register is also referred to as xPSR.

The Program Status Register (PSR) has three functions, and the register bits are assigned to the different functions:

- Application Program Status Register (APSR), bits 31:27, bits 19:16
- Execution Program Status Register (EPSR), bits 26:24, bits 15:10
- Interrupt Program Status Register (IPSR), bits 7:0

The PSR, IPSR, and EPSR registers can only be accessed in privileged mode; the APSR register can be accessed in either privileged or unprivileged mode.

APSR contains the current state of the condition flags from previous instruction executions.

EPSR contains the Thumb state bit and the execution state bits for the If-Then (IT) instruction or the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction. Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are always ignored. Fault handlers can examine the EPSR value in the stacked PSR to determine the operation that faulted (see [Section 1.6.7](#)).

IPSR contains the exception type number of the current Interrupt Service Routine (ISR).

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example, all of the registers can be read using PSR with the MRS instruction, or APSR only can be written to using APSR with the MSR instruction. [Table 1-8](#) lists the possible register combinations for the PSR. See the MRS and MSR instruction descriptions in the Cortex-M4 instruction set chapter in the [Arm® Cortex-M4 Devices Generic User Guide](#) for more information about how to access the program status registers.

Table 1-8. PSR Register Combinations

Register	Type	Combination
PSR	RW ⁽¹⁾⁽²⁾	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	RW ⁽¹⁾	APSR and IPSR
EAPSR	RW ⁽²⁾	APSR and EPSR

⁽¹⁾ The processor ignores writes to the IPSR bits.

⁽²⁾ Reads of the EPSR bits return zero, and the processor ignores writes to these bits.

Figure 1-8. PSR Register

31	30	29	28	27	26	25	24
N	Z	C	V	Q	ICI / IT		THUMB
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h		R-1h
23	22	21	20	19	18	17	16
RESERVED				GE			
R-0h				R/W-0h			
15	14	13	12	11	10	9	8
ICI / IT						RESERVED	
R-0h						R-0h	
7	6	5	4	3	2	1	0
ISRRNUM							
R-0h							

Table 1-9. PSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	N	R/W	0h	APSR Negative or Less Flag
30	Z	R/W	0h	APSR Zero Flag. The value of this bit is only meaningful when accessing PSR or APSR.
29	C	R/W	0h	APSR Carry or Borrow Flag. The value of this bit is only meaningful when accessing PSR or APSR.
28	V	R/W	0h	APSR Overflow Flag. The value of this bit is only meaningful when accessing PSR or APSR.
27	Q	R/W	0h	APSR DSP Overflow and Saturation Flag. The value of this bit is only meaningful when accessing PSR or APSR. This bit is cleared by software using an MRS instruction.
26:25	ICI / IT	R	0h	EPSR ICI / IT status. These bits, along with bits 15-10, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction. When EPSR holds the ICI execution state, bits 26-25 are zero. The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex-M4 instruction set chapter in the Arm Cortex-M4 Devices Generic User Guide for more information. The value of this field is only meaningful when accessing PSR or EPSR. These EPSR bits cannot be accessed using MRS and MSR instructions, but the definitions are provided to allow the stacked (E)PSR value to be decoded within an exception handler.
24	THUMB	R	1h	EPSR Thumb State. This bit indicates the Thumb state and should always be set. The following can clear the THUMB bit: The BLX, BX, and POP{PC} instructions Restoration from the stacked xPSR value on an exception return Bit 0 of the vector value on an exception entry or reset Attempting to execute instructions when this bit is clear results in a fault or lockup. See the <i>Lockup</i> section for more information. The value of this bit is only meaningful when accessing PSR or EPSR.
23:20	RESERVED	R	0h	
19:16	GE	R/W	0h	Greater Than or Equal Flags. See the description of the SEL instruction in the Cortex-M4 instruction set chapter in the Arm Cortex-M4 Devices Generic User Guide for more information. The value of this field is only meaningful when accessing PSR or APSR.

Table 1-9. PSR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15:10	ICI / IT	R	0h	<p>EPSR ICI / IT status.</p> <p>These bits, along with bits 26-25, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction.</p> <p>When an interrupt occurs during the execution of an LDM, STM, PUSH POP, VLDM, VSTM, VPUISH, or VPOP instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15-12. After servicing the interrupt, the processor returns to the register pointed to by bits 15-12 and resumes execution of the multiple load or store instruction. When EPSR holds the ICI execution state, bits 11-10 are zero.</p> <p>The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex-M4 instruction set chapter in the Arm Cortex-M4 Devices Generic User Guide for more information.</p> <p>The value of this field is only meaningful when accessing PSR or EPSR.</p>
9:8	RESERVED	R	0h	
7:0	ISRNUM	R	0h	<p>IPSR ISR Number. This field contains the exception type number of the current Interrupt Service Routine (ISR). The value of this field is only meaningful when accessing PSR or IPSR. 00h = Thread mode 01h = Reserved 02h = NMI 03h = Hard fault 04h = Memory management fault 05h = Bus fault 06h = Usage fault 07h-0Ah = Reserved 0Bh = SVCALL 0Ch = Reserved for Debug 0Dh = Reserved 0Eh = PendSV 0Fh = SysTick 10h = Interrupt Vector 0 11h = Interrupt Vector 1 ... 81h = Interrupt Vector 113</p>

1.4.2.1.6 Priority Mask Register (PRIMASK)

PRIMASK is shown in [Figure 1-9](#) and described in [Table 1-10](#).

Return to [Summary Table](#).

The PRIMASK register prevents activation of all exceptions with programmable priority. Reset, nonmaskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the PRIMASK register, and the CPS instruction may be used to change the value of the PRIMASK register. See the Cortex-M4 instruction set chapter in the [Arm® Cortex-M4 Devices Generic User Guide](#) for more information on these instructions. For more information on exception priority levels, see [Section 1.6.2](#).

Figure 1-9. PRIMASK Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PRIMASK
R-0h							R/W-0h

Table 1-10. PRIMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31:1	RESERVED	R	0h	
0	PRIMASK	R/W	0h	Priority Mask

1.4.2.1.7 Fault Mask Register (FAULTMASK)

FAULTMASK is shown in [Figure 1-10](#) and described in [Table 1-11](#).

Return to [Summary Table](#).

The FAULTMASK register prevents activation of all exceptions except for the nonmaskable interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the FAULTMASK register, and the CPS instruction may be used to change the value of the FAULTMASK register. See the Cortex-M4 instruction set chapter in the [Arm® Cortex-M4 Devices Generic User Guide](#) for more information on these instructions. For more information on exception priority levels, see [Section 1.6.2](#).

Figure 1-10. FAULTMASK Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							FAULTMASK
R-0h							R/W-0h

Table 1-11. FAULTMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31:1	RESERVED	R	0h	
0	FAULTMASK	R/W	0h	Fault Mask. The processor clears the FAULTMASK bit on exit from any exception handler except the NMI handler.

1.4.2.1.8 Base Priority Mask Register (BASEPRI)

BASEPRI is shown in [Figure 1-11](#) and described in [Table 1-12](#).

Return to [Summary Table](#).

The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the BASEPRI value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see [Section 1.6.2](#).

Figure 1-11. BASEPRI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								BASEPRI				RESERVED			
R-0h								R/W-0h				R-0h			

Table 1-12. BASEPRI Register Field Descriptions

Bit	Field	Type	Reset	Description
31:8	RESERVED	R	0h	
7:5	BASEPRI	R/W	0h	Base Priority. Any exception that has a programmable priority level with the same or lower priority as the value of this field is masked. The PRIMASK register can be used to mask all exceptions with programmable priority levels. Higher priority exceptions have lower priority levels.
4:0	RESERVED	R	0h	

1.4.2.1.9 Control Register (CONTROL)

CONTROL is shown in [Figure 1-12](#) and described in [Table 1-13](#).

Return to [Summary Table](#).

The CONTROL register controls the stack used and the privilege level for software execution when the processor is in thread mode, and indicates whether the FPU state is active. This register is only accessible in privileged mode.

Handler mode always uses the MSP, so the processor ignores explicit writes to the ASP bit of the CONTROL register when in handler mode. The exception entry and return mechanisms automatically update the CONTROL register based on the EXC_RETURN value (see [Table 1-20](#)). In an OS environment, threads running in thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, thread mode uses the MSP. To switch the stack pointer used in thread mode to the PSP, either use the MSR instruction to set the ASP bit, as detailed in the Cortex-M4 instruction set chapter in the [Arm® Cortex-M4 Devices Generic User Guide](#), or perform an exception return to thread mode with the appropriate EXC_RETURN value, as shown in [Table 1-20](#).

NOTE: When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction, ensuring that instructions after the ISB execute use the new stack pointer. See the Cortex-M4 instruction set chapter in the [Arm® Cortex-M4 Devices Generic User Guide](#).

Figure 1-12. CONTROL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					FPCA	ASP	TMPL
R-0h					R/W-0h	R/W-0h	R/W-0h

Table 1-13. CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31:3	RESERVED	R	0h	
2	FPCA	R/W	0h	Floating-Point Context Active. The Cortex-M4F uses this bit to determine whether to preserve floating-point state when processing an exception. NOTE: Two bits control when FPCA can be enabled: the ASPEN bit in the Floating-Point Context Control (FPCC) register and the DISFPCA bit in the Auxiliary Control (ACTLR) register.
1	ASP	R/W	0h	Active Stack Pointer In Handler mode. This bit reads as zero and ignores writes. The Cortex-M4F updates this bit automatically on exception return.
0	TMPL	R/W	0h	Thread Mode Privilege Level

1.4.2.1.10 Floating-Point Status Control Register (FPSC)

FPSC is shown in [Figure 1-13](#) and described in [Table 1-14](#).

Return to [Summary Table](#).

The FPSC register provides all necessary user-level control of the floating-point system.

Figure 1-13. FPSC Register

31	30	29	28	27	26	25	24
N	Z	C	V	RESERVED	AHP	DN	FZ
R/W-X	R/W-X	R/W-X	R/W-X	R-0h	R/W-X	R/W-X	R/W-X
23	22	21	20	19	18	17	16
RMODE		RESERVED					
R/W-X		R-0h					
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
IDC	RESERVED		IXC	UFC	OFC	DZC	IOC
R/W-X	R-0h		R/W-X	R/W-X	R/W-X	R/W-X	R/W-X

Table 1-14. FPSC Register Field Descriptions

Bit	Field	Type	Reset	Description
31	N	R/W	X	Negative Condition Code Flag Floating-point comparison operations update this condition code flag.
30	Z	R/W	X	Zero Condition Code Flag Floating-point comparison operations update this condition code flag.
29	C	R/W	X	Carry Condition Code Flag Floating-point comparison operations update this condition code flag.
28	V	R/W	X	Overflow Condition Code Flag Floating-point comparison operations update this condition code flag.
27	RESERVED	R	0h	
26	AHP	R/W	X	Alternative Half-Precision. When set, alternative half-precision format is selected. When clear, IEEE half-precision format is selected. The AHP bit in the FPDSC register holds the default value for this bit.
25	DN	R/W	X	Default NaN Mode. When set, any operation involving one or more NaNs returns the Default NaN. When clear, NaN operands propagate through to the output of a floating-point operation. The DN bit in the FPDSC register holds the default value for this bit.
24	FZ	R/W	X	Flush-to-Zero Mode. When set, Flush-to-Zero mode is enabled. When clear, Flush-to-Zero mode is disabled and the behavior of the floating-point system is fully compliant with the IEEE 754 standard. The FZ bit in the FPDSC register holds the default value for this bit.
23:22	RMODE	R/W	X	Rounding Mode. The specified rounding mode is used by almost all floating-point instructions. The RMODE bit in the FPDSC register holds the default value for this bit.
21:8	RESERVED	R	0h	
7	IDC	R/W	X	Input Denormal Cumulative Exception. When set, indicates this exception has occurred since 0 was last written to this bit.
6:5	RESERVED	R	0h	
4	IXC	R/W	X	Inexact Cumulative Exception. When set, indicates this exception has occurred since 0 was last written to this bit.
3	UFC	R/W	X	Underflow Cumulative Exception. When set, indicates this exception has occurred since 0 was last written to this bit.

Table 1-14. FPSC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	OFC	R/W	X	Overflow Cumulative Exception. When set, indicates this exception has occurred since 0 was last written to this bit.
1	DZC	R/W	X	Division by Zero Cumulative Exception. When set, indicates this exception has occurred since 0 was last written to this bit.
0	IOC	R/W	X	Invalid Operation Cumulative Exception. When set, indicates this exception has occurred since 0 was last written to this bit.

1.4.3 Exceptions and Interrupts

The Cortex-M4F processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See [Section 1.6.7](#) for more information.

The NVIC registers control interrupt handling. See [Section 2.2.2](#) for more information.

1.4.4 Data Types

The Cortex-M4F supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See [Section 1.5.1](#) for more information.

1.5 Memory Model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4GB of addressable memory.

[Table 1-15](#) provides the memory map for the MSP432E4 controller. In this manual, register addresses are given as a hexadecimal increment, relative to the base address of the module, as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see [Section 1.5.5](#)).

The processor reserves regions of the Private Peripheral Bus (PPB) address range for core peripheral registers (see [Chapter 2](#)).

NOTE: Within the memory map, attempts to read or write addresses in reserved spaces result in a bus fault. In addition, attempts to write addresses in the flash range also result in a bus fault.

Table 1-15. Memory Map

Start	End	Description
Memory		
0x0000.0000	0x000F.FFFF	On-chip flash
0x0010.0000	0x01FF.FFFF	Reserved
0x0200.0000	0x02FF.FFFF	On-chip ROM (16 MB)
0x0300.0000	0x1FFF.FFFF	Reserved
0x2000.0000	0x2006.FFFF	Bit-banded on-chip SRAM
0x2007.0000	0x21FF.FFFF	Reserved
0x2200.0000	0x2234.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000
0x2235.0000	0x3FFF.FFFF	Reserved
Peripherals		
0x4000.0000	0x4000.0FFF	Watchdog Timer 0
0x4000.1000	0x4000.1FFF	Watchdog Timer 1
0x4000.2000	0x4000.3FFF	Reserved
0x4000.4000	0x4000.4FFF	GPIO Port A
0x4000.5000	0x4000.5FFF	GPIO Port B
0x4000.6000	0x4000.6FFF	GPIO Port C
0x4000.7000	0x4000.7FFF	GPIO Port D
0x4000.8000	0x4000.8FFF	SSI0
0x4000.9000	0x4000.9FFF	SSI1
0x4000.A000	0x4000.AFFF	SSI2
0x4000.B000	0x4000.BFFF	SSI3

Table 1-15. Memory Map (continued)

Start	End	Description
0x4000.C000	0x4000.CFFF	UART0
0x4000.D000	0x4000.DFFF	UART1
0x4000.E000	0x4000.EFFF	UART2
0x4000.F000	0x4000.FFFF	UART3
0x4001.0000	0x4001.0FFF	UART4
0x4001.1000	0x4001.1FFF	UART5
0x4001.2000	0x4001.2FFF	UART6
0x4001.3000	0x4001.3FFF	UART7
0x4001.4000	0x4001.FFFF	Reserved
0x4002.0000	0x4002.0FFF	I2C 0
0x4002.1000	0x4002.1FFF	I2C 1
0x4002.2000	0x4002.2FFF	I2C 2
0x4002.3000	0x4002.3FFF	I2C 3
0x4002.4000	0x4002.4FFF	GPIO Port E
0x4002.5000	0x4002.5FFF	GPIO Port F
0x4002.6000	0x4002.6FFF	GPIO Port G
0x4002.7000	0x4002.7FFF	GPIO Port H
0x4002.8000	0x4002.8FFF	PWM 0
0x4002.9000	0x4002.BFFF	Reserved
0x4002.C000	0x4002.CFFF	QE10
0x4002.D000	0x4002.FFFF	Reserved
0x4003.0000	0x4003.0FFF	16/32-bit Timer 0
0x4003.1000	0x4003.1FFF	16/32-bit Timer 1
0x4003.2000	0x4003.2FFF	16/32-bit Timer 2
0x4003.3000	0x4003.3FFF	16/32-bit Timer 3
0x4003.4000	0x4003.4FFF	16/32-bit Timer 4
0x4003.5000	0x4003.5FFF	16/32-bit Timer 5
0x4003.6000	0x4003.7FFF	Reserved
0x4003.8000	0x4003.8FFF	ADC0
0x4003.9000	0x4003.9FFF	ADC1
0x4003.A000	0x4003.BFFF	Reserved
0x4003.C000	0x4003.CFFF	Analog Comparators
0x4003.D000	0x4003.DFFF	GPIO Port J
0x4003.E000	0x4003.FFFF	Reserved
0x4004.0000	0x4004.0FFF	CAN0 Controller
0x4004.1000	0x4004.1FFF	CAN1 Controller
0x4004.2000	0x4004.FFFF	Reserved
0x4005.0000	0x4005.0FFF	USB
0x4005.1000	0x4005.7FFF	Reserved
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)
0x4005.E000	0x4005.EFFF	GPIO Port G (AHB aperture)
0x4005.F000	0x4005.FFFF	GPIO Port H (AHB aperture)

Table 1-15. Memory Map (continued)

Start	End	Description
0x4006.0000	0x4006.0FFF	GPIO Port J (AHB aperture)
0x4006.1000	0x4006.1FFF	GPIO Port K (AHB aperture)
0x4006.2000	0x4006.2FFF	GPIO Port L (AHB aperture)
0x4006.3000	0x4006.3FFF	GPIO Port M (AHB aperture)
0x4006.4000	0x4006.4FFF	GPIO Port N (AHB aperture)
0x4006.5000	0x4006.5FFF	GPIO Port P (AHB aperture)
0x4006.6000	0x4006.6FFF	GPIO Port Q (AHB aperture)
0x4006.7000	0x4006.7FFF	GPIO Port R (AHB aperture)
0x4006.8000	0x4006.8FFF	GPIO Port S (AHB aperture)
0x4006.9000	0x4006.9FFF	GPIO Port T (AHB aperture)
0x4006.A000	0x400A.EFFF	Reserved
0x400A.F000	0x400A.FFFF	EEPROM and Key Locker
0x400B.0000	0x400B.5FFF	Reserved
0x400B.6000	0x400B.6FFF	1-Wire Master
0x400B.7000	0x400B.7FFF	Reserved
0x400B.8000	0x400B.8FFF	I2C 8
0x400B.9000	0x400B.9FFF	I2C 9
0x400B.A000	0x400B.FFFF	Reserved
0x400C.0000	0x400C.0FFF	I2C 4
0x400C.1000	0x400C.1FFF	I2C 5
0x400C.2000	0x400C.2FFF	I2C 6
0x400C.3000	0x400C.3FFF	I2C 7
0x400C.4000	0x400C.FFFF	Reserved
0x400D.0000	0x400D.0FFF	EPI0
0x400D.1000	0x400D.FFFF	Reserved
0x400E.0000	0x400E.0FFF	16/32-bit Timer 6
0x400E.1000	0x400E.1FFF	16/32-bit Timer 7
0x400E.2000	0x400E.BFFF	Reserved
0x400E.C000	0x400E.CFFF	Ethernet Controller
0x400E.D000	0x400F.8FFF	Reserved
0x400F.9000	0x400F.9FFF	System Exception Module
0x400F.A000	0x400F.BFFF	Reserved
0x400F.C000	0x400F.CFFF	Hibernation Module
0x400F.D000	0x400F.DFFF	Flash Memory Control
0x400F.E000	0x400F.EFFF	System Control
0x400F.F000	0x400F.FFFF	μDMA
0x4010.0000	0x41FF.FFFF	Reserved
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF
0x4400.0000	0x4402.FFFF	Reserved
0x4403.0000	0x4403.0FFF	CRC and Cryptographic Control Module
0x4403.1000	0x4403.1FFF	Reserved (4KB)
0x4403.2000	0x4403.3FFF	Reserved (8KB)
0x4403.4000	0x4403.5FFF	SHA/MD5
0x4403.6000	0x4403.7FFF	AES
0x4403.8000	0x4403.9FFF	DES
0x4403.A000	0x4403.EFFF	Reserved
0x4403.F000	0x4403.FFFF	Reserved (4KB)

Table 1-15. Memory Map (continued)

Start	End	Description
0x4404.0000	0x4404.FFFF	Reserved (64KB)
0x4405.0000	0x4405.0FFF	LCD
0x4405.1000	0x4405.3FFF	Reserved
0x4405.4000	0x4405.4FFF	EPHY 0
0x4405.5000	0x5FFF.FFFF	Reserved
0x6000.0000	0xDFFF.FFFF	EPI0 mapped peripheral and RAM
Private Peripheral Bus		
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)
0xE000.3000	0xE000.DFFF	Reserved
0xE000.E000	0xE000.EFFF	Cortex-M4F Peripherals (SysTick, NVIC, MPU, FPU, and SCB)
0xE000.F000	0xE003.FFFF	Reserved
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)
0xE004.1000	0xE004.1FFF	Embedded Trace Macrocell (ETM)
0xE004.2000	0xFFFF.FFFF	Reserved

1.5.1 Memory Regions, Types, and Attributes

The memory map and the programming of the MPU split the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

- Normal: The processor can order transactions for efficiency and perform speculative reads.
- Device: The processor preserves transaction order relative to other transactions to Device or Strongly Ordered memory.
- Strongly Ordered: The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly Ordered memory mean that the memory system can buffer a write to Device memory but must not buffer a write to Strongly Ordered memory.

An additional memory attribute is Execute Never (XN), which means the processor prevents instruction accesses. A fault exception is generated only when an instruction is executed from an XN region.

1.5.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not ensure that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see [Section 1.5.4](#)).

However, the memory system does ensure ordering of accesses to Device and Strongly Ordered memory. For two memory access instructions A1 and A2, if both A1 and A2 are accesses to either Device or Strongly Ordered memory, and if A1 occurs before A2 in program order, A1 is always observed before A2.

1.5.3 Behavior of Memory Accesses

[Table 1-16](#) shows the behavior of accesses to each region in the memory map. See [Section 1.5.1](#) for more information on memory types and the XN attribute. MSP432E4 devices may have reserved memory areas within the address ranges listed in [Table 1-16](#) (see [Table 1-15](#) for more information).

Table 1-16. Memory Access Behavior

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000.0000 to 0x1FFF.FFFF	Code	Normal	–	This executable region is for program code. Data can also be stored here.
0x2000.0000 to 0x3FFF.FFFF	SRAM	Normal	–	This executable region is for data. Code can also be stored here. This region includes bit-band and bit-band alias areas (see Table 1-17).
0x4000.0000 to 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit-band and bit-band alias areas (see Table 1-18).
0x6000.0000 to 0x9FFF.FFFF	External RAM	Normal	–	This executable region is for data.
0xA000.0000 to 0xDFFF.FFFF	External device	Device	XN	This region is for external device memory.
0xE000.0000 to 0xE00F.FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010.0000 to 0xFFFF.FFFF	Reserved	–	–	

The Code, SRAM, and external RAM regions can hold programs. However, TI recommends that programs always use the Code region because the Cortex-M4F has separate buses that can perform instruction fetches and data accesses simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see [Section 2.2.4](#).

The Cortex-M4F prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

1.5.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always ensure the order of the corresponding memory transactions for the following reasons:

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces.
- Memory or devices in the memory map have different wait states.
- Some memory accesses are buffered or speculative.

[Section 1.5.2](#) describes the cases where the memory system ensures the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M4F has the following memory barrier instructions:

- The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.
- The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.
- The Instruction Synchronization Barrier (ISB) instruction ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

Memory barrier instructions can be used in the following situations:

- MPU programming
 - If the MPU settings are changed and the change must be effective on the very next instruction, use a DSB instruction to make sure that the effect of the MPU takes place immediately at the end of context switching.
 - Use an ISB instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an ISB instruction is not

required.

- Vector table

If the program changes an entry in the vector table and then enables the corresponding exception, use a DMB instruction between the operations. The DMB instruction ensures that if the exception is taken immediately after being enabled, the processor uses the new exception vector.

- Self-modifying code

If a program contains self-modifying code, use an ISB instruction immediately after the code modification in the program. The ISB instruction ensures subsequent instruction execution uses the updated program.

- Memory map switching

If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. The DSB instruction ensures subsequent instruction execution uses the updated memory map.

- Dynamic exception priority change

When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. The change then takes effect on completion of the DSB instruction.

Memory accesses to Strongly Ordered memory, such as the System Control Block, do not require the use of DMB instructions.

For more information on the memory barrier instructions, see the Cortex-M4 instruction set chapter in the [Arm Cortex-M4 Devices Generic User Guide](#).

1.5.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1MB of the SRAM and peripheral memory regions. Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region, as shown in [Table 1-17](#). Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in [Table 1-18](#). For the specific address range of the bit-band regions, see [Table 1-15](#).

NOTE: A word access to the SRAM or the peripheral bit-band alias region maps to a single bit in the SRAM or peripheral bit-band region.

A word access to a bit-band address results in a word access to the underlying memory, and similarly for halfword and byte accesses. This allows bit-band accesses to match the access requirements of the underlying peripheral.

Table 1-17. SRAM Memory Bit-Banding Regions

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x2000.0000	0x2006.FFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM accesses, but this region is also bit addressable through bit-band alias.
0x2200.0000	0x2234.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit-band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 1-18. Peripheral Memory Bit-Banding Regions

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4000.0000	0x400F.FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.

Table 1-18. Peripheral Memory Bit-Banding Regions (continued)

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4200.0000	0x43FF.FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit-band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

The following formula shows how the alias region maps onto the bit-band region:

$$\text{bit_word_offset} = (\text{byte_offset} \times 32) + (\text{bit_number} \times 4)$$

$$\text{bit_word_addr} = \text{bit_band_base} + \text{bit_word_offset}$$

where:

bit_word_offset = The position of the target bit in the bit-band memory region.

bit_word_addr = The address of the word in the alias memory region that maps to the targeted bit.

bit_band_base = The starting address of the alias region.

byte_offset = The number of the byte in the bit-band region that contains the targeted bit.

bit_number = The bit position, 0-7, of the targeted bit.

Figure 1-14 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FF.FFE0 maps to bit 0 of the bit-band byte at 0x200F.FFFF:

$$0x23FF.FFE0 = 0x2200.0000 + (0x000F.FFFF \times 32) + (0 \times 4)$$

- The alias word at 0x23FF.FFFC maps to bit 7 of the bit-band byte at 0x200F.FFFF:

$$0x23FF.FFFC = 0x2200.0000 + (0x000F.FFFF \times 32) + (7 \times 4)$$

- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

$$0x2200.0000 = 0x2200.0000 + (0 \times 32) + (0 \times 4)$$

- The alias word at 0x2200.001C maps to bit 7 of the bit-band byte at 0x2000.0000:

$$0x2200.001C = 0x2200.0000 + (0 \times 32) + (7 \times 4)$$

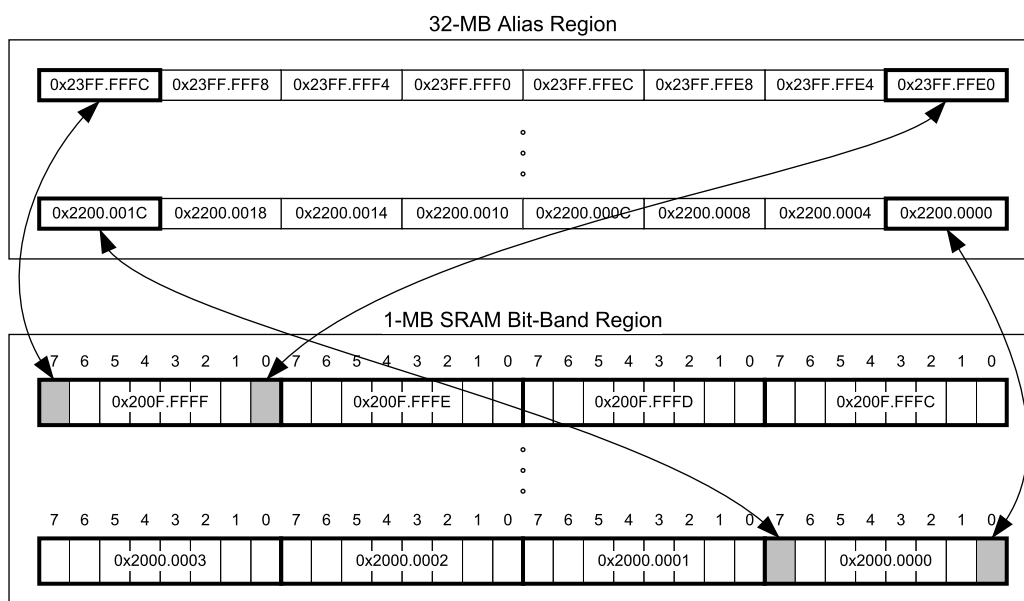


Figure 1-14. Bit-Band Mapping

1.5.5.1 Directly Accessing an Alias Region

Writing to a word in the alias region updates one bit in the bit-band region.

Bit 0 of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit 0 set writes 1 to the bit-band bit, and writing a value with bit 0 clear writes 0 to the bit-band bit.

Bits 31:1 of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

When reading a word in the alias region, 0x0000.0000 indicates that the targeted bit in the bit-band region is clear and 0x0000.0001 indicates that the targeted bit in the bit-band region is set.

1.5.5.2 Directly Accessing a Bit-Band Region

[Section 1.5.3](#) describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

1.5.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least significant byte (LSByte) of a word stored at the lowest-numbered byte, and the most significant byte (MSByte) stored at the highest-numbered byte. [Figure 1-15](#) shows how data is stored.

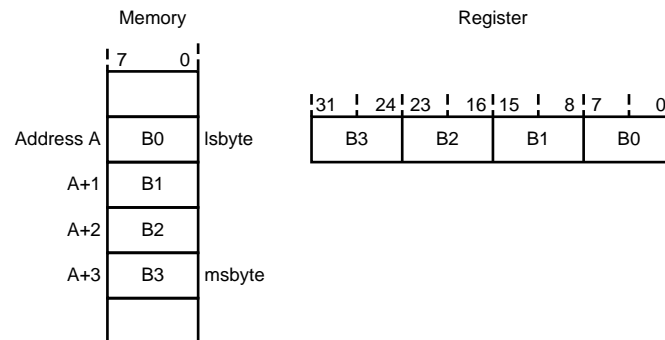


Figure 1-15. Data Storage

1.5.7 Synchronization Primitives

The Cortex-M4F instruction set includes pairs of synchronization primitives. These primitives provide a nonblocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use these primitives to perform an ensured read-modify-write memory update sequence or for a semaphore mechanism.

NOTE: The available pairs of synchronization primitives are available only for single processor use and should not be used with multiprocessor systems.

A pair of synchronization primitives consists of:

- A Load-Exclusive instruction, which is used to read the value of a memory location and requests exclusive access to that location.
- A Store-Exclusive instruction, which is used to try to write to the same memory location and returns a status bit to a register. If this status bit is clear, it indicates that the thread or process gained exclusive access to the memory and the write succeeds. If this status bit is set, it indicates that the thread or process did not gain exclusive access to the memory and no write was performed.

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- The word instructions LDREX and STREX

- The halfword instructions LDREXH and STREXH
- The byte instructions LDREXB and STREXB

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.

To perform an exclusive read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Modify the value, as required.
3. Use a Store-Exclusive instruction to try to write the new value back to the memory location.
4. Test the returned status bit.

If the status bit is clear, the read-modify-write completed successfully. If the status bit is set, no write was performed, which indicates that the value returned at step 1 might be out of date. The software must retry the entire read-modify-write sequence.

Software can use the synchronization primitives to implement a semaphore as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive instruction to write the claim value to the semaphore address.
3. If the returned status bit from Step 2 indicates that the Store-Exclusive succeeded, then the software has claimed the semaphore. However, if the Store-Exclusive instruction failed, another process might have claimed the semaphore after the software performed Step 1.

The Cortex-M4F includes an exclusive access monitor that tags the fact that the processor has executed a Load-Exclusive instruction. The processor removes its exclusive access tag if:

- It executes a CLREX instruction.
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs, which means the processor can resolve semaphore conflicts between different threads.

For more information about the synchronization primitive instructions, see the Cortex-M4 instruction set chapter in the [Arm Cortex-M4 Devices Generic User Guide](#).

1.6 Exception Model

The Arm Cortex-M4F processor and the NVIC prioritize and handle all exceptions in handler mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the ISR. The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables performance of back-to-back interrupts to be performed without the overhead of state saving and restoration.

[Table 1-19](#) lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on interrupts (see [Table 1-19](#)).

Priorities on the system handlers are set with the NVIC System Handler Priority *n* (SYSPRIn) registers. Interrupts are enabled through the NVIC Interrupt Set Enable *n* (ENn) register and prioritized with the NVIC Interrupt Priority *n* (PRIn) registers. Priorities can be grouped by splitting priority levels into preemption priorities and subpriorities. All the interrupt registers are described in [Section 2.2.2](#).

Internally, the highest user-programmable priority (0) is treated as fourth priority, after a Reset, Nonmaskable Interrupt (NMI), and a Hard Fault, in that order. The default priority is 0 for all the programmable priorities.

NOTE: After a write to clear an interrupt source, several processor cycles may be needed for the NVIC to identify the interrupt source deassert. Thus if the interrupt clear is done as the last action in an interrupt handler, it is possible for the interrupt handler to complete while the NVIC sees the interrupt as still asserted, causing errant re-entry of the interrupt handler. This situation can be avoided by either clearing the interrupt source at the beginning of the interrupt handler or by performing a read or write after the write to clear the interrupt source (and flush the write buffer).

See [Section 2.2.2](#) for more information on exceptions and interrupts.

1.6.1 Exception States

Each exception is in one of the following states:

- **Inactive**
The exception is not active and not pending.
- **Pending**
The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- **Active**
An exception is being serviced by the processor but has not completed.

NOTE: An exception handler can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.

- **Active and Pending**
The exception is being serviced by the processor, and there is a pending exception from the same source.

1.6.2 Exception Types

The exception types are:

- **Reset**
Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in thread mode.
- **NMI**
A non-maskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the Interrupt Control and State (INTCTRL) register. This exception has the highest priority other than reset. NMI is permanently enabled and has a fixed priority of -2. NMIs cannot be masked or prevented from activation by any other exception or preempted by any exception other than reset.
- **Hard Fault**
A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
- **Memory Management Fault**
A memory management fault is an exception that occurs because of a memory protection related fault, including access violation and no match. The MPU or the fixed memory protection constraints determine this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to Execute Never (XN) memory regions, even if the MPU is disabled.
- **Bus Fault**
A bus fault is an exception that occurs because of a memory-related fault for an instruction or data memory transaction such as a prefetch fault or a memory access fault. This fault can be enabled or disabled.

- **Usage Fault**

A usage fault is an exception that occurs because of a fault related to instruction execution, such as:

- An undefined instruction
- An illegal unaligned access
- Invalid state on instruction execution
- An error on exception return

An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.

- **SVCall**

A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.

- **Debug Monitor**

This exception is caused by the debug monitor (when not halting). This exception is active only when enabled. This exception does not activate if it is a lower priority than the current activation.

- **PendSV**

PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the Interrupt Control and State (INTCTRL) register.

- **SysTick**

A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the Interrupt Control and State (INTCTRL) register. In an OS environment, the processor can use this exception as system tick.

- **Interrupt (IRQ)**

An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.

For a list of the device-specific interrupts, see the data sheet.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that [Table 1-19](#) shows as having configurable priority (see the SYSHNDCTRL register in [Section 2.5.11](#) and the DIS0 register in [Section 2.4.2](#)).

For more information about hard faults, memory management faults, bus faults, and usage faults, see [Section 1.7](#).

Table 1-19. Exception Types

Exception Type	Vector Number	Priority ⁽¹⁾	Vector Address or Offset ⁽²⁾	Activation
–	0	–	0x0000.0000	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	–3 (highest)	0x0000.0004	Asynchronous
Nonmaskable Interrupt (NMI)	2	–2	0x0000.0008	Asynchronous
Hard Fault	3	–1	0x0000.000C	
Memory Management	4	Programmable ⁽³⁾	0x0000.0010	Synchronous
Bus Fault	5	Programmable ⁽³⁾	0x0000.0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	Programmable ⁽³⁾	0x0000.0018	Synchronous
–	7-10	–	–	Reserved

⁽¹⁾ 0 is the default priority for all the programmable priorities.

⁽²⁾ See [Section 1.6.4](#).

⁽³⁾ See SYSPRI1 in [Section 2.5.8](#).

Table 1-19. Exception Types (continued)

Exception Type	Vector Number	Priority ⁽¹⁾	Vector Address or Offset ⁽²⁾	Activation
SVCall	11	Programmable ⁽³⁾	0x0000.002C	Synchronous
Debug Monitor	12	Programmable ⁽³⁾	0x0000.0030	Synchronous
–	13	–	–	Reserved
PendSV	14	Programmable ⁽³⁾	0x0000.0038	Asynchronous
SysTick	15	Programmable ⁽³⁾	0x0000.003C	Asynchronous
Interrupts	16 and above	Programmable ⁽⁴⁾	0x0000.0040 and above	Asynchronous

⁽⁴⁾ See PRIn registers in [Section 2.4.6](#).

1.6.3 Exception Handlers

The processor handles exceptions using:

- Interrupt Service Routines (ISRs)
Interrupts (IRQx) are the exceptions handled by ISRs.
- Fault Handlers
Hard fault, memory management fault, usage fault, and bus fault are fault exceptions handled by the fault handlers.
- System Handlers
NMI, PendSV, SVCall, SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

1.6.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in [Table 1-19](#). [Figure 1-16](#) shows the order of the exception vectors in the vector table. The least significant bit of each vector must be 1, indicating that the exception handler is Thumb code.

Exception number	IRQ number	Offset	Vector
(N+16)	(N)	0x040 + 0x(N*4)	IRQ N
.	.	.	.
.	.	0x004C	.
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			
9			
8			Reserved
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
0		0x0000	Initial SP value

Figure 1-16. Vector Table

On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the Vector Table Offset (VTABLE) register to relocate the vector table start address to a different memory location, in the range 0x0000.0400 to 0x3FFF.FC00 (see [Section 1.6.4](#)). When configuring the VTABLE register, the offset must be aligned on a 1024-byte boundary.

1.6.5 Exception Priorities

As [Table 1-19](#) shows, all exceptions have an associated priority, with a lower priority value indicating a higher priority and configurable priorities for all exceptions except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see [Section 2.5.8](#) and [Section 2.4.6](#).

NOTE: Configurable priority values for the MSP432E4 implementation are in the range 0 to 7. This means that the Reset, Hard Fault, and NMI exceptions (with fixed negative priority values) always have higher priority than any other exception.

For example, assigning a higher priority value to `IRQ[0]` and a lower priority value to `IRQ[1]` means that `IRQ[1]` has higher priority than `IRQ[0]`. If both `IRQ[1]` and `IRQ[0]` are asserted, `IRQ[1]` is processed before `IRQ[0]`.

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both `IRQ[0]` and `IRQ[1]` are pending and have the same priority, then `IRQ[0]` is processed before `IRQ[1]`.

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, regardless of the exception number. However, the status of the new interrupt changes to pending.

1.6.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This grouping divides each interrupt priority register entry into two fields:

- An upper field that defines the group priority
- A lower field that defines a subpriority within the group

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see [Section 2.5.5](#).

1.6.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

- **Preemption**

When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See [Section 1.6.6](#) for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See [Section 1.6.7.1](#) for more information.

- **Return**

Return occurs when the exception handler is completed, and when there is no pending exception with sufficient priority to service and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See [Section 1.6.7.2](#) for more information.

- **Tail-Chaining**

This mechanism speeds up exception servicing. On completion of an exception handler, if a pending exception meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

- **Late-Arriving**

This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher-priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the saved state is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late-arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. When the late-arriving exception returns from the exception handler, the normal tail-chaining rules apply.

1.6.7.1 Exception Entry

Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers (see PRIMASK on , FAULTMASK on , and BASEPRI on). An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as *stacking* and the structure of eight data words is referred to as *stack frame*.

When using floating-point routines, the Cortex-M4F processor automatically stacks the architected floating-point state on exception entry. Figure 1-17 shows the Cortex-M4F stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

NOTE: Where stack space for floating-point state is not allocated, the stack frame is the same as that of Armv7-M implementations without an FPU. Figure 1-17 shows this stack frame.

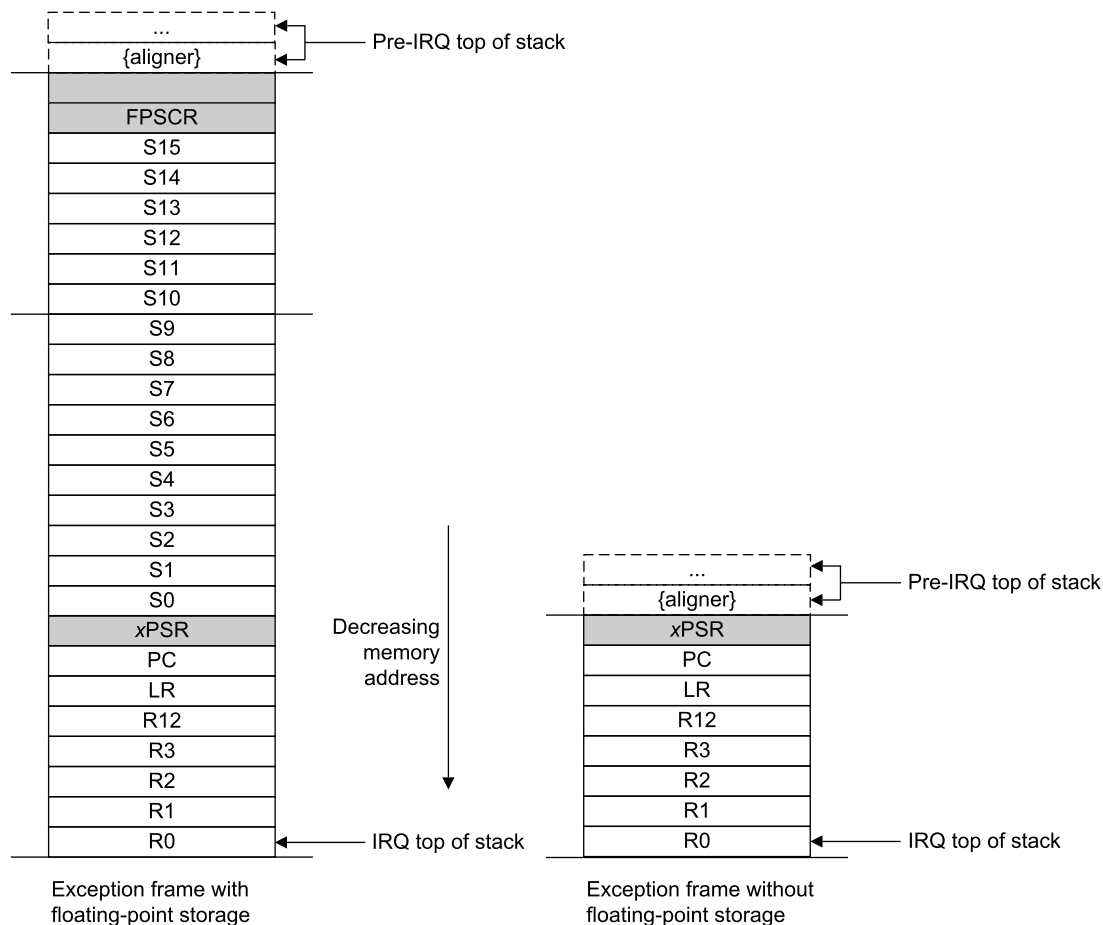


Figure 1-17. Exception Stack Frame

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame.

The stack frame includes the return address, which is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

In parallel with the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR, indicating which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher-priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher-priority exception occurs during exception entry, known as late arrival, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception.

1.6.7.2 Exception Return

Exception return occurs when the processor is in handler mode and executes one of the following instructions to load the EXC_RETURN value into the PC:

- An LDM or POP instruction that loads the PC
- A BX instruction using any register
- An LDR instruction with the PC as the destination

EXC_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest 5 bits of this value provide information on the return stack and processor mode. [Table 1-20](#) shows the EXC_RETURN values with a description of the exception return behavior.

EXC_RETURN bits 31:5 are all set. When this value is loaded into the PC, it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

Table 1-20. Exception Return Behavior

EXC_RETURN[31:0]	Description
0xFFFF.FFE0	Reserved
0xFFFF.FFE1	Return to handler mode. Exception return uses floating-point state from MSP. Execution uses MSP after return.
0xFFFF.FFE2 to 0xFFFF.FFE8	Reserved
0xFFFF.FFE9	Return to thread mode. Exception return uses floating-point state from MSP. Execution uses MSP after return.
0xFFFF.FFEA to 0xFFFF.FFEC	Reserved
0xFFFF.FFED	Return to thread mode. Exception return uses floating-point state from PSP. Execution uses PSP after return.
0xFFFF.FFEE to 0xFFFF.FFF0	Reserved
0xFFFF.FFF1	Return to handler mode. Exception return uses nonfloating-point state from MSP. Execution uses MSP after return.
0xFFFF.FFF2 to 0xFFFF.FFF8	Reserved
0xFFFF.FFF9	Return to thread mode. Exception return uses nonfloating-point state from MSP. Execution uses MSP after return.
0xFFFF.FFFA to 0xFFFF.FFFC	Reserved
0xFFFF.FFFD	Return to thread mode. Exception return uses nonfloating-point state from PSP. Execution uses PSP after return.
0xFFFF.FFFE to 0xFFFF.FFFF	Reserved

1.7 Fault Handling

Faults are a subset of the exceptions (see [Section 1.6](#)). The following conditions generate a fault:

- A bus error on an instruction fetch or vector table load or a data access
- An internally detected error such as an undefined instruction or an attempt to change state with a BX instruction
- Trying to execute an instruction from a memory region marked as Non-Executable (XN)
- An MPU fault because of a privilege violation or an attempt to access an unmanaged region

1.7.1 Fault Types

[Table 1-21](#) lists the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See [Section 2.5.12](#) for more information about the fault status registers.

Table 1-21. Faults

Fault	Handler	Fault Status Register	Bit Name
Bus error on a vector read	Hard fault	Hard Fault Status (HFAULTSTAT)	VECT
Fault escalated to a hard fault	Hard fault	Hard Fault Status (HFAULTSTAT)	FORCED
MPU or default memory mismatch on instruction access	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	IERR ⁽¹⁾
MPU or default memory mismatch on data access	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	DERR
MPU or default memory mismatch on exception stacking	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MSTKE
MPU or default memory mismatch on exception unstacking	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MUSTKE
MPU or default memory mismatch during lazy floating-point state preservation	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MLSPERR
Bus error during exception stacking	Bus fault	Bus Fault Status (BFAULTSTAT)	BSTKE
Bus error during exception unstacking	Bus fault	Bus Fault Status (BFAULTSTAT)	BUSTKE
Bus error during instruction prefetch	Bus fault	Bus Fault Status (BFAULTSTAT)	IBUS
Bus error during lazy floating-point state preservation	Bus fault	Bus Fault Status (BFAULTSTAT)	BLSPE
Precise data bus error	Bus fault	Bus Fault Status (BFAULTSTAT)	PRECISE
Imprecise data bus error	Bus fault	Bus Fault Status (BFAULTSTAT)	IMPRE
Attempt to access a coprocessor	Usage fault	Usage Fault Status (UFAULTSTAT)	NOCP
Undefined instruction	Usage fault	Usage Fault Status (UFAULTSTAT)	UNDEF
Attempt to enter an invalid instruction set state ⁽²⁾	Usage fault	Usage Fault Status (UFAULTSTAT)	INVSTAT
Invalid EXC_RETURN value	Usage fault	Usage Fault Status (UFAULTSTAT)	INVPC
Illegal unaligned load or store	Usage fault	Usage Fault Status (UFAULTSTAT)	UNALIGN
Divide by 0	Usage fault	Usage Fault Status (UFAULTSTAT)	DIV0

⁽¹⁾ Occurs on an access to an XN region even if the MPU is disabled.

⁽²⁾ Trying to use an instruction set other than the Thumb instruction set, or returning to a nonload-store-multiply instruction with ICI continuation.

1.7.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see SYSPRI1 in [Section 2.5.8](#)). Software can disable execution of the handlers for these faults (see SYSHNDCTRL in [Section 2.5.11](#)).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler, as described in [Section 1.6](#).

In some situations, a fault with configurable priority is treated as a hard fault. This process is called *priority escalation*, and the fault is described as escalated to hard fault. Escalation to hard fault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This situation occurs because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. Thus if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

NOTE: Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault.

1.7.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory-management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in [Table 1-22](#).

Table 1-22. Fault Status and Fault Address Registers

Handler	Status Register Name	Address Register Name	Section
Hard fault	Hard Fault Status (HFAULTSTAT)	–	
Memory management fault	Memory Management Fault Status (MFAULTSTAT)	Memory Management Fault Address (MMADDR)	
Bus fault	Bus Fault Status (BFAULTSTAT)	Bus Fault Address (FAULTADDR)	
Usage fault	Usage Fault Status (UFAULTSTAT)	–	

1.7.4 Lockup

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in the lockup state, it does not execute any instructions. The processor remains in lockup state until it is reset, an NMI occurs, or it is halted by a debugger.

NOTE: If the lockup state occurs from the NMI handler, a subsequent NMI does not cause the processor to leave the lockup state.

1.8 Power Management

The Cortex-M4F processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep-sleep mode stops the system clock and switches off the PLL and flash memory.

The SLEEPDEEP bit of the System Control (SYSCTRL) register selects which sleep mode is used (see [Section 2.5.6](#)). For more information about the behavior of the sleep modes, see [Section 4.1.6](#).

This section describes the mechanisms for entering sleep mode and the conditions for waking up from sleep mode, both of which apply to sleep mode and deep-sleep mode.

1.8.1 Entering Sleep Modes

This section describes the mechanisms software can use to put the processor into one of the sleep modes.

The system can generate spurious wake-up events; for example, a debug operation wakes up the processor. Therefore, software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

1.8.1.1 Wait for Interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true (see [Section 1.8.2.1](#)). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the Cortex-M4 instruction set chapter in the [Arm Cortex-M4 Devices Generic User Guide](#) for more information.

1.8.1.2 Wait for Event

The wait for event instruction, WFE, causes entry to sleep mode conditional on the value of a 1-bit event register. When the processor executes a WFE instruction, it checks the event register. If the register is 0, the processor stops executing instructions and enters sleep mode. If the register is 1, the processor clears the register and continues executing instructions without entering sleep mode.

If the event register is 1, the processor must not enter sleep mode when a WFE instruction is executed. Typically, this situation occurs if an SEV instruction has been executed. Software cannot access this register directly.

See the Cortex-M4 instruction set chapter in the [Arm Cortex-M4 Devices Generic User Guide](#) for more information.

1.8.1.3 Sleep-on-Exit

If the SLEEPEXIT bit of the SYSCTRL register is set, when the processor completes the execution of all exception handlers, it returns to Thread mode and immediately enters sleep mode. This mechanism can be used in applications that only require the processor to run when an exception occurs.

1.8.2 Wake Up From Sleep Mode

The conditions for the processor to wake up depend on the mechanism that caused it to enter sleep mode.

1.8.2.1 Wake Up From WFI or Sleep-on-Exit

Normally, the processor wakes up only when the NVIC detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears PRIMASK. For more information about PRIMASK and FAULTMASK, see [and](#) .

1.8.2.2 Wake Up From WFE

The processor wakes up if it detects an exception with sufficient priority to cause exception entry.

In addition, if the SEVONPEND bit in the SYSCTRL register is set, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about SYSCTRL, see [Section 2.5.6](#).

1.9 Instruction Set Summary

The processor implements a version of the Thumb instruction set. [Table 1-23](#) lists the supported instructions.

NOTE: The following conventions apply in Table 1-23:

- Angle brackets, <>, enclose alternative forms of the operand.
- Braces, {}, enclose optional operands.
- The Operands column is not exhaustive.
- Op2 is a flexible second operand that can be either a register or a constant.
- Most instructions can use an optional condition code suffix.

For more information on the instructions and operands, see the instruction descriptions in the Arm Cortex-M4 Technical Reference Manual.

Table 1-23. Cortex-M4F Instruction Summary

Mnemonic	Operands	Brief Description	Flags
ADC, ADCS	{Rd,} Rn, Op2	Add with carry	N,Z,C,V
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V
ADD, ADDW	{Rd,} Rn, #imm12	Add	–
ADR	Rd, label	Load PC-relative address	–
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C
ASR, ASRS	Rd, Rm, <Rs #n>	Arithmetic shift right	N,Z,C
B	label	Branch	–
BFC	Rd, #lsb, #width	Bit field clear	–
BFI	Rd, Rn, #lsb, #width	Bit field insert	–
BIC, BICS	{Rd,} Rn, Op2	Bit clear	N,Z,C
BKPT	#imm	Breakpoint	–
BL	label	Branch with link	–
BLX	Rm	Branch indirect with link	–
BX	Rm	Branch indirect	–
CBNZ	Rn, label	Compare and branch if nonzero	–
CBZ	Rn, label	Compare and branch if zero	–
CLREX	–	Clear exclusive	–
CLZ	Rd, Rm	Count leading zeros	–
CMN	Rn, Op2	Compare negative	N,Z,C,V
CMP	Rn, Op2	Compare	N,Z,C,V
CPSID	i	Change processor state, disable interrupts	–
CPSIE	i	Change processor state, enable interrupts	–
DMB	–	Data memory barrier	–
DSB	–	Data synchronization barrier	–
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N,Z,C
ISB	–	Instruction synchronization barrier	–
IT	–	If-Then condition block	–
LDM	Rn(!), reglist	Load multiple registers, increment after	–
LDMDB, LDMEA	Rn(!), reglist	Load multiple registers, decrement before	–
LDMFD, LDMIA	Rn(!), reglist	Load multiple registers, increment after	–
LDR	Rt, [Rn, #offset]	Load register with word	–
LDRB, LDRBT	Rt, [Rn, #offset]	Load register with byte	–
LDRD	Rt, Rt2, [Rn, #offset]	Load register with 2 bytes	–
LDREX	Rt, [Rn, #offset]	Load register exclusive	–
LDREXB	Rt, [Rn]	Load register exclusive with byte	–
LDREXH	Rt, [Rn]	Load register exclusive with halfword	–
LDRH, LDRHT	Rt, [Rn, #offset]	Load register with halfword	–

Table 1-23. Cortex-M4F Instruction Summary (continued)

Mnemonic	Operands	Brief Description	Flags
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load register with signed byte	–
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load register with signed halfword	–
LDRT	Rt, [Rn, #offset]	Load register with word	–
LSL, LSLS	Rd, Rm, <Rs #n>	Logical shift left	N,Z,C
LSR, LSRS	Rd, Rm, <Rs #n>	Logical shift right	N,Z,C
MLA	Rd, Rn, Rm, Ra	Multiply with accumulate, 32-bit result	–
MLS	Rd, Rn, Rm, Ra	Multiply and subtract, 32-bit result	–
MOV, MOVS	Rd, Op2	Move	N,Z,C
MOV, MOVW	Rd, #imm16	Move 16-bit constant	N,Z,C
MOVT	Rd, #imm16	Move top	–
MRS	Rd, spec_reg	Move from special register to general register	–
MSR	spec_reg, Rm	Move from general register to special register	N,Z,C,V
MUL, MULS	{Rd,} Rn, Rm	Multiply, 32-bit result	N,Z
MVN, MVNS	Rd, Op2	Move NOT	N,Z,C
NOP	–	No operation	–
ORN, ORNS	{Rd,} Rn, Op2	Logical OR NOT	N,Z,C
ORR, ORRS	{Rd,} Rn, Op2	Logical OR	N,Z,C
PKHTB, PKHBT	{Rd,} Rn, Rm, Op2	Pack halfword	–
POP	reglist	Pop registers from stack	–
PUSH	reglist	Push registers onto stack	–
QADD	{Rd,} Rn, Rm	Saturating add	Q
QADD16	{Rd,} Rn, Rm	Saturating add 16	–
QADD8	{Rd,} Rn, Rm	Saturating add 8	–
QASX	{Rd,} Rn, Rm	Saturating add and subtract with exchange	–
QDADD	{Rd,} Rn, Rm	Saturating double and add	Q
QDSUB	{Rd,} Rn, Rm	Saturating double and subtract	Q
QSAX	{Rd,} Rn, Rm	Saturating subtract and add with exchange	–
QSUB	{Rd,} Rn, Rm	Saturating subtract	Q
QSUB16	{Rd,} Rn, Rm	Saturating subtract 16	–
QSUB8	{Rd,} Rn, Rm	Saturating subtract 8	–
RBIT	Rd, Rn	Reverse bits	–
REV	Rd, Rn	Reverse byte order in a word	–
REV16	Rd, Rn	Reverse byte order in each halfword	–
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	–
ROR, RORS	Rd, Rm, <Rs #n>	Rotate right	N,Z,C
RRX, RRXS	Rd, Rm	Rotate right with extend	N,Z,C
RSB, RSBS	{Rd,} Rn, Op2	Reverse subtract	N,Z,C,V
SADD16	{Rd,} Rn, Rm	Signed add 16	GE
SADD8	{Rd,} Rn, Rm	Signed add 8	GE
SASX	{Rd,} Rn, Rm	Signed add and subtract with exchange	GE
SBC, SBCS	{Rd,} Rn, Op2	Subtract with carry	N,Z,C,V
SBFX	Rd, Rn, #lsb, #width	Signed bit field extract	–
SDIV	{Rd,} Rn, Rm	Signed divide	–
SEL	{Rd,} Rn, Rm	Select bytes	–
SEV	–	Send event	–
SHADD16	{Rd,} Rn, Rm	Signed halving add 16	–
SHADD8	{Rd,} Rn, Rm	Signed halving add 8	–

Table 1-23. Cortex-M4F Instruction Summary (continued)

Mnemonic	Operands	Brief Description	Flags
SHASX	{Rd,} Rn, Rm	Signed halving add and subtract with exchange	–
SHSAX	{Rd,} Rn, Rm	Signed halving add and subtract with exchange	–
SHSUB16	{Rd,} Rn, Rm	Signed halving subtract 16	–
SHSUB8	{Rd,} Rn, Rm	Signed halving subtract 8	–
SMLABB, SMLABT, SMLATB, SMLATT	Rd, Rn, Rm, Ra	Signed multiply accumulate long (halfwords)	Q
SMLAD, SMLADX	Rd, Rn, Rm, Ra	Signed multiply accumulate dual	Q
SMLAL	RdLo, RdHi, Rn, Rm	Signed multiply with accumulate (32×32+64), 64-bit result	–
SMLALBB, SMLALBT, SMLALTB, SMLALTT	RdLo, RdHi, Rn, Rm	Signed multiply accumulate long (halfwords)	–
SMLALD, SMLALDX	RdLo, RdHi, Rn, Rm	Signed multiply accumulate long dual	–
SMLAWB, SMLAWT	Rd, Rn, Rm, Ra	Signed multiply accumulate, word by halfword	Q
SMLSd SMLSdX	Rd, Rn, Rm, Ra	Signed multiply subtract dual	Q
SMLSd SMLSdX	RdLo, RdHi, Rn, Rm	Signed multiply subtract long dual	–
SMMLA	Rd, Rn, Rm, Ra	Signed most significant word multiply accumulate	–
SMMLS, SMMLR	Rd, Rn, Rm, Ra	Signed most significant word multiply subtract	–
SMMUL, SMMULR	{Rd,} Rn, Rm	Signed most significant word multiply	–
SMUAD SMUADX	{Rd,} Rn, Rm	Signed dual multiply add	Q
SMULBB, SMULBT, SMULTB, SMULTT	{Rd,} Rn, Rm	Signed multiply halfwords	–
SMULL	RdLo, RdHi, Rn, Rm	Signed multiply (32×32), 64-bit result	–
SMULWB, SMULWT	{Rd,} Rn, Rm	Signed multiply by halfword	–
SMUSD, SMUSDx	{Rd,} Rn, Rm	Signed dual multiply subtract	–
SSAT	Rd, #n, Rm {,shift #s}	Signed saturate	Q
SSAT16	Rd, #n, Rm	Signed saturate 16	Q
SSAX	{Rd,} Rn, Rm	Saturating subtract and add with exchange	GE
SSUB16	{Rd,} Rn, Rm	Signed subtract 16	–
SSUB8	{Rd,} Rn, Rm	Signed subtract 8	–
STM	Rn{!}, reglist	Store multiple registers, increment after	–
STMDB, STMEA	Rn{!}, reglist	Store multiple registers, decrement before	–
STMTD, STMIA	Rn{!}, reglist	Store multiple registers, increment after	–
STR	Rt, [Rn {, #offset}]	Store register word	–
STRB, STRBT	Rt, [Rn {, #offset}]	Store register byte	–
STRD	Rt, Rt2, [Rn {, #offset}]	Store register two words	–
STREX	Rt, Rt, [Rn {, #offset}]	Store register exclusive	–
STREXB	Rd, Rt, [Rn]	Store register exclusive byte	–
STREXH	Rd, Rt, [Rn]	Store register exclusive halfword	–
STRH, STRHT	Rt, [Rn {, #offset}]	Store register halfword	–
STRSB, STRSBT	Rt, [Rn {, #offset}]	Store register signed byte	–
STRSH, STRSHT	Rt, [Rn {, #offset}]	Store register signed halfword	–
STRT	Rt, [Rn {, #offset}]	Store register word	–
SUB, SUBS	{Rd,} Rn, Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd,} Rn, #imm12	Subtract 12-bit constant	N,Z,C,V
SVC	#imm	Supervisor call	–
SXTAB	{Rd,} Rn, Rm, {,ROR #}	Extend 8 bits to 32 and add	–
SXTAB16	{Rd,} Rn, Rm,{,ROR #}	Dual extend 8 bits to 16 and add	–
SXTAH	{Rd,} Rn, Rm,{,ROR #}	Extend 16 bits to 32 and add	–
SXTB16	{Rd,} Rm {,ROR #n}	Signed extend byte 16	–

Table 1-23. Cortex-M4F Instruction Summary (continued)

Mnemonic	Operands	Brief Description	Flags
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	–
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	–
TBB	[Rn, Rm]	Table branch byte	–
TBH	[Rn, Rm, LSL #1]	Table branch halfword	–
TEQ	Rn, Op2	Test equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C
UADD16	{Rd,} Rn, Rm	Unsigned add 16	GE
UADD8	{Rd,} Rn, Rm	Unsigned add 8	GE
UASX	{Rd,} Rn, Rm	Unsigned add and subtract with exchange	GE
UHADD16	{Rd,} Rn, Rm	Unsigned halving add 16	–
UHADD8	{Rd,} Rn, Rm	Unsigned halving add 8	–
UHASX	{Rd,} Rn, Rm	Unsigned halving add and subtract with exchange	–
UHSAX	{Rd,} Rn, Rm	Unsigned halving subtract and add with exchange	–
UHSUB16	{Rd,} Rn, Rm	Unsigned halving subtract 16	–
UHSUB8	{Rd,} Rn, Rm	Unsigned halving subtract 8	–
UBFX	Rd, Rn, #lsb, #width	Unsigned bit field extract	–
UDIV	{Rd,} Rn, Rm	Unsigned divide	–
UMAAL	RdLo, RdHi, Rn, Rm	Unsigned multiply accumulate accumulate long (32×32+64), 64-bit result	–
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned multiply with accumulate (32×32+32+32), 64-bit result	–
UMULL	RdLo, RdHi, Rn, Rm	Unsigned multiply (32×32), 64-bit result	–
UQADD16	{Rd,} Rn, Rm	Unsigned saturating add 16	–
UQADD8	{Rd,} Rn, Rm	Unsigned saturating add 8	–
UQASX	{Rd,} Rn, Rm	Unsigned saturating add and subtract with exchange	–
UQSAX	{Rd,} Rn, Rm	Unsigned saturating subtract and add with exchange	–
UQSUB16	{Rd,} Rn, Rm	Unsigned saturating subtract 16	–
UQSUB8	{Rd,} Rn, Rm	Unsigned saturating subtract 8	–
USAD8	{Rd,} Rn, Rm	Unsigned sum of absolute differences	–
USADA8	{Rd,} Rn, Rm, Ra	Unsigned sum of absolute differences and accumulate	–
USAT	Rd, #n, Rm {,shift #s}	Unsigned saturate	Q
USAT16	Rd, #n, Rm	Unsigned saturate 16	Q
USAX	{Rd,} Rn, Rm	Unsigned subtract and add with exchange	GE
USUB16	{Rd,} Rn, Rm	Unsigned subtract 16	GE
USUB8	{Rd,} Rn, Rm	Unsigned subtract 8	GE
UXTAB	{Rd,} Rn, Rm, {,ROR #}	Rotate, extend 8 bits to 32 and add	–
UXTAB16	{Rd,} Rn, Rm, {,ROR #}	Rotate, dual extend 8 bits to 16 and add	–
UXTAH	{Rd,} Rn, Rm, {,ROR #}	Rotate, unsigned extend and add halfword	–
UXTB	{Rd,} Rm, {,ROR #n}	Zero extend a Byte	–
UXTB16	{Rd,} Rm, {,ROR #n}	Unsigned extend byte 16	–
UXTH	{Rd,} Rm, {,ROR #n}	Zero extend a halfword	–
VABS.F32	Sd, Sm	Floating-point absolute	–
VADD.F32	{Sd,} Sn, Sm	Floating-point add	–
VCMP.F32	Sd, <Sm #0.0>	Compare two floating-point registers, or one floating-point register and zero	FPSCR

Table 1-23. Cortex-M4F Instruction Summary (continued)

Mnemonic	Operands	Brief Description	Flags
VCMPE.F32	Sd, <Sm #0.0>	Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check	FPSCR
VCVT.S32.F32	Sd, Sm	Convert between floating-point and integer	–
VCVT.S16.F32	Sd, Sd, #fbits	Convert between floating-point and fixed point	–
VCVTR.S32.F32	Sd, Sm	Convert between floating-point and integer with rounding	–
VCVT<B H>.F32.F16	Sd, Sm	Converts half-precision value to single-precision	–
VCVTT<B T>.F32.F16	Sd, Sm	Converts single-precision register to half-precision	–
VDIV.F32	{Sd,} Sn, Sm	Floating-point divide	–
VFMA.F32	{Sd,} Sn, Sm	Floating-point fused multiply accumulate	–
VFNMA.F32	{Sd,} Sn, Sm	Floating-point fused negate multiply accumulate	–
VFMS.F32	{Sd,} Sn, Sm	Floating-point fused multiply subtract	–
VFNMS.F32	{Sd,} Sn, Sm	Floating-point fused negate multiply subtract	–
VLDM.F<32 64>	Rn(!), list	Load multiple extension registers	–
VLDR.F<32 64>	<Dd Sd>, [Rn]	Load an extension register from memory	–
VLMA.F32	{Sd,} Sn, Sm	Floating-point multiply accumulate	–
VLMS.F32	{Sd,} Sn, Sm	Floating-point multiply subtract	–
VMOV.F32	Sd, #imm	Floating-point move immediate	–
VMOV	Sd, Sm	Floating-point move register	–
VMOV	Sn, Rt	Copy Arm core register to single precision	–
VMOV	Sm, Sm1, Rt, Rt2	Copy two Arm core registers to two single precision	–
VMOV	Dd[x], Rt	Copy Arm core register to scalar	–
VMOV	Rt, Dn[x]	Copy scalar to Arm core register	–
VMRS	Rt, FPSCR	Move FPSCR to Arm core register or APSR	N,Z,C,V
VMSR	FPSCR, Rt	Move to FPSCR from Arm Core register	FPSCR
VMUL.F32	{Sd,} Sn, Sm	Floating-point multiply	–
VNEG.F32	Sd, Sm	Floating-point negate	–
VNMLA.F32	{Sd,} Sn, Sm	Floating-point multiply and add	–
VNMLS.F32	{Sd,} Sn, Sm	Floating-point multiply and subtract	–
VNMUL	{Sd,} Sn, Sm	Floating-point multiply	–
VPOP	list	Pop extension registers	–
VPUSH	list	Push extension registers	–
VSQRT.F32	Sd, Sm	Calculates floating-point square root	–
VSTM	Rn(!), list	Floating-point register store multiple	–
VSTR.F3<32 64>	Sd, [Rn]	Stores an extension register to memory	–
VSUB.F<32 64>	{Sd,} Sn, Sm	Floating-point subtract	–
WFE	–	Wait for event	–
WFI	–	Wait for interrupt	–

Cortex-M4 Peripherals

This chapter describes the MSP432E4 implementation of the Cortex-M4 processor peripherals.

Topic	Page
2.1 Introduction	122
2.2 Functional Description	122
2.3 SysTick Registers.....	133
2.4 NVIC Registers	137
2.5 SCB Registers.....	147
2.6 MPU Registers	168
2.7 FPU Registers	177

2.1 Introduction

This chapter describes the following peripherals:

- SysTick [Section 2.2.1](#)
Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- Nested Vectored Interrupt Controller (NVIC) [Section 2.2.2](#)
 - Facilitates low-latency exception and interrupt handling
 - Controls power management
 - Implements system control registers
- System Control Block (SCB) [Section 2.2.3](#)
Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
- Memory Protection Unit (MPU) [Section 2.2.4](#)
Supports the standard Armv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.
- Floating-Point Unit (FPU) [Section 2.2.5](#)
Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

Table 2-1 shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

Table 2-1. Core Peripheral Register Regions

Address	Core Peripheral	Description
0xE000.E010 to 0xE000.E01F	System Timer	Section 2.2.1
0xE000.E100 to 0xE000.E4EF 0xE000.EF00 to 0xE000.EF03	Nested Vectored Interrupt Controller	Section 2.2.2
0xE000.E008-0xE000.E00F 0xE000.ED00 to 0xE000.ED3F	System Control Block	Section 2.2.3
0xE000.ED90 to 0xE000.EDB8	Memory Protection Unit	Section 2.2.4
0xE000.EF30 to 0xE000.EF44	Floating Point Unit	Section 2.2.5

2.2 Functional Description

This chapter provides information on the MSP432E4 implementation of the Cortex-M4 processor peripherals: SysTick, NVIC, SCB, MPU, FPU.

2.2.1 System Timer (SysTick)

Cortex-M4 includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example as:

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the system clock.
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter used to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNT bit in the STCTRL control and status register can be used to determine if an action completed within a set duration, as

part of a dynamic clock management control loop.

The timer consists of three registers:

- SysTick Control and Status (STCTRL) : A control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- SysTick Reload Value (STRELOAD) : The reload value for the counter, used to provide the counter's wrap value.
- SysTick Current Value (STCURRENT) : The current value of the counter.

When enabled, the timer counts down on each clock from the reload value to zero, reloads (wraps) to the value in the STRELOAD register on the next clock edge, then decrements on subsequent clocks. Clearing the STRELOAD register disables the counter on the next wrap. When the counter reaches zero, the COUNT status bit is set. The COUNT bit clears on reads.

Writing to the STCURRENT register clears the register and the COUNT status bit. The write does not trigger the SysTick exception logic. On a read, the current value is the value of the register at the time the register is accessed.

The SysTick counter runs on the system clock. If this clock signal is stopped for low power mode, the SysTick counter stops. Ensure software uses aligned word accesses to access the SysTick registers.

The SysTick counter reload and current value are undefined at reset; the correct initialization sequence for the SysTick counter is:

1. Program the value in the STRELOAD register.
2. Clear the STCURRENT register by writing to it with any value.
3. Configure the STCTRL register for the required operation.

NOTE: When the processor is halted for debugging, the counter does not decrement.

2.2.2 Nested Vectored Interrupt Controller (NVIC)

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports:

- 109 interrupts
- A programmable priority level of 0 to 7 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Low-latency exception and interrupt handling
- Level and pulse detection of interrupt signals
- Dynamic reprioritization of interrupts
- Grouping of priority values into group priority and subpriority fields
- Interrupt tail-chaining
- An external nonmaskable interrupt (NMI)

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead, providing low latency exception handling.

2.2.2.1 Level-Sensitive and Pulse Interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt. For more information, see [Section 2.2.2.2](#). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

2.2.2.2 Hardware and Software Control of Interrupts

The Cortex-M4 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- The NVIC detects that the interrupt signal is high and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the Software Trigger Interrupt (SWTRIG) register to make a Software-Generated Interrupt pending. For more information, see the INT bit in the PEND0 register on [Section 2.4.3](#) or SWTRIG on [Section 2.4.7](#).

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt, changing the state of the interrupt from pending to active. Then:
 - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
 - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit.
 - For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.
 - For a pulse interrupt, the state of the interrupt changes to inactive, if the state was pending or to active, if the state was active and pending.

2.2.3 System Control Block (SCB)

The System Control Block (SCB) provides system implementation information and system control, including configuration, control, and reporting of the system exceptions.

2.2.4 Memory Protection Unit (MPU)

This section describes the Memory protection unit (MPU). The MPU divides the memory map into a number of regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M4 MPU defines eight separate memory regions, 0 to 7, and a background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M4 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault, causing a fault exception and possibly causing termination of the process in an OS environment. In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types. For more information, see [Section 1.5.1](#).

[Table 2-2](#) shows the possible MPU region attributes. For programming a microcontroller implementation, see [Section 2.2.4.2.1](#) for guidelines.

Table 2-2. Memory Attributes Summary

Memory Type	Description
Strongly Ordered	All accesses to strongly ordered memory occur in program order.
Device	Memory-mapped peripherals
Normal	Normal memory

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- Except for the MPU Region Attribute and Size (MPUATTR) register, all MPU registers must be accessed with aligned word accesses.
- The MPUATTR register can be accessed with byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

2.2.4.1 Updating an MPU Region

To update the attributes for an MPU region, the MPU Region Number (MPUNUMBER), MPU Region Base Address (MPUBASE) and MPUATTR registers must be updated. Each register can be programmed separately or with a multiple-word write to program all of these registers. You can use the MPUBASEx and MPUATTRx aliases to program up to four regions simultaneously using an STM instruction.

2.2.4.1.1 Updating an MPU Region Using Separate Words

This example simple code configures one region:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER      ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]      ; Region Number
STR R4, [R0, #0x4]      ; Region Base Address
STRH R2, [R0, #0x8]     ; Region Size and Enable
STRH R3, [R0, #0xA]     ; Region Attribute
```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER      ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]      ; Region Number
BIC R2, R2, #1          ; Disable
STRH R2, [R0, #0x8]     ; Region Size and Enable
STR R4, [R0, #0x4]      ; Region Base Address
STRH R3, [R0, #0xA]     ; Region Attribute
ORR R2, #1              ; Enable
```

```
STRH R2, [R0, #0x8] ; Region Size and Enable
```

Software must use memory barrier instructions:

- Before MPU setup, if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings.
- After MPU setup, if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not need any memory barrier instructions during MPU setup, because it accesses the MPU through the Private Peripheral Bus (PPB), which is a Strongly Ordered memory region.

For example, if all of the memory access behavior is intended to take effect immediately after the programming sequence, then a DSB instruction and an ISB instruction should be used. A DSB is required after changing MPU settings, such as at the end of context switch. An ISB is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then an ISB is not required.

2.2.4.1.2 Updating an MPU Region Using Multi-Word Writes

The MPU can be programmed directly using multi-word writes, depending how the information is divided. Consider the following reprogramming:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0] ; Region Number
STR R2, [R0, #0x4] ; Region Base Address
STR R3, [R0, #0x8] ; Region Attribute, Size and Enable
```

An STM instruction can be used to optimize this:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER ; 0xE000ED98, MPU region number register
STM R0, {R1-R3} ; Region number, address, attribute, size and enable
```

This operation can be done in two words for prepacked information, meaning that the MPU Region Base Address (MPUBASE) register (see [Section 2.6.4](#)) contains the required region number and has the VALID bit set. This method can be used when the data is statically packed, for example in a boot loader:

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPUBASE ; 0xE000ED9C, MPU Region Base register
STR R1, [R0, #0x0] ; Region base address and region number combined
; with VALID (bit 4) set
STR R2, [R0, #0x4] ; Region Attribute, Size and Enable
```

2.2.4.1.3 Subregions

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the `SRD` field of the MPU Region Attribute and Size (MPUATTR) register (see [Section 2.6.5](#)) to disable a subregion. The least-significant bit of the `SRD` field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the `SRD` field must be configured to 0x00, otherwise the MPU behavior is unpredictable.

2.2.4.1.3.1 Example of SRD Use

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as [Figure 2-1](#) shows.

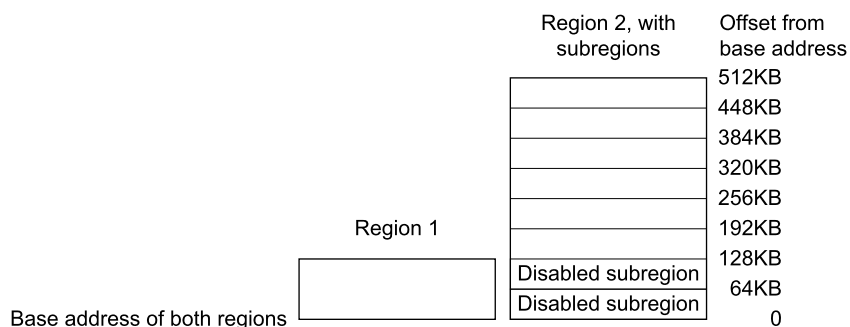


Figure 2-1. SRD Use Example

2.2.4.2 MPU Access Permission Attributes

The access permission bits, TEX, S, C, B, AP, and XN of the MPUATTR register, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

[Table 2-3](#) shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M4 does not support the concept of cacheability or shareability. For information on programming the MPU for MSP432E4 implementations, see [Section 2.2.4.2.1](#).

Table 2-3. TEX, S, C, and B Bit Field Encoding

TEX	S	C	B	Memory Type	Shareability	Other Attributes
000b	x ⁽¹⁾	0	0	Strongly Ordered	Shareable	–
000	x ⁽¹⁾	0	1	Device	Shareable	–
000	0	1	0	Normal	Not shareable	Outer and inner write-through. No write allocate.
000	1	1	0	Normal	Shareable	
000	0	1	1	Normal	Not shareable	
000	1	1	1	Normal	Shareable	
001	0	0	0	Normal	Not shareable	Outer and inner noncacheable.
001	1	0	0	Normal	Shareable	
001	x ⁽¹⁾	0	1	Reserved encoding	–	–
001	x ⁽¹⁾	1	0	Reserved encoding	–	–
001	0	1	1	Normal	Not shareable	Outer and inner write-back. Write and read allocate.
001	1	1	1	Normal	Shareable	
010	x ⁽¹⁾	0	0	Device	Not shareable	Nonshared Device.
010	x ⁽¹⁾	0	1	Reserved encoding	–	–
010	x ⁽¹⁾	1	x ⁽¹⁾	Reserved encoding	–	–
1BB	0	A	A	Normal	Not shareable	Cached memory (BB = outer policy, AA = inner policy). See Table 2-4 for the encoding of the AA and BB bits.
1BB	1	A	A	Normal	Shareable	

⁽¹⁾ The MPU ignores the value of this bit.

Table 2-4 shows the cache policy for memory attribute encodings with a TEX value in the range of 0x4 to 0x7.

Table 2-4. Cache Policy for Memory Attribute Encoding

Encoding, AA or BB	Corresponding Cache Policy
00	Noncacheable
01	Write back, write and read allocate
10	Write through, no write allocate
11	Write back, no write allocate

Table 2-5 shows the AP encodings in the MPUATTR register that define the access permissions for privileged and unprivileged software.

Table 2-5. AP Bit Field Encoding

AP Bit Field	Privileged Permissions	Unprivileged Permissions	Description
000	No access	No access	All accesses generate a permission fault.
001	RW	No access	Access from privileged software only.
010	RW	RO	Writes by unprivileged software generate a permission fault.
011	RW	RW	Full access.
100	Unpredictable	Unpredictable	Reserved.
101	RO	No access	Reads by privileged software only.
110	RO	RO	Read-only, by privileged or unprivileged software.
111	RO	RO	Read-only, by privileged or unprivileged software.

2.2.4.2.1 MPU Configuration for a MSP432E4 Microcontroller

MSP432E4 microcontrollers have only a single processor and no caches. As a result, the MPU should be programmed as shown in Table 2-6.

Table 2-6. Memory Region Attributes for MSP432E4 Microcontrollers

Memory Region	TEX	S	C	B	Memory Type and Attributes
Flash memory	000b	0	1	0	Normal memory, nonshareable, write-through
Internal SRAM	000b	1	1	0	Normal memory, shareable, write-through
External SRAM	000b	1	1	1	Normal memory, shareable, write-back, write-allocate
Peripherals	000b	1	0	1	Device memory, shareable

In current MSP432E4 microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations.

2.2.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault. For more information, see Section 1.4.3. The MFAULTSTAT register indicates the cause of the fault. For more information, see Section 2.5.12.

2.2.5 Floating-Point Unit (FPU)

This section describes the Floating-Point Unit (FPU) and the registers it uses. The FPU provides:

- 32-bit instructions for single-precision (C float) data-processing operations
- Combined multiply and accumulate instructions for increased precision (Fused MAC)

- Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
- Hardware support for denormals and all IEEE rounding modes
- 32 dedicated 32-bit single-precision registers, also addressable as 16 double-word registers
- Decoupled three stage pipeline

The Cortex-M4F FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions. The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard. The FPU's single-precision extension registers can also be accessed as 16 doubleword registers for load, store, and move operations.

2.2.5.1 FPU Views of the Register Bank

The FPU provides an extension register file containing 32 single-precision registers. These can be viewed as:

- Sixteen 64-bit doubleword registers, D0 to D15
- Thirty-two 32-bit single-word registers, S0 to S31
- A combination of registers from the above views

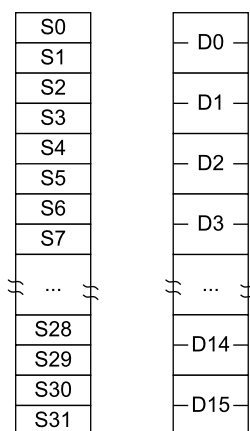


Figure 2-2. FPU Register Bank

The mapping between the registers is as follows:

- S_{2n} maps to the least significant half of $D_{n/2}$
- S_{2n+1} maps to the most significant half of $D_{n/2}$

For example, you can access the least significant half of the value in D6 by accessing S12, and the most significant half of the elements by accessing S13.

2.2.5.2 Modes of Operation

The FPU provides three modes of operation to accommodate a variety of applications.

Full-Compliance mode

In Full-Compliance mode, the FPU processes all operations according to the IEEE 754 standard in hardware.

Flush-to-Zero mode

Setting the FZ bit of the Floating-Point Status and Control (FPSC) register enables Flush-to-Zero mode. In this mode, the FPU treats all subnormal input operands of arithmetic CDP operations as zeros in the operation. Exceptions that result from a zero operand are signalled appropriately. VABS, VNEG, and VMOV are not considered arithmetic CDP operations and are not affected by Flush-to-Zero mode. A result that is tiny, as described in the IEEE 754 standard, where the destination precision is smaller in magnitude than the minimum normal value before rounding, is replaced with a zero. The IDC bit in FPSC indicates when an input flush occurs. The UFC bit in FPSC indicates when a result flush occurs.

Default NaN mode

Setting the DN bit in the FPSC register enables default NaN mode. In this mode, the result of any arithmetic data processing operation that involves an input NaN, or that generates a NaN result, returns the default NaN. Propagation of the fraction bits is maintained only by VABS, VNEG, and VMOV operations. All other CDP operations ignore any information in the fraction bits of an input NaN.

2.2.5.3 Compliance With the IEEE 754 Standard

When Default NaN (DN) and Flush-to-Zero (FZ) modes are disabled, FPv4 functionality is compliant with the IEEE 754 standard in hardware. No support code is required to achieve this compliance.

2.2.5.4 Complete Implementation of the IEEE 754 Standard

The Cortex-M4F floating point instruction set does not support all operations defined in the IEEE 754-2008 standard. Unsupported operations include, but are not limited to the following:

- Remainder
- Round floating-point number to integer-valued floating-point number
- Binary-to-decimal conversions
- Decimal-to-binary conversions
- Direct comparison of single-precision and double-precision values

The Cortex-M4FPU supports fused MAC operations as described in the IEEE standard. For complete implementation of the IEEE 754-2008 standard, floating-point functionality must be augmented with library functions.

2.2.5.5 IEEE 754 Standard Implementation Choices

2.2.5.5.1 NaN Handling

All single-precision values with the maximum exponent field value and a nonzero fraction field are valid NaNs. A most-significant fraction bit of zero indicates a Signaling NaN (SNaN). A one indicates a Quiet NaN (QNaN). Two NaN values are treated as different NaNs if they differ in any bit. The table below shows the default NaN values.

Table 2-7. NaN Handling

Sign	Fraction	Fraction
0	0xFF	bit [22] = 1, bits [21:0] are all zeros

Processing of input NaNs for Arm floating-point functionality and libraries is defined as follows:

- In full-compliance mode, NaNs are handled as described in the Arm Architecture Reference Manual. The hardware processes the NaNs directly for arithmetic CDP instructions. For data transfer operations, NaNs are transferred without raising the Invalid Operation exception. For the nonarithmetic CDP instructions, VABS, VNEG, and VMOV, NaNs are copied, with a change of sign if specified in the instructions, without causing the Invalid Operation exception.
- In default NaN mode, arithmetic CDP instructions involving NaN operands return the default NaN regardless of the fractions of any NaN operands. SNaNs in an arithmetic CDP operation set the IOC flag, FPSCR[0]. NaN handling by data transfer and nonarithmetic CDP instructions is the same as in full-compliance mode.

Table 2-8. QNaN and SNaN Handling

Instruction Type	Default NaN Mode	With QNaN Operand	With SNaN Operand
Arithmetic CDP	Off	The QNaN or one of the QNaN operands, if there is more than one, is returned according to the rules given in the	IOC ⁽¹⁾ set. The SNaN is quieted and the result NaN is determined by the rules given in the Arm Architecture Reference Manual.
	On	Default NaN returns.	IOC ⁽¹⁾ set. Default NaN returns.
Non-arithmetic CDP	Off/On	NaN passes to destination with sign changed as appropriate.	
FCMP(Z)	–	Unordered compare.	IOC set. Unordered compare.
FCMPE(Z)	–	IOC set. Unordered compare.	IOC set. Unordered compare.
All NaNs transferred	Off/On	All NaNs transferred	

⁽¹⁾ IOC is the Invalid Operation exception flag, FPSCR[0].

2.2.5.5.2 Comparisons

Comparison results modify the flags in the FPSCR. You can use the MVRSPSR_nzcv instruction (formerly FMSTAT) to transfer the current flags from the FPSCR to the APSR. For mapping of IEEE 754-2008 standard predicates to Arm conditions, see the Arm Architecture Reference Manual. The flags used are chosen so that subsequent conditional execution of Arm instructions can test the predicates defined in the IEEE standard.

2.2.5.5.3 Underflow

The Cortex-M4F FPU uses the before rounding form of tininess and the inexact result form of loss of accuracy as described in the IEEE 754-2008 standard to generate Underflow exceptions.

In flush-to-zero mode, results that are tiny before rounding, as described in the IEEE standard, are flushed to a zero, and the UFC flag, FPSCR[3], is set. For information on flush-to-zero mode, see the Arm Architecture Reference Manual.

When the FPU is not in flush-to-zero mode, operations are performed on subnormal operands. If the operation does not produce a tiny result, it returns the computed result, and the UFC flag, FPSCR[3], is not set. The IXC flag, FPSCR[4], is set if the operation is inexact. If the operation produces a tiny result, the result is a subnormal or zero value, and the UFC flag, FPSCR[3], is set if the result was also inexact.

2.2.5.6 Exceptions

The FPU sets the cumulative exception status flag in the FPSCR register as required for each instruction, in accordance with the FPv4 architecture. The FPU does not support user-mode traps. The exception enable bits in the FPSCR read-as-zero, and writes are ignored. The processor also has six output pins, FPIX, FPUFC, FPOFC, FPDZC, FPIDC, and FPIOC, that each reflect the status of one of the cumulative exception flags. For a description of these outputs, see the *Arm Cortex-M4 Integration and Implementation Manual*.

The processor can reduce the exception latency by using lazy stacking. For more information, see Auxiliary Control Register (ACTLR). This means that the processor reserves space on the stack for the FP state, but does not save that state information to the stack. For more information, see the *Armv7-M Architecture Reference Manual*.

2.2.5.7 Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions. The processor must be in privileged mode to read from and write to the Coprocessor Access Control (CPAC) register. The below example code sequence enables the FPU in both privileged and user modes.

```
; CPACR is located at address 0xE000ED88
LDR.W R0, =0xE000ED88
; Read CPACR
LDR R1, [R0]
; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR R1, R1, #(0xF << 20)
; Write back the modified value to the CPACR
STR R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
```

2.3 SysTick Registers

lists the Cortex-M4 Peripheral SysTick, NVIC, MPU, FPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000E000 (ending address of 0xE000EFFF).

NOTE: Register spaces that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

[Table 2-9](#) lists the memory-mapped registers for the SYSTICK. All register offset addresses not listed in [Table 2-9](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-9. SYSTICK Registers

Offset	Acronym	Register Name	Section
0x10	STCTRL	SysTick Control and Status Register	Section 2.3.1
0x14	STRELOAD	SysTick Reload Value Register	Section 2.3.2
0x18	STCURRENT	SysTick Current Value Register	Section 2.3.3

Complex bit access types are encoded to fit into small table cells. [Table 2-10](#) shows the codes that are used for access types in this section.

Table 2-10. SYSTICK Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
WC	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.3.1 STCTRL Register (Offset = 0x10) [reset = 0x0]

SysTick Control and Status Register (STCTRL)

NOTE: This register can be accessed only from privileged mode.

The SysTick STCTRL register enables the SysTick features.

STCTRL is shown in [Figure 2-3](#) and described in [Table 2-11](#).

Return to [Summary Table](#).

Figure 2-3. STCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							COUNT
R-0x0							R-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					CLK_SRC	INTEN	ENABLE
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 2-11. STCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	COUNT	R	0x0	Count Flag This bit is cleared by a read of the register or if the STCURRENT register is written with any value. If read by the debugger using the DAP, this bit is cleared only if the MasterType bit in the AHB-AP Control Register is clear. Otherwise, the COUNT bit is not changed by the debugger read. See the Arm Debug Interface V5 Architecture Specification for more information on MasterType. 0x0 = The SysTick timer has not counted to 0 since the last time this bit was read. 0x1 = The SysTick timer has counted to 0 since the last time this bit was read.
15-3	RESERVED	R	0x0	
2	CLK_SRC	R/W	0x0	Clock Source 0x0 = Precision internal oscillator (PIOSC) divided by 4 1 = System clock
1	INTEN	R/W	0x0	Interrupt Enable 0 = Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0. 1 = An interrupt is generated to the NVIC when SysTick counts to 0.
0	ENABLE	R/W	0x0	Enable 0 = The counter is disabled. 1 = Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.

2.3.2 STRELOAD Register (Offset = 0x14) [reset = 0x0]

SysTick Reload Value Register (STRELOAD)

NOTE: This register can only be accessed from privileged mode.

The STRELOAD register specifies the start value to load into the SysTick Current Value (STCURRENT) register when the counter reaches 0. The start value can be between 0x1 and 0x00FFFFFF. A start value of 0 is possible but has no effect because the SysTick interrupt and the COUNT bit are activated when counting from 1 to 0.

SysTick can be configured as a multi-shot timer, repeated over and over, firing every N+1 clock pulses, where N is any value from 1 to 0x00FFFFFF. For example, if a tick interrupt is required every 100 clock pulses, 99 must be written into the RELOAD field.

Note that in order to access this register correctly, the system clock must be faster than 8 MHz.

STRELOAD is shown in [Figure 2-4](#) and described in [Table 2-12](#).

Return to [Summary Table](#).

Figure 2-4. STRELOAD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RELOAD																							
R-0x0								R/W-0x0																							

Table 2-12. STRELOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23-0	RELOAD	R/W	0x0	Reload Value Value to load into the SysTick Current Value (STCURRENT) register when the counter reaches 0.

2.3.3 STCURRENT Register (Offset = 0x18) [reset = 0x0]

SysTick Current Value Register (STCURRENT)

NOTE: This register can only be accessed from privileged mode.

The STCURRENT register contains the current value of the SysTick counter.

STCURRENT is shown in [Figure 2-5](#) and described in [Table 2-13](#).

Return to [Summary Table](#).

Figure 2-5. STCURRENT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CURRENT																							
R-0x0								R/WC-0x0																							

Table 2-13. STCURRENT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23-0	CURRENT	R/WC	0x0	Current Value This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care. This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the COUNT bit of the STCTRL register.

2.4 NVIC Registers

The NVIC registers can only be fully accessed from privileged mode, but interrupts can be pended while in unprivileged mode by enabling the Configuration and Control (CFGCTRL) register. Any other unprivileged mode access causes a bus fault.

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers.

An interrupt can enter the pending state even if it is disabled.

Before programming the VTABLE register to relocate the vector table, ensure the vector table entries of the new vector table are set up for fault handlers, NMI, and all enabled exceptions such as interrupts. For more information, see [Section 2.5.4](#).

[Table 2-14](#) lists the memory-mapped registers for the NVIC. All register offset addresses not listed in [Table 2-14](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-14. NVIC Registers

Offset	Acronym	Register Name	Section
0x100	EN0	Interrupt 0-31 Set Enable	Section 2.4.1
0x104	EN1	Interrupt 32-63 Set Enable	Section 2.4.1
0x108	EN2	Interrupt 64-95 Set Enable	Section 2.4.1
0x10C	EN3	Interrupt 96-113 Set Enable	Section 2.4.1
0x180	DIS0	Interrupt 0-31 Clear Enable	Section 2.4.2
0x184	DIS1	Interrupt 32-63 Clear Enable	Section 2.4.2
0x188	DIS2	Interrupt 64-95 Clear Enable	Section 2.4.2
0x18C	DIS3	Interrupt 96-113 Clear Enable	Section 2.4.2
0x200	PEND0	Interrupt 0-31 Set Pending	Section 2.4.3
0x204	PEND1	Interrupt 32-63 Set Pending	Section 2.4.3
0x208	PEND2	Interrupt 64-95 Set Pending	Section 2.4.3
0x20C	PEND3	Interrupt 96-113 Set Pending	Section 2.4.3
0x280	UNPEND0	Interrupt 0-31 Clear Pending	Section 2.4.4
0x284	UNPEND1	Interrupt 32-63 Clear Pending	Section 2.4.4
0x288	UNPEND2	Interrupt 64-95 Clear Pending	Section 2.4.4
0x28C	UNPEND3	Interrupt 96-113 Clear Pending	Section 2.4.4
0x300	ACTIVE0	Interrupt 0-31 Active Bit	Section 2.4.5
0x304	ACTIVE1	Interrupt 32-63 Active Bit	Section 2.4.5
0x308	ACTIVE2	Interrupt 64-95 Active Bit	Section 2.4.5
0x30C	ACTIVE3	Interrupt 96-127 Active Bit	Section 2.4.5
0x400	PRI0	Interrupt 0-3 Priority	Section 2.4.6
0x404	PRI1	Interrupt 4-7 Priority	Section 2.4.6
0x408	PRI2	Interrupt 8-11 Priority	Section 2.4.6
0x40C	PRI3	Interrupt 12-15 Priority	Section 2.4.6
0x410	PRI4	Interrupt 16-19 Priority	Section 2.4.6
0x414	PRI5	Interrupt 20-23 Priority	Section 2.4.6
0x418	PRI6	Interrupt 24-27 Priority	Section 2.4.6
0x41C	PRI7	Interrupt 28-31 Priority	Section 2.4.6
0x420	PRI8	Interrupt 32-35 Priority	Section 2.4.6
0x424	PRI9	Interrupt 36-39 Priority	Section 2.4.6
0x428	PRI10	Interrupt 40-43 Priority	Section 2.4.6
0x42C	PRI11	Interrupt 44-47 Priority	Section 2.4.6
0x430	PRI12	Interrupt 48-51 Priority	Section 2.4.6
0x434	PRI13	Interrupt 52-55 Priority	Section 2.4.6

Table 2-14. NVIC Registers (continued)

Offset	Acronym	Register Name	Section
0x438	PRI14	Interrupt 56-59 Priority	Section 2.4.6
0x43C	PRI15	Interrupt 60-63 Priority	Section 2.4.6
0x440	PRI16	Interrupt 64-67 Priority	Section 2.4.6
0x444	PRI17	Interrupt 68-71 Priority	Section 2.4.6
0x448	PRI18	Interrupt 72-75 Priority	Section 2.4.6
0x44C	PRI19	Interrupt 76-79 Priority	Section 2.4.6
0x450	PRI20	Interrupt 80-83 Priority	Section 2.4.6
0x454	PRI21	Interrupt 84-87 Priority	Section 2.4.6
0x458	PRI22	Interrupt 88-91 Priority	Section 2.4.6
0x45C	PRI23	Interrupt 92-95 Priority	Section 2.4.6
0x460	PRI24	Interrupt 96-99 Priority	Section 2.4.6
0x464	PRI25	Interrupt 100-103 Priority	Section 2.4.6
0x468	PRI26	Interrupt 104-107 Priority	Section 2.4.6
0x46C	PRI27	Interrupt 108-111 Priority	Section 2.4.6
0x470	PRI28	Interrupt 112-113 Priority	Section 2.4.6
0xF00	SWTRIG	Software Trigger Interrupt	Section 2.4.7

Complex bit access types are encoded to fit into small table cells. [Table 2-15](#) shows the codes that are used for access types in this section.

Table 2-15. NVIC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
WO	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.4.1 EN0 to EN3 Registers

Interrupt 0-31 Set Enable (EN0), offset 0x100

Interrupt 32-63 Set Enable (EN1), offset 0x104

Interrupt 64-95 Set Enable (EN2), offset 0x108

Interrupt 96-113 Set Enable (EN3), offset 0x10C

NOTE: This register can only be accessed from privileged mode.

The ENn registers enable interrupts and show which interrupts are enabled. Bit 0 of EN0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of EN1 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of EN2 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of EN3 corresponds to Interrupt 96; bit 17 corresponds to Interrupt 113.

See for interrupt assignments.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

EN0 is shown in [Figure 2-6](#) and described in [Table 2-16](#).

Return to [Summary Table](#).

Figure 2-6. EN0 to EN3 Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT																															
R/W-0x0																															

Table 2-16. EN0 to EN3 Registers Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INT	R/W	0x0	<p>Interrupt Enable A bit can only be cleared by setting the corresponding INT[n] bit in the DISn register.</p> <p>0x0 = On a read, indicates the interrupt is disabled. On a write, no effect.</p> <p>0x1 = On a read, indicates the interrupt is enabled. On a write, enables the interrupt.</p>

2.4.2 DIS0 to DIS3 Registers

Interrupt 0-31 Clear Enable (DIS0), offset 0x180

Interrupt 32-63 Clear Enable (DIS1), offset 0x184

Interrupt 64-95 Clear Enable (DIS2), offset 0x188

Interrupt 96- 113 Clear Enable (DIS3), offset 0x18C

NOTE: This register can only be accessed from privileged mode.

The DISn registers disable interrupts. Bit 0 of DIS0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of DIS1 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of DIS2 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of DIS3 corresponds to Interrupt 96.

See for interrupt assignments.

DISn is shown in [Figure 2-7](#) and described in [Table 2-17](#).

Return to [Summary Table](#).

Figure 2-7. DISn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT																															
R/W-0x0																															

Table 2-17. DISn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INT	R/W	0x0	<p>Interrupt Disable</p> <p>0x0 = On a read, indicates the interrupt is disabled.On a write, no effect.</p> <p>0x1 = On a read, indicates the interrupt is enabled.On a write, clears the corresponding INT[n] bit in the EN0 register, disabling interrupt [n].</p>

2.4.3 PEND0 to PEND 3 Registers

Interrupt 0-31 Set Pending (PEND0), offset 0x200

Interrupt 32-63 Set Pending (PEND1), offset 0x204

Interrupt 64-95 Set Pending (PEND2), offset 0x208

Interrupt 96-113 Set Pending (PEND3), offset 0x20C

NOTE: This register can only be accessed from privileged mode.

The PENDn registers force interrupts into the pending state and show which interrupts are pending. Bit 0 of PEND0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of PEND1 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of PEND2 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of PEND3 corresponds to Interrupt 96; bit 17 corresponds to interrupt 113.

See for interrupt assignments.

PENDn is shown in [Figure 2-8](#) and described in [Table 2-18](#).

Return to [Summary Table](#).

Figure 2-8. PENDn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT																															
R/W-0x0																															

Table 2-18. PENDn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INT	R/W	0x0	Interrupt Set Pending If the corresponding interrupt is already pending, setting a bit has no effect. A bit can only be cleared by setting the corresponding INT[n] bit in the UNPEND0 register. 0x0 = On a read, indicates that the interrupt is not pending.On a write, no effect. 0x1 = On a read, indicates that the interrupt is pending.On a write, the corresponding interrupt is set to pending even if it is disabled.

2.4.4 UNPEND0 to UNPEND3 Registers

Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280

Interrupt 32-63 Clear Pending (UNPEND1), offset 0x284

Interrupt 64-95 Clear Pending (UNPEND2), offset 0x288

Interrupt 96- 113 Clear Pending (UNPEND3), offset 0x28C

NOTE: This register can only be accessed from privileged mode.

The UNPENDn registers show which interrupts are pending and remove the pending state from interrupts. Bit 0 of UNPEND0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of UNPEND1 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of UNPEND2 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of UNPEND3 corresponds to Interrupt 96; bit 31 corresponds to Interrupt 113.

See for interrupt assignments.

UNPENDn is shown in [Figure 2-9](#) and described in [Table 2-19](#).

Return to [Summary Table](#).

Figure 2-9. UNPENDn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT																															
R/W-0x0																															

Table 2-19. UNPENDn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INT	R/W	0x0	<p>Interrupt Clear Pending</p> <p>0x0 = On a read, indicates that the interrupt is not pending. On a write, no effect.</p> <p>0x1 = On a read, indicates that the interrupt is pending. On a write, clears the corresponding INT[n] bit in the PEND0 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.</p>

2.4.5 ACTIVE0 to ACTIVE3 Registers

Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300

Interrupt 32-63 Active Bit (ACTIVE1), offset 0x304

Interrupt 64-95 Active Bit (ACTIVE2), offset 0x308

Interrupt 96-127 Active Bit (ACTIVE3), offset 0x30C

NOTE: This register can only be accessed from privileged mode.

The ACTIVE_n registers indicate which interrupts are active. Bit 0 of ACTIVE0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of ACTIVE1 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of ACTIVE2 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of ACTIVE3 corresponds to Interrupt 96; bit 17 corresponds to Interrupt 113.

See for interrupt assignments.

NOTE: Do not manually set or clear the bits in this register.

ACTIVE_n is shown in [Figure 2-10](#) and described in [Table 2-20](#).

Return to [Summary Table](#).

Figure 2-10. ACTIVE_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT																															
R-0x0																															

Table 2-20. ACTIVE_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INT	R	0x0	Interrupt Active 0x0 = The corresponding interrupt is not active. 0x1 = The corresponding interrupt is active, or active and pending.

2.4.6 PRI0 to PRI28 Registers

Interrupt 0-3 Priority (PRI0), offset 0x400
 Interrupt 4-7 Priority (PRI1), offset 0x404
 Interrupt 8-11 Priority (PRI2), offset 0x408
 Interrupt 12-15 Priority (PRI3), offset 0x40C
 Interrupt 16-19 Priority (PRI4), offset 0x410
 Interrupt 20-23 Priority (PRI5), offset 0x414
 Interrupt 24-27 Priority (PRI6), offset 0x418
 Interrupt 28-31 Priority (PRI7), offset 0x41C
 Interrupt 32-35 Priority (PRI8), offset 0x420
 Interrupt 36-39 Priority (PRI9), offset 0x424
 Interrupt 40-43 Priority (PRI10), offset 0x428
 Interrupt 44-47 Priority (PRI11), offset 0x42C
 Interrupt 48-51 Priority (PRI12), offset 0x430
 Interrupt 52-55 Priority (PRI13), offset 0x434
 Interrupt 56-59 Priority (PRI14), offset 0x438
 Interrupt 60-63 Priority (PRI15), offset 0x43C
 Interrupt 64-67 Priority (PRI16), offset 0x440
 Interrupt 68-71 Priority (PRI17), offset 0x444
 Interrupt 72-75 Priority (PRI18), offset 0x448
 Interrupt 76-79 Priority (PRI19), offset 0x44C
 Interrupt 80-83 Priority (PRI20), offset 0x450
 Interrupt 84-87 Priority (PRI21), offset 0x454
 Interrupt 88-91 Priority (PRI22), offset 0x458
 Interrupt 92-95 Priority (PRI23), offset 0x45C
 Interrupt 96-99 Priority (PRI24), offset 0x460
 Interrupt 100-103 Priority (PRI25), offset 0x464
 Interrupt 104-107 Priority (PRI26), offset 0x468
 Interrupt 108-111 Priority (PRI27), offset 0x46C
 Interrupt 112- 113 Priority (PRI28), offset 0x470

NOTE: This register can only be accessed from privileged mode.

The PRIn registers provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29	Interrupt [4n+3]
Bits 23:21	Interrupt [4n+2]
Bits 15:13	Interrupt [4n+1]
Bits 7:5	Interrupt [4n]

See for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the Application Interrupt and Reset Control (APINT) register (see [Section 2.5.5](#)) indicates the position of the binary point that splits the priority and subpriority fields.

These registers can only be accessed from privileged mode.

PRIn is shown in [Figure 2-11](#) and described in [Table 2-21](#).

Return to [Summary Table](#).

Figure 2-11. PRIn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTD				RESERVED								INTC			
R/W-0x0				R-0x0								R/W-0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTB				RESERVED								INTA			
R/W-0x0				R-0x0								R/W-0x0			

Table 2-21. PRIn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	INTD	R/W	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the Interrupt Priority register (n=0 for PRI0, and so on). The lower the value, the greater the priority of the corresponding interrupt.
28-24	RESERVED	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23-21	INTC	R/W	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the Interrupt Priority register (n=0 for PRI0, and so on). The lower the value, the greater the priority of the corresponding interrupt.
20-16	RESERVED	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15-13	INTB	R/W	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the Interrupt Priority register (n=0 for PRI0, and so on). The lower the value, the greater the priority of the corresponding interrupt.
12-8	RESERVED	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7-5	INTA	R/W	0x0	Interrupt Priority for Interrupt [4n] This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the Interrupt Priority register (n=0 for PRI0, and so on). The lower the value, the greater the priority of the corresponding interrupt.
4-0	RESERVED	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

2.4.7 SWTRIG Register (Offset = 0xF00) [reset = 0x0]

Software Trigger Interrupt (SWTRIG), offset 0xF00

NOTE: Only privileged software can enable unprivileged access to the SWTRIG register.

Writing an interrupt number to the SWTRIG register generates a Software Generated Interrupt (SGI). See for interrupt assignments.

When the MAINPEND bit in the Configuration and Control (CFGCTRL) register (see [Section 2.5.7](#)) is set, unprivileged software can access the SWTRIG register.

SWTRIG is shown in [Figure 2-12](#) and described in [Table 2-22](#).

Return to [Summary Table](#).

Figure 2-12. SWTRIG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								INTID							
R-0x0																								WO-0x0							

Table 2-22. SWTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7-0	INTID	WO	0x0	Interrupt ID This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3.

2.5 SCB Registers

This section lists and describes the System Control Block (SCB) registers, in numerical order by address offset. The SCB registers can only be accessed from privileged mode.

All registers must be accessed with aligned word accesses except for the FAULTSTAT, SYSPRI1, SYSPRI2, and SYSPRI3 registers, which can be accessed with byte or aligned halfword or word accesses. The processor does not support unaligned accesses to system control block registers.

[Table 2-23](#) lists the memory-mapped registers for the SCB. All register offset addresses not listed in [Table 2-23](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-23. SCB Registers

Offset	Acronym	Register Name	Section
0x8	ACTLR	Auxiliary Control	Section 2.5.1
0xD00	CPUID	CPU ID Base	Section 2.5.2
0xD04	INTCTRL	Interrupt Control and State	Section 2.5.3
0xD08	VTABLE	Vector Table Offset	Section 2.5.4
0xD0C	APINT	Application Interrupt and Reset Control	Section 2.5.5
0xD10	SYSCTRL	System Control	Section 2.5.6
0xD14	CFGCTRL	Configuration and Control	Section 2.5.7
0xD18	SYSPRI1	System Handler Priority 1	Section 2.5.8
0xD1C	SYSPRI2	System Handler Priority 2	Section 2.5.9
0xD20	SYSPRI3	System Handler Priority 3	Section 2.5.10
0xD24	SYSHNDCTRL	System Handler Control and State	Section 2.5.11
0xD28	FAULTSTAT	Configurable Fault Status	Section 2.5.12
0xD2C	HFAULTSTAT	Hard Fault Status	Section 2.5.13
0xD34	MMADDR	Memory Management Fault Address	Section 2.5.14
0xD38	FAULTADDR	Bus Fault Address	Section 2.5.15

Complex bit access types are encoded to fit into small table cells. [Table 2-24](#) shows the codes that are used for access types in this section.

Table 2-24. SCB Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.1 ACTLR Register (Offset = 0x8) [reset = 0x0]

Auxiliary Control (ACTLR)

NOTE: This register can only be accessed from privileged mode.

The ACTLR register provides disable bits for IT folding, write buffer use for accesses to the default memory map, and interruption of multi-cycle instructions. By default, this register is set to provide optimum performance from the Cortex-M4 processor and does not normally require modification.

ACTLR is shown in [Figure 2-13](#) and described in [Table 2-25](#).

Return to [Summary Table](#).

Figure 2-13. ACTLR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						DISOFP	DISFPCA
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED					DISFOLD	DISWBUF	DISMCYC
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 2-25. ACTLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	DISOFP	R/W	0x0	Disable Out-Of-Order Floating Point Disables floating-point instructions completing out of order with respect to integer instructions.
8	DISFPCA	R/W	0x0	Disable CONTROL FPCA Disable automatic update of the FPCA bit in the CONTROL register. NOTE: Two bits control when FPCA can be enabled: the ASPEN bit in the Floating-Point Context Control (FPCC) register and the DISFPCA bit in the Auxiliary Control (ACTLR) register.
7-3	RESERVED	R	0x0	
2	DISFOLD	R/W	0x0	Disable IT Folding In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called IT folding, and improves performance. However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit before executing the task, to disable IT folding.
1	DISWBUF	R/W	0x0	Disable Write Buffer
0	DISMCYC	R/W	0x0	Disable Interrupts of Multiple Cycle Instructions

2.5.2 CPUID Register (Offset = 0xD00) [reset = 0x410FC241]

CPU ID Base (CPUID)

NOTE: This register can only be accessed from privileged mode.

The CPUID register contains the Arm Cortex-M4 processor part number, version, and implementation information.

CPUID is shown in [Figure 2-14](#) and described in [Table 2-26](#).

Return to [Summary Table](#).

Figure 2-14. CPUID Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMP								VAR				CON				PARTNO								REV							
R-0x41								R-0x0				R-0xF				R-0xC24								R-0x1							

Table 2-26. CPUID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	IMP	R	0x41	Implementer Code
23-20	VAR	R	0x0	Variant Number
19-16	CON	R	0xF	Constant
15-4	PARTNO	R	0xC24	Part Number
3-0	REV	R	0x1	Revision Number

2.5.3 INTCTRL Register (Offset = 0xD04) [reset = 0x0]

Interrupt Control and State (INTCTRL)

NOTE: This register can only be accessed from privileged mode.

The INCTRL register provides a set-pending bit for the NMI exception, and set-pending and clear-pending bits for the PendSV and SysTick exceptions. In addition, bits in this register indicate the exception number of the exception being processed, whether there are preempted active exceptions, the exception number of the highest priority pending exception, and whether any interrupts are pending.

When writing to INCTRL, the effect is unpredictable when writing a 1 to both the PENDSV and UNPENDSV bits, or writing a 1 to both the PENDSTSET and PENDSTCLR bits.

INTCTRL is shown in [Figure 2-15](#) and described in [Table 2-27](#).

Return to [Summary Table](#).

Figure 2-15. INTCTRL Register

31	30	29	28	27	26	25	24
NMISSET	RESERVED		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR	RESERVED
R/W-0x0	R-0x0		R/W-0x0	W-0x0	R/W-0x0	W-0x0	R-0x0
23	22	21	20	19	18	17	16
ISRPRE	ISRPEND	RESERVED		VECPEND			
R-0x0	R-0x0	R-0x0		R-0x0			
15	14	13	12	11	10	9	8
VECPEND				RETBASE	RESERVED		
R-0x0				R-0x0	R-0x0		
7	6	5	4	3	2	1	0
VECACT							
R-0x0							

Table 2-27. INTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NMISSET	R/W	0x0	NMI Set Pending Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers the setting of this bit, and clears this bit on entering the interrupt handler. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
30-29	RESERVED	R	0x0	
28	PENDSV	R/W	0x0	PendSV Set Pending Setting this bit is the only way to set the PendSV exception state to pending. This bit is cleared by writing a 1 to the UNPENDSV bit.
27	UNPENDSV	W	0x0	PendSV Clear Pending This bit is write only; on a register read, its value is unknown.
26	PENDSTSET	R/W	0x0	SysTick Set Pending This bit is cleared by writing a 1 to the PENDSTCLR bit.
25	PENDSTCLR	W	0x0	SysTick Clear Pending This bit is write only; on a register read, its value is unknown.
24	RESERVED	R	0x0	
23	ISRPRE	R	0x0	Debug Interrupt Handling This bit is only meaningful in Debug mode and reads as zero when the processor is not in Debug mode.

Table 2-27. INTCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
22	ISRPEND	R	0x0	Interrupt Pending This bit provides status for all interrupts excluding NMI and Faults.
21-20	RESERVED	R	0x0	
19-12	VECPEND	R	0x0	Interrupt Pending Vector Number This field contains the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.
11	RETBASE	R	0x0	Return to Base This bit provides status for all interrupts excluding NMI and Faults. This bit only has meaning if the processor is currently executing an ISR (the Interrupt Program Status (IPSR) register is non-zero).
10-8	RESERVED	R	0x0	
7-0	VECACT	R	0x0	Interrupt Pending Vector Number This field contains the active exception number. The exception numbers can be found in the description for the VECPEND field. If this field is clear, the processor is in Thread mode. This field contains the same value as the ISRNUM field in the IPSR register. Subtract 16 from this value to obtain the IRQ number required to index into the Interrupt Set Enable (ENn), Interrupt Clear Enable (DISn), Interrupt Set Pending (PENDn), Interrupt Clear Pending (UNPENDn), and Interrupt Priority (PRIn) registers (see).

2.5.4 VTABLE Register (Offset = 0xD08) [reset = 0x0]

Vector Table Offset (VTABLE)

NOTE: This register can only be accessed from privileged mode.

The VTABLE register indicates the offset of the vector table base address from memory address 0x00000000.

VTABLE is shown in [Figure 2-16](#) and described in [Table 2-28](#).

Return to [Summary Table](#).

Figure 2-16. VTABLE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET												RESERVED																			
R/W-0x0												R-0x0																			

Table 2-28. VTABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	OFFSET	R/W	0x0	Vector Table Offset When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 112 interrupts, the offset must be aligned on a 1024-byte boundary.
9-0	RESERVED	R	0x0	

2.5.5 APINT Register (Offset = 0xD0C) [reset = 0xFA050000]

Application Interrupt and Reset Control (APINT)

NOTE: This register can only be accessed from privileged mode.

The APINT register provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, 0x05FA must be written to the VECTKEY field, otherwise the write is ignored.

The PRIGROUP field indicates the position of the binary point that splits the INTx fields in the Interrupt Priority (PRIx) registers into separate group priority and subpriority fields. Table 2-29 shows how the PRIGROUP value controls this split. The bit numbers in the Group Priority Field and Subpriority Field columns in the table refer to the bits in the INTA field. For the INTB field, the corresponding bits are 15:13; for INTC, 23:21; and for INTD, 31:29.

Determining preemption of an exception uses only the group priority field.

Table 2-29. Interrupt Priority Levels

PRIGROUP Bit Field	Binary Point ⁽¹⁾	Group Priority Field	Subpriority Field	Group Priorities	Subpriorities
0x0 to 0x4	bxxx.	[7:5]	None	8	1
0x5	bxx.y	[7:6]	[5]	4	2
0x6	bx.yy	[7]	[6:5]	2	4
0x7	b.yyy	None	[7:5]	1	8

⁽¹⁾ INTx field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

APINT is shown in Figure 2-17 and described in Table 2-30.

Return to [Summary Table](#).

Figure 2-17. APINT Register

31	30	29	28	27	26	25	24
VECTKEY							
R/W-0xFA05							
23	22	21	20	19	18	17	16
VECTKEY							
R/W-0xFA05							
15	14	13	12	11	10	9	8
ENDIANESS	RESERVED				PRIGROUP		
R-0x0	R-0x0				R/W-0x0		
7	6	5	4	3	2	1	0
RESERVED					SYSRESREQ	VECTCLRACT	VECTRESET
R-0x0					W-0x0	W-0x0	W-0x0

Table 2-30. APINT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	VECTKEY	R/W	0xFA05	Register Key This field is used to guard against accidental writes to this register. 0x05FA must be written to this field in order to change the bits in this register. On a read, 0xFA05 is returned.
15	ENDIANESS	R	0x0	Data Endianess The MSP432E4 implementation uses only little-endian mode so this is cleared to 0.
14-11	RESERVED	R	0x0	

Table 2-30. APINT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10-8	PRIGROUP	R/W	0x0	Interrupt Priority Grouping This field determines the split of group priority from subpriority (see for more information).
7-3	RESERVED	R	0x0	
2	SYSRESREQ	W	0x0	System Reset Request This bit is automatically cleared during the reset of the core and reads as 0.
1	VECTCLRACT	W	0x0	Clear Active NMI / Fault This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.
0	VECTRESET	W	0x0	System Reset This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.

2.5.6 SYSCTRL Register (Offset = 0xD10) [reset = 0x0]

System Control (SYSCTRL)

NOTE: This register can only be accessed from privileged mode.

The SYSCTRL register controls features of entry to and exit from low-power state.

SYSCTRL is shown in [Figure 2-18](#) and described in [Table 2-31](#).

Return to [Summary Table](#).

Figure 2-18. SYSCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			SEVONPEND	RESERVED	SLEEPDEEP	SLEEPEXIT	RESERVED
R-0x0			R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 2-31. SYSCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	SEVONPEND	R/W	0x0	Wake Up on Pending When an event or interrupt enters the pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of a SEV instruction or an external event.
3	RESERVED	R	0x0	
2	SLEEPDEEP	R/W	0x0	Deep Sleep Enable
1	SLEEPEXIT	R/W	0x0	Sleep on ISR Exit Setting this bit enables an interrupt-driven application to avoid returning to an empty main application.
0	RESERVED	R	0x0	

2.5.7 CFGCTRL Register (Offset = 0xD14) [reset = 0x200]

Configuration and Control (CFGCTRL)

NOTE: This register can only be accessed from privileged mode.

The CFGCTRL register controls entry to Thread mode and enables: the handlers for NMI, hard fault and faults escalated by the FAULTMASK register to ignore bus faults; trapping of divide by zero and unaligned accesses; and access to the SWTRIG register by unprivileged software (see).

CFGCTRL is shown in [Figure 2-19](#) and described in [Table 2-32](#).

Return to [Summary Table](#).

Figure 2-19. CFGCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						STKALIGN	BFHFNMI
R-0x0						R/W-0x1	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			DIV0	UNALIGNED	RESERVED	MAINPEND	BASETHR
R-0x0			R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0

Table 2-32. CFGCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	STKALIGN	R/W	0x1	Stack Alignment on Exception Entry On exception entry, the processor uses bit 9 of the stacked PSR to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.
8	BFHFNMI	R/W	0x0	Ignore Bus Fault in NMI and Fault This bit enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. The setting of this bit applies to the hard fault, NMI, and FAULTMASK escalated handlers. Set this bit only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.
7-5	RESERVED	R	0x0	
4	DIV0	R/W	0x0	Trap on Divide by 0 This bit enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0.
3	UNALIGNED	R/W	0x0	Trap on Unaligned Access Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of whether UNALIGNED is set.
2	RESERVED	R	0x0	
1	MAINPEND	R/W	0x0	Allow Main Interrupt Trigger
0	BASETHR	R/W	0x0	Thread State Control

2.5.8 SYSPRI1 Register (Offset = 0xD18) [reset = 0x0]

System Handler Priority 1 (SYSPRI1)

NOTE: This register can only be accessed from privileged mode.

The SYSPRI1 register configures the priority level, 0 to 7 of the usage fault, bus fault, and memory management fault exception handlers. This register is byte-accessible.

SYSPRI1 is shown in [Figure 2-20](#) and described in [Table 2-33](#).

Return to [Summary Table](#).

Figure 2-20. SYSPRI1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								USAGE				RESERVED			
R-0x0								R/W-0x0				R-0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUS				RESERVED				MEM				RESERVED			
R/W-0x0				R-0x0				R/W-0x0				R-0x0			

Table 2-33. SYSPRI1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23-21	USAGE	R/W	0x0	Usage Fault Priority This field configures the priority level of the usage fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
20-16	RESERVED	R	0x0	
15-13	BUS	R/W	0x0	Bus Fault Priority This field configures the priority level of the bus fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
12-8	RESERVED	R	0x0	
7-5	MEM	R/W	0x0	Memory Management Fault Priority This field configures the priority level of the memory management fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
4-0	RESERVED	R	0x0	

2.5.9 SYSPRI2 Register (Offset = 0xD1C) [reset = 0x0]

System Handler Priority 2 (SYSPRI2)

NOTE: This register can only be accessed from privileged mode.

The SYSPRI2 register configures the priority level, 0 to 7 of the SVCcall handler. This register is byte-accessible.

SYSPRI2 is shown in [Figure 2-21](#) and described in [Table 2-34](#).

Return to [Summary Table](#).

Figure 2-21. SYSPRI2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVC			RESERVED																												
R/W-0x0			R-0x0																												

Table 2-34. SYSPRI2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	SVC	R/W	0x0	SVCcall Priority This field configures the priority level of SVCcall. Configurable priority values are in the range 0-7, with lower values having higher priority.
28-0	RESERVED	R	0x0	

2.5.10 SYSPRI3 Register (Offset = 0xD20) [reset = 0x0]

System Handler Priority 3 (SYSPRI3)

NOTE: This register can only be accessed from privileged mode.

The SYSPRI3 register configures the priority level, 0 to 7 of the SysTick exception and PendSV handlers. This register is byte-accessible.

SYSPRI3 is shown in [Figure 2-22](#) and described in [Table 2-35](#).

Return to [Summary Table](#).

Figure 2-22. SYSPRI3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TICK				RESERVED								PENDSV			
R/W-0x0				R-0x0								R/W-0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								DEBUG				RESERVED			
R-0x0								R/W-0x0				R-0x0			

Table 2-35. SYSPRI3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	TICK	R/W	0x0	SysTick Exception Priority This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority.
28-24	RESERVED	R	0x0	
23-21	PENDSV	R/W	0x0	PendSV Priority This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority.
20-8	RESERVED	R	0x0	
7-5	DEBUG	R/W	0x0	Debug Priority This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority.
4-0	RESERVED	R	0x0	

2.5.11 SYSHNDCTRL Register (Offset = 0xD24) [reset = 0x0]

System Handler Control and State (SYSHNDCTRL)

NOTE: This register can only be accessed from privileged mode.

The SYSHNDCTRL register enables the system handlers, and indicates the pending status of the usage fault, bus fault, memory management fault, and SVC exceptions as well as the active status of the system handlers.

If a system handler is disabled and the corresponding fault occurs, the processor treats the fault as a hard fault.

This register can be modified to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

NOTE: Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.

If the value of a bit in this register must be modified after enabling the system handlers, a read-modify-write procedure must be used to ensure that only the required bit is modified.

SYSHNDCTRL is shown in [Figure 2-23](#) and described in [Table 2-36](#).

Return to [Summary Table](#).

Figure 2-23. SYSHNDCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED					USAGE	BUS	MEM
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
SVC	BUSP	MEMP	USAGEP	TICK	PNDSP	RESERVED	MON
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0
7	6	5	4	3	2	1	0
SVCA	RESERVED			USGA	RESERVED	BUSA	MEMA
R/W-0x0	R-0x0			R/W-0x0	R-0x0	R/W-0x0	R/W-0x0

Table 2-36. SYSHNDCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0x0	
18	USAGE	R/W	0x0	Usage Fault Enable
17	BUS	R/W	0x0	Bus Fault Enable
16	MEM	R/W	0x0	Memory Management Fault Enable
15	SVC	R/W	0x0	SVC Call Pending This bit can be modified to change the pending status of the SVC call exception.
14	BUSP	R/W	0x0	Bus Fault Pending This bit can be modified to change the pending status of the bus fault exception.

Table 2-36. SYSHNDCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13	MEMP	R/W	0x0	Memory Management Fault Pending This bit can be modified to change the pending status of the memory management fault exception.
12	USAGEP	R/W	0x0	Usage Fault Pending This bit can be modified to change the pending status of the usage fault exception.
11	TICK	R/W	0x0	SysTick Exception Active This bit can be modified to change the active status of the SysTick exception, however, see the Caution above before setting this bit.
10	PND SV	R/W	0x0	PendSV Exception Active This bit can be modified to change the active status of the PendSV exception, however, see the Caution above before setting this bit.
9	RESERVED	R	0x0	
8	MON	R/W	0x0	Debug Monitor Active
7	SVCA	R/W	0x0	SVC Call Active This bit can be modified to change the active status of the SVC call exception, however, see the Caution above before setting this bit.
6-4	RESERVED	R	0x0	
3	USGA	R/W	0x0	Usage Fault Active This bit can be modified to change the active status of the usage fault exception, however, see the Caution above before setting this bit.
2	RESERVED	R	0x0	
1	BUSA	R/W	0x0	Bus Fault Active This bit can be modified to change the active status of the bus fault exception, however, see the Caution above before setting this bit.
0	MEMA	R/W	0x0	Memory Management Fault Active This bit can be modified to change the active status of the memory management fault exception, however, see the Caution above before setting this bit.

2.5.12 FAULTSTAT Register (Offset = 0xD28) [reset = 0x0]

Configurable Fault Status (FAULTSTAT)

NOTE: This register can only be accessed from privileged mode.

The FAULTSTAT register indicates the cause of a memory management fault, bus fault, or usage fault. Each of these functions is assigned to a subregister as follows:

- Usage Fault Status (UFAULTSTAT), bits 31:16
- Bus Fault Status (BFAULTSTAT), bits 15:8
- Memory Management Fault Status (MFAULTSTAT), bits 7:0

FAULTSTAT is byte accessible. FAULTSTAT or its subregisters can be accessed as follows:

- The complete FAULTSTAT register, with a word access to offset 0xD28
- The MFAULTSTAT, with a byte access to offset 0xD28
- The MFAULTSTAT and BFAULTSTAT, with a halfword access to offset 0xD28
- The BFAULTSTAT, with a byte access to offset 0xD29
- The UFAULTSTAT, with a halfword access to offset 0xD2A

Bits are cleared by writing a 1 to them.

In a fault handler, the true faulting address can be determined by:

1. Read and save the Memory Management Fault Address (MMADDR) or Bus Fault Address (FAULTADDR) value.
2. Read the MMARV bit in MFAULTSTAT, or the BFARV bit in BFAULTSTAT to determine if the MMADDR or FAULTADDR contents are valid.

Software must follow this sequence because another higher priority exception might change the MMADDR or FAULTADDR value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the MMADDR or FAULTADDR value.

FAULTSTAT is shown in [Figure 2-24](#) and described in [Table 2-37](#).

Return to [Summary Table](#).

Figure 2-24. FAULTSTAT Register

31	30	29	28	27	26	25	24
RESERVED						DIV0	UNALIGN
R-0x0						R/W1C-0x0	R/W1C-0x0
23	22	21	20	19	18	17	16
RESERVED				NOCP	INVPC	INVSTAT	UNDEF
R-0x0				R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0
15	14	13	12	11	10	9	8
BFARV	RESERVED	BLSPERR	BSTKE	BUSTKE	IMPRE	PRECISE	IBUS
R/W1C-0x0	R-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0
7	6	5	4	3	2	1	0
MMARV	RESERVED	MLSPERR	MSTKE	MUSTKE	RESERVED	DERR	IERR
R/W1C-0x0	R-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R-0x0	R/W1C-0x0	R/W1C-0x0

Table 2-37. FAULTSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25	DIV0	R/W1C	0x0	Divide-by-Zero Usage Fault When this bit is set, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Trapping on divide-by-zero is enabled by setting the DIV0 bit in the Configuration and Control (CFGCTRL) register (see). This bit is cleared by writing a 1 to it.
24	UNALIGN	R/W1C	0x0	Unaligned Access Usage Fault Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the configuration of this bit. Trapping on unaligned access is enabled by setting the UNALIGNED bit in the CFGCTRL register (see). This bit is cleared by writing a 1 to it.
23-20	RESERVED	R	0x0	
19	NOCP	R/W1C	0x0	No Coprocessor Usage Fault This bit is cleared by writing a 1 to it.
18	INVPC	R/W1C	0x0	Invalid PC Load Usage Fault When this bit is set, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC. This bit is cleared by writing a 1 to it.
17	INVSTAT	R/W1C	0x0	Invalid State Usage Fault When this bit is set, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the Execution Program Status Register (EPSR) register. This bit is not set if an undefined instruction uses the EPSR register. This bit is cleared by writing a 1 to it.
16	UNDEF	R/W1C	0x0	Undefined Instruction Usage Fault When this bit is set, the PC value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode. This bit is cleared by writing a 1 to it.
15	BFARV	R/W1C	0x0	Bus Fault Address Register Valid This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose FAULTADDR register value has been overwritten. This bit is cleared by writing a 1 to it.
14	RESERVED	R	0x0	
13	BLSPERR	R/W1C	0x0	Bus Fault on Floating-Point Lazy State Preservation This bit is cleared by writing a 1 to it.
12	BSTKE	R/W1C	0x0	Stack Bus Fault When this bit is set, the SP is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the FAULTADDR register. This bit is cleared by writing a 1 to it.
11	BUSTKE	R/W1C	0x0	Unstack Bus Fault This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The SP is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the FAULTADDR register. This bit is cleared by writing a 1 to it.

Table 2-37. FAULTSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	IMPRE	R/W1C	0x0	<p>Imprecise Data Bus Error</p> <p>When this bit is set, a fault address is not written to the FAULTADDR register. This fault is asynchronous. Therefore, if the fault is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher-priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects that both the IMPRE bit is set and one of the precise fault status bits is set. This bit is cleared by writing a 1 to it.</p>
9	PRECISE	R/W1C	0x0	<p>Precise Data Bus Error</p> <p>When this bit is set, the fault address is written to the FAULTADDR register. This bit is cleared by writing a 1 to it.</p>
8	IBUS	R/W1C	0x0	<p>Instruction Bus Error</p> <p>The processor detects the instruction bus error on prefetching an instruction, but sets this bit only if it attempts to issue the faulting instruction. When this bit is set, a fault address is not written to the FAULTADDR register. This bit is cleared by writing a 1 to it.</p>
7	MMARV	R/W1C	0x0	<p>Memory Management Fault Address Register Valid</p> <p>If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose MMADDR register value has been overwritten. This bit is cleared by writing a 1 to it.</p>
6	RESERVED	R	0x0	
5	MLSPERR	R/W1C	0x0	<p>Memory Management Fault on Floating-Point Lazy State Preservation</p> <p>This bit is cleared by writing a 1 to it.</p>
4	MSTKE	R/W1C	0x0	<p>Stack Access Violation</p> <p>When this bit is set, the SP is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the MMADDR register. This bit is cleared by writing a 1 to it.</p>
3	MUSTKE	R/W1C	0x0	<p>Unstack Access Violation</p> <p>This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The SP is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the MMADDR register. This bit is cleared by writing a 1 to it.</p>
2	RESERVED	R	0x0	
1	DERR	R/W1C	0x0	<p>Data Access Violation</p> <p>When this bit is set, the PC value stacked for the exception return points to the faulting instruction and the address of the attempted access is written to the MMADDR register. This bit is cleared by writing a 1 to it.</p>
0	IERR	R/W1C	0x0	<p>Instruction Access Violation</p> <p>This fault occurs on any access to an XN region, even when the MPU is disabled or not present. When this bit is set, the PC value stacked for the exception return points to the faulting instruction and the address of the attempted access is not written to the MMADDR register. This bit is cleared by writing a 1 to it.</p>

2.5.13 HFAULTSTAT Register (Offset = 0xD2C) [reset = 0x0]

Hard Fault Status (HFAULTSTAT)

NOTE: This register can only be accessed from privileged mode.

The HFAULTSTAT register gives information about events that activate the hard fault handler.

Bits are cleared by writing a 1 to them.

HFAULTSTAT is shown in [Figure 2-25](#) and described in [Table 2-38](#).

Return to [Summary Table](#).

Figure 2-25. HFAULTSTAT Register

31	30	29	28	27	26	25	24
DBG	FORCED	RESERVED					
R/W-0x0	R/W1C-0x0	R-0x0					
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						VECT	RESERVED
R-0x0						R/W1C-0x0	R-0x0

Table 2-38. HFAULTSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DBG	R/W	0x0	Debug Event This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable.
30	FORCED	R/W1C	0x0	Forced Hard Fault When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault. This bit is cleared by writing a 1 to it.
29-2	RESERVED	R	0x0	
1	VECT	R/W1C	0x0	Vector Table Read Fault This error is always handled by the hard fault handler. When this bit is set, the PC value stacked for the exception return points to the instruction that was preempted by the exception. This bit is cleared by writing a 1 to it.
0	RESERVED	R	0x0	

2.5.14 MMADDR Register (Offset = 0xD34) [reset = X]

Memory Management Fault Address (MMADDR)

NOTE: This register can only be accessed from privileged mode.

The MMADDR register contains the address of the location that generated a memory management fault. When an unaligned access faults, the address in the MMADDR register is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size. Bits in the Memory Management Fault Status (MFAULTSTAT) register indicate the cause of the fault and whether the value in the MMADDR register is valid (see [Section 2.5.12](#)).

MMADDR is shown in [Figure 2-26](#) and described in [Table 2-39](#).

Return to [Summary Table](#).

Figure 2-26. MMADDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
R/W-X																															

Table 2-39. MMADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	X	Fault Address When the MMARV bit of MFAULTSTAT is set, this field holds the address of the location that generated the memory management fault.

2.5.15 FAULTADDR Register (Offset = 0xD38) [reset = X]

Bus Fault Address (FAULTADDR)

NOTE: This register can only be accessed from privileged mode.

The FAULTADDR register contains the address of the location that generated a bus fault. When an unaligned access faults, the address in the FAULTADDR register is the one requested by the instruction, even if it is not the address of the fault. Bits in the Bus Fault Status (BFAULTSTAT) register indicate the cause of the fault and whether the value in the FAULTADDR register is valid (see [Section 2.5.12](#)).

FAULTADDR is shown in [Figure 2-27](#) and described in [Table 2-40](#).

Return to [Summary Table](#).

Figure 2-27. FAULTADDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
R/W-X																															

Table 2-40. FAULTADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	X	Fault Address When the FAULTADDRV bit of BFAULTSTAT is set, this field holds the address of the location that generated the bus fault.

2.6 MPU Registers

This section lists and describes the Memory Protection Unit (MPU) registers, in numerical order by address offset.

The MPU registers can only be accessed from privileged mode.

[Table 2-41](#) lists the memory-mapped registers for the MPU. All register offset addresses not listed in [Table 2-41](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-41. MPU Registers

Offset	Acronym	Register Name	Section
0xD90	MPUTYPE	MPU Type	Section 2.6.1
0xD94	MPUCTRL	MPU Control	Section 2.6.2
0xD98	MPUNUMBER	MPU Region Number	Section 2.6.3
0xD9C	MPUBASE	MPU Region Base Address	Section 2.6.4
0xDA0	MPUATTR	MPU Region Attribute and Size	Section 2.6.5
0xDA4	MPUBASE1	MPU Region Base Address 1	Section 2.6.4
0xDA8	MPUATTR1	MPU Region Attribute and Size 1	Section 2.6.5
0xDAC	MPUBASE2	MPU Region Base Address 2	Section 2.6.4
0xDB0	MPUATTR2	MPU Region Attribute and Size 2	Section 2.6.5
0xDB4	MPUBASE3	MPU Region Base Address 3	Section 2.6.4
0xDB8	MPUATTR3	MPU Region Attribute and Size 3	Section 2.6.5

Complex bit access types are encoded to fit into small table cells. [Table 2-42](#) shows the codes that are used for access types in this section.

Table 2-42. MPU Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
WO	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.6.1 MPUType Register (Offset = 0xD90) [reset = 0x800]

MPU Type (MPUTYPE), offset 0xD90

NOTE: This register can only be accessed from privileged mode.

The MPUType register indicates whether the MPU is present, and if so, how many regions it supports.

MPUTYPE is shown in [Figure 2-28](#) and described in [Table 2-43](#).

Return to [Summary Table](#).

Figure 2-28. MPUType Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
IREGION							
R-0x0							
15	14	13	12	11	10	9	8
DREGION							
R-0x8							
7	6	5	4	3	2	1	0
RESERVED							SEPARATE
R-0x0							R-0x0

Table 2-43. MPUType Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23-16	IREGION	R	0x0	Number of I Regions This field indicates the number of supported MPU instruction regions. This field always contains 0x00. The MPU memory map is unified and is described by the DREGION field.
15-8	DREGION	R	0x8	Number of D Regions
7-1	RESERVED	R	0x0	
0	SEPARATE	R	0x0	Separate or Unified MPU

2.6.2 MPUCTRL Register (Offset = 0xD94) [reset = 0x0]

MPU Control (MPUCTRL), offset 0xD94

NOTE: This register can only be accessed from privileged mode.

The MPUCTRL register enables the MPU, enables the default memory map background region, and enables use of the MPU when in the hard fault, Non-maskable Interrupt (NMI), and Fault Mask Register (FAULTMASK) escalated handlers.

When the ENABLE and PRIVDEFEN bits are both set:

- For privileged accesses, the default memory map is as described in [Section 1.5](#). Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

Execute Never (XN) and Strongly Ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFEN bit is set. If the PRIVDEFEN bit is set and no regions are enabled, then only privileged software can operate.

When the ENABLE bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see [Table 1-16](#) for more information). The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFEN is set.

Unless HFNMIENA is set, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception or when FAULTMASK is enabled. Setting the HFNMIENA bit enables the MPU when operating with these two priorities.

MPUCTRL is shown in [Figure 2-29](#) and described in [Table 2-44](#).

Return to [Summary Table](#).

Figure 2-29. MPUCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					PRIVDEFEN	HFNMIENA	ENABLE
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 2-44. MPUCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	PRIVDEFEN	R/W	0x0	<p>MPU Default Region</p> <p>This bit enables privileged software access to the default memory map.</p> <p>When this bit is set, the background region acts as if it is region number -1.</p> <p>Any region that is defined and enabled has priority over this default map.</p> <p>If the MPU is disabled, the processor ignores this bit.</p>
1	HFNMIENA	R/W	0x0	<p>MPU Enabled During Faults</p> <p>This bit controls the operation of the MPU during hard fault, NMI, and FAULTMASK handlers.</p> <p>When the MPU is disabled and this bit is set, the resulting behavior is unpredictable.</p>
0	ENABLE	R/W	0x0	<p>MPU Enable</p> <p>When the MPU is disabled and the HFNMIENA bit is set, the resulting behavior is unpredictable.</p>

2.6.3 MPUNUMBER Register (Offset = 0xD98) [reset = 0x0]

MPU Region Number (MPUNUMBER), offset 0xD98

NOTE: This register can only be accessed from privileged mode.

The MPUNUMBER register selects which memory region is referenced by the MPU Region Base Address (MPUBASE) and MPU Region Attribute and Size (MPUATTR) registers. Normally, the required region number should be written to this register before accessing the MPUBASE or the MPUATTR register. However, the region number can be changed by writing to the MPUBASE register with the VALID bit set (see [Section 2.6.4](#)). This write updates the value of the REGION field.

MPUNUMBER is shown in [Figure 2-30](#) and described in [Table 2-45](#).

Return to [Summary Table](#).

Figure 2-30. MPUNUMBER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												NUMBER			
R-0x0												R/W-0x0			

Table 2-45. MPUNUMBER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2-0	NUMBER	R/W	0x0	MPU Region to Access This field indicates the MPU region referenced by the MPUBASE and MPUATTR registers. The MPU supports eight memory regions.

2.6.4 MPUBASEn Registers

MPU Region Base Address (MPUBASE), offset 0xD9C

MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4

MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC

MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4

NOTE: This register can only be accessed from privileged mode.

The MPUBASE register defines the base address of the MPU region selected by the MPU Region Number (MPUNUMBER) register and can update the value of the MPUNUMBER register. To change the current region number and update the MPUNUMBER register, write the MPUBASE register with the VALID bit set.

The ADDR field is bits 31: N of the MPUBASE register. Bits(N-1):5 are reserved. The region size, as specified by the SIZE field in the MPU Region Attribute and Size (MPUATTR) register, defines the value of N where:

$$N = \log_2(\text{Region size in bytes})$$

If the region size is configured to 4 GB in the MPUATTR register, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x00000000.

The base address is aligned to the size of the region. For example, a 64KB region must be aligned on a multiple of 64KB, for example, at 0x00010000 or 0x00020000.

MPUBASEn is shown in [Figure 2-31](#) and described in [Table 2-46](#).

Return to [Summary Table](#).

Figure 2-31. MPUBASEn Register

31	30	29	28	27	26	25	24
ADDR							
R/W-0x0							
23	22	21	20	19	18	17	16
ADDR							
R/W-0x0							
15	14	13	12	11	10	9	8
ADDR							
R/W-0x0							
7	6	5	4	3	2	1	0
ADDR			VALID	RESERVED	REGION		
R/W-0x0			WO-0x0	R-0x0	R/W-0x0		

Table 2-46. MPUBASEn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	ADDR	R/W	0x0	Base Address Mask Bits 31:N in this field contain the region base address. The value of N depends on the region size, as shown above. The remaining bits (N-1):5 are reserved. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	VALID	WO	0x0	Region Number Valid This bit is always read as 0.
3	RESERVED	R	0x0	

Table 2-46. MPUBASEn Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2-0	REGION	R/W	0x0	<p>Region Number</p> <p>On a write, contains the value to be written to the MPUNUMBER register.</p> <p>On a read, returns the current region number in the MPUNUMBER register.</p>

2.6.5 MPUATTRn Registers

MPU Region Attribute and Size (MPUATTR), offset 0xDA0

MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8

MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0

MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8

NOTE: This register can only be accessed from privileged mode.

The MPUATTR register defines the region size and memory attributes of the MPU region specified by the MPU Region Number (MPUNUMBER) register and enables that region and any subregions.

The MPUATTR register is accessible using word or halfword accesses with the most-significant halfword holding the region attributes and the least-significant halfword holds the region size and the region and subregion enable bits.

The MPU access permission attribute bits, XN, AP, TEX, S, C, and B, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

The SIZE field defines the size of the MPU memory region specified by the MPUNUMBER register as follows:

$$(\text{Region size in bytes}) = 2^{(\text{SIZE}+1)}$$

The smallest permitted region size is 32 bytes, corresponding to a SIZE value of 4. [Table 2-47](#) gives example SIZE values with the corresponding region size and value of N in the MPU Region Base Address (MPUBASE) register.

Table 2-47. Example SIZE Field Values

SIZE Encoding	Region Size	Value of N ⁽¹⁾	Note
00100b (0x4)	32 B	5	Minimum permitted size
01001b (0x9)	1 KB	10	-
10011b (0x13)	1 MB	20	-
11101b (0x1D)	1 GB	30	-
11111b (0x1F)	4 GB	No valid ADDR field in MPUBASE; the region occupies the complete memory map.	Maximum possible size

⁽¹⁾ Refers to the N parameter in the MPUBASE register (see [Section 2.6.4](#)).

MPUATTRn is shown in [Figure 2-32](#) and described in [Table 2-48](#).

Return to [Summary Table](#).

Figure 2-32. MPUATTRn Register

31	30	29	28	27	26	25	24
RESERVED			XN	RESERVED	AP		
R-0x0			R/W-0x0	R-0x0	R/W-0x0		
23	22	21	20	19	18	17	16
RESERVED		TEX		S		C	B
R-0x0		R/W-0x0		R/W-0x0		R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
SRD							
R/W-0x0							
7	6	5	4	3	2	1	0
RESERVED		SIZE					ENABLE
R-0x0		R/W-0x0					R/W-0x0

Table 2-48. MPUATTRn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0x0	
28	XN	R/W	0x0	Instruction Access Disable
27	RESERVED	R	0x0	
26-24	AP	R/W	0x0	Access Privilege For information on using this bit field, see Table 2-5 .
23-22	RESERVED	R	0x0	
21-19	TEX	R/W	0x0	Type Extension Mask For information on using this bit field, see Table 2-3 .
18	S	R/W	0x0	Shareable For information on using this bit, see Table 2-3 .
17	C	R/W	0x0	Cacheable For information on using this bit, see .
16	B	R/W	0x0	Bufferable For information on using this bit, see Table 2-3 .
15-8	SRD	R/W	0x0	Subregion Disable Bits Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, configure the SRD field as 0x00. See for more information.
7-6	RESERVED	R	0x0	
5-1	SIZE	R/W	0x0	Region Size Mask The SIZE field defines the size of the MPU memory region specified by the MPUNUMBER register. Refer to Table 2-47 for more information.
0	ENABLE	R/W	0x0	Region Enable

2.7 FPU Registers

[Table 2-49](#) lists the memory-mapped registers for the FPU. All register offset addresses not listed in [Table 2-49](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-49. FPU Registers

Offset	Acronym	Register Name	Section
0xD88	CPAC	Coprocessor Access Control	Section 2.7.1
0xF34	FPCC	Floating-Point Context Control	Section 2.7.2
0xF38	FPCA	Floating-Point Context Address	Section 2.7.3
0xF3C	FPDSC	Floating-Point Default Status Control	Section 2.7.4

Complex bit access types are encoded to fit into small table cells. [Table 2-50](#) shows the codes that are used for access types in this section.

Table 2-50. FPU Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.7.1 CPAC Register (Offset = 0xD88) [reset = 0x0]

Coprocessor Access Control (CPAC)

The CPAC register specifies the access privileges for coprocessors.

CPAC is shown in [Figure 2-33](#) and described in [Table 2-51](#).

Return to [Summary Table](#).

Figure 2-33. CPAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								CP11		CP10		RESERVED			
R-0x0								R/W-0x0		R/W-0x0		R-0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															
R-0x0															

Table 2-51. CPAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23-22	CP11	R/W	0x0	CP11 Coprocessor Access Privilege 0x0 = Access Denied. Any attempted access generates a NOCP Usage Fault. 0x1 = Privileged Access Only. An unprivileged access generates a NOCP fault. 0x2 = Reserved. The result of any access is unpredictable. 0x3 = Full Access
21-20	CP10	R/W	0x0	CP10 Coprocessor Access Privilege 0x0 = Access Denied. Any attempted access generates a NOCP Usage Fault. 0x1 = Privileged Access Only. An unprivileged access generates a NOCP fault. 0x2 = Reserved. The result of any access is unpredictable. 0x3 = Full Access
19-0	RESERVED	R	0x0	

2.7.2 FPCC Register (Offset = 0xF34) [reset = 0xC0000000]

Floating-Point Context Control (FPCC)

The FPCC register sets or returns FPU control data.

FPCC is shown in [Figure 2-34](#) and described in [Table 2-52](#).

Return to [Summary Table](#).

Figure 2-34. FPCC Register

31	30	29	28	27	26	25	24
ASPEN	LSPEN	RESERVED					
R/W-0x1	R/W-0x1	R-0x0					
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							MONRDY
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	BFRDY	MMRDY	HFRDY	THREAD	RESERVED_3	USER	LSPACT
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0

Table 2-52. FPCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31	ASPEN	R/W	0x1	Automatic State Preservation Enable When set, enables the use of the FRACTV bit in the CONTROL register on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit. NOTE: Two bits control when FPCA can be enabled: the ASPEN bit in the Floating-Point Context Control (FPCC) register and the DISFPCA bit in the Auxiliary Control (ACTLR) register.
30	LSPEN	R/W	0x1	Lazy State Preservation Enable When set, enables automatic lazy state preservation for floating-point context.
29-9	RESERVED	R	0x0	
8	MONRDY	R/W	0x0	Monitor Ready When set, DebugMonitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated.
7	RESERVED	R	0x0	
6	BFRDY	R/W	0x0	Bus Fault Ready When set, BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated.
5	MMRDY	R/W	0x0	Memory Management Fault Ready When set, MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated.
4	HFRDY	R/W	0x0	Hard Fault Ready When set, priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated.
3	THREAD	R/W	0x0	Thread Mode When set, mode was Thread Mode when the floating-point stack frame was allocated.
2	RESERVED	R	0x0	

Table 2-52. FPCC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	USER	R/W	0x0	User Privilege Level When set, privilege level was user when the floating-point stack frame was allocated.
0	LSPACT	R/W	0x0	Lazy State Preservation Active When set, Lazy State preservation is active. Floating-point stack frame has been allocated but saving state to it has been deferred.

2.7.3 FPCA Register (Offset = 0xF38) [reset = X]

Floating-Point Context Address (FPCA)

The FPCA register holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

FPCA is shown in [Figure 2-35](#) and described in [Table 2-53](#).

Return to [Summary Table](#).

Figure 2-35. FPCA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS												RESERVED			
R/W-X												R-0x0			

Table 2-53. FPCA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	ADDRESS	R/W	X	Address The location of the unpopulated floating-point register space allocated on an exception stack frame.
2-0	RESERVED	R	0x0	

2.7.4 FPDSC Register (Offset = 0xF3C) [reset = X]

Floating-Point Default Status Control (FPDSC)

The FPDSC register holds the default values for the Floating-Point Status Control (FPSC) register.

FPDSC is shown in [Figure 2-36](#) and described in [Table 2-54](#).

Return to [Summary Table](#).

Figure 2-36. FPDSC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					AHP	DN	FZ	RMODE			RESERVED				
R-0x0					R/W-X	R/W-X	R/W-X	R/W-X			R-0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															
R-0x0															

Table 2-54. FPDSC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0x0	
26	AHP	R/W	X	AHP Bit Default This bit holds the default value for the AHP bit in the FPSC register.
25	DN	R/W	X	DN Bit Default This bit holds the default value for the DN bit in the FPSC register.
24	FZ	R/W	X	FZ Bit Default This bit holds the default value for the FZ bit in the FPSC register.
23-22	RMODE	R/W	X	RMODE Bit Default This bit holds the default value for the RMODE bit field in the FPSC register. 0x0 = Round to Nearest (RN) mode 0x1 = Round toward Plus Infinity (RP) mode 0x2 = Round toward Minus Infinity (RM) mode 0x3 = Round toward Zero (RZ) mode
21-0	RESERVED	R	0x0	

JTAG Interface

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port (TAP) and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Register (DR) can be used to test the interconnections of assembled printed-circuit boards and obtain manufacturing information on the components. The JTAG port also provides a means of accessing and controlling design-for-test features such as I/O pin observation and control, scan testing, and debugging.

Topic	Page
3.1 Introduction	184
3.2 Block Diagram.....	184
3.3 Functional Description	185
3.4 Initialization and Configuration	190
3.5 Register Descriptions.....	190

3.1 Introduction

The JTAG port is composed of four pins: TCK, TMS, TDI, and TDO. Data is transmitted serially into the controller on TDI and out of the controller on TDO. The interpretation of this data is dependent on the current state of the TAP controller. For detailed information on the operation of the JTAG port and test access port (TAP) controller, see the IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture.

The MSP432E4 JTAG controller works with the Arm® JTAG controller built into the Cortex®-M4F core by multiplexing the TDO outputs from both JTAG controllers. Arm JTAG instructions select the Arm TDO output while JTAG instructions select the TDO output. The multiplexer is controlled by the JTAG controller, which has comprehensive programming for the Arm core, MSP432E4 microcontroller, and unimplemented JTAG instructions.

The JTAG module has the following features:

- IEEE 1149.1-1990 compatible TAP controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, and EXTEST
- Arm additional instructions: APACC, DPACC, and ABORT
- Integrated Arm Serial Wire Debug (SWD)
 - Serial Wire JTAG Debug Port (SWJ-DP)
 - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
 - Data Watchpoint and Trace (DWT) unit for implementing watchpoints, trigger resources, and system profiling
 - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
 - Embedded Trace Macrocell (ETM) for instruction trace capture
 - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

See the Arm Debug Interface V5 Architecture Specification for more information on the Arm JTAG controller.

3.2 Block Diagram

Figure 3-1 shows the block diagram of the JTAG module.

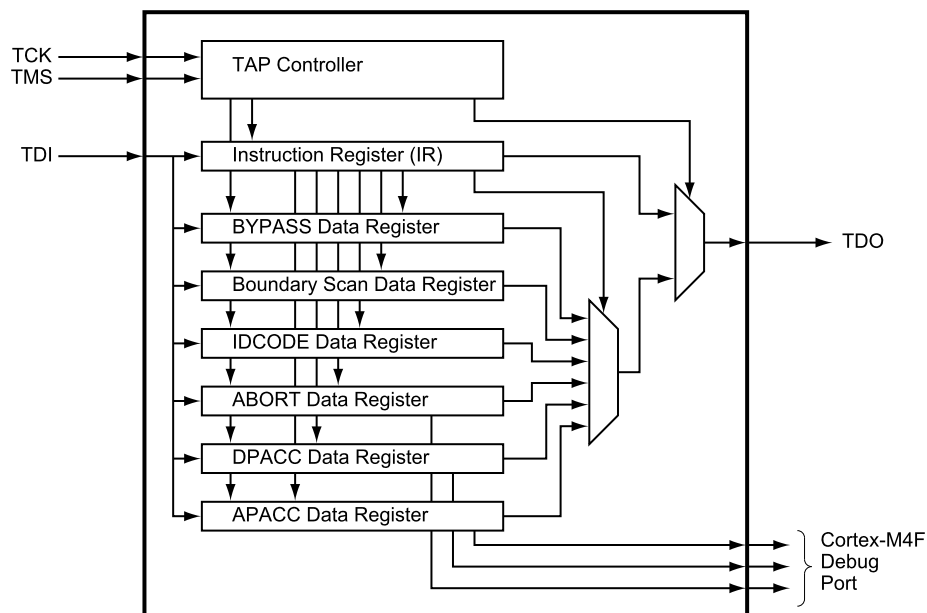


Figure 3-1. JTAG Module Block Diagram

3.3 Functional Description

Figure 3-1 shows a high-level conceptual drawing of the JTAG module. The JTAG module is composed of the TAP controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TCK and TMS inputs. The current state of the TAP controller depends on the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI toward TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the IR chain or one of the DR chains is being accessed.

The serial shift chains with parallel load registers are comprised of one IR chain and multiple DR chains. The current instruction loaded in the parallel load register determines which DR chain is captured, shifted, or updated during the sequencing of the TAP controller.

Some instructions, like EXTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see [Table 3-2](#) for a list of implemented instructions).

Depending on the reset source, the effect on the JTAG module varies. The following reset sources reset the entire JTAG module:

- Externally generated power-on reset (POR)

The following reset sources reset only the JTAG pin configuration:

- RST pin POR
- Brownout POR
- Watchdog POR
- HIB module POR
- RST pin system reset
- Brownout system reset
- Software system reset request (using the SYSRESREQ bit in the APINT register)
- Software peripheral reset
- Watchdog system reset
- HIB module system reset

3.3.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. [Table 3-1](#) lists these pins and their associated state after a power-on reset or reset caused by the RST input. Detailed information on each pin follows.

NOTE: The following pins are configured as JTAG port pins out of reset. See [Chapter 17](#) for information on how to reprogram the configuration of these pins.

Table 3-1. JTAG Port Pins State After POR or RST Assertion

Pin Name	Data Direction	Internal Pullup	Internal Pulldown	Drive Strength	Drive Value
TCK	Input	Enabled	Disabled	N/A	N/A
TMS	Input	Enabled	Disabled	N/A	N/A
TDI	Input	Enabled	Disabled	N/A	N/A
TDO	Output	Enabled	Disabled	2-mA driver	Hi-Z

3.3.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks and to ensure that multiple JTAG TAP controllers that are daisy-chained together can synchronously communicate serial test data between components. During normal operation, TCK is driven by a free-running clock with a nominal 50% duty cycle. When necessary, TCK can be stopped at 0 or 1 for extended periods of time. While TCK is stopped at 0 or 1, the state of the TAP controller does not change and data in the JTAG Instruction and Data Registers is not lost.

By default, the internal pullup resistor on the TCK pin is enabled after reset, assuring that no clocking occurs if the pin is not driven from an external source. The internal pullup and pulldown resistors can be turned off to save internal power as long as the TCK pin is constantly being driven by an external source (see [Section 17.5.15](#) and [Section 17.5.16](#)).

3.3.1.2 Test Mode Select (TMS)

The TMS pin selects the next state of the JTAG TAP controller. TMS is sampled on the rising edge of TCK. Depending on the current TAP state and the sampled value of TMS, the next state may be entered. Because the TMS pin is sampled on the rising edge of TCK, the IEEE Standard 1149.1 expects the value on TMS to change on the falling edge of TCK.

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in [Figure 3-2](#).

By default, the internal pullup resistor on the TMS pin is enabled after reset. Changes to the pullup resistor settings on GPIO Port C should ensure that the internal pullup resistor remains enabled on PC1/TMS; otherwise JTAG communication could be lost (see [Section 17.5.15](#)).

3.3.1.3 Test Data Input (TDI)

The TDI pin provides a stream of serial information to the IR chain and the DR chains. TDI is sampled on the rising edge of TCK and, depending on the current TAP state and the current instruction, may present this data to the proper shift register chain. Because the TDI pin is sampled on the rising edge of TCK, the IEEE Standard 1149.1 expects the value on TDI to change on the falling edge of TCK.

By default, the internal pullup resistor on the TDI pin is enabled after reset. Changes to the pullup resistor settings on GPIO Port C should ensure that the internal pullup resistor remains enabled on PC2/TDI; otherwise JTAG communication could be lost (see [Section 17.5.15](#)).

3.3.1.4 Test Data Output (TDO)

The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the IEEE Standard 1149.1 expects the value on TDO to change on the falling edge of TCK.

By default, the internal pullup resistor on the TDO pin is enabled after reset, assuring that the pin remains at a constant logic level when the JTAG port is not being used. The internal pullup and pulldown resistors can be turned off to save internal power if a Hi-Z output value is acceptable during certain TAP controller states (see [Section 17.5.15](#) and [Section 17.5.16](#)).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

3.3.2 JTAG TAP Controller

Figure 3-2 shows the JTAG TAP controller state machine. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). To reset the JTAG module after the microcontroller has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, see IEEE Standard 1149.1.

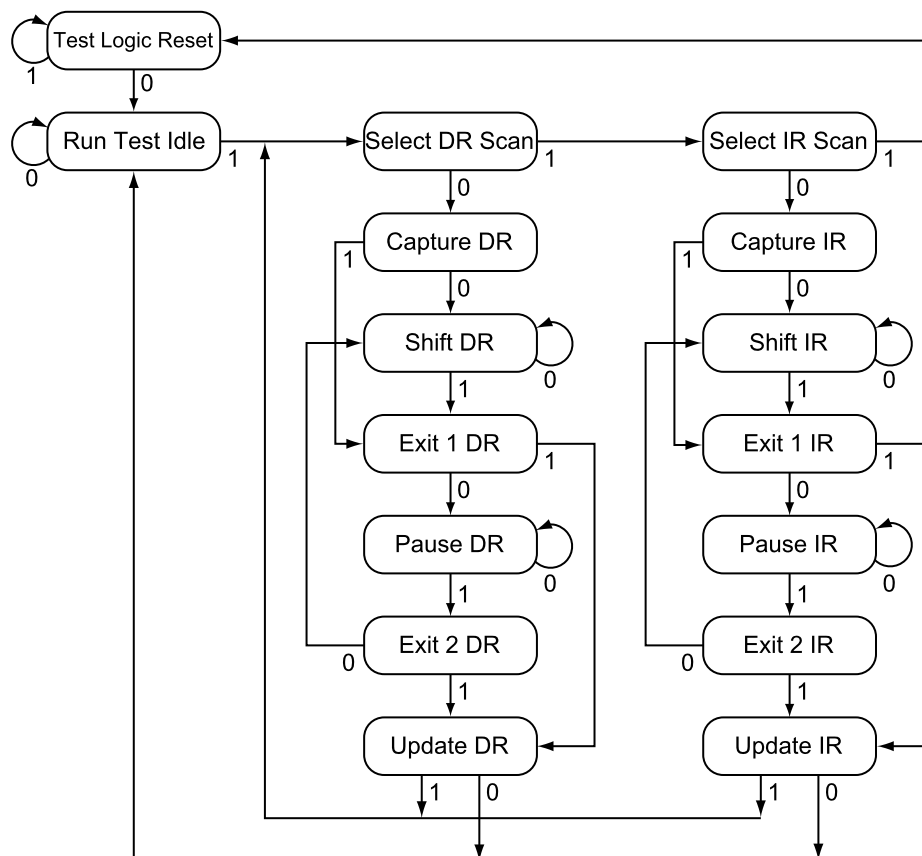


Figure 3-2. Test Access Port State Machine

3.3.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out on TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in [Section 3.5](#).

3.3.4 Operational Considerations

Certain operational parameters must be considered when using the JTAG module. Because the JTAG pins can be programmed to be GPIOs, board configuration and reset conditions on these pins must be considered. In addition, because the JTAG module has integrated Arm Serial Wire Debug, the method for switching between these two operational modes is described below.

3.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or RST, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (DEN[3:0] set in the Port C GPIO Digital Enable (GPIODEN) register), enabling the pullup resistors (PUE[3:0] set in the Port C GPIO Pullup Select (GPIOPUR) register), disabling the pulldown resistors (PDE[3:0] cleared in the Port C GPIO Pulldown Select (GPIOPDR) register) and enabling the alternate hardware function (AFSEL[3:0] set in the Port C GPIO Alternate Function Select (GPIOAFSEL) register) on the JTAG/SWD pins. See [Section 17.5.10](#), [Section 17.5.15](#), [Section 17.5.16](#), and [Section 17.5.18](#).

It is possible for software to configure these pins as GPIOs after reset by clearing AFSEL[3:0] in the Port C GPIOAFSEL register. If the user does not require the JTAG/SWD port for debugging or board-level testing, this provides four more GPIOs for use in the design.

NOTE: It is possible to create a software sequence that prevents the debugger from connecting to the MSP432E4 microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the MSP432E4 flash programmer "Unlock" feature.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see for pin numbers). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), GPIO Pullup Select (GPIOPUR) register (see [Section 17.5.15](#)), GPIO Pulldown Select (GPIOPDR) register (see [Section 17.5.16](#)), and GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)) are not committed to storage unless the GPIO Lock (GPIOLOCK) register (see [Section 17.5.19](#)) has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register (see [Section 17.5.20](#)) have been set.

3.3.4.2 Communication With JTAG/SWD

Because the debug clock and the system clock can be running at different frequencies, care must be taken to maintain reliable communication with the JTAG/SWD interface. In the Capture-DR state, the result of the previous transaction, if any, is returned, together with a 3-bit ACK response. Software should check the ACK response to see if the previous operation has completed before initiating a new transaction. Alternatively, if the system clock is at least 8 times faster than the debug clock (TCK or SWCLK), the previous operation has enough time to complete and the ACK bits do not have to be checked.

3.3.4.3 Recovering a "Locked" Microcontroller

NOTE: Performing the following sequence restores the nonvolatile registers discussed in [Section 7.2.3.12](#) to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the nonvolatile registers being restored.

In addition, the EEPROM is erased and its wear-leveling counters are returned to factory default values when performing the sequence below.

If software configures any of the JTAG/SWD pins as GPIO and loses the ability to communicate with the debugger, there is a debug port unlock sequence that can be used to recover the microcontroller. Performing a total of ten JTAG-to-SWD and SWD-to-JTAG switch sequences while holding the microcontroller in reset mass erases the Flash memory. The debug port unlock sequence is:

1. Assert and hold the RST signal.
2. Apply power to the device.
3. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on [Section 3.3.4.4.1](#).

4. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on [Section 3.3.4.4.2](#).
5. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
6. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
7. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
8. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
9. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
10. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
11. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
12. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
13. Release the RST signal.
14. Wait 400 ms.
15. Power-cycle the microcontroller.

3.3.4.4 Arm Serial Wire Debug (SWD)

In order to seamlessly integrate the Arm Serial Wire Debug (SWD) functionality, a serial-wire debugger must be able to connect to the Cortex-M4F core without having to perform, or have any knowledge of, JTAG cycles. This integration is accomplished with a SWD preamble that is issued before the SWD session begins.

The switching preamble used to enable the SWD interface of the SWJ-DP module starts with the TAP controller in the Test-Logic-Reset state. From here, the preamble sequences the TAP controller through the following states: Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, and Test Logic Reset states.

Stepping through this sequence of the TAP state machine enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the Arm Debug Interface V5 Architecture Specification.

Because this sequence is a valid series of JTAG operations that could be issued, the Arm JTAG TAP controller is not fully compliant to the IEEE Standard 1149.1. This instance is the only one where the Arm JTAG TAP controller does not meet full compliance with the specification. Due to the low probability of this sequence occurring during normal operation of the TAP controller, it should not affect normal performance of the JTAG interface.

3.3.4.4.1 JTAG-to-SWD Switching

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send the switching preamble to the microcontroller. The 16-bit TMS/SWDIO command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first. This command can also be represented as 0xE79E when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK / SWCLK and TMS / SWDIO signals:

1. Send at least 50 TCK / SWCLK cycles with TMS / SWDIO High to ensure that both JTAG and SWD are in their reset states.
2. Send the 16-bit JTAG-to-SWD switch command, 0xE79E, on TMS/SWDIO.
3. Send at least 50 TCK / SWCLK cycles with TMS / SWDIO High to ensure that if SWJ-DP was already in SWD mode before sending the switch sequence, the SWD goes into the line reset state.

To verify that the DAP has switched to the Serial Wire Debug (SWD) operating mode, perform a SWD READID operation. The ID value can be compared against the device's known ID to verify the switch.

3.3.4.4.2 SWD-to-JTAG Switching

To switch the operating mode of the DAP from SWD to JTAG mode, the external debug hardware must send a switch command to the microcontroller. The 16-bit TMS/SWDIO command for switching to JTAG mode is defined as b1110.0111.0011.1100, transmitted LSB first. This command can also be represented as 0xE73C when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK / SWCLK and TMS / SWDIO signals:

1. Send at least 50 TCK / SWCLK cycles with TMS / SWDIO High to ensure that both JTAG and SWD are in their reset states.
2. Send the 16-bit SWD-to-JTAG switch command, 0xE73C, on TMS/SWDIO.
3. Send at least 50 TCK / SWCLK cycles with TMS / SWDIO High to ensure that if SWJ-DP was already in JTAG mode before sending the switch sequence, the JTAG goes into the Test Logic Reset state.

To verify that the DAP has switched to the JTAG operating mode, set the JTAG IR to the IDCODE instruction and shift out the DR. The DR value can be compared against the device's known IDCODE to verify the switch.

3.4 Initialization and Configuration

After a Power-On-Reset or an external reset (RST), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. To return the pins to their JTAG functions, enable the four JTAG pins (PC[3:0]) for their alternate function using the GPIOAFSEL register. In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the four JTAG pins (PC[3:0]) should be returned to their default settings.

3.5 Register Descriptions

The registers in the JTAG TAP Controller or Shift Register chains are not memory mapped and are not accessible through the on-chip Advanced Peripheral Bus (APB). Instead, the registers within the JTAG controller are all accessed serially through the TAP Controller. These registers include the Instruction Register and the six Data Registers.

3.5.1 Instruction Register (IR)

The JTAG TAP IR is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in [Table 3-2](#). A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 3-2. JTAG Instruction Register Commands

IR[3:0]	Instruction	Description
0x0	EXTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads.
0x2	SAMPLE / PRELOAD	Captures the current I/O values and shifts the sampled values out of the Boundary Scan Chain while new preload data is shifted in.
0x8	ABORT	Shifts data into the Arm Debug Port Abort Register.
0xA	DPACC	Shifts data into and out of the Arm DP Access Register.
0xB	APACC	Shifts data into and out of the Arm AC Access Register.
0xE	IDCODE	Loads manufacturing information defined by the IEEE Standard 1149.1 into the IDCODE chain and shifts it out.
0xF	BYPASS	Connects TDI to TDO through a single Shift Register chain.
All Others	Reserved	Defaults to the BYPASS instruction to ensure that TDI is always connected to TDO.

3.5.1.1 EXTEST Instruction

The EXTEST instruction is not associated with its own Data Register chain. Instead, the EXTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the EXTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the outputs and output enables are used to drive the GPIO pads rather than the signals coming from the core. With tests that drive known values out of the controller, this instruction can be used to verify connectivity. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

3.5.1.2 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction connects the Boundary Scan Data Register chain between TDI and TDO. This instruction samples the current state of the pad pins for observation and preloads new test data. Each GPIO pad has an associated input, output, and output enable signal. When the TAP controller enters the Capture DR state during this instruction, the input, output, and output-enable signals to each of the GPIO pads are captured. These samples are serially shifted out on TDO while the TAP controller is in the Shift DR state and can be used for observation or comparison in various tests.

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST instruction to drive data into or out of the controller. See [Section 3.5.2.3](#) for more information.

3.5.1.3 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the Arm DAP. Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the [Section 3.5.2.6](#) for more information.

3.5.1.4 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the Arm DAP. Shifting the proper data into this register and reading the data output from this register allows read and write access to the Arm debug and status registers. See [Section 3.5.2.5](#) for more information.

3.5.1.5 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the Arm DAP. Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See [Section 3.5.2.4](#) for more information.

3.5.1.6 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the Arm core. This information can be used by testing equipment and debuggers to automatically configure input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See [Section 3.5.2.1](#) for more information.

3.5.1.7 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See [Section 3.5.2.2](#) for more information.

3.5.2 Data Registers

The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

3.5.2.1 IDCODE Data Register

[Figure 3-3](#) shows the format for the 32-bit IDCODE Data Register defined by the IEEE Standard 1149.1. The standard requires that every JTAG-compliant microcontroller implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

The major uses of the JTAG port are for manufacturer testing of component assembly and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x4BA0.0477. This value allows the debuggers to automatically configure themselves to work correctly with the Cortex-M4F during debug.

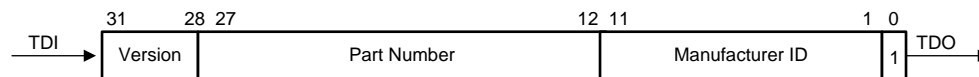


Figure 3-3. IDCODE Register Format

3.5.2.2 BYPASS Data Register

[Figure 3-4](#) shows the format for the 1-bit BYPASS Data Register defined by the IEEE Standard 1149.1. The standard requires that every JTAG-compliant microcontroller implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

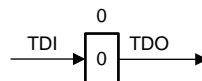


Figure 3-4. BYPASS Register Format

3.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in [Figure 3-5](#). Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure.

When the Boundary Scan Data Register is accessed with the SAMPLE/PRELOAD instruction, the input, output, and output enable from each digital pad are sampled and then shifted out of the chain to be verified. The sampling of these values occurs on the rising edge of TCK in the Capture DR state of the TAP controller. While the sampled data is being shifted out of the Boundary Scan chain in the Shift DR state of the TAP controller, new data can be preloaded into the chain for use with the EXTEST instruction. The EXTEST instruction forces data out of the controller.

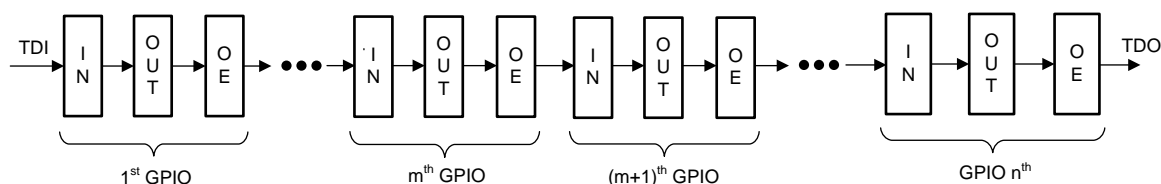


Figure 3-5. Boundary Scan Register Format

3.5.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by Arm is described in the Arm Debug Interface V5 Architecture Specification.

3.5.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by Arm is described in the Arm Debug Interface V5 Architecture Specification.

3.5.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by Arm is described in the Arm Debug Interface V5 Architecture Specification.

System Control

System control configures the overall operation of the device and provides information about the device. Configurable features include reset control, NMI operation, power control, clock control, and low-power modes.

Topic	Page
4.1 Functional Description	195
4.2 System Control Registers	217
4.3 Cryptographic System Control (CCM) Registers	466

4.1 Functional Description

The System Control module provides the following capabilities:

- Device identification (see [Section 4.1.1](#))
- Configurable control of reset, power, and clock sources
- System control (run, sleep, and deep-sleep modes) (see [Section 4.1.6](#))

4.1.1 Device Identification

Read-only registers in the system control module provide information about the microcontroller, such as version, part number, pin count, operating temperature range and available peripherals on the device. The Device Identification 0 (see [DID0](#)) and Device Identification 1 (see [DID1](#)) registers provide details about the device version, package, temperature range, and so on. The Peripheral Present registers starting at system control offset 0x300, such as the Watchdog Timer Peripheral Present (PPWD) register, provide information on how many of each type of module are included on the device. Finally, information about the capabilities of the on-chip peripherals are provided at offset 0xFC0 in the register space of each peripheral in the Peripheral Properties registers, such as the GPTM Peripheral Properties (GPTMPP). In addition, four unique identifier registers, Unique Identifier n (UNIQUEIDn), provide a 128-bit unique identifier for each device that cannot be modified.

4.1.2 Reset Control

This section discusses aspects of hardware functions during reset as well as system software requirements following the reset sequence.

4.1.2.1 Reset Sources

The MSP432E4 microcontroller has the following reset sources:

1. Power-on reset (POR) (see [Section 4.1.2.3](#))
2. External reset input pin (RST) assertion (see [Section 4.1.2.4](#))
3. A brownout detection of V_{DDA} (analog voltage source) or V_{DD} (external voltage source) dropping below its acceptable operating range (see [Section 4.1.2.5](#))
4. Software-initiated reset (with the software reset registers) (see [Section 4.1.2.6](#))
5. A watchdog timer reset condition violation (see [Section 4.1.2.7](#))
6. Hibernation module event
7. A software restart initiated through a Hardware System Service Request (HSSR)
8. MOSC failure (see [Section 4.1.3.2](#))

[Table 4-1](#) provides a summary of results of the various reset operations. The external RST pin, the brownout detection unit, the HIB module, and watchdog timer can all be programmed to generate either a power-on reset (POR) or system reset depending on how the Reset Behavior Control (RESBEHAVCTL) register at offset 0x1D8 is programmed.

Table 4-1. Reset Sources

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
Externally generated POR	Yes	Yes	Yes
RST pin POR	Yes	Pin configuration only	Yes
RST pin system reset	Yes	Pin configuration only	Yes
Brownout POR	Yes	Pin configuration only	Yes
Brownout system reset	Yes	Pin configuration only	Yes
Software system reset request using the SYSRESREQ bit in the APINT register	Yes	Pin configuration only	Yes
Software system reset request using the VECTRESET bit in the APINT register	Yes	No	No

Table 4-1. Reset Sources (continued)

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
Software peripheral reset	No	Pin configuration only	Yes ⁽¹⁾
Watchdog POR	Yes	Pin configuration only	Yes
Watchdog system reset	Yes	Pin configuration only	Yes
HIB module POR	Yes	Pin configuration only	Yes
HIB module system reset	Yes	Pin configuration only	Yes
HSSR reset	Yes	Pin configuration only	Yes
MOSC failure reset	Yes	Pin configuration only	Yes

⁽¹⁾ Programmable on a module-by-module basis by using the individual peripheral Software Reset registers starting at system control offset 0x500.

After a reset, the Reset Cause (RESC) register is set with the reset cause. The bits in this register are sticky and maintain their state across multiple reset sequences. A bit in the RESC register can be cleared by writing 0.

4.1.2.2 Boot Configuration

After POR and device initialization occurs, the hardware loads the stack pointer from either flash or ROM based on the presence of an application in flash and the state of the EN bit in the BOOTCFG register. If the flash address 0x0000.0004 contains an erased word (value 0xFFFF.FFFF) or the EN bit is of the BOOTCFG register is clear, the stack pointer and reset vector pointer are loaded from ROM at address 0x0100.0000 and 0x0100.0004, respectively. The bootloader executes and configures the available boot slave interfaces and waits for a programmer, host PC or boot server to load its software. The bootloader uses a simple packet interface to provide synchronous communication with the device for I²C, SSI, and UART. The speed of the bootloader is determined by the frequency of the internal oscillator (PIOSC) or external crystal (if connected).

If the flash at address 0x0000.0004 contains a valid reset vector value and the BOOTCFG register does not indicate the bootloader, the boot sequence causes the stack pointer and reset vector fetch from flash. This application stack pointer and reset vector is loaded and the processor executes the application directly.

NOTE: If the device fails the initialization phase, it toggles the TDO output pin to indicate that the device is not executing. This feature is provided for debug purposes.

4.1.2.3 Externally Generated POR

NOTE: The JTAG controller can be reset by a POR or by holding the TMS pin high for 5 clock cycles.

During an externally generated POR, the internal POR circuit monitors the power supply voltage (V_{DD}) and generates a reset signal to all of the internal logic including JTAG when the power supply ramp reaches a threshold value (V_{POR}). Reset does not complete if specific voltage parameters are not met as defined in the device-specific data sheet. For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the RST input may be used as discussed in [Section 4.1.2.4](#). Holding this pin active can keep the initialization process from starting even though a POR has occurred. This is useful for in-circuit testing and other situations where it is desirable to delay the operation of the device until an external supervisor has released.

The POR sequence is:

1. The microcontroller waits for internal POR to go inactive.
2. The internal reset is released and the core executes a full initialization of the device.
3. When initialization is complete, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.

4. Execution begins.

The internal POR is active only on the initial power-up of the microcontroller, when the microcontroller wakes from hibernation, and when the VDD supply drops below the its defined operating limit. See the device-specific data sheet for information on exact values.

4.1.2.4 External Reset Pin

When the external reset pin (RST) is asserted, it initiates a system reset or POR depending on what has been configured in the Reset Behavior Control (RESBEHAVCTL) Register. If the EXTRES bit field in RESBEHAVCTL is set to 0x3, a simulated full initialization begins when RST is asserted. If these bits are programmed to 0x2, a system reset is issued. When EXTRES is set to a 0x0 or 0x1, the external RST pin performs its default operation when it is asserted, which is issuing a full simulated POR.

An external reset pin (RST) that is configured to generate a POR resets the microcontroller including the core and all the on-chip peripherals. The external reset sequence is:

1. The external RST pin is asserted for the duration specified by t_{MIN} and then deasserted (see the *Specifications* chapter in the device-specific data sheet). This generates an internal POR signal.
2. The microcontroller waits for internal POR to go inactive.
3. The internal reset is released and the core executes a full initialization of the device.
4. When initialization is complete, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter
5. Execution begins.

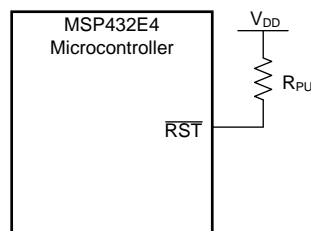
An external RST pin that is configured to generate a system reset will reset the microcontroller including the core and all the on-chip peripherals. The external reset sequence is:

1. The external reset pin (RST) is asserted for the duration specified by t_{MIN} and then deasserted (see the *Specifications* section in the device-specific data sheet).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.
3. Execution begins.

NOTE: Make the trace for the RST signal as short as possible. Place any components connected to the RST signal as close as possible to the microcontroller.

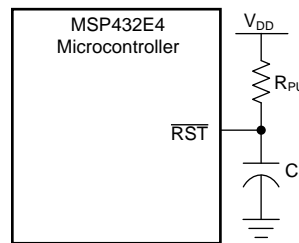
If the application only uses the internal POR circuit, the RST input must be connected to the power supply (V_{DD}) through an optional pullup resistor (0 to 100 k Ω) (see [Figure 4-1](#)). The RST input has filtering that requires a minimum pulse duration for the reset pulse to be recognized (see the device-specific data sheet).

To improve noise immunity or to delay reset at power up, the RST input can be connected to an RC network (see [Figure 4-2](#)). If the application requires the use of an external reset switch, [Figure 4-3](#) shows the proper circuitry. In the figures, the R_{PU} and C_1 components define the power-on delay.



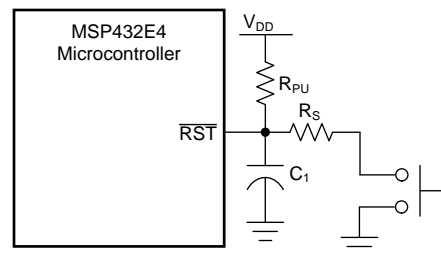
NOTE: $R_{\text{PU}} = 0$ to 100 k Ω

Figure 4-1. Basic RST Configuration



NOTE: $R_{PU} = 1\text{ k}\Omega$ to $100\text{ k}\Omega$, $C_1 = 1\text{ nF}$ to $10\text{ }\mu\text{F}$

Figure 4-2. External Circuitry to Extend POR



NOTE: Typical $R_{PU} = 10\text{ k}\Omega$, typical $R_S = 470\text{ }\Omega$, $C_1 = 10\text{ nF}$

Figure 4-3. Reset Circuit Controlled by Switch

4.1.2.5 Brownout Reset (BOR)

The microcontroller provides a brownout detection circuit that triggers if the V_{DD} (external) or V_{DDA} (analog) power supply drops below its corresponding brownout threshold voltage. If a brownout condition is detected, the system may generate an interrupt, a system reset or a POR. The default value at reset is to generate an interrupt.

The application can identify the type of BOR event that occurred by reading the Power-Temperature Cause (PWRTC) register. The BOR detection circuits can be programmed to generate a reset, System Control interrupt, or NMI in the Power-Temp Brownout Control (PTBOCTL) register. The default settings at reset are:

- V_{DDA} under BOR detection default setting is for no action to occur.
- V_{DD} under BOR detection default setting is to execute a full POR.

If the user has programmed a field in the PTBOCTL register to generate a reset, then the BOR bit of the RESBEHAVCTL register can be programmed to further define what type of reset is generated:

- If the BOR field is set to 0x3, a full POR is initiated.
- If the BOR field is set to 0x2, then a system reset is issued.
- If the BOR field is set to 0x0 or 0x1, then the brownout detection circuit performs its default operation upon assertion, which is issuing an interrupt.

NOTE: V_{DDA} BOR and V_{DD} BOR events are a combined BOR to the system logic, such that if either BOR event occurs, the following bits are affected:

- BORRIS bit in the RIS register (see [Section 4.2.4](#)).
- BORMIS bit in the MISC register (see [Section 4.2.6](#)). This bit is set only if the BORIM bit in the Interrupt Mask Control (IMC) register has been set (see [Section 4.2.5](#)).
- BOR bit in the RESC register (see [Section 4.2.7](#)). This bit is set only if either of the BOR events have been configured to initiate a reset.

In addition, the following bits control both BOR events:

- BORIM bit in the IMC register (see [Section 4.2.5](#)).
- VDDA_UBOR0 and VDD_UBOR0 bits in the PWRTC register (see [Section 4.2.8](#)).

The brownout POR reset sequence is:

1. When one of the BOR event triggers occurs, an internal BOR condition is set.
2. If the BOR event has been programmed to generate a reset in the PTBOCTL register and the BOR bit of the RESBEHAVCTL has been set to 0x3, an internal POR reset is asserted.
3. The internal reset is released and the core executes a full initialization of the device. Upon completion, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution. The application starts after deassertion of internal POR. See the device-specific data sheet for BOR internal reset deassertion timing.

The brownout system reset sequence is:

1. When one of the BOR event triggers occurs, an internal BOR condition is set.
2. If the BOR event has been programmed to generate a reset in the PTBOCTL register and the BOR bit of the RESBEHAVCTL has been set to 0x2, an internal reset is asserted.
3. The internal reset is released and the microcontroller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter
4. Execution begins.

The result of a BOR is equivalent to that of an assertion of the external RST input, and the reset is held active until the proper voltage level is restored. The RESC register can be read in the reset interrupt handler to determine if a brownout condition was the cause of the reset, thus allowing software to determine which actions are required to recover.

4.1.2.6 Software Reset

Software can reset a specific peripheral or generate a reset to the entire microcontroller.

Peripherals can be individually reset by software through peripheral-specific reset registers available beginning at system control offset 0x500 [for example, the Watchdog Timer Software Reset (SRWD) register; see [Section 4.2.64](#)]. If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset.

The entire microcontroller, including the core, can be reset by setting the SYSRESREQ bit in the Application Interrupt and Reset Control (APINT) register in the core peripheral memory map space. The software-initiated system reset sequence is:

1. A software microcontroller reset is initiated by setting the SYSRESREQ bit.
2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.
4. Execution begins.

The core only can be reset by setting the VECTRESET bit in the APINT register. The software-initiated core reset sequence is:

1. A core reset is initiated by setting the VECTRESET bit.
2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.
4. Execution begins.

4.1.2.7 Watchdog Timer Reset

The function of the watchdog timer module is to prevent system hangs. The MSP432E4 microcontroller has two watchdog timer modules in case one watchdog clock source fails. One watchdog is run off the system clock, and the other is run off the precision internal oscillator (PIOSC). The watchdog timer can be configured to generate an interrupt or a nonmaskable interrupt to the microcontroller on its first time-out and to generate a system reset or POR on its second time-out.

After the first time-out event of the watchdog, the 32-bit watchdog counter is reloaded with the value of the Watchdog Timer Load (WDTLOAD) register and resumes counting down from that value. If the timer counts down to zero again before the first time-out interrupt is cleared, and watchdog reset generation has been enabled through the RESEN bit in the Watchdog Control (WDTCTL) register, the watchdog timer asserts its reset signal to the microcontroller. The reset generated can be a full POR or a system reset depending on the value programmed in WDOGN bit field of the RESBEHAVCTL register:

- If the RESEN bit of the WDTCTL register is 1 and the WDOGN bit field of the RESBEHAVCTL register is 0x3, a full POR is initiated.
- If WDOGN is 0x2, a system reset is issued.
- If WDOGN is 0x0 or 0x1, the watchdog timer performs its default operation upon assertion, which is issuing a full POR.

The watchdog timer POR sequence is:

1. The watchdog timer times out for the second time without being serviced.
2. An internal POR reset is asserted.
3. The internal reset is released and the core executes a full initialization of the device. Upon completion, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.
4. Execution begins.

See the device-specific data sheet for watchdog time-out internal reset deassertion timing.

The watchdog timer system reset sequence is:

1. The watchdog timer times out for the second time without being serviced.
2. An internal reset is asserted.
3. The internal reset is released and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.
4. Execution begins.

For more information on the Watchdog Timer module, see [Chapter 28](#).

4.1.2.8 Hibernation Module Reset

When the Hibernation module has been configured and powered by an initial cold POR and is subsequently put into hibernation mode, a wake event (not including an external reset pin wake) causes the module to generate a system reset. This reset signal resets all circuitry on the device with the exception of the Hibernation module. All registers of the Hibernation module retain their values after this reset.

When the Hibernation module receives a wake event and V_{DD} is enabled, a system reset sequence occurs:

1. The POR or EXT bit in the RESC register is set.
2. An internal reset is asserted.
3. The internal reset is released and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter.
4. Execution begins.
5. The HIBRIS register in the Hibernation module can be read to determine the cause of the reset.
6. The POR or EXT bit in the RESC register is cleared by writing 0.

4.1.2.9 HSSR Reset

The Hardware System Service Request (HSSR) register can be used to restore the device back to factory settings. A successful write to the HSSR register initiates a system reset. The reset initialization process executes before examining the HSSR register and processing the command. The HSSR register can be accessed only in privileged mode.

Before the return-to-factory settings routine has completed, a system reset sequence executes and the HSSR bit in the RESC register is set. After the HSSR function has been processed, the CDOFF field in the HSSR register is written with the outcome of the function processing and another HSSR system reset is executed. The HSSR bit can be cleared in the RESC register by writing 0.

For more information regarding use of the HSSR register, see [Section 4.1.6.6](#).

4.1.3 Nonmaskable Interrupt

The microcontroller has multiple sources of nonmaskable interrupt (NMI):

- The assertion of the NMI signal.
- A main oscillator verification error.
- The NMISSET bit in the Interrupt Control and State (INTCTRL) register in the Cortex-M4F (see [Section 2.5.3](#)).
- The Watchdog module time-out interrupt when the INTTYPE bit in the WDTCTL register is set (see [Section 28.5.3](#)).
- Tamper event (see [Chapter 6](#) for more information).
- Any of the following BOR trigger events:
 - V_{DDA} under BOR setting
 - V_{DD} under BOR setting

Software must read the cause of the interrupt in the NMI Cause (NMIC) register to determine the source.

4.1.3.1 NMI Pin

The NMI signal is an alternate function for the GPIO port pins specified in the device-specific data sheet.

The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in [Chapter 17](#). Enabling the NMI alternate function requires the use of the GPIO lock and commit function, similar to the requirements of the GPIO port pins associated with JTAG/SWD functionality (see [Section 17.5.20](#)). The active sense of the NMI signal is high; asserting the enabled NMI signal above V_{IH} initiates the NMI interrupt sequence.

4.1.3.2 Main Oscillator Verification Failure

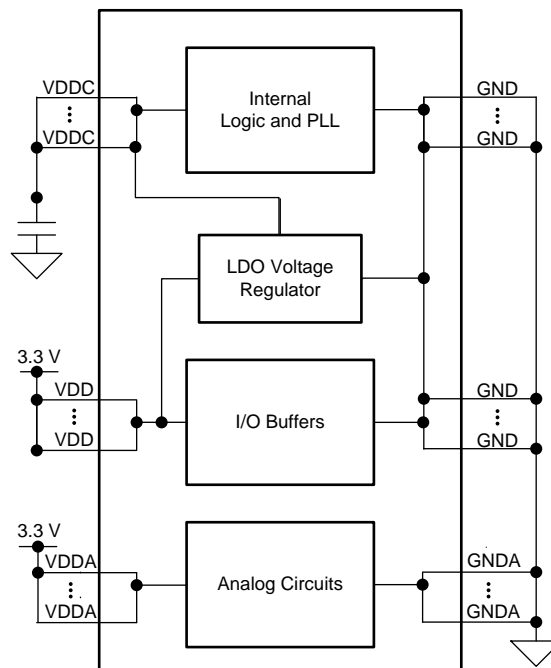
The MSP432E4 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. If the main oscillator verification circuit is enabled and a failure occurs, either a POR is generated and control is transferred to the NMI handler, or an interrupt is generated. The MOSCIM bit in the MOSCCTL register determines which action occurs. In either case, the system clock source is automatically switched to the PIOSC. If a MOSC failure reset occurs, the NMI handler is used to address the main oscillator verification failure, because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the CVAL bit in the Main Oscillator Control (MOSCCTL) register. The main oscillator verification error is indicated in the main oscillator fail status (MOSCFAIL) bit in the RESC register. The main oscillator verification circuit action is described in more detail in [Section 4.1.5.4.1](#).

4.1.4 Power Control

An integrated LDO regulator provides power to most of the internal logic of the microcontroller. [Figure 4-4](#) shows the power architecture. The voltage output has a maximum voltage of 1.2 V. See [Section 4.1.6.4](#) for more information on the LDO operation.

An external LDO may not be used.

NOTE: VDDA must be supplied with 3.3 V, or the microcontroller does not function properly. VDDA is the supply for all of the analog circuitry on the device, including the clock circuitry.



A The VDDA voltage source is typically connected to a filtered voltage source or regulator.

Figure 4-4. Power Architecture

4.1.5 Clock Control

The system control module determines the control of clocks in this part.

4.1.5.1 Fundamental Clock Sources

Multiple clock sources are available for use in the microcontroller. The Run and Sleep Mode Configuration (RSCLKCFG) register can be used to configure the required clock source for the device after POR, as well as the system clock divisor encodings. The available clock sources are:

- Precision Internal Oscillator (PIOSC)

The PIOSC is an on-chip clock source that the microcontroller uses during and following POR. The PIOSC is the clock source in effect at the start of reset vector fetch and the start of code application execution. The PIOSC does not require the use of any external components and provides a clock that is $16 \text{ MHz} \pm f_{\text{PIOSC}}$ across temperature (see the PIOSC specifications in the device-specific data sheet). The PIOSC allows for a reduced system cost for applications in which a high-precision clock source is not required. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference. If the Hibernation module clock source is a 32.768-kHz oscillator, the PIOSC can be trimmed by software based on a reference clock for increased accuracy. Regardless of whether or not the PIOSC is the source for the system clock, the PIOSC can be configured to be an alternate clock source for some of the peripherals. See [Section 4.1.5.2.1](#) for more information on peripherals that can use the PIOSC as an alternate clock.

- Main Oscillator (MOSC)

The MOSC provides a frequency-accurate clock sourced by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the frequency of the crystal can be 5 MHz to 25 MHz (inclusive). See [Table 4-6](#) for recommended crystal values and PLL register programming. If the PLL is not being used, the frequency of the crystal can be 4 MHz to 25 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller.

- Low-Frequency Internal Oscillator (LFIOSC)

The LFIOSC provides a nominal frequency of 33 kHz with (see the device-specific data sheet for specifications). The LFIOSC is intended for use during deep-sleep power-saving modes. This power-savings mode provides reduced internal switching and the ability to power down the MOSC or PIOSC while in deep-sleep mode through configuration of the Deep Sleep Clock Configuration (DSCLKCFG) register.

- Hibernation Module RTC Oscillator (RTCOSC) Clock Source

The Hibernation module provides a multiplexed output of two clocks to the System Control module: an external 32.768-kHz clock or a low-frequency clock. The Hibernation module can be clocked by a 32.768-kHz oscillator connected to the XOSC0 pin. The 32.768-kHz oscillator can be used for the system clock, thus eliminating the need for an additional crystal or oscillator. Alternatively, the Hibernation module contains a low-frequency oscillator (HIB LFIOSC), which is intended to provide the system with a real-time clock source and may also provide an accurate source of deep-sleep or hibernate mode power savings. The HIB LFIOSC is a different clock source than the LFIOSC. See the device-specific data sheet for more information on frequency range.

The internal system clock (SysClk), is derived from any of the preceding sources. An internal PLL can also be used by the PIOSC or MOSC clock to generate the system clock and peripheral clocks. [Table 4-2](#) shows how the various clock sources can be used in a system.

Table 4-2. Clock Source Options

Clock Source	Drive PLL Capability?	PLL Enabled, RSCLKCFG Bit Encodings	SysClk Generation Capability?	SysClk Generation Enabled, RSCLKCFG Bit Encodings
PIOSC	Yes	USEPLL = 1, PLLSRC = 0x0	Yes	USEPLL = 0, OSCSRC = 0x0
MOSC	Yes	USEPLL = 1, PLLSRC = 0x3	Yes	USEPLL = 0, OSCSRC = 0x3
LFIOSC ⁽¹⁾	No	—	Yes	USEPLL = 0, OSCSRC = 0x2
RTCOSC (32.768-kHz oscillator or HIB LFIOSC)	No	—	Yes	USEPLL = 0, OSCSRC = 0x4

⁽¹⁾ LFIOSC frequency is characterized as 33 kHz nominal, 10 kHz minimum, and 90 kHz maximum.

4.1.5.2 Clock Configuration

The Run and Sleep Mode Configuration (RSCLKCFG) register provides control for the system clock in run and sleep mode. The DSCLKCFG register specifies the behavior of the clock system while in deep-sleep mode. These registers control the following clock functionality:

- Source of system clock in run and sleep mode
- Source of system clock in deep-sleep mode
- Enabling and disabling of PLL-derived system clock
- Clock divisors for PLL or oscillator, depending on what is enabled
- Enabling of memory timing parameters for flash

Providing further configuration, the PLL Frequency n (PLLFREQn) registers allow multiplication or division of the PLL VCO frequency (f_{VCO}) by programmable values, depending on the system clock speed required.

[Table 4-3](#) lists the state of the clock sources following a POR.

Table 4-3. Clock Source State Following POR

Clock Source	POR State
PLL	Disabled or powered off
MOSC	Disabled or powered off
LFIOSC	Enabled
PIOSC	Enabled
HIB RTCOSC	Disabled

[Figure 4-5](#) shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled or disabled.

NOTE: The clock sources in [Figure 4-5](#) include a superset of peripherals available in the family. Some peripheral clock sources may not be present on your specific device.

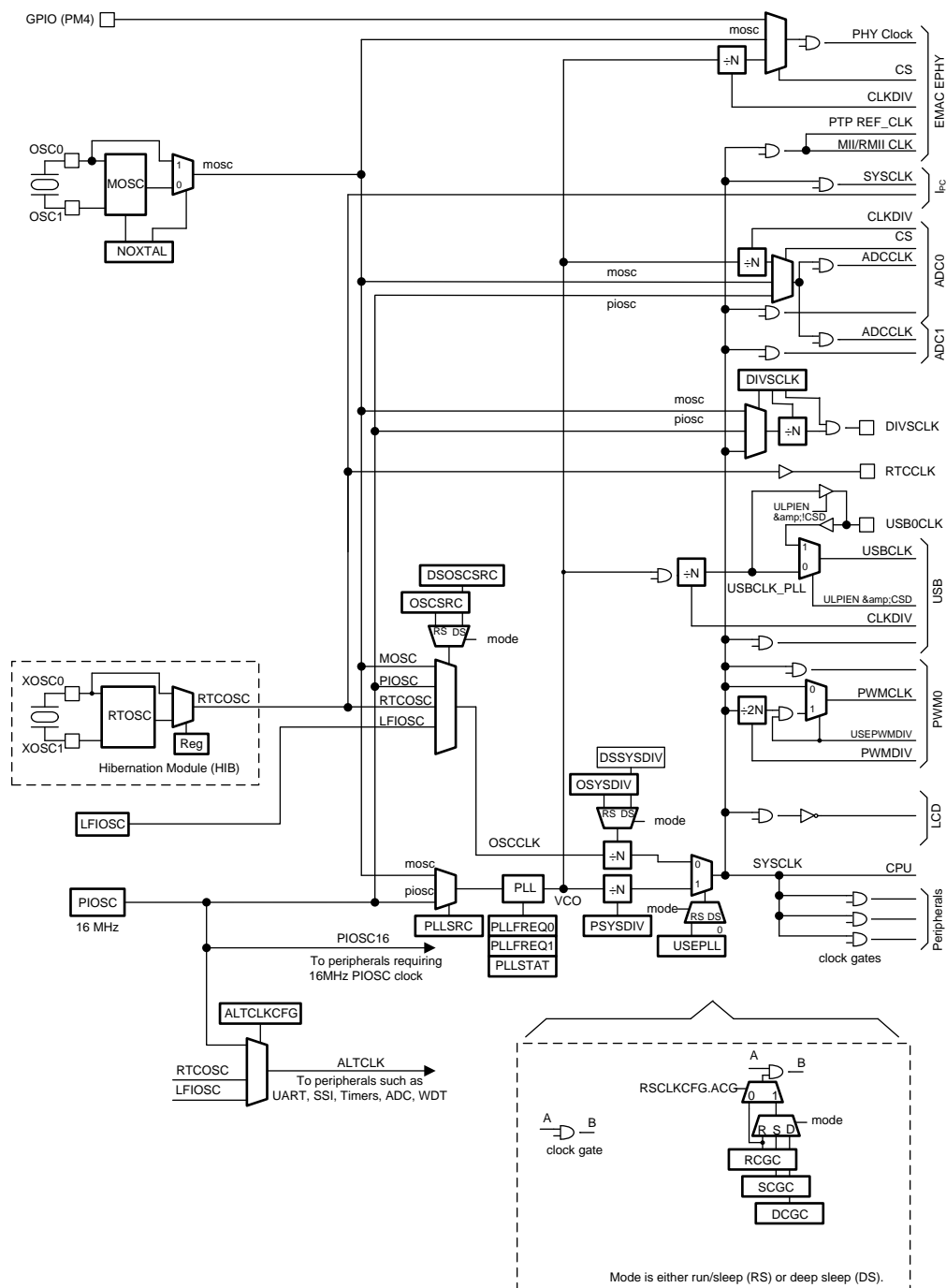


Figure 4-5. Main Clock Tree

4.1.5.2.1 Peripheral Clock Sources

In addition to the main clock tree shown in [Figure 4-5](#), the ADC, USB, Ethernet, PWM, UART, and QSSI all have a clock control register in their register map at offset 0xFC8 that can be used to control the clock generation for the module.

4.1.5.2.1.1 ADC Clock Control

The ADC digital block is clocked by the system clock and the ADC analog block is clocked from a separate conversion clock (ADC clock). The ADC clock frequency can be up to 32 MHz to generate a conversion rate of 2 Msps. A 16-MHz ADC clock provides a 1-Msps sampling rate. There are three sources for the ADC clock:

- The PLL VCO (f_{VCO}) can be used if the CS bit field is 0x0 in the ADC Clock Configuration (ADCCC) register and the CLKDIV bit field is configured in the same register.
- The PIOSC can be used directly to provide a conversion rate near 1 Msps. To use the PIOSC, the CS field in the ADCCC register must be 0x1, and the ALTCLK field in the Alternate Clock Configuration (ALTCLKCFG) register must be 0x0.
- The MOSC clock source must be 16 MHz for a 1-Msps conversion rate and 32 MHz for a 2-Msps conversion rate.

NOTE: If the ADC module is not using the PIOSC as the clock source, the system clock must be at least 16 MHz.

4.1.5.2.1.2 USB Clock Control

When the USB module uses the integrated USB PHY, the MOSC must be the clock source, either with or without the PLL, and the system clock must be at least 30 MHz. In addition, only integer divisors should be used to achieve the 60-MHz USB clock source. Fractional divisors may increase jitter and compromise USB function. Program the CLKDIV bit field in the USB Clock Control (USBCC) register to specify the divisor used to reduce the PLL VCO output to the 60-MHz clock source required for the serialization and deserialization module of the USB controller.

In ULPI mode, if the clock source to the USB is internal, the USB0CLK pin is an output to the external ULPI PHY. If the USB clock source is external, the USB0CLK pin functions as an input from the external ULPI PHY.

4.1.5.2.1.3 Ethernet Clock Control

Available clock sources are dependent on the interface chosen. The following sections describe the clock control for the various interfaces.

The Ethernet Controller module and integrated PHY receive two clock inputs. A gated system clock acts as the clock source to the CSRs of the Ethernet MAC and must be 20 MHz or greater for correct operation. The SYSCLK frequency for run, sleep, and deep-sleep modes is programmed in the System Control module. See [Section 15.3.1](#) for more information.

4.1.5.2.1.3.1 PHY Interface Clocking

The Ethernet Controller module and Integrated PHY receive two clock inputs (see [Section 15.3.1.1](#) for more information):

- A gated system clock acts as the clock source to the CSRs of the Ethernet MAC. The SysClk frequency for run, sleep, and deep-sleep modes is programmed in the System Control module.
- The PHY receives the MOSC, which must be 25 MHz \pm 50 ppm for proper operation. The MOSC source can be a single-ended source or a crystal.

4.1.5.2.1.3.2 MII Interface Clocking

Four clock inputs are driven into the Ethernet MAC when the MII configuration is enabled. The clocks are described as follows (see [Section 15.3.1.2](#) for more information):

- **Gated system clock (SysClk):** The SysClk signal acts as the clock source to the CSRs of the Ethernet MAC. The SysClk frequency for run, sleep, and deep-sleep modes is programmed in the System Control module.
- **MOSC:** A gated version of the MOSC clock is provided as the Precision Time Protocol (PTP) reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the PTP module has been enabled by setting the PTPCEN bit in the EMACCC register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. See [Section 15.3.6](#) for more information.
- **EN0RXCK:** This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz depending on whether the device is operating at 10 or 100 Mbps.
- **EN0TXCK:** This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz depending on whether the device is operating at 10 or 100 Mbps.

4.1.5.2.1.3.3 RMII Interface Clocking

Three clock sources interface to the Ethernet MAC in an RMII configuration (see [Section 15.3.1.3](#) for more information):

- **Gated system clock (SysClk):** The SysClk signal acts as the clock source to the CSRs of the Ethernet MAC. The SysClk frequency for run, sleep, and deep-sleep modes is programmed in the System Control module.
- **MOSC:** A gated version of the MOSC clock is provided as the PTP reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the PTP module has been enabled by setting the PTPCEN bit in the EMACCC register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. See [Section 15.3.6](#) for more information.
- **EN0REF_CLK:** When using RMII, a 50-MHz external reference clock must drive the reference clock input signal (EN0REF_CLK) and the external PHY. Depending on the configuration of the FES bit in the Ethernet MAC Configuration (EMACCFG) register, the EN0REF_CLK is divided by 20 for 10-Mbps operation or by 2 for 100-Mbps operation and is used as the clock for receive and transmit data.

4.1.5.2.1.4 PWM Clock Control

The PWMCC register can be used to select the system clock as the PWM clock source or a divided system clock. For more information, see [Section 21.5.32](#).

4.1.5.2.1.5 Other Peripheral Clock Control

In the UART and QSSI Clock Control registers, users can choose between the system clock (SysClk), which is the default source for the baud clock, and an alternate clock. There may be special considerations when configuring the baud clock.

4.1.5.2.2 Optional Clock Output Signal (DIVSCLK)

An optional clock output, DIVSCLK, can be used as a clock source to an external device but bears no timing relationship to other signals. DIVSCLK is not synchronized to the system clock. By programming the SRC field in the Divisor and Source Clock Configuration (DIVSCLK) register, the following clock outputs can be selected for DIVSCLK:

- System clock
- PIOSC
- MOSC

The DIV field in the DIVSCLK register controls the divided output clock frequency. The DIVSCLK signal is selected as an alternate function of a GPIO signal and has the electrical characteristics of a GPIO (see the device-specific data sheet).

4.1.5.2.3 System Clock (SysClk) Frequency

The SysClk is distributed to the processor and the integrated peripherals after clock gating. The SysClk frequency is based on the frequency of the clock source and the divisor factor. For example, if the PLL is not being used and the device is not in deep-sleep mode, then the OSYSDIV bit field in the RSCLKCFG register is the divisor used to determine the system clock. If the PLL is being used, then PSYSDIV bit field in the RSCLKCFG register must be programmed as well as the values in the PLLFREQ0 and PLLFREQ1 registers. If the device is in deep-sleep mode, then the DSCLKCFG register can be programmed with the divisor bit field DSSYSDIV to modify the clock source frequency. Table 4-4 lists the different system clock frequency calculations based on the operation mode, clock source, and PLL encoding.

Table 4-4. System Clock Frequency

Clock Mode	USEPLL (RSCLKCFG)	SYSCLK Value	Divisor Factors Used
Run or sleep	1	$f_{VCO} / (PSYSDIV + 1)$	PSYSDIV bit field in RSCLKCFG; MINT, MDIV in PLLFREQ0; Q, N bits in PLLFREQ1
Run or sleep	0	$f_{OSCCLK} / (OSYSDIV + 1)$	OSYSDIV bit field in RSCLKCFG
Deep sleep	PLL not enabled in deep-sleep mode	$f_{OSCCLK} / (DSSYSDIV + 1)$	DSSYSDIV bit field in DSCLKCFG

4.1.5.3 PIOSC Operation

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC must remain enabled, because it is used for internal functions. The PIOSC can only be disabled during deep-sleep mode. It can be powered down by setting the PIOSCPD bit in the DSCLKCFG register.

The PIOSC generates a 16-MHz (typical) clock. At the factory, the PIOSC is set to 16 MHz at room temperature; however, the frequency can be trimmed for other voltage or temperature conditions using software in the following ways:

- Default calibration: Clear the UTEN bit and set the UPDATE bit in the Precision Internal Oscillator Calibration (PIOSCCAL) register.
- User-defined calibration: The user can program the UT value to adjust the PIOSC frequency. As the UT value increases, the generated period increases. To commit a new UT value, first set the UTEN bit, then program the UT field, and then set the UPDATE bit. The adjustment finishes within a few clock periods and is glitch free.
- Automatic calibration using the enable 32.768-kHz oscillator from the Hibernation module: Set the CAL bit in the PIOSCCAL register; the results of the calibration are shown in the RESULT field in the Precision Internal Oscillator Statistic (PIOSCSTAT) register. When calibration is complete, the PIOSC is trimmed using the trimmed value returned in the CT field.

4.1.5.4 MOSC Operation

The MOSC supports the use of crystals with a frequency of 5 to 25 MHz. The RSCLKCFG register can be configured to specify the MOSC as the system clock or as the PLL input source. The MOSC can be selected as the oscillator source by programming the OSCRC bit in the RSCLKCFG register. The NOXTAL bit in the MOSCCTL register lets the user turn off power to the MOSC if no crystal is connected, which reduces power draw from the MOSC circuit.

4.1.5.4.1 MOSC Verification Circuit

The clock control includes circuitry to ensure that the MOSC is running at the appropriate frequency. The circuit monitors the MOSC frequency and signals if the frequency is outside of the allowable band of attached crystals.

The detection circuit is enabled using the CVAL bit in the Main Oscillator Control (MOSCCTL) register. If this circuit is enabled and detects an error, and if the MOSCIM bit in the MOSCCTL register is clear, then the following sequence is performed by the hardware:

1. The MOSCFAIL bit in the RESC register is set.
2. The system clock is switched from the main oscillator to the PIOSC.

3. An internal system reset is initiated.
4. Reset is deasserted and the processor is directed to the NMI handler during the reset sequence.

4.1.5.5 PLL

The PLL has two modes of operation: normal and power-down. The modes are programmed using the PLLPWR bit in the PLLFREQ0 register (see [Section 4.2.19](#)).

- Normal: The PLL oscillates based on the values in the PLLFREQ0 and PLLFREQ1 registers and drives the output.
- Power-down: Most of the PLL internal circuitry is disabled and the PLL does not drive the output.

4.1.5.5.1 PLL Configuration

The PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency and enables the PLL to drive the output. The PLL is controlled using the PLLFREQ0, PLLFREQ1, and PLLSTAT registers. Changes made to these registers do not become active until after the NEWFREQ bit in the RSCLKCFG register is enabled. The clock source for the main PLL is selected by configuring the PLLSRC field in the Run and Sleep Clock Configuration (RSCLKCFG) register. The PLL allows for the generation of system clock frequencies in excess of the reference clock provided. The reference clocks for the PLL are the PIOSC and the MOSC.

The PLL is controlled by two registers, PLLFREQ0 and PLLFREQ1. The PLL VCO frequency (f_{VCO}) is determined through [Equation 1](#).

$$f_{VCO} = f_{IN} \times MDIV$$

where

- $f_{IN} = f_{XTAL} / (Q+1)(N+1)$ or $f_{PIOSC} / (Q+1)(N+1)$
 - $MDIV = MINT + (MFRAC / 1024)$
- (1)

The Q and N values are programmed in the PLLFREQ1 register. To reduce jitter, program MFRAC to 0x0.

When the PLL is active, the system clock frequency (SysClk) is calculated using [Equation 2](#).

$$SysClk = f_{VCO} / (1 + 1)$$
(2)

The PLL system divisor factor (PSYSDIV) must be set as 1. [Table 4-5](#) lists examples of the system clock frequency.

Table 4-5. Examples of System Clock Frequencies

f_{VCO} (MHz)	Q	PSYSDIV + 1	System Clock (SYSCLK) Frequency (MHz)
480	2	2	120
480	3	2	80
480	4	2	60
480	5	2	48
320	2	2	80
320	3	2	53
320	4	2	40

If the MOSC provides the clock reference to the PLL, the translation provided by hardware and used to program the PLL is available for software in the PLL Frequency n (PLLFREQn) registers (see [Section 4.2.19](#)). The internal translation provides a translation within $\pm 1\%$ of the targeted PLL VCO frequency. [Table 4-6](#) shows the actual PLL frequency and error for a given crystal choice.

[Table 4-6](#) provides examples of the programming expected for the PLLFREQ0 and PLLFREQ1 registers. The Crystal Frequency column specifies the input crystal frequency, and the PLL Frequency column lists the PLL frequency given the values of MINT and N, when Q = 0.

Table 4-6. Actual PLL Frequency⁽¹⁾

Crystal Frequency (MHz)	MINT		N	Reference Frequency (MHz) ⁽²⁾	PLL Frequency (MHz)
	(Decimal)	(Hex)			
5	64	0x40	0x0	5	320
6	160	0x35	0x2	2	320
8	40	0x28	0x0	8	320
10	32	0x20	0x0	10	320
12	80	0x50	0x2	4	320
16	20	0x14	0x0	16	320
18	160	0xA0	0x8	2	320
20	16	0x10	0x0	20	320
24	40	0x28	0x2	8	320
25	64	0x40	0x4	5	320
5	96	0x60	0x0	5	480
6	80	0x50	0x0	6	480
8	60	0x3C	0x0	8	480
10	48	0x30	0x0	10	480
12	40	0x28	0x0	12	480
16	30	0x1E	0x0	16	480
18	80	0x50	0x2	6	480
20	24	0x18	0x0	20	480
24	20	0x14	0x0	24	480
25	96	0x60	0x4	5	480

⁽¹⁾ For all examples listed, Q = 0

⁽²⁾ For a given crystal frequency, N should be chosen such that the reference frequency is within 4 to 30 MHz.

4.1.5.2 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is t_{READY} (see the device-specific data sheet). During the relock time, the affected PLL is not usable as a clock reference. Software can poll the LOCK bit in the PLL Status (PLLSTAT) register to determine when the PLL has locked.

Do not modify the PLL VCO frequency while the PLL serves as a clock source to the system. All changes to the PLL must be performed using a different clock source until the PLL has locked frequency. Thus, changing the PLL VCO frequency must be done as a sequence from the PLL to PIOSC or MOSC and then PIOSC or MOSC to the new PLL.

Hardware is provided to keep the PLL from being used as a system clock until the t_{READY} condition is met after one of the previous changes. Software must ensure that the system is using a stable clock source (like the main oscillator) before the RSCLKCFG register is reprogrammed to enable the PLL. Software can use many methods to ensure that the system is clocked from the PLL, including periodically polling the PLLLRIS bit in the RIS register at offset 0x050, and enabling the PLL Lock interrupt in the IMC register at offset 0x054.

4.1.6 System Control

Four levels of operation are defined for the microcontroller:

- Run mode (see [Section 4.1.6.1](#))
- Sleep mode (see [Section 4.1.6.2](#))
- Deep-sleep mode (see [Section 4.1.6.3](#))
- Hibernation mode (see [Section 4.1.6.5](#))

For power-savings purposes, the peripheral-specific RCGCx, SCGCx, and DCGCx registers (for example, RCGCWD) control the clock-gating logic for that peripheral or block in the system while the microcontroller is in Run, Sleep, and deep-sleep mode, respectively. These registers are located in the system control register map starting at offsets 0x600, 0x700, and 0x800, respectively.

NOTE: A change in the RCGCx (or SCGCx, DCGCx, PCx, or SRx) registers may not have an immediate effect on the clock in all situations. Poll the Peripheral Ready (PRx) register to determine when a peripheral is ready to be accessed.

NOTE: If a peripheral is configured to be clock-gated during run, sleep, or deep-sleep mode, software should ensure that there are no pending transfers or register accesses before or immediately after entering the clock-gated mode.

4.1.6.1 Run Mode

In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the peripheral-specific RCGC registers. In run mode (and in sleep mode), the Run and Sleep Clock Configuration (RSCLKCFG) register specifies the source of SysClk. The source is either from the VCO output of the PLL divided down (using the Q divider) or from the output of an oscillator divided down by a dedicated divisor (divisor value specified by the OSYSDIV field). The source is selected using the USEPLL bit in the RSCLKCFG register. The PLL has two sources of reference clock as an input: the main oscillator (MOSC) or the precision internal oscillator (PIOSC). The PLL input select is specified by PLLSRC. If the PLL VCO output is not selected as the source of SysClk then the following reference clocks can be programmed as an input:

- MOSC
- PIOSC
- LFIOSC
- RTCOSC: The source of this signal can be either a 32.768-kHz oscillator source, an external 32.768-kHz clock source, or the internal Hibernation module low-frequency oscillator (HIB LFIOSC). If this clock source is selected, it must also be enabled in the Hibernation module.

These alternate sources are selected through the OSCSRC field in the RSCLKCFG register.

4.1.6.2 Sleep Mode

In sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M4F core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system returns the processor to run mode. See [Section 1.8](#) for more details.

Peripherals are clocked if enabled in the peripheral-specific SCGC registers when automatic clock gating is enabled or in the peripheral-specific RCGC registers when automatic clock gating is disabled. The system clock has the same source and frequency in sleep mode as it does during run mode.

The option to use the PLL VCO or an alternate oscillator source such as MOSC, PIOSC, Hibernation module RTC, or the LFIOSC is the same as described in [Section 4.1.6.1](#). The RSCLKCFG register programming applies to sleep mode.

Additional sleep modes are available that lower the power consumption of the SRAM and flash memory. However, the lower power consumption modes have slower sleep and wake-up times.

NOTE: If the Cortex-M4F Debug Access Port (DAP) has been enabled, and the device wakes from a low-power sleep or deep-sleep mode, the core may start executing code before all clocks to peripherals have been restored to their run mode configuration. The DAP is usually enabled by software tools accessing the JTAG or SWD interface when debugging or flash programming. If this condition occurs, a Hard Fault is triggered when software accesses a peripheral with an invalid clock.

A software delay loop can be used at the beginning of the interrupt routine that is used to wake up a system from a WFI instruction. This stalls the execution of any code that accesses a peripheral register that might cause a fault. This loop can be removed for production software, because the DAP is most likely not enabled during normal execution.

Because the DAP is disabled by default (after a POR), the user can also reset the device. The DAP is not enabled unless it is enabled through the JTAG or SWD interface.

4.1.6.3 Deep-Sleep Mode

In deep-sleep mode, the clock frequency of the active peripherals may change (depending on the deep-sleep mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-sleep mode is entered by first setting the SLEEPDEEP bit in the System Control (SYSCTRL) register and then executing a WFI instruction. Any properly configured interrupt event in the system returns the processor to run mode. See [Section 1.8](#) for more details.

NOTE: If the DAP is enabled in run mode and the device tries to transition into deep-sleep mode, the device is prevented from entering deep-sleep mode.

The Cortex-M4F processor core and the memory subsystem are not clocked in deep-sleep mode. Peripherals are clocked if enabled in the peripheral-specific DCGC registers when automatic clock gating is enabled or in the peripheral-specific RCGC registers when automatic clock gating is disabled.

The system clock source is specified in the DSCLKCFG register. When the DSCLKCFG register is used, the internal oscillator source is powered up, if necessary, and other clocks are powered down. If the PLL is running at the time of the WFI instruction, hardware shuts down the PLL for power savings. For additional power savings, the PIOSC can be disabled through the PIOSCPD bit in the DSCLKCFG register. When the deep-sleep exit event occurs, hardware returns the system clock to the source and frequency it had at the start of deep-sleep mode before enabling the clocks that had been stopped during deep-sleep mode. If the PIOSC is used as the PLL reference clock source, it may continue to provide the clock during deep-sleep mode (see [Section 4.2.14](#)).

NOTE: If the MOSC is chosen as the deep-sleep clock source in the DSCLKCFG register, the MOSC must also be configured as the run and sleep clock source in the RSCLKCFG register before entering deep-sleep mode. If the PIOSC, LFIOOSC, or Hibernation RTC module oscillator (HIBLFIOOSC or 32-kHz crystal) is configured as the run and sleep clock source in the RSCLKCFG register, and the MOSC is configured as the deep-sleep clock source in the DSCLKCFG register, then two outcomes are possible:

- If the PIOSC is still powered in deep sleep (using the PIOSCPD bit in the DSCLKCFG register), the PIOSC is used as the clock source when entering deep sleep, and the device enters and exits deep-sleep mode normally. The MOSC is not used as the clock source in deep-sleep mode.
 - If the PIOSC has been configured to be powered down in deep-sleep mode, the device can enter deep-sleep mode but cannot exit properly. This situation can be avoided by programming the MOSC as the run and sleep clock source in the RSCLKCFG register before entering deep-sleep mode.
-

To provide the lowest possible deep-sleep power consumption and the ability to wake the processor from a peripheral without reconfiguring the peripheral for a change in clock, some of the communications modules have a clock control register at offset 0xFC8 in the module register space. The CS field in the clock control register lets the user select the PIOSC or ALTCLK as the clock source for the baud clock of the module. When the microcontroller enters deep-sleep mode, the PIOSC or ALTCLK becomes the source for the module clock as well, which allows the transmit and receive FIFOs to continue operation while the microcontroller is in deep-sleep mode. Figure 4-6 shows how the clocks are selected.

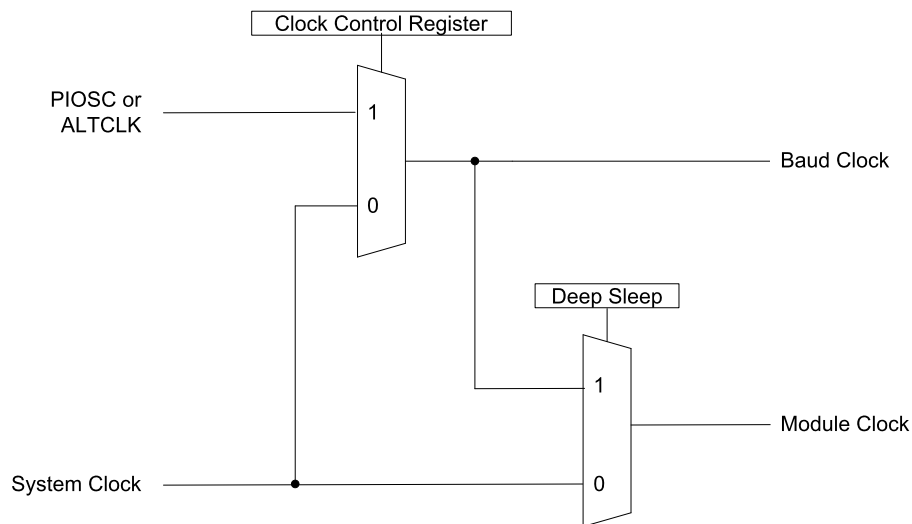


Figure 4-6. Module Clock Selection

Additional power-management modes are available that lower the power consumption of the peripheral memory, flash memory, and SRAM. However, the lower power consumption modes have slower deep-sleep and wake-up times.

NOTE: If one or more wait states are configured for run mode, when the device enters deep-sleep mode, it achieves its lowest possible current. If no wait states are applied in run mode, the lowest possible current is not achieved.

4.1.6.4 Dynamic Power Management

In addition to the sleep and deep-sleep modes and the clock gating for the on-chip modules, other power mode options let the LDO, flash memory, and SRAM enter different levels of power savings while in sleep or deep-sleep mode. In addition, software can control the LDO settings to gain a power advantage when running at slower speeds. These features may not be available on all devices; the System Properties (SYSPROP) register provides information on whether a mode is supported on a given MCU. The following registers provide these capabilities:

- Peripheral Power Control (PCx): Controls power to peripheral if that peripheral has the ability to respond to a power request
- Peripheral Memory Power Control (xMPC): Provides power control to some the peripheral memory arrays
- LDO Sleep Power Control (LDOSPCTL): Controls the LDO value in sleep mode
- LDO Deep-Sleep Power Control (LDODPCTL): Controls the LDO value in deep-sleep mode
- LDO Sleep Power Calibration (LDOSPCAL): Provides factory recommendations for the LDO value in sleep mode
- LDO Deep-Sleep Power Calibration (LDODPCAL): Provides factory recommendations for the LDO

value in deep-sleep mode

- Sleep Power Configuration (SLPPWRCFG): Controls the power-saving modes for flash memory and SRAM in sleep mode
- Deep-Sleep Power Configuration (DSLPPWRCFG): Controls the power-saving modes for flash memory and SRAM in deep-sleep mode
- Deep-Sleep Clock Configuration (DSCLKCFG): Controls the clocking in deep-sleep mode
- Sleep / Deep-Sleep Power Mode Status (SDPMST): Provides status information on the various power saving events

4.1.6.4.1 Peripheral Power Control

The Peripheral Power Control (PCx) registers reside at offset 0x900 in the System Control module register space. For modules that reside in a separate power domain, software can power down the module by setting the appropriate Pn bit to 0x0. This configuration provides the lowest power consumption for the module. The following registers can be programmed to disable power to the module:

- PCCAN
- PCLCD
- PCEMAC
- PCEPHY
- PCUSB
- PCCCM

Modification to other PCx registers have no effect, because other modules are not on their own power domain.

4.1.6.4.2 Peripheral Memory Power Control

When the device enters deep-sleep mode, software can further reduce power in peripheral modules that have their own associated memory array. Many of these peripherals can be programmed to enable a low-power retention mode or a power down of their associated peripheral SRAM array. If retention is supported and the PWRCTL bit field in the xMPC register is programmed to 0x1, the associated peripheral SRAM array enters retention mode and no accesses can be performed. When the PWRCTL bit is set to 0x0 in deep-sleep mode, the memory is powered off, the contents are lost, and the SRAM is not accessible. The Power Domain Status (xPDS) register of each peripheral can be read to determine the status of the memory array as well as the current power domain status of the peripheral. [Table 4-7](#) lists the peripherals with SRAM arrays and their capabilities during low-power modes.

Table 4-7. Peripheral Memory Power Control

Module	Memory Retention Capability?	Memory Array Power Down Capability?
USB	Yes	Yes
EMAC	No	Yes (only when power domain is off, PCEMAC register = 0x0)
LCD	No	No
CAN	No	Yes

4.1.6.4.3 LDO Power Control

NOTE: While the device is connected through JTAG, the LDO control settings for sleep and deep-sleep modes are not available and cannot be applied.

Software can configure the LDOSPCTL register (see [Section 4.2.25](#)) or the LDODPCTL register (see [Section 4.2.27](#)) to dynamically raise or lower the LDO voltage in sleep or deep-sleep mode, depending on whether an increase in performance or reduction in power consumption is required. The VLDO field in the LDOSPCTL register is set to 1.2 V by default. The LDODPCTL register is set to an LDO voltage of 0.9 V by default. If an application requires performance over power consumption in deep-sleep mode, the deep-sleep LDO voltage can be configured to a voltage greater than 0.9 V during system control initialization by setting the VADJEN bit and programming the VLDO field of the LDODPCTL register.

Before the LDO level is lowered during sleep or deep-sleep mode, the system clock must be configured to an acceptable frequency in the RSCLKCFG register for sleep mode and in DSLPCLKCFG for deep-sleep mode. [Table 4-8](#) lists the maximum system clock and PIOSC frequencies with respect to the LDO voltage.

Table 4-8. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage

Operating Voltage (LDO)	Maximum System Clock Frequency	PIOSC
1.2 V	120 MHz	16 MHz
0.9 V	30 MHz	16 MHz

The LDO power calibration registers, LDOSPCAL and LDODPCAL, provide suggested values for the LDO in the various modes. If software requests an LDO value that is too low or too high, the value is not accepted and an error is reported in the SDPMST register.

NOTE: When using the USB, Ethernet, EPI, and QSSI interfaces, the LDO must be configured to 1.2 V.

4.1.6.4.4 Flash Memory and SRAM Power Control

During sleep or deep-sleep mode, flash memory can be in either the default active mode or the low-power mode, while SRAM can be in the default active mode, standby mode, or low-power mode. The active mode in each case provides the fastest times to sleep and wake up, but consumes more power. Low-power mode provides the lowest power consumption, but the device takes longer to sleep and wake up.

The SRAM can be programmed to prohibit any power management by configuring the SRAMPM bit in the Sleep Power Configuration (SLPPWRCFG) register. This configuration provides the fastest sleep and wake-up times but consumes the most power while in sleep and deep-sleep modes.

The following power saving options are available in sleep and deep-sleep modes:

- The clocks can be gated according to the settings in the peripheral-specific SCGC or DCGC registers.
- In deep-sleep mode, the clock source can be changed and the PIOSC can be powered off (if no active peripheral requires it) using the DSCLKCFG register. These options are not available for Sleep mode.
- The LDO voltage can be changed using the LDOSPCTL or LDODPCTL register.
- The flash memory can be put into low-power mode.
- The SRAM can be put into standby or low-power mode.

The SDPMST register provides results on the dynamic power management command issued. The SDPMST register also reports some real-time status that can be viewed by a debugger or by the core if it is running. These events do not trigger an interrupt and are meant to provide information to help tune software for power management. The status register is written at the beginning of every dynamic power management event request that provides error checking. There is no mechanism to clear the bits; they are overwritten on the next event. The data is real time, and no event registers that information.

4.1.6.5 Hibernation Mode

In this mode, the power supplies are turned off to the main part of the microcontroller and only the circuitry of the Hibernation module is active. An external wake event or RTC event is required to return the microcontroller to run mode. When exiting hibernation mode, the Cortex-M4F processor and peripherals other than the Hibernation module see a normal "power on" sequence and the processor starts running code. If the Hibernation module has been put in hibernation mode and a reset occurs, the reset handler should read the HIB Raw Interrupt Status (HIBRIS) register in the Hibernation module to determine the cause of the reset.

4.1.6.6 Hardware System Service Request

The Hardware System Service Request (HSSR) register can issue a request that returns a device to factory settings. An HSSR consists of writing the appropriate key and data structure address offset to the HSSR register in the System Control module. Any HSSR initiates a reset event as the first event in the process. Then the HSSR register is evaluated.

To write to the HSSR register, the KEY field must be 0xCA. The CDOFF field in the HSSR register can have one of the following values:

- 0x00.0000: No request or the previous request completed successfully
- 0xFF.FFFF: No request and the previous request failed
- Anything else: The offset into SRAM of a HSSR request structure

During the HSSR routine, if any value other than 0x00.0000 or 0xFF.FFFF is in the CDOFF field, the offset is examined for validity, and the structure to which it points is examined for validity. If either is invalid, the request fails, and 0xFF.FFFF is written to the CDOFF field.

The offset is valid if all the following conditions are met:

- The CDOFF value is word aligned (that is, the two LSBs are both zero).
- The CDOFF value is at least 0x2000.4000.
- The CDOFF value is at most 0x2003.FFF0.

After a valid HSSR offset is determined, the following structure is examined in the SRAM that is indicated by the CDOFF field in the HSSR register. To initiate a return-to-factory settings function, the data structure must be:

- Request (32 bits) = 0xFEED.0001
- Data 1 (32 bits) = 0x0201.0100
- Data 2 (32 bits) = 0x0D08.0503
- Data 3 (32 bits) = 0x5937.2215

If the data bytes are correct, the device is returned to factory condition. During the return-to-factory settings function, the following events occur:

- The RAM is erased in the Hibernation module.
- The system SRAM is erased.
- The FMPPEn registers are set to 0xFFFF.FFFF (to allow a flash erase operation to occur).
- The EEPROM pages are erased.
- A mass erase of the flash array occurs.
- The BOOTCFG register is written with 0xFFFF.FFFE.

When the return-to-factory settings sequence is completed, the CDOFF field of the HSSR register is written with 0x00.0000, indicating a successful completion and activating a system reset.

4.2 System Control Registers

Table 4-9 lists the memory-mapped registers for the System Control. All register offset addresses not listed in Table 4-9 should be considered as reserved locations and the register contents should not be modified.

Additional flash and ROM registers defined in the System Control register space are described in Chapter 7.

All address offsets given are relative to the System Control base address of 0x400FE000.

Table 4-9. System Control Registers

Offset	Acronym	Register Name	Section
0x0	DID0	Device Identification	Section 4.2.1
0x4	DID1	Device Identification	Section 4.2.2
0x38	PTBOCTL	Power-Temp Brownout Control	Section 4.2.3
0x50	RIS	Raw Interrupt Status	Section 4.2.4
0x54	IMC	Interrupt Mask Control	Section 4.2.5
0x58	MISC	RW1C Masked Interrupt Status and Clear	Section 4.2.6
0x5C	RESC	Reset Cause	Section 4.2.7
0x60	PWRTC	RW1C Power-Temperature Cause	Section 4.2.8
0x64	NMIC	NMI Cause Register	Section 4.2.9
0x7C	MOSCCTL	Main Oscillator Control	Section 4.2.10
0xB0	RSCLKCFG	Run and Sleep Mode Configuration Register	Section 4.2.11
0xC0	MENTIM0	Memory Timing Parameter Register 0 for Main Flash and EEPROM	Section 4.2.12
0x138	ALTCLKCFG	Alternate Clock Configuration	Section 4.2.13
0x144	DSCLKCFG	Deep Sleep Clock Configuration Register	Section 4.2.14
0x148	DIVCLK	Divisor and Source Clock Configuration	Section 4.2.15
0x14C	SYSPROP	System Properties	Section 4.2.16
0x150	PIOSCCAL	Precision Internal Oscillator Calibration	Section 4.2.17
0x154	PIOSCCSTAT	Precision Internal Oscillator Statistics	Section 4.2.18
0x160	PLLREQ0	PLL Frequency 0	Section 4.2.19
0x164	PLLREQ1	PLL Frequency 1	Section 4.2.20
0x168	PLLSTAT	PLL Status	Section 4.2.21
0x188	SLPPWRCFG	Sleep Power Configuration	Section 4.2.22
0x18C	DSLPPWRCFG	Deep-Sleep Power Configuration	Section 4.2.23
0x1A0	NVMSTAT	Non-Volatile Memory Information	Section 4.2.24
0x1B4	LDOSPCTL	LDO Sleep Power Control	Section 4.2.25
0x1B8	LDOSPCAL	LDO Sleep Power Calibration	Section 4.2.26
0x1BC	LDODPCTL	LDO Deep-Sleep Power Control	Section 4.2.27
0x1C0	LDODPCAL	LDO Deep-Sleep Power Calibration	Section 4.2.28
0x1CC	SDPMST	Sleep / Deep-Sleep Power Mode Status	Section 4.2.29
0x1D8	RESBEHAVCTL	Reset Behavior Control Register	Section 4.2.30
0x1F4	HSSR	Hardware System Service Request	Section 4.2.31
0x280	USBPDS	USB Power Domain Status	Section 4.2.32
0x284	USBMPC	USB Memory Power Control	Section 4.2.33
0x288	EMACPDS	Ethernet MAC Power Domain Status	Section 4.2.34
0x28C	EMACMPC	Ethernet MAC Memory Power Control	Section 4.2.35
0x290	LCDPDS	LCD Power Domain Status	Section 4.2.36
0x294	LCDMPC	LCD Memory Power Control	Section 4.2.37
0x298	CAN0PDS	CAN 0 Power Domain Status	Section 4.2.38
0x29C	CAN0MPC	CAN 0 Memory Power Control	Section 4.2.39

Table 4-9. System Control Registers (continued)

Offset	Acronym	Register Name	Section
0x2A0	CAN1PDS	CAN 1 Power Domain Status	Section 4.2.40
0x2A4	CAN1MPC	CAN 1 Memory Power Control	Section 4.2.41
0x300	PPWD	Watchdog Timer Peripheral Present	Section 4.2.42
0x304	PPTIMER	16/32-Bit General-Purpose Timer Peripheral Present	Section 4.2.43
0x308	PPGPIO	General-Purpose Input/Output Peripheral Present	Section 4.2.44
0x30C	PPDMA	Micro Direct Memory Access Peripheral Present	Section 4.2.45
0x310	PPEPI	EPI Peripheral Present	Section 4.2.46
0x314	PPHIB	Hibernation Peripheral Present	Section 4.2.47
0x318	PPUART	Universal Asynchronous Receiver/Transmitter Peripheral Present	Section 4.2.48
0x31C	PPSSI	Synchronous Serial Interface Peripheral Present	Section 4.2.49
0x320	PPI2C	Inter-Integrated Circuit Peripheral Present	Section 4.2.50
0x328	PPUSB	Universal Serial Bus Peripheral Present	Section 4.2.51
0x330	PPEPHY	Ethernet PHY Peripheral Present	Section 4.2.52
0x334	PPCAN	Controller Area Network Peripheral Present	Section 4.2.53
0x338	PPADC	Analog-to-Digital Converter Peripheral Present	Section 4.2.54
0x33C	PPACMP	Analog Comparator Peripheral Present	Section 4.2.55
0x340	PPPWM	Pulse Width Modulator Peripheral Present	Section 4.2.56
0x344	PPQEI	Quadrature Encoder Interface Peripheral Present	Section 4.2.57
0x358	PPEEPROM	EEPROM Peripheral Present	Section 4.2.58
0x374	PPCCM	CRC and Cryptographic Modules Peripheral Present	Section 4.2.59
0x390	PPLCD	LCD Peripheral Present	Section 4.2.60
0x398	PPOWIRE	1-Wire Peripheral Present	Section 4.2.61
0x39C	PPEMAC	Ethernet MAC Peripheral Present	Section 4.2.62
0x3A0	PPPRB	Power Regulator Bus Peripheral Present	Section 4.2.63
0x500	SRWD	Watchdog Timer Software Reset	Section 4.2.64
0x504	SRTIMER	16/32-Bit General-Purpose Timer Software Reset	Section 4.2.65
0x508	SRGPIO	General-Purpose Input/Output Software Reset	Section 4.2.66
0x50C	SRDMA	Micro Direct Memory Access Software Reset	Section 4.2.67
0x510	SREPI	EPI Software Reset	Section 4.2.68
0x514	SRHIB	Hibernation Software Reset	Section 4.2.69
0x518	SRUART	Universal Asynchronous Receiver/Transmitter Software Reset	Section 4.2.70
0x51C	SRSSI	Synchronous Serial Interface Software Reset	Section 4.2.71
0x520	SRI2C	Inter-Integrated Circuit Software Reset	Section 4.2.72
0x528	SRUSB	Universal Serial Bus Software Reset	Section 4.2.73
0x530	SREPHY	Ethernet PHY Software Reset	Section 4.2.74
0x534	SRCAN	Controller Area Network Software Reset	Section 4.2.75
0x538	SRADC	Analog-to-Digital Converter Software Reset	Section 4.2.76
0x53C	SRACMP	Analog Comparator Software Reset	Section 4.2.77
0x540	SRPWM	Pulse Width Modulator Software Reset	Section 4.2.78
0x544	SRQEI	Quadrature Encoder Interface Software Reset	Section 4.2.79
0x558	SREEEPROM	EEPROM Software Reset	Section 4.2.80
0x574	SRCCM	CRC and Cryptographic Modules Software Reset	Section 4.2.81
0x590	SRLCD	LCD Controller Software Reset	Section 4.2.82
0x598	SROWIRE	1-Wire Software Reset	Section 4.2.83
0x59C	SREMAC	Ethernet MAC Software Reset	Section 4.2.84
0x600	RCGCWD	Watchdog Timer Run Mode Clock Gating Control	Section 4.2.85
0x604	RCGCTIMER	16/32-Bit General-Purpose Timer Run Mode Clock Gating Control	Section 4.2.86

Table 4-9. System Control Registers (continued)

Offset	Acronym	Register Name	Section
0x608	RCGCGPIO	General-Purpose Input/Output Run Mode Clock Gating Control	Section 4.2.87
0x60C	RCGCDMA	Micro Direct Memory Access Run Mode Clock Gating Control	Section 4.2.88
0x610	RCGCEPI	EPI Run Mode Clock Gating Control	Section 4.2.89
0x614	RCGCHIB	Hibernation Run Mode Clock Gating Control	Section 4.2.90
0x618	RCGCUART	Universal Asynchronous Receiver/Transmitter RunMode Clock Gating Control	Section 4.2.91
0x61C	RCGCSSI	Synchronous Serial Interface Run Mode Clock Gating Control	Section 4.2.92
0x620	RCGCI2C	Inter-Integrated Circuit Run Mode Clock Gating Control	Section 4.2.93
0x628	RCGUSB	Universal Serial Bus Run Mode Clock Gating Control	Section 4.2.94
0x630	RCGCEPHY	Ethernet PHY Run Mode Clock Gating Control	Section 4.2.95
0x634	RCGCCAN	Controller Area Network RunMode Clock Gating Control	Section 4.2.96
0x638	RCGCADC	Analog-to-Digital Converter Run Mode Clock Gating Control	Section 4.2.97
0x63C	RCGCACMP	Analog Comparator Run Mode Clock Gating Control	Section 4.2.98
0x640	RCGCPWM	Pulse Width Modulator Run Mode Clock Gating Control	Section 4.2.99
0x644	RCGCQEI	Quadrature Encoder Interface Run Mode Clock Gating Control	Section 4.2.100
0x658	RCGCEEPROM	EEPROM Run Mode Clock Gating Control	Section 4.2.101
0x674	RCGCCCM	CRC and CryptographicModules RunMode ClockGating Control	Section 4.2.102
0x690	RCGCLCD	LCD Controller Run Mode Clock Gating Control	Section 4.2.103
0x698	RCGCOWIRE	1-Wire Run Mode Clock Gating Control	Section 4.2.104
0x69C	RCGCEMAC	Ethernet MAC Run Mode Clock Gating Control	Section 4.2.105
0x700	SCGCWD	Watchdog Timer Sleep Mode Clock Gating Control	Section 4.2.106
0x704	SCGCTIMER	16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control	Section 4.2.107
0x708	SCGCGPIO	General-Purpose Input/Output Sleep Mode Clock Gating Control	Section 4.2.108
0x70C	SCGCDMA	Micro Direct Memory Access Sleep Mode Clock Gating Control	Section 4.2.109
0x710	SCGCEPI	EPI Sleep Mode Clock Gating Control	Section 4.2.110
0x714	SCGCHIB	Hibernation Sleep Mode Clock Gating Control	Section 4.2.111
0x718	SCGCUART	Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control	Section 4.2.112
0x71C	SCGCSSI	Synchronous Serial Interface Sleep Mode Clock Gating Control	Section 4.2.113
0x720	SCGCI2C	Inter-Integrated Circuit Sleep Mode Clock Gating Control	Section 4.2.114
0x728	SCGUSB	Universal Serial Bus Sleep Mode Clock Gating Control	Section 4.2.115
0x730	SCGCEPHY	Ethernet PHY Sleep Mode Clock Gating Control	Section 4.2.116
0x734	SCGCCAN	Controller Area Network Sleep Mode Clock Gating Control	Section 4.2.117
0x738	SCGCADC	Analog-to-Digital Converter Sleep Mode Clock Gating Control	Section 4.2.118
0x73C	SCGCACMP	Analog Comparator Sleep Mode Clock Gating Control	Section 4.2.119
0x740	SCGCPWM	PulseWidthModulator Sleep Mode Clock Gating Control	Section 4.2.120
0x744	SCGCQEI	Quadrature Encoder Interface Sleep Mode Clock Gating Control	Section 4.2.121
0x758	SCGCEEPROM	EEPROM Sleep Mode Clock Gating Control	Section 4.2.122
0x774	SCGCCCM	CRC and Cryptographic Modules Sleep Mode Clock Gating Control	Section 4.2.123
0x790	SCGCLCD	LCD Controller Sleep Mode Clock Gating Control	Section 4.2.124
0x798	SCGCOWIRE	1-Wire Sleep Mode Clock Gating Control	Section 4.2.125
0x79C	SCGCEMAC	Ethernet MAC Sleep Mode Clock Gating Control	Section 4.2.126
0x800	DCGCWD	Watchdog Timer Deep-SleepMode Clock Gating Control	Section 4.2.127
0x804	DCGCTIMER	Clock 16/32-Bit General-Purpose Timer Deep-Sleep Mode Gating Control	Section 4.2.128
0x808	DCGCGPIO	General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control	Section 4.2.129
0x80C	DCGCDMA	Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control	Section 4.2.130
0x810	DCGCEPI	EPI Deep-Sleep Mode Clock Gating Control	Section 4.2.131
0x814	DCGCHIB	Hibernation Deep-Sleep Mode Clock Gating Control	Section 4.2.132

Table 4-9. System Control Registers (continued)

Offset	Acronym	Register Name	Section
0x818	DCGCUART	Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control	Section 4.2.133
0x81C	DCGCSSI	Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control	Section 4.2.134
0x820	DCGCI2C	Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control	Section 4.2.135
0x828	DCGCUSB	Universal Serial Bus Deep-Sleep Mode Clock Gating Control	Section 4.2.136
0x830	DCGCEPHY	Ethernet PHY Deep-Sleep Mode Clock Gating Control	Section 4.2.137
0x834	DCGCCAN	Controller Area Network Deep-Sleep Mode Clock Gating Control	Section 4.2.138
0x838	DCGCADC	Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control	Section 4.2.139
0x83C	DCGCACMP	Analog Comparator Deep-Sleep Mode Clock Gating Control	Section 4.2.140
0x840	DCGCPWM	Pulse Width Modulator Deep-Sleep Mode Clock Gating Control	Section 4.2.141
0x844	DCGCQEI	Quadrature Encoder Interface Deep-Sleep Mode Clock Gating Control	Section 4.2.142
0x858	DCGCEEPROM	EEPROM Deep-Sleep Mode Clock Gating Control	Section 4.2.143
0x874	DCGCCCM	CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control	Section 4.2.144
0x890	DCGCLCD	LCD Controller Deep-Sleep Mode Clock Gating Control	Section 4.2.145
0x898	DCGCOWIRE	1-Wire Deep-Sleep Mode Clock Gating Control	Section 4.2.146
0x89C	DCGCEMAC	Ethernet MAC Deep-Sleep Mode Clock Gating Control	Section 4.2.147
0x900	PCWD	Watchdog Timer Power Control	Section 4.2.148
0x904	PCTIMER	16/32-Bit General-Purpose Timer Power Control	Section 4.2.149
0x908	PCGPIO	General-Purpose Input/Output Power Control	Section 4.2.150
0x90C	PCDMA	Micro Direct Memory Access Power Control	Section 4.2.151
0x910	PCEPI	External Peripheral Interface Power Control	Section 4.2.152
0x914	PCHIB	Hibernation Power Control	Section 4.2.153
0x918	PCUART	Universal Asynchronous Receiver/Transmitter Power Control	Section 4.2.154
0x91C	PCSSI	Synchronous Serial Interface Power Control	Section 4.2.155
0x920	PCI2C	Inter-Integrated Circuit Power Control	Section 4.2.156
0x928	PCUSB	Universal Serial Bus Power Control	Section 4.2.157
0x930	PCEPHY	Ethernet PHY Power Control	Section 4.2.158
0x934	PCCAN	Controller Area Network Power Control	Section 4.2.159
0x938	PCADC	Analog-to-Digital Converter Power Control	Section 4.2.160
0x93C	PCACMP	Analog Comparator Power Control	Section 4.2.161
0x940	PCPWM	Pulse Width Modulator Power Control	Section 4.2.162
0x944	PCQEI	Quadrature Encoder Interface Power Control	Section 4.2.163
0x958	PCEEPROM	EEPROM Power Control	Section 4.2.164
0x974	PCCCM	CRC and Cryptographic Modules Power Control	Section 4.2.165
0x990	PCLCD	LCD Controller Power Control	Section 4.2.166
0x998	PCOWIRE	1-Wire Power Control	Section 4.2.167
0x99C	PCEMAC	Ethernet MAC Power Control	Section 4.2.168
0xA00	PRWD	Watchdog Timer Peripheral Ready	Section 4.2.169
0xA04	PRTIMER	16/32-Bit General-Purpose Timer Peripheral Ready	Section 4.2.170
0xA08	PRGPIO	General-Purpose Input/Output Peripheral Ready	Section 4.2.171
0xA0C	PRDMA	Micro Direct Memory Access Peripheral Ready	Section 4.2.172
0xA10	PREPI	EPI Peripheral Ready	Section 4.2.173
0xA14	PRHIB	Hibernation Peripheral Ready	Section 4.2.174
0xA18	PRUART	Universal Asynchronous Receiver/Transmitter Peripheral Ready	Section 4.2.175
0xA1C	PRSSI	Synchronous Serial Interface Peripheral Ready	Section 4.2.176
0xA20	PRI2C	Inter-Integrated Circuit Peripheral Ready	Section 4.2.177
0xA28	PRUSB	Universal Serial Bus Peripheral Ready	Section 4.2.178
0xA30	PREPHY	Ethernet PHY Peripheral Ready	Section 4.2.179

Table 4-9. System Control Registers (continued)

Offset	Acronym	Register Name	Section
0xA34	PRCAN	Controller Area Network Peripheral Ready	Section 4.2.180
0xA38	PRADC	Analog-to-Digital Converter Peripheral Ready	Section 4.2.181
0xA3C	PRACMP	Analog Comparator Peripheral Ready	Section 4.2.182
0xA40	PRPWM	Pulse Width Modulator Peripheral Ready	Section 4.2.183
0xA44	PRQEI	Quadrature Encoder Interface Peripheral Ready	Section 4.2.184
0xA58	PREEPROM	EEPROM Peripheral Ready	Section 4.2.185
0xA74	PRCCM	CRC and Cryptographic Modules Peripheral Ready	Section 4.2.186
0xA90	PRLCD	LCD Controller Peripheral Ready	Section 4.2.187
0xA98	PROWIRE	1-Wire Peripheral Ready	Section 4.2.188
0xA9C	PREMAC	Ethernet MAC Peripheral Ready	Section 4.2.189
0xF20	UNIQUEID0	Unique ID 0	Section 4.2.190
0xF24	UNIQUEID1	Unique ID 1	Section 4.2.190
0xF28	UNIQUEID2	Unique ID 2	Section 4.2.190
0xF2C	UNIQUEID3	Unique ID 3	Section 4.2.190

Complex bit access types are encoded to fit into small table cells. [Table 4-10](#) lists the codes that are used for access types in this section.

Table 4-10. System Control Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

4.2.1 DID0 Register (Offset = 0x0) [reset = X]

Device Identification 0 (DID0)

This register identifies the version of the microcontroller. Each microcontroller is uniquely identified by the combined values of the CLASS field in the DID0 register and the PARTNO field in the DID1 register. The MAJOR and MINOR bit fields indicate the die revision number. Combined, the MAJOR and MINOR bit fields indicate the part revision number.

MAJOR Bit Field Value	MINOR Bit Field Value	Die Revision	Part Revision
0x0	0x0	A0	1
0x0	0x1	A1	2
0x0	0x2	A2	3

DID0 is shown in [Figure 4-7](#) and described in [Table 4-11](#).

Return to [Summary Table](#).

Figure 4-7. DID0 Register

31	30	29	28	27	26	25	24
RESERVED	VER			RESERVED			
R-0x0	R-0x1			R-0x8			
23	22	21	20	19	18	17	16
CLASS							
R-0x0C							
15	14	13	12	11	10	9	8
MAJOR							
R-X							
7	6	5	4	3	2	1	0
MINOR							
R-X							

Table 4-11. DID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	Reserved
30-28	VER	R	0x1	DID0 Version. This field defines the DID0 register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): 0x1 = Second version of the DID0 register format.
27-24	RESERVED	R	0x8	Reserved
23-16	CLASS	R	0x0C	Device Class. The value of the CLASS field identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The value of the CLASS field is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the MAJOR or MINOR fields require differentiation from prior microcontrollers. The value of the CLASS field is encoded as follows (all other encodings are reserved): 0x0C = MSP432E4 microcontrollers

Table 4-11. DID0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-8	MAJOR	R	X	<p>Major Revision.</p> <p>This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows:</p> <p>0x0 = Revision A (initial device)</p> <p>0x1 = Revision B (first base layer revision)</p> <p>0x2 = Revision C (second base layer revision)</p> <p>... and so on.</p>
7-0	MINOR	R	X	<p>Minor Revision.</p> <p>This field specifies the minor revision number of the microcontroller. The minor revision reflects changes to the metal layers of the design. The MINOR field value is reset when the MAJOR field is changed. The MINOR field is numeric and is encoded as follows:</p> <p>0x0 = Initial device, or a major revision update.</p> <p>0x1 = First metal layer change.</p> <p>0x2 = Second metal layer change.</p> <p>... and so on.</p>

4.2.2 DID1 Register (Offset = 0x4) [reset = X]

Device Identification 1 (DID1)

This register identifies the device family, part number, temperature range, pin count, and package type. Each microcontroller is uniquely identified by the combined values of the CLASS field in the DID0 register and the PARTNO field in the DID1 register.

DID1 is shown in [Figure 4-8](#) and described in [Table 4-12](#).

Return to [Summary Table](#).

Figure 4-8. DID1 Register

31	30	29	28	27	26	25	24
VER				FAM			
R-0x1				R-0x0			
23	22	21	20	19	18	17	16
PARTNO							
R-0x32							
15	14	13	12	11	10	9	8
PINCOUNT				RESERVED			
R-X				R-0x0			
7	6	5	4	3	2	1	0
TEMP			PKG		ROHS	QUAL	
R-0x3			R-0x2		R-0x1	R-0x2	

Table 4-12. DID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	VER	R	0x1	DID1 Version. This field defines the DID1 register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): 0x0 = Reserved 0x1 = Second version of the DID1 register format.
27-24	FAM	R	0x0	Family. This field provides the family identification of the device within the product portfolio. The value is encoded as follows (all other encodings are reserved): 0x0 = MSP432E4 microcontrollers
23-16	PARTNO	R	X	Part Number. This field provides the part number of the device within the family.
15-13	PINCOUNT	R	X	Package Pin Count. This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): 0x0 = Reserved 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved 0x4 = Reserved 0x5 = Reserved 0x6 = 128-pin TQFP package 0x7 = 212-pin BGA package
12-8	RESERVED	R	0x0	
7-5	TEMP	R	0x3	Temperature Range. This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved): 0x0 = Commercial temperature range 0x1 = Industrial temperature range 0x2 = Extended temperature range

Table 4-12. DID1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4-3	PKG	R	X	Package Type. This field specifies the package type. The value is encoded as follows (all other encodings are reserved): 0x0 = Reserved 0x1 = QFP package 0x2 = BGA package
2	ROHS	R	0x1	RoHS-Compliance. This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.
1-0	QUAL	R	0x2	Qualification Status. This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved): 0x0 = Engineering Sample (unqualified) 0x1 = Pilot Production (unqualified) 0x2 = Fully Qualified

4.2.3 PTBOCTL Register (Offset = 0x38) [reset = 0x3]

Power-Temp Brownout Control (PTBOCTL)

This register determines, based on an individual event level, the appropriate next level of action (for example, NONE, System Control Interrupt, NMI, or reset) when an event occurs.

Power-temperature event actions are directed to the core as a System Control Interrupt or NMI. When a reset occurs, its behavior is controlled by the Reset Behavior Control (RESBEHAVCTL) register. If one of the events configured in the PTBOCTL register causes a reset, it is registered as a BOR interrupt in the Reset Cause (RESC) register.

NOTE: V_{DDA} is the supply voltage to the analog components of the device and V_{DD} is the supply voltage to the digital components of the device.

PTBOCTL is shown in [Figure 4-9](#) and described in [Table 4-13](#).

Return to [Summary Table](#).

Figure 4-9. PTBOCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						VDDA_UBOR	
R-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
RESERVED						VDD_UBOR	
R-0x0						R/W-0x3	

Table 4-13. PTBOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9-8	VDDA_UBOR	R/W	0x0	V_{DDA} Under BOR Event Action. An event occurs when V_{DDA} trips under the V_{DDA_BOR0} threshold found in the device-specific data sheet. This field determines the action to take on the event. 0x0 = No Action 0x1 = System control interrupt 0x2 = NMI 0x3 = Reset
7-2	RESERVED	R	0x0	
1-0	VDD_UBOR	R/W	0x3	V_{DD} Under BOR Event Action. An event occurs when V_{DD} trips under the V_{DD_BOR} threshold found in the device-specific data sheet. This field determines the action to take on the event. 0x0 = No Action 0x1 = System control interrupt 0x2 = NMI 0x3 = Reset

4.2.4 RIS Register (Offset = 0x50) [reset = 0x0]

Raw Interrupt Status (RIS)

This register indicates the status for system control raw interrupts. An interrupt is sent to the interrupt controller if the corresponding bit in the Interrupt Mask Control (IMC) register is set. Writing 1 to the corresponding bit in the Masked Interrupt Status and Clear (MISC) register clears an interrupt status bit.

RIS is shown in [Figure 4-10](#) and described in [Table 4-14](#).

Return to [Summary Table](#).

Figure 4-10. RIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							MOSCPUPRIS
R-0x0							R-0x0
7	6	5	4	3	2	1	0
RESERVED	PLLLRIS	RESERVED	MOFRIS	RESERVED	BORRIS	RESERVED	RESERVED
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 4-14. RIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	MOSCPUPRIS	R	0x0	MOSC Power Up Raw Interrupt Status. This bit is cleared by writing 1 to the MOSCPUPMIS bit in the MISC register. 0x0 = Sufficient time has not passed for the MOSC to reach the expected frequency. 0x1 = Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by TMOSC_START.
7	RESERVED	R	0x0	
6	PLLLRIS	R	0x0	PLL Lock Raw Interrupt Status. This bit is cleared by writing 1 to the PLLLMIS bit in the MISC register. 0x0 = The PLL timer has not reached TREADY. 0x1 = The PLL timer has reached TREADY indicating that sufficient time has passed for the PLL to lock.
5-4	RESERVED	R	0x0	
3	MOFRIS	R	0x0	Main Oscillator Failure Raw Interrupt Status. This bit is cleared by writing 1 to the MOFMIS bit in the MISC register. 0x0 = The main oscillator has not failed. 0x1 = The MOSCIM bit in the MOSCCTL register is set and the main oscillator has failed.
2	RESERVED	R	0x0	

Table 4-14. RIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	BORRIS	R	0x0	Brownout Reset Raw Interrupt Status. The appropriate BOR bit in the PTBOCTL register must be set to an interrupt (0x1) encoding to generate an interrupt. . This bit is cleared by writing 1 to the BORMIS bit in the MISC register. 0x0 = A brownout condition is not currently active. 0x1 = A brownout condition is currently active.
0	RESERVED	R	0x0	

4.2.5 IMC Register (Offset = 0x54) [reset = 0x0]

Interrupt Mask Control (IMC)

This register contains the mask bits for system control raw interrupts. A raw interrupt, indicated by a bit being set in the Raw Interrupt Status (RIS) register, is sent to the interrupt controller if the corresponding bit in this register is set.

IMC is shown in [Figure 4-11](#) and described in [Table 4-15](#).

Return to [Summary Table](#).

Figure 4-11. IMC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							MOSCPUPIM
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	PLLLIM	RESERVED	MOFIM	RESERVED	BORIM	RESERVED	RESERVED
R-0x0	R/W-0x0	R-0x0	R/W-0x0	R-0x0	R/W-0x0	R-0x0	R-0x0

Table 4-15. IMC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	MOSCPUPIM	R/W	0x0	MOSC Power Up Interrupt Mask 0x0 = The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the RIS register is set.
7	RESERVED	R	0x0	
6	PLLLIM	R/W	0x0	PLL Lock Interrupt Mask 0x0 = The PLLLRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PLLLRIS bit in the RIS register is set.
5-4	RESERVED	R	0x0	
3	MOFIM	R/W	0x0	Main Oscillator Failure Interrupt Mask 0x0 = The MOFRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the MOFRIS bit in the RIS register is set.
2	RESERVED	R	0x0	
1	BORIM	R/W	0x0	Brownout Reset Interrupt Mask 0x0 = The BORRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the BORRIS bit in the RIS register is set.
0	RESERVED	R	0x0	

4.2.6 MISC Register (Offset = 0x58) [reset = 0x0]

Masked Interrupt Status and Clear (MISC)

On a read, this register gives the current masked status value of the corresponding interrupt in the Raw Interrupt Status (RIS) register. All of the bits are RW1C, thus writing 1 to a bit clears the corresponding raw interrupt bit in the RIS register (see [Section 4.2.4](#)).

MISC is shown in [Figure 4-12](#) and described in [Table 4-16](#).

Return to [Summary Table](#).

Figure 4-12. MISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							MOSCPUPMIS
R-0x0							R/W1C-0x0
7	6	5	4	3	2	1	0
RESERVED	PLLLMIS	RESERVED	MOFMIS	RESERVED	BORMIS	RESERVED	RESERVED
R-0x0	R/W1C-0x0	R-0x0	R/W1C-0x0	R-0x0	R/W1C-0x0	R-0x0	R-0x0

Table 4-16. MISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	MOSCPUPMIS	R/W1C	0x0	<p>MOSC Power Up Masked Interrupt Status</p> <p>0x0 = When read, 0 indicates that sufficient time has not passed for the MOSC PLL to lock. Writing 0 has no effect on the state of this bit.</p> <p>0x1 = When read, 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock. Writing 1 to this bit clears it and also the MOSCPUPRIS bit in the RIS register.</p>
7	RESERVED	R	0x0	
6	PLLLMIS	R/W1C	0x0	<p>PLL Lock Masked Interrupt Status</p> <p>0x0 = When read, 0 indicates that sufficient time has not passed for the PLL to lock. Writing 0 has no effect on the state of this bit.</p> <p>0x1 = When read, 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock. Writing 1 to this bit clears it and also the PLLLRIS bit in the RIS register.</p>
5-4	RESERVED	R	0x0	
3	MOFMIS	R/W1C	0x0	<p>Main Oscillator Failure Masked Interrupt Status</p> <p>0x0 = When read, 0 indicates that the main oscillator has not failed. Writing 0 has no effect on the state of this bit.</p> <p>0x1 = When read, 1 indicates that an unmasked interrupt was signaled because the main oscillator failed. Writing 1 to this bit clears it and also the MOFRIS bit in the RIS register.</p>
2	RESERVED	R	0x0	

Table 4-16. MISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	BORMIS	R/W1C	0x0	<p>BOR Masked Interrupt Status</p> <p>0x0 = When read, 0 indicates that a brownout condition has not occurred. Writing 0 has no effect on the state of this bit.</p> <p>0x1 = When read, 1 indicates that an unmasked interrupt was signaled because of a brownout condition. Writing 1 to this bit clears it and also the BORRIS bit in the RIS register.</p>
0	RESERVED	R	0x0	

4.2.7 RESC Register (Offset = 0x5C) [reset = X]

Reset Cause (RESC)

This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences. If a full POK-POR is initiated, the POR bit in the RESC register is set and all other bits are cleared. If the WDOGN, BOR or EXTRES configuration fields are set to 0x3 in the RESBEHAVCTL register and a simulated POR is initiated, the cause of the reset is reflected in the RESC register.

NOTE: After the RESC register is read, the Hibernate Raw Interrupt Status (HIBRIS) register in the Hibernation module must be evaluated to determine the full cause of the reset. Although an external reset assertion or POR resulting from a wake event is registered in the RESC register, the specific external wake source, including a low battery detect, is only registered in the HIBRIS register.

RESC is shown in [Figure 4-13](#) and described in [Table 4-17](#).

Return to [Summary Table](#).

Figure 4-13. RESC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							MOSCFAIL
R-0x0							R/W-X
15	14	13	12	11	10	9	8
RESERVED			HSSR	RESERVED			
R-0x0			R/W-X	R-0x0			
7	6	5	4	3	2	1	0
RESERVED		WDT1	SW	WDT0	BOR	POR	EXT
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0

Table 4-17. RESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	MOSCFAIL	R/W	X	MOSC Failure Reset. Writing 0 to this bit clears it. 0x0 = When read, this bit indicates that a MOSC failure has not generated a reset since the previous power-on reset. Writing 0 to this bit clears it. 0x1 = When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed while the MOSCIM bit in the MOSCCTL register is clear, generating a reset event.
15-13	RESERVED	R	0x0	
12	HSSR	R/W	X	HSSR Reset 0x0 = When read, this bit indicates that a HSSR request has not generated a reset since the previous power-on reset. Writing 0 to this bit clears it. 0x1 = When read, this bit indicates that a HSSR request has generated a reset.
11-6	RESERVED	R	0x0	

Table 4-17. RESC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	WDT1	R/W	0x0	<p>Watchdog Timer 1 Reset</p> <p>0x0 = When read, this bit indicates that Watchdog Timer 1 has not generated a reset since the previous power-on reset. Writing 0 to this bit clears it.</p> <p>0x1 = When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset.</p>
4	SW	R/W	0x0	<p>Software Reset</p> <p>0x0 = When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset. Writing 0 to this bit clears it.</p> <p>0x1 = When read, this bit indicates that a software reset has caused a reset event.</p>
3	WDT0	R/W	0x0	<p>Watchdog Timer 0 Reset</p> <p>0x0 = When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset. Writing 0 to this bit clears it.</p> <p>0x1 = When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.</p>
2	BOR	R/W	0x0	<p>Brownout Reset.</p> <p>For this bit, the BOR event that causes the brownout reset can be either:</p> <p>The V_{DD} supply drops below its acceptable operating range.</p> <p>The V_{DDA} supply drops below its acceptable operating range.</p> <p>0x0 = When read, this bit indicates that a brownout reset has not generated a reset since the previous power-on reset. Writing 0 to this bit clears it.</p> <p>0x1 = When read, this bit indicates that a brownout reset has caused a reset event.</p>
1	POR	R/W	0x1	<p>Power-On Reset</p> <p>0x0 = When read, this bit indicates that a power-on reset has not generated a reset. Writing 0 to this bit clears it.</p> <p>0x1 = When read, this bit indicates that a power-on reset has caused a reset event.</p>
0	EXT	R/W	0x0	<p>External Reset</p> <p>0x0 = When read, this bit indicates that an external reset (RST assertion) has not caused a reset event since the previous power-on reset. Writing 0 to this bit clears it.</p> <p>0x1 = When read, this bit indicates that an external reset (RST assertion) has caused a reset event.</p>

4.2.8 PWRTC Register (Offset = 0x60) [reset = 0x0]

Power-Temperature Cause (PWRTC)

This register provides detailed information on the power subsystem event that caused a reset or interrupt. The event sets the condition in this register without regard to whether it is used to generate a system control interrupt, reset, NMI, or no action. The PTBOCTL register contains the action to be taken on the specific events. The combination of the PWRTC register outputs and the PTBOCTL register causes the appropriate interrupt or reset condition to occur and the corresponding status bits to be set.

PWRTC is shown in [Figure 4-14](#) and described in [Table 4-18](#).

Return to [Summary Table](#).

Figure 4-14. PWRTC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			VDDA_UBOR	RESERVED			VDD_UBOR
R-0x0			R/W1C-0x0	R-0x0			R/W1C-0x0

Table 4-18. PWRTC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	VDDA_UBOR	R/W1C	0x0	V _{DDA} Under BOR Status 0x0 = VDDA has not tripped undervoltage BOR comparison. 0x1 = VDDA has tripped undervoltage BOR comparison.
3-1	RESERVED	R	0x0	
0	VDD_UBOR	R/W1C	0x0	V _{DD} Under BOR Status 0x0 = VDD has not tripped undervoltage BOR comparison. 0x1 = VDD has tripped undervoltage BOR comparison.

4.2.9 NMIC Register (Offset = 0x64) [reset = 0x0]

NMI Cause Register (NMIC)

This register provides the detailed information on the cause of an NMI interrupt. These bits are set through hardware when the event occurs and the higher level control indicates that it should be NMI event.

NOTE: The NMIC register must be cleared by the following sequence:

1. Read the NMIC register to identify the source of the NMI.
2. Clear the source of the NMI.
3. Read the NMIC register again to check the status.
4. Write a 0 into the NMIC register bit that corresponds with the NMI source.
5. Read the NMIC to determine if it is cleared. If not, repeat Step 3 and Step 4.

NMIC is shown in [Figure 4-15](#) and described in [Table 4-19](#).

Return to [Summary Table](#).

Figure 4-15. NMIC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							MOSCFAIL
R-0x0							R/W-0x0
15	14	13	12	11	10	9	8
RESERVED						TAMPER	RESERVED
R-0x0						R/W-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED		WDT1	RESERVED	WDT0	POWER	RESERVED	EXTERNAL
R-0x0		R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0

Table 4-19. NMIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	MOSCFAIL	R/W	0x0	MOSC Failure NMI 0x0 = No MOSC failure has occurred. 0x1 = An NMI has occurred due to a MOSC failure.
15-10	RESERVED	R	0x0	
9	TAMPER	R/W	0x0	Tamper Event NMI See the HIB module tamper registers for more details on the tamper event. 0x0 = No tamper event has occurred. 0x1 = An NMI has occurred due to a tamper event.
8-6	RESERVED	R	0x0	
5	WDT1	R/W	0x0	Watch Dog Timer (WDT) 1 NMI 0x0 = No WDT 1 time-out has occurred. 0x1 = An NMI has occurred due to a WDT1 time-out event.
4	RESERVED	R	0x0	
3	WDT0	R/W	0x0	Watch Dog Timer (WDT) 0 NMI 0x0 = No WDT 0 time-out has occurred. 0x1 = An NMI has occurred due to a WDT0 time-out event.

Table 4-19. NMIC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	POWER	R/W	0x0	Power/Brownout Event NMI See PWRTC register for exact cause of power out or brownout event. 0x0 = No power event has occurred. 0x1 = An NMI has occurred due to a power event.
1	RESERVED	R	0x0	
0	EXTERNAL	R/W	0x0	External Pin NMI 0x0 = No NMI pin event has occurred. 0x1 = The NMI pin was asserted by external hardware.

4.2.10 MOSCCTL Register (Offset = 0x7C) [reset = 0xC]

Main Oscillator Control (MOSCCTL)

This register provides control over the features of the main oscillator, including the ability to enable the MOSC clock verification circuit, what action to take when the MOSC fails, and whether or not a crystal is connected. When enabled, this circuit monitors the frequency of the MOSC to verify that the oscillator is operating within specified limits. If the clock goes invalid after being enabled, the microcontroller issues a power-on reset and reboots to the NMI handler or generates an interrupt.

NOTE: If the MOSC is chosen as the clock to the Ethernet PHY then software must enable the MOSC before enabling the Ethernet PHY by setting the P0 bit in the PCEPHY.

MOSCCTL is shown in [Figure 4-16](#) and described in [Table 4-20](#).

Return to [Summary Table](#).

Figure 4-16. MOSCCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			OSCRNG	PWRDN	NOXTAL	MOSCIM	CVAL
R-0x0			R/W-0x0	R/W-0x1	R/W-0x1	R/W-0x0	R/W-0x0

Table 4-20. MOSCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	OSCRNG	R/W	0x0	Oscillator Range Specifies the frequency range of operation of the oscillator. 0x0 = Low-frequency range 0x1 = High-frequency range (equal to or greater than 10 MHz).
3	PWRDN	R/W	0x1	Power Down Provides user control over powering down the main oscillator circuit. This bit should be cleared when using a crystal and set for single-ended mode. 0x0 = Power to MOSC circuit is enabled. 0x1 = MOSC circuit is powered down.
2	NOXTAL	R/W	0x1	No MOSC or Crystal Connected Provides the user control over the power drawn from the main oscillator circuit. This bit should be set when either crystal or single-ended mode is being used. If the application needs MOSC, this bit should be cleared. 0x0 = This bit should be cleared when a crystal or oscillator is connected to the OSC0 and OSC1 inputs, regardless of whether or not the MOSC is used or powered down. For proper clock functionality when switching to crystal mode, software must clear this bit and set the PWRDN bit in a single write access. 0x1 = This bit should be set when a crystal or external oscillator is not connected to the OSC0 and OSC1 inputs to reduce power consumption.

Table 4-20. MOSCCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	MOSCIM	R/W	0x0	<p>MOSC Failure Action</p> <p>Regardless of the action taken, if the MOSC fails, the oscillator source is switched to the PIOSC automatically.</p> <p>0x0 = If the MOSC fails, a MOSC failure reset is generated and reboots to the NMI handler.</p> <p>0x1 = If the MOSC fails, an interrupt is generated as indicated by the MOSRIS bit in the RIS register.</p>
0	CVAL	R/W	0x0	<p>Clock Validation for MOSC</p> <p>0x0 = The MOSC monitor circuit is disabled.</p> <p>0x1 = The MOSC monitor circuit is enabled.</p>

4.2.11 RSCLKCFG Register (Offset = 0xB0) [reset = 0x0]

Run and Sleep Mode Configuration Register (RSCLKCFG)

NOTE: When transitioning the system clock configuration to use the MOSC as the fundamental clock source, the PWRDN bit must be set in the MOSCCTL register before reselecting the MOSC for proper operation.

RSCLKCFG is shown in [Figure 4-17](#) and described in [Table 4-21](#).

Return to [Summary Table](#).

Figure 4-17. RSCLKCFG Register

31	30	29	28	27	26	25	24
MEMTIMU	NEWFREQ	ACG	USEPLL	PLLSRC			
R0/W-0x0	R0/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0			
23	22	21	20	19	18	17	16
OSCSRC				OSYSDIV			
R/W-0x0				R/W-0x0			
15	14	13	12	11	10	9	8
OSYSDIV						PSYSDIV	
R/W-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
PSYSDIV							
R/W-0x0							

Table 4-21. RSCLKCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MEMTIMU	R0/W	0x0	Memory Timing Register Update. Setting this bit causes the MEMTIMU register value to be applied, and the memory timing to be updated. Execution and access is suspended during the change. This bit is automatically cleared by hardware.
30	NEWFREQ	R0/W	0x0	New PLLFREQ Accept. This bit controls the activation of the values in the PLLFREQ0 and PLLFREQ1 registers as applied to the PLL. Until NEWFREQ is written to a 1, writes to the PLLFREQ0 and PLLFREQ1 are deferred. When written with a 1, the values stored in PLLFREQ0 and PLLFREQ1 are applied to the PLL. This bit is automatically cleared by hardware. Software will not check the value after being set.
29	ACG	R/W	0x0	Auto Clock Gating. This bit specifies whether the system uses the Sleep-Mode Clock Gating Control (SCGCn) registers and Deep-Sleep-Mode Clock Gating Control (DCGCn) registers if the microcontroller enters a sleep or deep-sleep mode (respectively). The RCGCn registers are always used to control the clocks in run mode. 0x0 = The Run-Mode Clock Gating Control (RCGCn) registers are used when the microcontroller enters a sleep mode. 0x1 = If the microcontroller is in sleep mode, the SCGCn registers are used to control the clocks distributed to the peripherals. If the microcontroller is in deep-sleep mode, the DCGCn registers are used to control the clocks distributed to the peripherals. The SCGCn and DCGCn registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.
28	USEPLL	R/W	0x0	Use PLL. This bit controls whether the clock source is specified by the OSCSRC field or the output of the PLL is provided to the system clock divider and serves as the system clock source. 0x0 = Clock source is specified by the OSCSRC field. 0x1 = Clock source is specified by the PLL.

Table 4-21. RSCLKCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
27-24	PLLSRC	R/W	0x0	PLL Source. This field specifies the PLL input clock source. 0x0 = Reserved 0x3 = MOSC is the PLL input clock source
23-20	OSCSRC	R/W	0x0	Oscillator Source. This field specifies the oscillator source that becomes the oscillator clock (OSCCLK) source, which is used when the PLL is bypassed during run or sleep modes. 0x0 = Reserved 0x1 = Reserved 0x2 = LFIOOSC is the oscillator source. 0x3 = MOSC is the oscillator source. 0x4 = Hibernation module RTC oscillator (RTCOSC)
19-10	OSYSDIV	R/W	0x0	Oscillator System Clock Divisor. This field specifies the system clock divisor value for the oscillator path. This field is used when the USEPLL bit is 0. $f_{\text{sysclk}} = f_{\text{oscclk}} / (\text{OSYSDIV} + 1)$ The divisor value is the OSYSDIV field value + 1
9-0	PSYSDIV	R/W	0x0	PLL System Clock Divisor. This field specifies the system clock divisor value for the PLL. This field is used when the USEPLL bit is 1. $f_{\text{sysclk}} = f_{\text{VCO}} / (\text{PSYSDIV} + 1)$

4.2.12 MEMTIM0 Register (Offset = 0xC0) [reset = 0x00200030]

Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0)

The MEMTIM0 register provides timing parameters for the main Flash and EEPROM memories. The timing parameters apply to the memory while the system is in run or sleep mode; the clocking for these modes is consistent and unchanged, because the system clock frequency and source remains unchanged during transitions between run-to-sleep and sleep-back-to-run. Writes to MEMTIM0 do not have any effect on system state; the register contents are applied only when the MEMTIMU bit in the RSCLKCFG register is set. Doing so allows the software to execute out of the same memory system for which the timing parameters are being modified.

Depending on the CPU frequency, the application must program specific values into the fields of the MEMTIM0 register. [Table 4-22](#) details the bit field values that are required for the given CPU frequency ranges.

Table 4-22. MEMTIM0 Register Configuration versus Frequency

CPU Frequency Range (f) in MHz	Time Period Range (t) in ns	FBCHT and EBCHT	FBCE and EBCE	FWS and EWS
16	62.5	0x0	1	0x0
16 < f ≤ 40	62.5 > t ≥ 25	0x2	0	0x1
40 < f ≤ 60	25 > t ≥ 16.67	0x3	0	0x2
60 < f ≤ 80	16.67 > t ≥ 12.5	0x4	0	0x3
80 < f ≤ 100	12.5 > t ≥ 10	0x5	0	0x4
100 < f ≤ 120	10 > t ≥ 8.33	0x6	0	0x5

NOTE: The associated flash and EEPROM fields in the MEMTIM0 register must be programmed to the same values. For example, the FWS field must be programmed to the same value as the EWS field.

MEMTIM0 is shown in [Figure 4-18](#) and described in [Table 4-23](#).

Return to [Summary Table](#).

Figure 4-18. MEMTIM0 Register

31	30	29	28	27	26	25	24
RESERVED						EBCHT	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
EBCHT		EBCE		RESERVED		EWS	
R/W-0x0		R/W-0x1		R-0x0		R/W-0x0	
15	14	13	12	11	10	9	8
RESERVED						FBCHT	
R/W-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
FBCHT		FBCE		RESERVED		FWS	
R/W-0x0		R/W-0x1		R/W-0x1		R/W-0x0	

Table 4-23. MEMTIM0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-22	EBCHT	R/W	0x0	EEPROM Clock High Time Specifies the length of the EEPROM bank clock high time. 0x0 = 1/2 system clock period 0x1 = 1 system clock period 0x2 = 1.5 system clock periods 0x3 = 2 system clock periods 0x4 = 2.5 system clock periods 0x5 = 3 system clock periods 0x6 = 3.5 system clock periods 0x7 = 4 system clock periods 0x8 = 4.5 system clock periods
21	EBCE	R/W	0x1	EEPROM Bank Clock Edge Specifies the relationship of EEPROM clock to system clock. 0x0 = EEPROM clock rising aligns with system clock rising 0x1 = EEPROM clock rising aligns with system clock falling
20	RESERVED	R	0x0	
19-16	EWS	R/W	0x0	EEPROM Wait States. This field specifies the number of wait states inserted. Note: The value of the EWS bit must match the value of the FWS bit. 0x0 = Reserved 0x1 = 1 wait state 0x2 = 2 wait states 0x3 = 3 wait states 0x4 = 4 wait states 0x5 = 5 wait states 0x6 = 6 wait states 0x7 = 7 wait states
15-10	RESERVED	R/W	0x0	
9-6	FBCHT	R/W	0x0	Flash Bank Clock High Time Specifies the length of the flash bank clock high time. 0x0 = 1/2 system clock period 0x1 = 1 system clock period 0x2 = 1.5 system clock periods 0x3 = 2 system clock periods 0x4 = 2.5 system clock periods 0x5 = 3 system clock periods 0x6 = 3.5 system clock periods 0x7 = 4 system clock periods 0x8 = 4.5 system clock periods
5	FBCE	R/W	0x1	Flash Bank Clock Edge Specifies the relationship of flash clock to system clock. 0x0 = Flash clock rising aligns with system clock rising. 0x1 = Flash clock rising aligns with system clock falling.
4	RESERVED	R/W	0x1	

Table 4-23. MEMTIM0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	FWS	R/W	0x0	Flash Wait State. This field specifies the number of wait states inserted. Note: The value of the FWS bit must match the value of the EWS bit. 0x0 = Reserved 0x1 = 1 wait state 0x2 = 2 wait states 0x3 = 3 wait states 0x4 = 4 wait states 0x5 = 5 wait states 0x6 = 6 wait states 0x7 = 7 wait states

4.2.13 ALTCLKCFG Register (Offset = 0x138) [reset = 0x0]

Alternate Clock Configuration (ALTCLKCFG)

The ALTCLKCFG register specifies the alternate clock source used by many of the peripherals.

ALTCLKCFG is shown in [Figure 4-19](#) and described in [Table 4-24](#).

Return to [Summary Table](#).

Figure 4-19. ALTCLKCFG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												ALTCLK			
R-0x0												R/W-0x0			

Table 4-24. ALTCLKCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	ALTCLK	R/W	0x0	<p>Alternate Clock Source</p> <p>This provides a clock source of numerous frequencies to the general-purpose timer, SSI, and UART modules. If the Hibernation real-time clock output is selected, the clock source must also be enabled in the Hibernation module.</p> <p>0x0 = Precision Internal Oscillator (PIOSC)</p> <p>0x1 = Reserved</p> <p>0x2 = Reserved</p> <p>0x3 = Hibernation module real-time clock output (RTCOSC)</p> <p>0x4 = Low-frequency internal oscillator (LFIOSC)</p>

4.2.14 DSCLKCFG Register (Offset = 0x144) [reset = 0x0]

Deep Sleep Clock Configuration Register (DSCLKCFG)

The DSCLKCFG register specifies the behavior of the clock system while in deep sleep.

The MOSCDPD bit affects not only deep-sleep mode, but all other modes as well depending on the value of the bit. See [Table 4-25](#) when programming this bit:

Table 4-25. MOSC Configurations

PWRDN Bit	MOSCDPD Field	Result
0	0	MOSC is powered on in run and sleep modes, but is disabled in accidental power down, when the PWRDN bit is set in the MOSCTL register, or in deep-sleep mode only if it is not the deep-sleep clock source (DSOSCSRC != 0x3).
0	1	MOSC is powered and running in run, sleep, and deep-sleep modes.
1	0	MOSC is powered off and does not run in any mode. In this configuration, when the MOSC is disabled, choosing the MOSC as a clock source causes indeterminate results.
1	1	MOSC runs and does not disable itself in run, sleep, and deep-sleep modes regardless of the whether or not the PWRDN bit is set.

NOTE: The MOSCDPD bit has an effect in all modes of operation

NOTE: If the MOSC is chosen as the deep-sleep clock source in the DSCLKCFG register, the MOSC must also be configured as the run and sleep clock source in the RSCLKCFG register before entering deep sleep. If the PIOSC, LFIOSC, or Hibernation RTC module oscillator (HIBLFIOSC or 32-kHz crystal) is configured as the run and sleep clock source in the RSCLKCFG register, and the MOSC is configured as the deep-sleep clock source in the DSCLKCFG register, then two outcomes are possible:

- If the PIOSC is still powered in deep sleep (using the PIOSCPD bit in the DSCLKCFG register) then the PIOSC is used as the clock source when entering deep sleep and the device enters and exits the deep-sleep mode normally. The MOSC is not used as the clock source in deep sleep.
- If the PIOSC has been configured to be powered down in deep sleep, then the device can enter the deep-sleep mode, but cannot exit properly. This situation can be avoided by programming the MOSC as the run and sleep clock source in the RSCLKCFG register before entering deep sleep.

DSCLKCFG is shown in [Figure 4-20](#) and described in [Table 4-26](#).

Return to [Summary Table](#).

Figure 4-20. DSCLKCFG Register

31	30	29	28	27	26	25	24
PIOSCPD	MOSCDPD	RESERVED					
R/W-0x0	R/W-0x0	R-0x0					
23	22	21	20	19	18	17	16
DSOSCSRC				RESERVED			
R/W-0x0				R-0x0			
15	14	13	12	11	10	9	8
RESERVED						DSSYS DIV	
R-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
DSSYS DIV							
R/W-0x0							

Table 4-26. DSCLKCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PIOSCPD	R/W	0x0	<p>PIOSC Power Down</p> <p>0x0 = The PIOSC is active during deep-sleep mode.</p> <p>0x1 = The PIOSC is disabled during sleep mode for additional power savings.</p>
30	MOSCDPD	R/W	0x0	<p>MOSC Disable Power Down.</p> <p>This bit inhibits the MOSC from automatic or accidental power down. This bit is defined to ensure the MOSC circuit cannot be interrupted in uses where MOSC supplies a clock to the peripherals (for example, Ethernet PHY).</p> <p>0x0 = During deep-sleep (if DSOSCSRC is not MOSC), accidental power down or when the PWRDWN bit is set in the MOSCCTL register, the MOSC is powered down.</p> <p>0x1 = MOSC is not powered off during automatic or accidental power down. MOSC is also not powered off if DSOSCSRC is programmed to be MOSC.</p> <p>This bit should be set only after software configures the MOSCCTL register. Setting the MOSCDPD bit masks writes to PWRDN bit in the MOSCCTL register.</p>
29-24	RESERVED	R	0x0	
23-20	DSOSCSRC	R/W	0x0	<p>Deep Sleep Oscillator Source.</p> <p>This field specifies the oscillator source that becomes the oscillator clock (OSCCLK) source, which is used when the PLL is bypassed during deep-sleep mode.</p> <p>0x0 = Reserved</p> <p>0x1 = Reserved</p> <p>0x2 = LFIOSC</p> <p>0x3 = MOSC</p> <p>0x4 = Hibernation module RTCOSC</p>
19-10	RESERVED	R	0x0	
9-0	DSSYSDIV	R/W	0x0	<p>Deep Sleep Clock Divisor.</p> <p>This field specifies the system clock divisor value during deep-sleep mode. The clock source selected by DSOSCSRC is divided by DSSYSDIV + 1:</p> $f_{\text{SYSCLK}} = f_{\text{OSCCLK}} / (\text{DSSYSDIV} + 1)$ <p>Values 0x0 and 0x1 should not be used.</p> <p>If deep-sleep clock divide by 1 or divide by 2 is desired, the OSYSDIV bit field of the RSCLKCFG register must be configured for the desired deep-sleep divider before entering deep sleep. In this case, the Q post-divider bit field in the PLLFREQ1 register may need to be adjusted to keep the system clock frequency within the maximum clock frequency before entering deep sleep.</p>

4.2.15 DIVSCLK Register (Offset = 0x148) [reset = 0x0]

Divisor and Source Clock Configuration (DIVSCLK)

The DIVSCLK register specifies the source and divisor of the DIVSCLK reference clock output. This signal can be used as a clock source to an external device but bears no timing relationship to other signals.

NOTE: The DIVSCLK signal output is not synchronized to the System Clock.

DIVSCLK is shown in [Figure 4-21](#) and described in [Table 4-27](#).

Return to [Summary Table](#).

Figure 4-21. DIVSCLK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN	RESERVED														SRC
R/W-0x0	R-0x0														R/W-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								DIV							
R-0x0								R/W-0x0							

Table 4-27. DIVSCLK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	EN	R/W	0x0	DIVSCLK Enable. This bit enables the generation of the DIVSCLK clock output. It resets to 0 to disable the output thereby reducing initial current/power consumption. 0x0 = The clock output is disabled. 0x1 = Clock output is enabled.
30-18	RESERVED	R	0x0	
17-16	SRC	R/W	0x0	Clock Source. Selects the reference clock used to generate the output. 0x0 = System clock 0x1 = PIOSC 0x2 = MOSC 0x3 = Reserved
15-8	RESERVED	R	0x0	
7-0	DIV	R/W	0x0	Divisor Value. This field controls the ratio of the source clock to the output clock. The output clock frequency is equal to the source clock frequency divided by the DIV field value plus 1. 0x0 = Divided by 1 0x1 = Divided by 2

4.2.16 SYSPROP Register (Offset = 0x14C) [reset = 0x00031F31]

System Properties (SYSPROP)

This register provides information on whether certain System Control properties are present on the microcontroller.

SYSPROP is shown in [Figure 4-22](#) and described in [Table 4-28](#).

Return to [Summary Table](#).

Figure 4-22. SYSPROP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						LDOSME	TSPDE
R-0x0						R-0x1	R-0x1
15	14	13	12	11	10	9	8
RESERVED			PIOSCPDE	SRAMSM	SRAMLPM	RESERVED	FLASHLPM
R-0x0			R-0x1	R-0x1	R-0x1	R-0x0	R-0x1
7	6	5	4	3	2	1	0
RESERVED		LDOSQ	RESERVED				FPU
R-0x0		R-0x1	R-0x0				R-0x1

Table 4-28. SYSPROP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	LDOSME	R	0x1	LDO Sleep Mode Enable 0x0 = The LDOSM bit of the DSLPPWRCFG register is ignored. 0x1 = The LDOSM bit of the DSLPPWRCFG register can be set to place the LDO in a low-power mode when deep-sleep mode is entered.
16	TSPDE	R	0x1	Temp Sense Power Down Enable. This bit allows the internal temperature sensor in the ADC to be powered off in deep-sleep mode. 0x0 = The TSPD bit of the DSLPPWRCFG register is ignored. 0x1 = The TSPD bit of the DSLPPWRCFG register can be set to power off the temperature sensor in deep-sleep mode.
15-13	RESERVED	R	0x0	
12	PIOSCPDE	R	0x1	PIOSC Power Down Present. This bit determines whether the PIOSCPD bit in the DSCLKCFG register can be set to power down the PIOSC in deep-sleep mode. 0x0 = The status of the PIOSCPD bit is ignored. 0x1 = The PIOSCPD bit can be set to power down the PIOSC in deep-sleep mode.
11	SRAMSM	R	0x1	SRAM Sleep/Deep-Sleep Standby Mode Present. This bit determines whether the SRAMP field in the SLPPWRCFG and DSLPPWRCFG registers can be configured to put the SRAM into standby mode while in sleep or deep-sleep mode. 0x0 = A value of 0x1 in the SRAMP fields is ignored. 0x1 = The SRAMP fields can be configured to put the SRAM into standby mode while in sleep or deep-sleep mode.

Table 4-28. SYSPROP Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	SRAMLPM	R	0x1	SRAM Sleep/Deep-Sleep Low Power Mode Present. This bit determines whether the SRAMPMP field in the SLPPWRCFG and DSLPPWRCFG registers can be configured to put the SRAM into low-power mode while in sleep or deep-sleep mode. 0x0 = A value of 0x3 in the SRAMPMP fields is ignored. 0x1 = The SRAMPMP fields can be configured to put the SRAM into low-power mode while in sleep or deep-sleep mode.
9	RESERVED	R	0x0	
8	FLASHLPM	R	0x1	Flash Memory Sleep/Deep-Sleep Low Power Mode Present. This bit determines whether the FLASHPM field in the SLPPWRCFG and DSLPPWRCFG registers can be configured to put the flash memory into low-power mode while in sleep or deep-sleep mode. 0x0 = A value of 0x2 in the FLASHPM fields is ignored. 0x1 = The FLASHPM fields can be configured to put the flash memory into low-power mode while in sleep or deep-sleep mode.
7-6	RESERVED	R	0x0	
5	LDOSQ	R	0x1	Automatic LDO Sequence Control Present. This bit indicates that the ability to sequence the LDO output voltage is available during sleep and deep-sleep modes. 0x0 = Software cannot set the VADJEN bit in the LDOSPCTL and LDODPCTL registers. 0x1 = Software can set the VADJEN bit in the LDOSPCTL and LDODPCTL registers.
4-1	RESERVED	R	0x0	
0	FPU	R	0x1	FPU Present. This bit indicates if the FPU is present in the Cortex -M4 core. 0x0 = FPU is not present. 0x1 = FPU is present.

4.2.17 PIOSCCAL Register (Offset = 0x150) [reset = 0x0]

Precision Internal Oscillator Calibration (PIOSCCAL)

This register provides the ability to update or recalibrate the precision internal oscillator. A 32.768-kHz oscillator must be used as the clock source of the Hibernation module for the user to calibrate the PIOSC.

PIOSCCAL is shown in [Figure 4-23](#) and described in [Table 4-29](#).

Return to [Summary Table](#).

Figure 4-23. PIOSCCAL Register

31	30	29	28	27	26	25	24
UTEN	RESERVED						
R/W-0x0	R-0x0						
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						CAL	UPDATE
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	UT						
R-0x0	R/W-0x0						

Table 4-29. PIOSCCAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	UTEN	R/W	0x0	Use User Trim Value 0x0 = The factory calibration value is used for an update trim operation. 0x1 = The trim value in bits [6:0] of this register are used for any update trim operation.
30-10	RESERVED	R	0x0	
9	CAL	R/W	0x0	Start Calibration. This bit is auto-cleared after it is set. 0x0 = No action 0x1 = Starts a new calibration of the PIOSC. Results are in the PIOSCCAL register. The resulting trim value from the operation is active in the PIOSC after the calibration completes. The result overrides any previous update trim operation whether the calibration passes or fails.
8	UPDATE	R/W	0x0	Update Trim. This bit is automatically cleared after the update. 0x0 = No action 0x1 = Updates the PIOSC trim value with the UT bit or the DT bit in the PIOSCCAL register. Used with UTEN.
7	RESERVED	R	0x0	
6-0	UT	R/W	0x0	User Trim Value. User trim value that can be loaded into the PIOSC.

4.2.18 PIOSCSTAT Register (Offset = 0x154) [reset = 0x00400040]

Precision Internal Oscillator Statistics (PIOSCSTAT)

This register provides the user information on the PIOSC calibration. A 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to calibrate the PIOSC.

PIOSCSTAT is shown in [Figure 4-24](#) and described in [Table 4-30](#).

Return to [Summary Table](#).

Figure 4-24. PIOSCSTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				DT			
R-0x0				R-0x40			
15	14	13	12	11	10	9	8
RESERVED						RESULT	
R-0x0						R-0x0	
7	6	5	4	3	2	1	0
RESERVED				CT			
R-0x0				R-0x40			

Table 4-30. PIOSCSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0x0	
22-16	DT	R	0x40	Default Trim Value. This field contains the default trim value. This value is loaded into the PIOSC after every full power up.
15-10	RESERVED	R	0x0	
9-8	RESULT	R	0x0	Calibration Result 0x0 = Calibration has not been attempted. 0x1 = The last calibration operation completed to meet 1% accuracy. 0x2 = The last calibration operation failed to meet 1% accuracy. 0x3 = Reserved
7	RESERVED	R	0x0	
6-0	CT	R	0x40	Calibration Trim Value. This field contains the trim value from the last calibration operation. After factory calibration, CT and DT are the same.

4.2.19 PLLFREQ0 Register (Offset = 0x160) [reset = 0x0]

PLL Frequency 0 (PLLFREQ0)

This register always contains the variables used to configure the PLL. If the PLL is reprogrammed, it must go through a relock sequence which is defined by the parameter t_{READY} in the device-specific data sheet. When controlling this register directly, software must change this value while the PLL is powered down. Writes to PLLFREQ0 are delayed from affecting the PLL until the RSCLKCFG register NEWFREQ bit is written with a 1.

The PLL frequency can be calculated using Equation 3.

$$f_{\text{VCO}} = (f_{\text{IN}} \times \text{MDIV})$$

where

- $f_{\text{IN}} = f_{\text{XTAL}} / (Q+1)(N+1)$ or $f_{\text{PIOSC}} / (Q+1)(N+1)$
 - $\text{MDIV} = \text{MINT} + (\text{MFRAC} / 1024)$
- (3)

The Q and N values are programmed in the PLLFREQ1 register. To reduce jitter, program MFRAC to 0x0.

PLLFREQ0 is shown in Figure 4-25 and described in Table 4-31.

Return to [Summary Table](#).

Figure 4-25. PLLFREQ0 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
PLLWPR	RESERVED			MFRAC			
R/W-0x0	R-0x0			R/W-0x0			
15	14	13	12	11	10	9	8
MFRAC						MINT	
R/W-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
MINT							
R/W-0x0							

Table 4-31. PLLFREQ0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23	PLLWPR	R/W	0x0	PLL Power. This bit controls power to the PLL. If set, the PLL power is applied and the PLL oscillates based on the values in the PLLFREQ0 and PLLFREQ1 registers.
22-20	RESERVED	R	0x0	
19-10	MFRAC	R/W	0x0	PLL M Fractional Value
9-0	MINT	R/W	0x0	PLL M Integer Value. This field contains the integer value of the PLL M value.

4.2.20 PLLFREQ1 Register (Offset = 0x164) [reset = 0x0]

PLL Frequency 1 (PLLFREQ1)

This register always contains the current Q and N values presented to the system PLL. If the PLL is reconfigured, it must go through a relock sequence, which requires approximately 128 PIOSC clocks. When controlling this register directly, software must change this value while the PLL is powered down. Writes to PLLFREQ0 are delayed from affecting the PLL until the RSCLKCFG register NEWFREQ bit is written as 1.

The MINT and MFRAC fields are present in the PLLFREQ0 register.

PLLFREQ1 is shown in [Figure 4-26](#) and described in [Table 4-32](#).

Return to [Summary Table](#).

Figure 4-26. PLLFREQ1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				Q				RESERVED				N			
R-0x0				R/W-0x0				R-0x0				R/W-0x0			

Table 4-32. PLLFREQ1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12-8	Q	R/W	0x0	PLL Q Value. This field contains the PLL Q value.
7-5	RESERVED	R	0x0	
4-0	N	R/W	0x0	PLL N Value. This field contains the PLL N value.

4.2.21 PLLSTAT Register (Offset = 0x168) [reset = 0x0]

PLL Status (PLLSTAT)

This register shows the direct status of the PLL lock.

PLLSTAT is shown in [Figure 4-27](#) and described in [Table 4-33](#).

Return to [Summary Table](#).

Figure 4-27. PLLSTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							LOCK
R-0x0							R-0x0

Table 4-33. PLLSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	LOCK	R	0x0	PLL Lock 0x0 = The PLL is unpowered or is not yet locked. 0x1 = The PLL powered and locked.

4.2.22 SLPPWRCFG Register (Offset = 0x188) [reset = 0x0]

Sleep Power Configuration (SLPPWRCFG)

This register provides configuration information for the power control of the SRAM and flash memory while in sleep mode.

SLPPWRCFG is shown in [Figure 4-28](#) and described in [Table 4-34](#).

Return to [Summary Table](#).

Figure 4-28. SLPPWRCFG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		FLASHPM		RESERVED		SRAMPM	
R-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 4-34. SLPPWRCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	FLASHPM	R/W	0x0	Flash Power Modes. 0x0 = Active mode. flash memory is not placed in a lower-power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in sleep mode. 0x1 = Reserved 0x2 = Low-power mode. flash memory is placed in low-power mode. This mode provides the lowers power consumption but requires more time to come out of sleep mode. 0x3 = Reserved
3-2	RESERVED	R	0x0	
1-0	SRAMPM	R/W	0x0	SRAM Power Modes. This field controls the low-power modes of the on-chip SRAM, including the USB SRAM while the microcontroller is in sleep mode. 0x0 = Active mode. SRAM is not placed in a lower-power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in sleep mode. 0x1 = Standby mode. SRAM is placed in standby mode while in sleep mode. 0x2 = Reserved 0x3 = Low-power mode. SRAM is placed in low-power mode. This mode provides the slowest time to sleep and wakeup but the lowest power consumption while in sleep mode.

4.2.23 DSLPPWRCFG Register (Offset = 0x18C) [reset = 0x0]

Deep-Sleep Power Configuration (DSLPPWRCFG)

This register provides configuration information for the power control of the SRAM and flash memory while in deep-sleep mode.

DSLPPWRCFG is shown in [Figure 4-29](#) and described in [Table 4-35](#).

Return to [Summary Table](#).

Figure 4-29. DSLPPWRCFG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						LDOSM	TSPD
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED		FLASHPM		RESERVED		SRAMPM	
R-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 4-35. DSLPPWRCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	LDOSM	R/W	0x0	LDO Sleep Mode 0x0 = LDO is disabled in sleep mode. 0x1 = LDO is placed in a low-power mode when deep-sleep mode is entered.
8	TSPD	R/W	0x0	Temperature Sense Power Down. This bit controls low-power mode for the internal temperature sensor in the ADC. 0x0 = Temperature sensor in the ADC is disabled in sleep-mode. 0x1 = The internal temperature sensor in the ADC is placed in a low-power mode when deep-sleep mode is entered.
7-6	RESERVED	R	0x0	
5-4	FLASHPM	R/W	0x0	Flash Power Modes. This field enables the flash to be placed in a low-power mode. See the device-specific data sheet for information regarding wake times from Sleep and deep sleep. If using the LFIOOSC as the deep-sleep clock source, FLASHPM = 0x2 must be used. If FLASHPM = 0x0 and the LFIOOSC is used, current could be higher and could vary. 0x0 = Active Mode. Flash memory is not placed in a lower-power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in deep-sleep mode. 0x1 = Reserved 0x2 = Low-power mode. Flash memory is placed in low-power mode. This mode provides the lowers power consumption but requires more time to come out of deep-sleep mode. 0x3 = Reserved
3-2	RESERVED	R	0x0	

Table 4-35. DSLPPWRCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	SRAMPM	R/W	0x0	<p>SRAM Power Modes.</p> <p>This field controls the low-power modes of the on-chip SRAM, including the USB SRAM while the microcontroller is in deep-sleep mode. See the device-specific data sheet for information regarding wake times from sleep and deep sleep.</p> <p>0x0 = Active Mode. SRAM is not placed in a lower-power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in deep-sleep mode.</p> <p>0x1 = Standby Mode. SRAM is place in standby mode while in deep-sleep mode.</p> <p>0x2 = Reserved</p> <p>0x3 = Low Power Mode. SRAM is placed in low-power mode. This mode provides the slowest time to sleep and wakeup but the lowest power consumption while in deep-sleep mode.</p>

4.2.24 NVMSTAT Register (Offset = 0x1A0) [reset = 0x1]

Non-Volatile Memory Information (NVMSTAT)

This register is predefined by the part and can be used to verify features.

NVMSTAT is shown in [Figure 4-30](#) and described in [Table 4-36](#).

Return to [Summary Table](#).

Figure 4-30. NVMSTAT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															FWB
R-0x0															R-0x1

Table 4-36. NVMSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	FWB	R	0x1	32 Word Flash Write Buffer Available. When set, indicates that the 32-word flash memory write buffer feature is available.

4.2.25 LDOSPCTL Register (Offset = 0x1B4) [reset = 0x18]

LDO Sleep Power Control (LDOSPCTL)

This register specifies the LDO output voltage in sleep mode. This register should be configured while in run mode. If the VADJEN bit is set, writes can be made to the VLDO field within the provided encodings. [Table 4-37](#) lists the maximum clock frequencies with respect to LDO voltage.

Table 4-37. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage

Operating Voltage (LDO)	Maximum System Clock Frequency	PIOSC
1.2 V	120 MHz	16 MHz
0.9 V	30 MHz	16 MHz

LDOSPCTL is shown in [Figure 4-31](#) and described in [Table 4-38](#).

Return to [Summary Table](#).

Figure 4-31. LDOSPCTL Register

31	30	29	28	27	26	25	24
VADJEN	RESERVED						
R/W-0x0	R-0x0						
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
VLDO							
R/W-0x18							

Table 4-38. LDOSPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	VADJEN	R/W	0x0	Voltage Adjust Enable. This bit enables the value of the VLDO field to be used to specify the output voltage of the LDO in sleep mode. 0x0 = The LDO output voltage is set to the factory default value in sleep mode. The value of the VLDO field does not affect the LDO operation. 0x1 = The LDO output value in sleep mode is configured by the value in the VLDO field.
30-8	RESERVED	R	0x0	
7-0	VLDO	R/W	0x18	LDO Output Voltage. This field provides program control of the LDO output voltage in sleep mode. The value of the field is used only for the LDO voltage when the VADJEN bit is set. When using the USB module, the LDO must be configured to 1.2 V. 0x12 = 0.90 V 0x13 = 0.95 V 0x14 = 1.00 V 0x15 = 1.05 V 0x16 = 1.10 V 0x17 = 1.15 V 0x18 = 1.20 V All other values are reserved.

4.2.26 LDOSPCAL Register (Offset = 0x1B8) [reset = 0x1818]

LDO Sleep Power Calibration (LDOSPCAL)

This register provides factory determined values that are recommended for the VLDO field in the LDOSPCTL register while in sleep mode. The reset value of this register cannot be determined until the product has been characterized.

LDOSPCAL is shown in [Figure 4-32](#) and described in [Table 4-39](#).

Return to [Summary Table](#).

Figure 4-32. LDOSPCAL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																WITHPLL								NOPLL							
R-0x0																R-0x18								R-0x18							

Table 4-39. LDOSPCAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-8	WITHPLL	R	0x18	Sleep with PLL. The value in this field is the suggested value for the VLDO field in the LDOSPCTL register when using the PLL. This value provides the lowest recommended LDO output voltage for use with the PLL at the maximum specified value.
7-0	NOPLL	R	0x18	Sleep without PLL. The value in this field is the suggested value for the VLDO field in the LDOSPCTL register when not using the PLL. This value provides the lowest recommended LDO output voltage for use without the PLL.

4.2.27 LDODPCTL Register (Offset = 0x1BC) [reset = 0x12]

LDO Deep-Sleep Power Control (LDODPCTL)

This register specifies the LDO output voltage in sleep mode. This register should be configured while in run mode. If the VADJEN bit is set, writes can be made to the VLDO field within the provided encodings. The following table shows the maximum clock frequencies with respect to LDO Voltage.

LDODPCTL is shown in [Figure 4-33](#) and described in [Table 4-40](#).

Return to [Summary Table](#).

Figure 4-33. LDODPCTL Register

31	30	29	28	27	26	25	24
VADJEN	RESERVED						
R/W-0x0				R-0x0			
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
VLDO							
R/W-0x12							

Table 4-40. LDODPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	VADJEN	R/W	0x0	Voltage Adjust Enable. This bit enables the value of the VLDO field to be used to specify the output voltage of the LDO in deep-sleep mode. 0x0 = The LDO output voltage is set to the factory default value in deep-sleep mode. The value of the VLDO field does not affect the LDO operation. 0x1 = The LDO output value in deep-sleep mode is configured by the value in the VLDO field.
30-8	RESERVED	R	0x0	
7-0	VLDO	R/W	0x12	LDO Output Voltage. This field provides program control of the LDO output voltage in deep-sleep mode. The value of the field is only used for the LDO voltage when the VADJEN bit is set. 0x12 = 0.90 V 0x13 = 0.95 V 0x14 = 1.00 V 0x15 = 1.05 V 0x16 = 1.10 V 0x17 = 1.15 V 0x18 = 1.20 V All other values are reserved.

4.2.28 LDODPCAL Register (Offset = 0x1C0) [reset = 0x1212]

LDO Deep-Sleep Power Calibration (LDODPCAL)

This register provides factory determined values that are recommended for the VLDO field in the LDODPCTL register while in deep-sleep mode. The reset value of this register cannot be determined until the product has been characterized.

LDODPCAL is shown in [Figure 4-34](#) and described in [Table 4-41](#).

Return to [Summary Table](#).

Figure 4-34. LDODPCAL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																NOPLL								30KHZ							
R-0x0																R-0x12								R-0x12							

Table 4-41. LDODPCAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-8	NOPLL	R	0x12	Deep-Sleep Without PLL. The value in this field is the suggested value for the VLDO field in the LDODPCTL register when not using the PLL. This value provides the lowest recommended LDO output voltage for use with the system clock.
7-0	30KHZ	R	0x12	Deep-Sleep With IOSC. The value in this field is the suggested value for the VLDO field in the LDODPCTL register when not using the PLL. This value provides the lowest recommended LDO output voltage for use with the low-frequency internal oscillator.

4.2.29 SDPMST Register (Offset = 0x1CC) [reset = 0x0]

Sleep / Deep-Sleep Power Mode Status (SDPMST)

This register provides status information on the Sleep and Deep-Sleep power modes as well as some real time status that can be viewed by a debugger or the core if it is running. These events do not trigger an interrupt and are meant to provide information that can help tune software for power management. The status register gets written at the beginning of every Dynamic Power Management event request with the results of any error checking. There is no mechanism to clear the bits; they are overwritten on the next event. The LDOUA, FLASHLP, LOWPWR, and PRACT bits provide real-time data and there are no events to register that information.

SDPMST is shown in [Figure 4-35](#) and described in [Table 4-42](#).

Return to [Summary Table](#).

Figure 4-35. SDPMST Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				LDOUA	FLASHLP	LOWPWR	PRACT
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
PPDW	LMAXERR	RESERVED	LSMINERR	LDMINERR	PPDERR	FPDERR	SPDERR
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 4-42. SDPMST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	LDOUA	R	0x0	LDO Update Active 0x0 = The LDO voltage level is not changing. 0x1 = The LDO voltage level is changing.
18	FLASHLP	R	0x0	Flash Memory in Low Power State 0x0 = The flash memory is currently in the active state. 0x1 = The flash memory is currently in the low power state as programmed in the SLPPWRCFG or DSLPPWRCFG register.
17	LOWPWR	R	0x0	Sleep or Deep-Sleep Mode 0x0 = The microcontroller is currently in run mode. 0x1 = The microcontroller is currently in sleep or deep-sleep mode and is waiting for an interrupt or is in the process of powering up. The status of this bit is not affected by the power state of the flash memory or SRAM.
16	PRACT	R	0x0	Sleep or Deep-Sleep Power Request Active 0x0 = A power request is not active. 0x1 = The microcontroller is currently in deep-sleep mode or is in sleep mode and a request to put the SRAM or flash memory into a lower-power mode is currently active as configured by the SLPPWRCFG register.
15-8	RESERVED	R	0x0	

Table 4-42. SDPMST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	PPDW	R	0x0	<p>PIOSC Power Down Request Warning</p> <p>0x0 = No error.</p> <p>0x1 = This bit indicates that the PIOSC was not powered off even though the PIOSCPD bit was set in the DSLCLKCFG register because the PIOSC was in use by a peripheral.</p>
6	LMAXERR	R	0x0	<p>VLDO Value Above Maximum Error</p> <p>0x0 = No error.</p> <p>0x1 = An error has occurred because software has requested that the LDO voltage be above the maximum value allowed using the VLDO bit in the LDOSPCTL, or LDODPCTL register. In this situation, the LDO is set to the factory default value.</p>
5	RESERVED	R	0x0	
4	LSMINERR	R	0x0	<p>VLDO Value Below Minimum Error in Sleep Mode</p> <p>0x0 = No error.</p> <p>0x1 = An error has occurred because software has requested that the LDO voltage be below the minimum value allowed using the VLDO bit in the LDOSPCTL register. In this situation, the LDO voltage is not changed when entering sleep mode.</p>
3	LDMINERR	R	0x0	<p>VLDO Value Below Minimum Error in Deep-Sleep Mode</p> <p>0x0 = No error.</p> <p>0x1 = An error has occurred because software has requested that the LDO voltage be below the minimum value allowed using the VLDO bit in the LDODPCTL register. In this situation, the LDO voltage is not changed when entering deep-sleep mode.</p>
2	PPDERR	R	0x0	<p>PIOSC Power Down Request Error</p> <p>0x0 = No error.</p> <p>0x1 = An error has occurred because software has requested that the PIOSC be powered down during deep sleep and it is not possible to power down the PIOSC. In this situation, the PIOSC is not powered down when entering deep-sleep mode.</p>
1	FPDERR	R	0x0	<p>Flash Memory Power Down Request Error</p> <p>0x0 = No error.</p> <p>0x1 = An error has occurred because software has requested a flash memory power down mode that is not available using the FLASHPM field in the SLPPWRCFG or the DSLPPWRCFG register.</p>
0	SPDERR	R	0x0	<p>SRAM Power Down Request Error</p> <p>0x0 = No error.</p> <p>0x1 = An error has occurred because software has requested an SRAM power down mode that is not available using the SRAMP field in the SLPPWRCFG or the DSLPPWRCFG register.</p>

4.2.30 RESBEHAVCTL Register (Offset = 0x1D8) [reset = 0x00FFFFFF]

Reset Behavior Control Register (RESBEHAVCTL)

The Reset Behavior Control Register contains system management controls.

The RESBEHAVCTL register effect occurs immediately when the register is changed. The next power-on reset sequence returns the reset value.

If any of the following bit fields are set to 0x3 when a reset occurs, a simulated POR is generated and the appropriate reset cause is set in the Reset Cause (RESC) register. During a simulated POR, registers are reloaded and the bootloader is executed. If a full POR is initiated, the POR bit in the RESC register is set and all other bits are cleared.

RESBEHAVCTL is shown in [Figure 4-36](#) and described in [Table 4-43](#).

Return to [Summary Table](#).

Figure 4-36. RESBEHAVCTL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0xFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								WDOG1		WDOG0		BOR		EXTRES	
R-0xFFFF								R/W-0x3		R/W-0x3		R/W-0x3		R/W-0x3	

Table 4-43. RESBEHAVCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0xFFFF	
7-6	WDOG1	R/W	0x3	Watchdog 1 Reset Operation 0x0 = Reserved. Default operation is performed. 0x2 = Watchdog 1 issues a system reset. 0x3 = Watchdog 1 issues a simulated POR sequence (default).
5-4	WDOG0	R/W	0x3	Watchdog 0 Reset Operation 0x0 = Reserved. Default operation is performed. 0x2 = Watchdog 0 issues a system reset. 0x3 = Watchdog 0 issues a simulated POR sequence (default).
3-2	BOR	R/W	0x3	BOR Reset operation. This field defines operation of BOR when the user has defined the BOR operation to be a reset. If the BOR operation is defined as an interrupt, this setting has no effect. 0x0 = Reserved. Default operation is performed. 0x2 = Brownout reset issues system reset. 0x3 = Brownout reset issues a simulated POR sequence (default).
1-0	EXTRES	R/W	0x3	External RST Pin Operation 0x0 = Reserved. Default operation is performed. 0x2 = External RST assertion issues a system reset. 0x3 = External RST assertion issues a simulated POR sequence (default).

4.2.31 HSSR Register (Offset = 0x1F4) [reset = 0x0]

Hardware System Service Request (HSSR)

The HSSR register is used to control system configuration functions, such as return-to-factory settings. A write to the HSSR register stores a command descriptor pointer (CDOFF) value if the KEY field is correct (0xCA). A successful write to this register also initiates a system reset. The initialization process executes before examining the HSSR register and processing the command. This register can only be accessed in privilege mode. See [Section 4.1.6.6](#) for more information on how to use the HSSR register.

HSSR is shown in [Figure 4-37](#) and described in [Table 4-44](#).

Return to [Summary Table](#).

Figure 4-37. HSSR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY								CDOFF																							
R0/W-0x0								R/W-0x0																							

Table 4-44. HSSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	KEY	R0/W	0x0	Write Key. When read, this field returns zero. When written, this field must contain the value of 0xCA to register the CDOFF field and initiate a HSSR request. Writes with KEY values other than 0xCA are ignored.
23-0	CDOFF	R/W	0x0	Command Descriptor Pointer. This field contains either the status result from the previous HSSR request, or it contains a (word-aligned) memory address where the command descriptor is located. If CDOFF = 0x000000, there is no request and no processing is performed. If CDOFF = 0xFFFFFFFF, the previous request through HSSR did not complete due to an error. Otherwise, CDOFF contains the offset for a data structure in SRAM.

4.2.32 USBPDS Register (Offset = 0x280) [reset = 0x3F]

USB Power Domain Status (USBPDS)

This register provides the status of power to the USB SRAM array.

NOTE: If the PWRCTL field in the USBMPC register is set to 0x3 and the power domain to the USB is turned off by writing 0 to the P0 bit of the PCUSB register, then the SRAM goes into retention and the MEMSTAT field of the USBPDS register reads as 0x1 (retention).

USBPDS is shown in [Figure 4-38](#) and described in [Table 4-45](#).

Return to [Summary Table](#).

Figure 4-38. USBPDS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		MEMSTAT		PWRSTAT	
R-0x0		R-0x3		R-0x3		R-0x3	

Table 4-45. USBPDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	RESERVED	R	0x3	
3-2	MEMSTAT	R	0x3	Memory Array Power Status. Displays status of USB SRAM. 0x0 = Array off 0x1 = SRAM retention 0x2 = Reserved 0x3 = Array on
1-0	PWRSTAT	R	0x3	Power Domain Status 0x0 = Off 0x1 = Reserved 0x2 = Reserved 0x3 = On

4.2.33 USBMPC Register (Offset = 0x284) [reset = 0x3]

USB Memory Power Control (USBMPC)

This register provides power control to the peripheral memory array.

NOTE: If the PWRCTL field of the USBMPC register is set to 0x3 and the power domain to the USB is turned off by writing 0 to the P0 bit of the PCUSB register, then the SRAM goes into retention and the MEMSTAT field of the USBPDS register reads as 0x1 (retention).

USBMPC is shown in [Figure 4-39](#) and described in [Table 4-46](#).

Return to [Summary Table](#).

Figure 4-39. USBMPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PWRCTL	
R-0x0														R/W-0x3	

Table 4-46. USBMPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1-0	PWRCTL	R/W	0x3	Memory Array Power Control. Allows multiple levels of power control in the peripheral SRAM space. 0x0 = Array off 0x1 = SRAM retention 0x2 = Reserved 0x3 = Array on

4.2.34 EMACPDS Register (Offset = 0x288) [reset = 0x3F]

Ethernet MAC Power Domain Status (EMACPDS)

This register provides the status of power to the EMAC SRAM array.

NOTE: The EMAC memory array does not support retention and can only be turned on and off. Memory array off is supported only when the power domain is off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to the EMAC is subsequently removed by clearing the P0 bit of the PCEMAC register, the event causes the memory array to turn off and the MEMSTAT bit in the EMACPDS register to be 0x0 (array off).

EMACPDS is shown in [Figure 4-40](#) and described in [Table 4-47](#).

Return to [Summary Table](#).

Figure 4-40. EMACPDS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		MEMSTAT		PWRSTAT	
R-0x0		R-0x3		R-0x3		R-0x3	

Table 4-47. EMACPDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	RESERVED	R	0x3	
3-2	MEMSTAT	R	0x3	Memory Array Power Status. Displays the status of the EMAC SRAM. 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on
1-0	PWRSTAT	R	0x3	Power Domain Status 0x0 = Off 0x1 = Reserved 0x2 = Reserved 0x3 = On

4.2.35 EMACMPC Register (Offset = 0x28C) [reset = 0x3]

Ethernet MAC Memory Power Control (EMACMPC)

This register provides power control to the peripheral memory array.

NOTE: The EMAC memory array does not support retention and can only be turned on and off. Memory array off is supported only when the power domain is off. If the memory array is turned on (PWRCTL = 0x3) and the power control to the EMAC is removed by clearing the P0 bit of the PCEMAC register, the memory array is turned off and the MEMSTAT bit in the EMACPDS register is 0x0.

EMACMPC is shown in [Figure 4-41](#) and described in [Table 4-48](#).

Return to [Summary Table](#).

Figure 4-41. EMACMPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PWRCTL	
R-0x0														R/W-0x3	

Table 4-48. EMACMPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1-0	PWRCTL	R/W	0x3	Memory Array Power Control 0x0 = Array off. Array off mode is supported only when the P0 bit of the PCEMAC register at offset 0x99C is set to 0. 0x1 = Reserved 0x2 = Reserved 0x3 = Array on

4.2.36 LCDPDS Register (Offset = 0x290) [reset = 0x3F]

LCD Power Domain Status (LCDPDS)

This register provides the status of power to the LCD SRAM array.

NOTE: The LCD memory array does not support retention and can only be turned on and off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to the LCD is subsequently removed by clearing the P0 bit of the PCLCD register, the event causes the memory array to turn off and the MEMSTAT bit in the LCDPDS register to be 0x0 (array off).

LCDPDS is shown in [Figure 4-42](#) and described in [Table 4-49](#).

Return to [Summary Table](#).

Figure 4-42. LCDPDS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		MEMSTAT		PWRSTAT	
R-0x0		R-0x3		R-0x3		R-0x3	

Table 4-49. LCDPDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	RESERVED	R	0x3	
3-2	MEMSTAT	R	0x3	Memory Array Power Status. Displays status of LCD SRAM. 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on
1-0	PWRSTAT	R	0x3	Power Domain Status 0x0 = Off 0x1 = Reserved 0x2 = Reserved 0x3 = On

4.2.37 LCDMPC Register (Offset = 0x294) [reset = 0x3]

LCD Memory Power Control (LCDMPC)

This register provides power control to the peripheral memory array.

NOTE: The LCD memory array does not support retention and can only be turned on and off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to the LCD is subsequently removed by clearing the P0 bit of the PCLCD register, the event causes the memory array to turn off and the MEMSTAT bit in the LCDPDS register to be 0x0 (array off).

LCDMPC is shown in [Figure 4-43](#) and described in [Table 4-50](#).

Return to [Summary Table](#).

Figure 4-43. LCDMPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PWRCTL	
R-0x0														R/W-0x3	

Table 4-50. LCDMPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1-0	PWRCTL	R/W	0x3	Memory Array Power Control. Allows multiple levels of power control in peripheral's SRAM space 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on

4.2.38 CAN0PDS Register (Offset = 0x298) [reset = 0x3F]

CAN 0 Power Domain Status (CAN0PDS)

This register provides the status of power to the CAN0 SRAM array.

NOTE: The CAN0 memory array does not support retention and can only be turned on and off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to CAN0 is subsequently removed by clearing the P0 bit of the PCCAN register, the event causes the memory array to turn off and the MEMSTAT bit in the CAN0PDS register to be 0x0 (array off).

CAN0PDS is shown in [Figure 4-44](#) and described in [Table 4-51](#).

Return to [Summary Table](#).

Figure 4-44. CAN0PDS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		MEMSTAT		PWRSTAT	
R-0x0		R-0x3		R-0x3		R-0x3	

Table 4-51. CAN0PDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	RESERVED	R	0x3	
3-2	MEMSTAT	R	0x3	Memory Array Power Status. Displays status of the CAN0 SRAM. 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on
1-0	PWRSTAT	R	0x3	Power Domain Status 0x0 = Off 0x1 = Reserved 0x2 = Reserved 0x3 = On

4.2.39 CAN0MPC Register (Offset = 0x29C) [reset = 0x3]

CAN 0 Memory Power Control (CAN0MPC)

This register provides power control to the peripheral memory array.

NOTE: The CAN0 memory array does not support retention and can only be turned on and off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to CAN0 is subsequently removed by clearing the P0 bit of the PCCAN register, the event causes the memory array to turn off and the MEMSTAT bit in the CAN0PDS register to be 0x0 (array off).

CAN0MPC is shown in [Figure 4-45](#) and described in [Table 4-52](#).

Return to [Summary Table](#).

Figure 4-45. CAN0MPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PWRCTL	
R-0x0														R/W-0x3	

Table 4-52. CAN0MPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1-0	PWRCTL	R/W	0x3	Memory Array Power Control. Allows multiple levels of power control in peripheral SRAM space. 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on

4.2.40 CAN1PDS Register (Offset = 0x2A0) [reset = 0x3F]

CAN 1 Power Domain Status (CAN1PDS)

This register provides the status of power to the CAN 1 SRAM array.

NOTE: The CAN1 memory array does not support retention and can only be turned on and off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to CAN1 is subsequently removed by clearing the P1 bit of the PCCAN register, the event causes the memory array to turn off and the MEMSTAT bit in the CAN1PDS register to be 0x0 (array off).

CAN1PDS is shown in [Figure 4-46](#) and described in [Table 4-53](#).

Return to [Summary Table](#).

Figure 4-46. CAN1PDS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		MEMSTAT		PWRSTAT	
R-0x0		R-0x3		R-0x3		R-0x3	

Table 4-53. CAN1PDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	RESERVED	R	0x3	
3-2	MEMSTAT	R	0x3	Memory Array Power Status. Displays status of CAN1 SRAM. 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on
1-0	PWRSTAT	R	0x3	Power Domain Status 0x0 = Off 0x1 = Reserved 0x2 = Reserved 0x3 = On

4.2.41 CAN1MPC Register (Offset = 0x2A4) [reset = 0x3]

CAN 1 Memory Power Control (CAN1MPC)

This register provides power control to the peripheral memory array.

NOTE: The CAN1 memory array does not support retention and can only be turned on and off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to CAN1 is subsequently removed by clearing the P1 bit of the PCCAN register, the event causes the memory array to turn off and the MEMSTAT bit in the CAN1PDS register to be 0x0 (array off).

CAN1MPC is shown in [Figure 4-47](#) and described in [Table 4-54](#).

Return to [Summary Table](#).

Figure 4-47. CAN1MPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PWRCTL	
R-0x0														R/W-0x3	

Table 4-54. CAN1MPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1-0	PWRCTL	R/W	0x3	Memory Array Power Control. Allows multiple levels of power control in peripheral SRAM space. 0x0 = Array off 0x1 = Reserved 0x2 = Reserved 0x3 = Array on

4.2.42 PPWD Register (Offset = 0x300) [reset = 0x3]

Watchdog Timer Peripheral Present (PPWD)

The PPWD register provides software information regarding the watchdog modules.

NOTE: This register reports which watchdog timers are implemented on this microcontroller.

PPWD is shown in [Figure 4-48](#) and described in [Table 4-55](#).

Return to [Summary Table](#).

Figure 4-48. PPWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														P1	P0
R-0x0														R-0x1	R-0x1

Table 4-55. PPWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	P1	R	0x1	Watchdog Timer 1 Present 0x0 = Watchdog module 1 is not present. 0x1 = Watchdog module 1 is present.
0	P0	R	0x1	Watchdog Timer 0 Present 0x0 = Watchdog module 0 is not present. 0x1 = Watchdog module 0 is present.

4.2.43 PPTIMER Register (Offset = 0x304) [reset = 0xFF]

16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER)

The PPTIMER register provides software information regarding the 16/32-bit general-purpose timer modules.

NOTE: This register reports which timers are implemented on this microcontroller.

PPTIMER is shown in [Figure 4-49](#) and described in [Table 4-56](#).

Return to [Summary Table](#).

Figure 4-49. PPTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								P7	P6	P5	P4	P3	P2	P1	P0
R-0x0								R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1

Table 4-56. PPTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	P7	R	0x1	16/32-Bit General-Purpose Timer 7 Present 0x0 = 16/32-bit general-purpose timer module 7 is not present. 0x1 = 16/32-bit general-purpose timer module 7 is present.
6	P6	R	0x1	16/32-Bit General-Purpose Timer 6 Present 0x0 = 16/32-bit general-purpose timer module 6 is not present. 0x1 = 16/32-bit general-purpose timer module 6 is present.
5	P5	R	0x1	16/32-Bit General-Purpose Timer 5 Present 0x0 = 16/32-bit general-purpose timer module 5 is not present. 0x1 = 16/32-bit general-purpose timer module 5 is present.
4	P4	R	0x1	16/32-Bit General-Purpose Timer 4 Present 0x0 = 16/32-bit general-purpose timer module 4 is not present. 0x1 = 16/32-bit general-purpose timer module 4 is present.
3	P3	R	0x1	16/32-Bit General-Purpose Timer 3 Present 0x0 = 16/32-bit general-purpose timer module 3 is not present. 0x1 = 16/32-bit general-purpose timer module 3 is present.
2	P2	R	0x1	16/32-Bit General-Purpose Timer 2 Present 0x0 = 16/32-bit general-purpose timer module 2 is not present. 0x1 = 16/32-bit general-purpose timer module 2 is present.
1	P1	R	0x1	16/32-Bit General-Purpose Timer 1 Present 0x0 = 16/32-bit general-purpose timer module 1 is not present. 0x1 = 16/32-bit general-purpose timer module 1 is present.
0	P0	R	0x1	16/32-Bit General-Purpose Timer 0 Present 0x0 = 16/32-bit general-purpose timer module 0 is not present. 0x1 = 16/32-bit general-purpose timer module 0 is present.

4.2.44 PPGPIO Register (Offset = 0x308) [reset = 0x3FFFF]

General-Purpose Input/Output Peripheral Present (PPGPIO)

The PPGPIO register provides software information regarding the general-purpose input/output modules.

NOTE: This register reports which GPIO ports are implemented on this microcontroller.

PPGPIO is shown in [Figure 4-50](#) and described in [Table 4-57](#).

Return to [Summary Table](#).

Figure 4-50. PPGPIO Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED														P17	P16
R-0x0														R-0x1	R-0x1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1

Table 4-57. PPGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	X	
17	P17	R	X	GPIO Port T Present 0x0 = GPIO port T is not present. 0x1 = GPIO port T is present.
16	P16	R	X	GPIO Port S Present 0x0 = GPIO port S is not present. 0x1 = GPIO port S is present.
15	P15	R	X	GPIO Port R Present 0x0 = GPIO port R is not present. 0x1 = GPIO port R is present.
14	P14	R	X	GPIO Port Q Present 0x0 = GPIO port Q is not present. 0x1 = GPIO port Q is present.
13	P13	R	X	GPIO Port P Present 0x0 = GPIO port P is not present. 0x1 = GPIO port P is present.
12	P12	R	X	GPIO Port N Present 0x0 = GPIO port N is not present. 0x1 = GPIO port N is present.
11	P11	R	X	GPIO Port M Present 0x0 = GPIO port M is not present. 0x1 = GPIO port M is present.
10	P10	R	X	GPIO Port L Present 0x0 = GPIO port L is not present. 0x1 = GPIO port L is present.
9	P9	R	X	GPIO Port K Present 0x0 = GPIO port K is not present. 0x1 = GPIO port K is present.
8	P8	R	X	GPIO Port J Present 0x0 = GPIO port J is not present. 0x1 = GPIO port J is present.

Table 4-57. PPGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	P7	R	0x1	GPIO Port H Present 0x0 = GPIO port H is not present. 0x1 = GPIO port H is present.
6	P6	R	0x1	GPIO Port G Present 0x0 = GPIO port G is not present. 0x1 = GPIO port G is present.
5	P5	R	0x1	GPIO Port F Present 0x0 = GPIO port F is not present. 0x1 = GPIO port F is present.
4	P4	R	0x1	GPIO Port E Present 0x0 = GPIO port E is not present. 0x1 = GPIO port E is present.
3	P3	R	0x1	GPIO Port D Present 0x0 = GPIO port D is not present. 0x1 = GPIO port D is present.
2	P2	R	0x1	GPIO Port C Present 0x0 = GPIO port C is not present. 0x1 = GPIO port C is present.
1	P1	R	0x1	GPIO Port B Present 0x0 = GPIO port B is not present. 0x1 = GPIO port B is present.
0	P0	R	0x1	GPIO Port A Present 0x0 = GPIO port A is not present. 0x1 = GPIO port A is present.

4.2.45 PPDMA Register (Offset = 0x30C) [reset = 0x1]

Micro Direct Memory Access Peripheral Present (PPDMA)

The PPDMA register provides software information regarding the μ DMA module.

NOTE: This register reports if the μ DMA module is implemented on this microcontroller.

PPDMA is shown in [Figure 4-51](#) and described in [Table 4-58](#).

Return to [Summary Table](#).

Figure 4-51. PPDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-58. PPDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	μ DMA Module Present 0x0 = μ DMA module is not present. 0x1 = μ DMA module is present.

4.2.46 PPEPI Register (Offset = 0x310) [reset = 0x1]

EPI Peripheral Present (PPEPI)

The PPEPI register provides software information regarding the EPI module.

NOTE: This register reports if the EPI module is implemented on this microcontroller.

PPEPI is shown in [Figure 4-52](#) and described in [Table 4-59](#).

Return to [Summary Table](#).

Figure 4-52. PPEPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-59. PPEPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	EPI Module Present 0x0 = EPI module is not present. 0x1 = EPI module is present.

4.2.47 PPHIB Register (Offset = 0x314) [reset = 0x01]

Hibernation Peripheral Present (PPHIB)

The PPHIB register provides software information regarding the Hibernation module.

NOTE: This register reports if the Hibernation module is implemented on this microcontroller.

PPHIB is shown in [Figure 4-53](#) and described in [Table 4-60](#).

Return to [Summary Table](#).

Figure 4-53. PPHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-60. PPHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	Hibernation Module Present 0x0 = Hibernation module is not present. 0x1 = Hibernation module is present.

4.2.48 PPUART Register (Offset = 0x318) [reset = 0xFF]

Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART)

The PPUART register provides software information regarding the UART modules.

NOTE: This register reports which UART modules are implemented on this microcontroller.

PPUART is shown in [Figure 4-54](#) and described in [Table 4-61](#).

Return to [Summary Table](#).

Figure 4-54. PPUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								P7	P6	P5	P4	P3	P2	P1	P0
R-0x0								R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1

Table 4-61. PPUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	P7	R	0x1	UART Module 7 Present 0x0 = UART module 7 is not present. 0x1 = UART module 7 is present.
6	P6	R	0x1	UART Module 6 Present 0x0 = UART module 6 is not present. 0x1 = UART module 6 is present.
5	P5	R	0x1	UART Module 5 Present 0x0 = UART module 5 is not present. 0x1 = UART module 5 is present.
4	P4	R	0x1	UART Module 4 Present 0x0 = UART module 4 is not present. 0x1 = UART module 4 is present.
3	P3	R	0x1	UART Module 3 Present 0x0 = UART module 3 is not present. 0x1 = UART module 3 is present.
2	P2	R	0x1	UART Module 2 Present 0x0 = UART module 2 is not present. 0x1 = UART module 2 is present.
1	P1	R	0x1	UART Module 1 Present 0x0 = UART module 1 is not present. 0x1 = UART module 1 is present.
0	P0	R	0x1	UART Module 0 Present 0x0 = UART module 0 is not present. 0x1 = UART module 0 is present.

4.2.49 PPSSI Register (Offset = 0x31C) [reset = 0xF]

Synchronous Serial Interface Peripheral Present (PPSSI)

The PPSSI register provides software information regarding the SSI modules.

NOTE: This register reports which SSI modules are implemented on this microcontroller. However, to support legacy software, the DC2 register is available. A read of the DC2 register correctly identifies if a legacy SSI module is present. Software must use this register to determine if a module that is not supported by the DC2 register is present.

PPSSI is shown in [Figure 4-55](#) and described in [Table 4-62](#).

Return to [Summary Table](#).

Figure 4-55. PPSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												P3	P2	P1	P0
R-0x0												R-0x1	R-0x1	R-0x1	R-0x1

Table 4-62. PPSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	P3	R	0x1	SSI Module 3 Present 0x0 = SSI module 3 is not present. 0x1 = SSI module 3 is present.
2	P2	R	0x1	SSI Module 2 Present 0x0 = SSI module 2 is not present. 0x1 = SSI module 2 is present.
1	P1	R	0x1	SSI Module 1 Present 0x0 = SSI module 1 is not present. 0x1 = SSI module 1 is present.
0	P0	R	0x1	SSI Module 0 Present 0x0 = SSI module 0 is not present. 0x1 = SSI module 0 is present.

4.2.50 PPI2C Register (Offset = 0x320) [reset = 0x3FF]

Inter-Integrated Circuit Peripheral Present (PPI2C)

The PPI2C register provides software information regarding the I2C modules.

NOTE: This register reports which I2C modules are implemented on this microcontroller.

PPI2C is shown in [Figure 4-56](#) and described in [Table 4-63](#).

Return to [Summary Table](#).

Figure 4-56. PPI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
R-0x0						R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1	R-0x1

Table 4-63. PPI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	P9	R	0x1	I2C Module 9 Present 0x0 = I2C module 9 is not present. 0x1 = I2C module 9 is present.
8	P8	R	0x1	I2C Module 8 Present 0x0 = I2C module 8 is not present. 0x1 = I2C module 8 is present.
7	P7	R	0x1	I2C Module 7 Present 0x0 = I2C module 7 is not present. 0x1 = I2C module 7 is present.
6	P6	R	0x1	I2C Module 6 Present 0x0 = I2C module 6 is not present. 0x1 = I2C module 6 is present.
5	P5	R	0x1	I2C Module 5 Present 0x0 = I2C module 5 is not present. 0x1 = I2C module 5 is present.
4	P4	R	0x1	I2C Module 4 Present 0x0 = I2C module 4 is not present. 0x1 = I2C module 4 is present.
3	P3	R	0x1	I2C Module 3 Present 0x0 = I2C module 3 is not present. 0x1 = I2C module 3 is present.
2	P2	R	0x1	I2C Module 2 Present 0x0 = I2C module 2 is not present. 0x1 = I2C module 2 is present.
1	P1	R	0x1	I2C Module 1 Present 0x0 = I2C module 1 is not present. 0x1 = I2C module 1 is present.
0	P0	R	0x1	I2C Module 0 Present 0x0 = I2C module 0 is not present. 0x1 = I2C module 0 is present.

4.2.51 PPUSB Register (Offset = 0x328) [reset = 0x1]

Universal Serial Bus Peripheral Present (PPUSB)

The PPUSB register provides software information regarding the USB module.

NOTE: This register reports if the USB module is implemented on this microcontroller.

PPUSB is shown in [Figure 4-57](#) and described in [Table 4-64](#).

Return to [Summary Table](#).

Figure 4-57. PPUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-64. PPUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	USB Module Present 0x0 = USB module is not present. 0x1 = USB module is present.

4.2.52 PPEPHY Register (Offset = 0x330) [reset = 0x1]

Ethernet PHY Peripheral Present (PPEPHY)

The PPEPHY register provides software information regarding the Ethernet PHY module.

NOTE: This register reports if the Ethernet PHY module is implemented on this microcontroller.

PPEPHY is shown in [Figure 4-58](#) and described in [Table 4-65](#).

Return to [Summary Table](#).

Figure 4-58. PPEPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-65. PPEPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	Ethernet PHY Module Present 0x0 = Ethernet PHY module is not present. 0x1 = Ethernet PHY module is present.

4.2.53 PPCAN Register (Offset = 0x334) [reset = 0x3]

Controller Area Network Peripheral Present (PPCAN)

The PPCAN register provides software information regarding the CAN modules.

NOTE: This register reports which CAN modules are implemented on this microcontroller.

PPCAN is shown in [Figure 4-59](#) and described in [Table 4-66](#).

Return to [Summary Table](#).

Figure 4-59. PPCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														P1	P0
R-0x0														R-0x1	R-0x1

Table 4-66. PPCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	P1	R	0x1	CAN Module 1 Present 0x0 = CAN module 1 is not present. 0x1 = CAN module 1 is present.
0	P0	R	0x1	CAN Module 0 Present 0x0 = CAN module 0 is not present. 0x1 = CAN module 0 is present.

4.2.54 PPADC Register (Offset = 0x338) [reset = 0x3]

Analog-to-Digital Converter Peripheral Present (PPADC)

The PPADC register provides software information regarding the ADC modules.

NOTE: This register reports which ADC modules are implemented on this microcontroller.

PPADC is shown in [Figure 4-60](#) and described in [Table 4-67](#).

[Return to Summary Table.](#)

Figure 4-60. PPADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														P1	P0
R-0x0														R-0x1	R-0x1

Table 4-67. PPADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	P1	R	0x1	ADC Module 1 Present 0x0 = ADC module 1 is not present. 0x1 = ADC module 1 is present.
0	P0	R	0x1	ADC Module 0 Present 0x0 = ADC module 0 is not present. 0x1 = ADC module 0 is present.

4.2.55 PPACMP Register (Offset = 0x33C) [reset = 0x1]

Analog Comparator Peripheral Present (PPACMP)

The PPACMP register provides software information regarding the analog comparator module.

NOTE: This register reports if the analog comparator module is implemented on this microcontroller.

The Analog Comparator Peripheral Properties (ACMPPP) register indicates how many analog comparator blocks are included in the module.

PPACMP is shown in [Figure 4-61](#) and described in [Table 4-68](#).

Return to [Summary Table](#).

Figure 4-61. PPACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-68. PPACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	Analog Comparator Module Present 0x0 = Analog comparator module is not present. 0x1 = Analog comparator module is present.

4.2.56 PPPWM Register (Offset = 0x340) [reset = 0x1]

Pulse Width Modulator Peripheral Present (PPPWM)

The PPPWM register provides software information regarding the PWM modules.

NOTE: This register reports which PWM modules are implemented on this microcontroller.

PPPWM is shown in [Figure 4-62](#) and described in [Table 4-69](#).

Return to [Summary Table](#).

Figure 4-62. PPPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-69. PPPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	PWM Module 0 Present 0x0 = PWM module 0 is not present. 0x1 = PWM module 0 is present.

4.2.57 PPQEI Register (Offset = 0x344) [reset = 0x1]

Quadrature Encoder Interface Peripheral Present (PPQEI)

The PPQEI register provides software information regarding the QEI modules.

NOTE: This register reports which QEI modules are implemented on this microcontroller.

PPQEI is shown in [Figure 4-63](#) and described in [Table 4-70](#).

Return to [Summary Table](#).

Figure 4-63. PPQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-70. PPQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	QEI Module 0 Present 0x0 = QEI module 0 is not present. 0x1 = QEI module 0 is present.

4.2.58 PEEPROM Register (Offset = 0x358) [reset = 0x01]

EEPROM Peripheral Present (PEEPROM)

The PEEPROM register provides software information regarding the EEPROM module.

PEEPROM is shown in [Figure 4-64](#) and described in [Table 4-71](#).

Return to [Summary Table](#).

Figure 4-64. PEEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-71. PEEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	EEPROM 0 Module Present 0x0 = EEPROM module is not present. 0x1 = EEPROM module is present.

4.2.59 PPCCM Register (Offset = 0x374) [reset = 0x01]

CRC and Cryptographic Modules Peripheral Present (PPCCM)

The PPCCM register provides software information regarding the CRC and Cryptographic Modules (AES, DES, and SHA).

NOTE: This register reports if the CRC and Cryptographic Modules (AES, DES, SHA/MD5) are implemented on this microcontroller.

PPCCM is shown in [Figure 4-65](#) and described in [Table 4-72](#).

Return to [Summary Table](#).

Figure 4-65. PPCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-72. PPCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	CRC and Cryptographic Modules Present 0x0 = The CRC, AES, DES, and SHA/MD modules are not present. 0x1 = The CRC, AES, DES, and SHA/MD modules are present.

4.2.60 PPLCD Register (Offset = 0x390) [reset = 0x01]

LCD Peripheral Present (PPLCD)

The PPLCD register provides software information regarding the LCD module.

NOTE: This register reports if an LCD controller is implemented on this microcontroller.

PPLCD is shown in [Figure 4-66](#) and described in [Table 4-73](#).

Return to [Summary Table](#).

Figure 4-66. PPLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-73. PPLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	LCD Module Present 0x0 = LCD module is not present. 0x1 = LCD module is present.

4.2.61 PPOWIRE Register (Offset = 0x398) [reset = 0x01]

1-Wire Peripheral Present (PPOWIRE)

The PPOWIRE register provides software information regarding the 1-Wire module.

NOTE: This register reports which 1-Wire modules are implemented on this microcontroller.

PPOWIRE is shown in [Figure 4-67](#) and described in [Table 4-74](#).

Return to [Summary Table](#).

Figure 4-67. PPOWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-74. PPOWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	1-Wire Module Present 0x0 = 1-Wire module is not present. 0x1 = 1-Wire module is present.

4.2.62 PPEMAC Register (Offset = 0x39C) [reset = 0x01]

Ethernet MAC Peripheral Present (PPEMAC)

The PPEMAC register provides software information regarding the Ethernet controller module.

NOTE: This register reports which Ethernet controller modules are implemented on this microcontroller.

PPEMAC is shown in [Figure 4-68](#) and described in [Table 4-75](#).

Return to [Summary Table](#).

Figure 4-68. PPEMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x1

Table 4-75. PPEMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x1	Ethernet Controller Module Present 0x0 = Ethernet Controller MAC module is not present. 0x1 = Ethernet Controller MAC module is present.

4.2.63 PPPRB Register (Offset = 0x3A0) [reset = 0x00]

Power Regulator Bus Peripheral Present (PPPRB)

The PPPRB register provides software information regarding the Power Regulator Bus module.

NOTE: This register reports which Power Regulator Bus modules are implemented on this microcontroller.

PPPRB is shown in [Figure 4-69](#) and described in [Table 4-76](#).

Return to [Summary Table](#).

Figure 4-69. PPPRB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R-0x0

Table 4-76. PPPRB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R	0x0	PRB Module Present 0x0 = PRB module is not present. 0x1 = PRB module is present.

4.2.64 SRWD Register (Offset = 0x500) [reset = 0x00]

Watchdog Timer Software Reset (SRWD)

The SRWD register lets software reset the available watchdog modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRWD register. While the SRWD bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRWD bit.

There may be latency from the clearing of the SRWD bit to when the peripheral is ready for use. Software should check the corresponding PRWD bit to verify that the Watchdog Timer module registers are ready to be accessed.

NOTE: Use this register to reset the watchdog modules.

SRWD is shown in [Figure 4-70](#) and described in [Table 4-77](#).

Return to [Summary Table](#).

Figure 4-70. SRWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R/W-0x0	R/W-0x0

Table 4-77. SRWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R/W	0x0	Watchdog Timer 1 Software Reset 0x0 = Watchdog module 1 is not reset. 0x1 = Watchdog module 1 is reset.
0	R0	R/W	0x0	Watchdog Timer 0 Software Reset 0x0 = Watchdog module 0 is not reset. 0x1 = Watchdog module 0 is reset.

4.2.65 SRTIMER Register (Offset = 0x504) [reset = 0x00]

16/32-Bit General-Purpose Timer Software Reset (SRTIMER)

The SRTIMER register lets software reset the available 16/32-bit timer modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRTIMER register. While the SRTIMER bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRTIMER bit.

There may be latency from the clearing of the SRTIMER bit to when the peripheral is ready for use. Software should check the corresponding PRTIMER bit to verify that the Timer module registers are ready to be accessed.

NOTE: Use this register to reset the timer modules.

SRTIMER is shown in [Figure 4-71](#) and described in [Table 4-78](#).

Return to [Summary Table](#).

Figure 4-71. SRTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								R7	R6	R5	R4	R3	R2	R1	R0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-78. SRTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	R7	R/W	0x0	16/32-Bit General-Purpose Timer 7 Software Reset 0x0 = 16/32-bit general-purpose timer module 7 is not reset. 0x1 = 16/32-bit general-purpose timer module 7 is reset.
6	R6	R/W	0x0	16/32-Bit General-Purpose Timer 6 Software Reset 0x0 = 16/32-bit general-purpose timer module 6 is not reset. 0x1 = 16/32-bit general-purpose timer module 6 is reset.
5	R5	R/W	0x0	16/32-Bit General-Purpose Timer 5 Software Reset 0x0 = 16/32-bit general-purpose timer module 5 is not reset. 0x1 = 16/32-bit general-purpose timer module 5 is reset.
4	R4	R/W	0x0	16/32-Bit General-Purpose Timer 4 Software Reset 0x0 = 16/32-bit general-purpose timer module 4 is not reset. 0x1 = 16/32-bit general-purpose timer module 4 is reset.
3	R3	R/W	0x0	16/32-Bit General-Purpose Timer 3 Software Reset 0x0 = 16/32-bit general-purpose timer module 3 is not reset. 0x1 = 16/32-bit general-purpose timer module 3 is reset.
2	R2	R/W	0x0	16/32-Bit General-Purpose Timer 2 Software Reset 0x0 = 16/32-bit general-purpose timer module 2 is not reset. 0x1 = 16/32-bit general-purpose timer module 2 is reset.
1	R1	R/W	0x0	16/32-Bit General-Purpose Timer 1 Software Reset 0x0 = 16/32-bit general-purpose timer module 1 is not reset. 0x1 = 16/32-bit general-purpose timer module 1 is reset.
0	R0	R/W	0x0	16/32-Bit General-Purpose Timer 0 Software Reset 0x0 = 16/32-bit general-purpose timer module 0 is not reset. 0x1 = 16/32-bit general-purpose timer module 0 is reset.

4.2.66 SRGPIO Register (Offset = 0x508) [reset = 0x00]

General-Purpose Input/Output Software Reset (SRGPIO)

The SRGPIO register lets software reset the available GPIO modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRGPIO register. While the SRGPIO bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRGPIO bit.

There may be latency from the clearing of the SRGPIO bit to when the peripheral is ready for use. Software should check the corresponding PRGPIO bit to verify that the GPIO module registers are ready to be accessed.

NOTE: Use this register to reset the GPIO modules.

SRGPIO is shown in [Figure 4-72](#) and described in [Table 4-79](#).

Return to [Summary Table](#).

Figure 4-72. SRGPIO Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED														R17	R16
R-0x0														R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-79. SRGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	R17	R/W	0x0	GPIO Port T Software Reset 0x0 = GPIO port T is not reset. 0x1 = GPIO port T is reset.
16	R16	R/W	0x0	GPIO Port S Software Reset 0x0 = GPIO port S is not reset. 0x1 = GPIO port S is reset.
15	R15	R/W	0x0	GPIO Port R Software Reset 0x0 = GPIO port R is not reset. 0x1 = GPIO port R is reset.
14	R14	R/W	0x0	GPIO Port Q Software Reset 0x0 = GPIO port Q is not reset. 0x1 = GPIO port Q is reset.
13	R13	R/W	0x0	GPIO Port P Software Reset 0x0 = GPIO port P is not reset. 0x1 = GPIO port P is reset.
12	R12	R/W	0x0	GPIO Port N Software Reset 0x0 = GPIO port N is not reset. 0x1 = GPIO port N is reset.
11	R11	R/W	0x0	GPIO Port M Software Reset 0x0 = GPIO port M is not reset. 0x1 = GPIO port M is reset.

Table 4-79. SRGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	R10	R/W	0x0	GPIO Port L Software Reset 0x0 = GPIO port L is not reset. 0x1 = GPIO port L is reset.
9	R9	R/W	0x0	GPIO Port K Software Reset 0x0 = GPIO port K is not reset. 0x1 = GPIO port K is reset.
8	R8	R/W	0x0	GPIO Port J Software Reset 0x0 = GPIO port J is not reset. 0x1 = GPIO port J is reset.
7	R7	R/W	0x0	GPIO Port H Software Reset 0x0 = GPIO port H is not reset. 0x1 = GPIO port H is reset.
6	R6	R/W	0x0	GPIO Port G Software Reset 0x0 = GPIO port G is not reset. 0x1 = GPIO port G is reset.
5	R5	R/W	0x0	GPIO Port F Software Reset 0x0 = GPIO port F is not reset. 0x1 = GPIO port F is reset.
4	R4	R/W	0x0	GPIO Port E Software Reset 0x0 = GPIO port E is not reset. 0x1 = GPIO port E is reset.
3	R3	R/W	0x0	GPIO Port D Software Reset 0x0 = GPIO port D is not reset. 0x1 = GPIO port D is reset.
2	R2	R/W	0x0	GPIO Port C Software Reset 0x0 = GPIO port C is not reset. 0x1 = GPIO port C is reset.
1	R1	R/W	0x0	GPIO Port B Software Reset 0x0 = GPIO port B is not reset. 0x1 = GPIO port B is reset.
0	R0	R/W	0x0	GPIO Port A Software Reset 0x0 = GPIO port A is not reset. 0x1 = GPIO port A is reset.

4.2.67 SRDMA Register (Offset = 0x50C) [reset = 0x00]

Micro Direct Memory Access Software Reset (SRDMA)

The SRDMA register lets software reset the available μ DMA module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRDMA register. While the SRDMA bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRDMA bit.

There may be latency from the clearing of the SRDMA bit to when the peripheral is ready for use. Software should check the corresponding PRDMA bit to verify that the μ DMA module registers are ready to be accessed.

NOTE: Use this register to reset the μ DMA module.

SRDMA is shown in [Figure 4-73](#) and described in [Table 4-80](#).

Return to [Summary Table](#).

Figure 4-73. SRDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-80. SRDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	μ DMA Module Software Reset 0x0 = μ DMA module is not reset. 0x1 = μ DMA module is reset.

4.2.68 SREPI Register (Offset = 0x510) [reset = 0x00]

EPI Software Reset (SREPI)

The SREPI register lets software reset the available EPI module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SREPI register. While the SREPI bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SREPI bit.

There may be latency from the clearing of the SREPI bit to when the peripheral is ready for use. Software should check the corresponding PREPI bit to verify that the EPI module registers are ready to be accessed.

NOTE: Use this register to reset the EPI module.

SREPI is shown in [Figure 4-74](#) and described in [Table 4-81](#).

Return to [Summary Table](#).

Figure 4-74. SREPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-81. SREPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	EPI Module Software Reset 0x0 = EPI module is not reset. 0x1 = EPI module is reset.

4.2.69 SRHIB Register (Offset = 0x514) [reset = 0x00]

Hibernation Software Reset (SRHIB)

The SRHIB register lets software reset the available Hibernation module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRHIB register. While the SRHIB bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRHIB bit.

There may be latency from the clearing of the SRHIB bit to when the peripheral is ready for use. Software should check the corresponding PRHIB bit to verify that the Hibernation module registers are ready to be accessed.

NOTE: Use this register to reset the Hibernation module.

SRHIB is shown in [Figure 4-75](#) and described in [Table 4-82](#).

Return to [Summary Table](#).

Figure 4-75. SRHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-82. SRHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Hibernation Module Software Reset 0x0 = Hibernation module is not reset. 0x1 = Hibernation module is reset.

4.2.70 SRUART Register (Offset = 0x518) [reset = 0x00]

Universal Asynchronous Receiver/Transmitter Software Reset (SRUART)

The SRUART register lets software reset the available UART modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRUART register. While the SRUART bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRUART bit.

There may be latency from the clearing of the SRUART bit to when the peripheral is ready for use. Software should check the corresponding PRUART bit to verify that the UART module registers are ready to be accessed.

NOTE: Use this register to reset the UART modules.

SRUART is shown in [Figure 4-76](#) and described in [Table 4-83](#).

Return to [Summary Table](#).

Figure 4-76. SRUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								R7	R6	R5	R4	R3	R2	R1	R0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-83. SRUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	X	
7	R7	R/W	0x0	UART Module 7 Software Reset 0x0 = UART module 7 is not reset. 0x1 = UART module 7 is reset.
6	R6	R/W	0x0	UART Module 6 Software Reset 0x0 = UART module 6 is not reset. 0x1 = UART module 6 is reset.
5	R5	R/W	0x0	UART Module 5 Software Reset 0x0 = UART module 5 is not reset. 0x1 = UART module 5 is reset.
4	R4	R/W	0x0	UART Module 4 Software Reset 0x0 = UART module 4 is not reset. 0x1 = UART module 4 is reset.
3	R3	R/W	0x0	UART Module 3 Software Reset 0x0 = UART module 3 is not reset. 0x1 = UART module 3 is reset.
2	R2	R/W	0x0	UART Module 2 Software Reset 0x0 = UART module 2 is not reset. 0x1 = UART module 2 is reset.
1	R1	R/W	0x0	UART Module 1 Software Reset 0x0 = UART module 1 is not reset. 0x1 = UART module 1 is reset.

Table 4-83. SRUART Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	R0	R/W	0x0	UART Module 0 Software Reset 0x0 = UART module 0 is not reset. 0x1 = UART module 0 is reset.

4.2.71 SRSSI Register (Offset = 0x51C) [reset = 0x0]

Synchronous Serial Interface Software Reset (SRSSI)

The SRSSI register lets software reset the available SSI modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRSSI register. While the SRSSI bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRSSI bit.

There may be latency from the clearing of the SRSSI bit to when the peripheral is ready for use. Software should check the corresponding PRSSI bit to verify that the SSI module registers are ready to be accessed.

NOTE: Use this register to reset the SSI modules.

SRSSI is shown in [Figure 4-77](#) and described in [Table 4-84](#).

Return to [Summary Table](#).

Figure 4-77. SRSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												R3	R2	R1	R0
R-0x0												R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-84. SRSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	R3	R/W	0x0	SSI Module 3 Software Reset 0x0 = SSI module 3 is not reset. 0x1 = SSI module 3 is reset.
2	R2	R/W	0x0	SSI Module 2 Software Reset 0x0 = SSI module 2 is not reset. 0x1 = SSI module 2 is reset.
1	R1	R/W	0x0	SSI Module 1 Software Reset 0x0 = SSI module 1 is not reset. 0x1 = SSI module 1 is reset.
0	R0	R/W	0x0	SSI Module 0 Software Reset 0x0 = SSI module 0 is not reset. 0x1 = SSI module 0 is reset.

4.2.72 SRI2C Register (Offset = 0x520) [reset = 0x0]

Inter-Integrated Circuit Software Reset (SRI2C)

The SRI2C register lets software reset the available I2C modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRI2C register. While the SRI2C bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRI2C bit.

There may be latency from the clearing of the SRI2C bit to when the peripheral is ready for use. Software should check the corresponding PRI2C bit to verify that the I2C module registers are ready to be accessed.

NOTE: Use this register to reset the I2C modules.

SRI2C is shown in [Figure 4-78](#) and described in [Table 4-85](#).

Return to [Summary Table](#).

Figure 4-78. SRI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
R-0x0						R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0	R/W- 0x0

Table 4-85. SRI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	R9	R/W	0x0	I2C Module 9 Software Reset 0x0 = I2C module 9 is not reset. 0x1 = I2C module 9 is reset.
8	R8	R/W	0x0	I2C Module 8 Software Reset 0x0 = I2C module 8 is not reset. 0x1 = I2C module 8 is reset.
7	R7	R/W	0x0	I2C Module 7 Software Reset 0x0 = I2C module 7 is not reset. 0x1 = I2C module 7 is reset.
6	R6	R/W	0x0	I2C Module 6 Software Reset 0x0 = I2C module 6 is not reset. 0x1 = I2C module 6 is reset.
5	R5	R/W	0x0	I2C Module 5 Software Reset 0x0 = I2C module 5 is not reset. 0x1 = I2C module 5 is reset.
4	R4	R/W	0x0	I2C Module 4 Software Reset 0x0 = I2C module 4 is not reset. 0x1 = I2C module 4 is reset.
3	R3	R/W	0x0	I2C Module 3 Software Reset 0x0 = I2C module 3 is not reset. 0x1 = I2C module 3 is reset.

Table 4-85. SRI2C Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	R2	R/W	0x0	I2C Module 2 Software Reset 0x0 = I2C module 2 is not reset. 0x1 = I2C module 2 is reset.
1	R1	R/W	0x0	I2C Module 1 Software Reset 0x0 = I2C module 1 is not reset. 0x1 = I2C module 1 is reset.
0	R0	R/W	0x0	I2C Module 0 Software Reset 0x0 = I2C module 0 is not reset. 0x1 = I2C module 0 is reset.

4.2.73 SRUSB Register (Offset = 0x528) [reset = 0x0]

Universal Serial Bus Software Reset (SRUSB)

The SRUSB register lets software reset the available USB module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRUSB register. While the SRUSB bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRUSB bit.

There may be latency from the clearing of the SRUSB bit to when the peripheral is ready for use. Software should check the corresponding PRUSB bit to verify that the USB module registers are ready to be accessed.

NOTE: Use this register to reset the USB module.

SRUSB is shown in [Figure 4-79](#) and described in [Table 4-86](#).

Return to [Summary Table](#).

Figure 4-79. SRUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-86. SRUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	USB Module Software Reset 0x0 = USB module is not reset. 0x1 = USB module is reset.

4.2.74 SREPHY Register (Offset = 0x530) [reset = 0x0]

Ethernet PHY Software Reset (SREPHY)

The SREPHY register lets software reset the available PHY module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SREPHY register. While the SREPHY bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SREPHY bit.

There may be latency from the clearing of the SREPHY bit to when the peripheral is ready for use. Software should check the corresponding PREPHY bit to verify that the EPHY module registers are ready to be accessed.

NOTE: Use this register to reset the Ethernet PHY module.

SREPHY is shown in [Figure 4-80](#) and described in [Table 4-87](#).

Return to [Summary Table](#).

Figure 4-80. SREPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-87. SREPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Ethernet PHY Module Software Reset 0x0 = Ethernet PHY module is not reset. 0x1 = Ethernet PHY module is reset.

4.2.75 SRCAN Register (Offset = 0x534) [reset = 0x0]

Controller Area Network Software Reset (SRCAN)

The SRCAN register lets software reset the available CAN modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRCAN register. While the SRCAN bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRCAN bit.

There may be latency from the clearing of the SRCAN bit to when the peripheral is ready for use. Software should check the corresponding PRCAN bit to verify that the CAN module registers are ready to be accessed.

NOTE: Use this register to reset the CAN modules.

SRCAN is shown in [Figure 4-81](#) and described in [Table 4-88](#).

Return to [Summary Table](#).

Figure 4-81. SRCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R/W-0x0	R/W-0x0

Table 4-88. SRCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R/W	0x0	CAN Module 1 Software Reset 0x0 = CAN module 1 is not reset. 0x1 = CAN module 1 is reset.
0	R0	R/W	0x0	CAN Module 0 Software Reset 0x0 = CAN module 0 is not reset. 0x1 = CAN module 0 is reset.

4.2.76 SRADC Register (Offset = 0x538) [reset = 0x0]

Analog-to-Digital Converter Software Reset (SRADC)

The SRADC register lets software reset the available ADC modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRADC register. While the SRADC bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRADC bit.

There may be latency from the clearing of the SRADC bit to when the peripheral is ready for use. Software should check the corresponding PRADC bit to verify that the ADC module registers are ready to be accessed.

NOTE: Use this register to reset the ADC modules.

SRADC is shown in [Figure 4-82](#) and described in [Table 4-89](#).

Return to [Summary Table](#).

Figure 4-82. SRADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R/W-0x0	R/W-0x0

Table 4-89. SRADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R/W	0x0	ADC Module 1 Software Reset 0x0 = ADC module 1 is not reset. 0x1 = ADC module 1 is reset.
0	R0	R/W	0x0	ADC Module 0 Software Reset 0x0 = ADC module 0 is not reset. 0x1 = ADC module 0 is reset.

4.2.77 SRACMP Register (Offset = 0x53C) [reset = 0x0]

Analog Comparator Software Reset (SRACMP)

The SRACMP register lets software reset the available analog comparator module.

A block is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRACMP register. While the SRACMP bit is 1, the module is held in reset.
2. Software completes the reset process by clearing the SRACMP bit.

There may be latency from the clearing of the SRACMP bit to when the module is ready for use. Software should check the corresponding PRACMP bit to verify that the Analog Comparator module registers are ready to be accessed.

NOTE: Use this register to reset the analog comparator module.

SRACMP is shown in [Figure 4-83](#) and described in [Table 4-90](#).

Return to [Summary Table](#).

Figure 4-83. SRACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-90. SRACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Analog Comparator Module 0 Software Reset 0x0 = Analog comparator module is not reset. 0x1 = Analog comparator module is reset.

4.2.78 SRPWM Register (Offset = 0x540) [reset = 0x0]

Pulse Width Modulator Software Reset (SRPWM)

The SRPWM register lets software reset the available PWM modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRPWM register. While the SRPWM bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRPWM bit.

There may be latency from the clearing of the SRPWM bit to when the peripheral is ready for use. Software should check the corresponding PRPWM bit to verify that the PWM module registers are ready to be accessed.

NOTE: Use this register to reset the PWM modules.

SRPWM is shown in [Figure 4-84](#) and described in [Table 4-91](#).

Return to [Summary Table](#).

Figure 4-84. SRPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-91. SRPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	PWM Module 0 Software Reset 0x0 = PWM module 0 is not reset. 0x1 = PWM module 0 is reset.

4.2.79 SRQEI Register (Offset = 0x544) [reset = 0x0]

Quadrature Encoder Interface Software Reset (SRQEI)

The SRQEI register lets software reset the available QEI modules.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRQEI register. While the SRQEI bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRQEI bit.

There may be latency from the clearing of the SRQEI bit to when the peripheral is ready for use. Software should check the corresponding PRQEI bit to verify that the QEI module registers are ready to be accessed.

NOTE: Use this register to reset the QEI modules.

SRQEI is shown in [Figure 4-85](#) and described in [Table 4-92](#).

Return to [Summary Table](#).

Figure 4-85. SRQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-92. SRQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	QEI Module 0 Software Reset 0x0 = QEI module 0 is not reset. 0x1 = QEI module 0 is reset.

4.2.80 SREEPROM Register (Offset = 0x558) [reset = 0x0]

EEPROM Software Reset (SREEPROM)

The SREEPROM register lets software reset the available EEPROM module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SREEPROM register. While the SREEPROM bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SREEPROM bit.

There may be latency from the clearing of the SREEPROM bit to when the peripheral is ready for use. Software should check the corresponding PREEEPROM bit to verify that the EEPROM module registers are ready to be accessed.

SREEPROM is shown in [Figure 4-86](#) and described in [Table 4-93](#).

Return to [Summary Table](#).

Figure 4-86. SREEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-93. SREEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	EEPROM Module 0 Software Reset 0x0 = EEPROM module is not reset. 0x1 = EEPROM module is reset.

4.2.81 SRCCM Register (Offset = 0x574) [reset = 0x0]

CRC and Cryptographic Modules Software Reset (SRCCM)

The SRCCM register lets software reset the CRC and Cryptographic modules (AES, DES, and SHA/MD5).

A module is reset by software using a simple 2-step process:

1. Software sets the bit in the SRCCM register. While the SRCCM bit is 1, the peripherals are held in reset.
2. Software completes the reset process by clearing the SRCCM bit.

There may be latency from the clearing of the SRCCM bit to when the peripheral is ready for use. Software should read the corresponding PRCCM bit to verify that the CRC and Cryptographic module (AES, DES, and SHA/MD5) registers are ready to be accessed.

NOTE: Use this register to reset the CRC, AES, DES, and SHA/MD5 modules.

SRCCM is shown in [Figure 4-87](#) and described in [Table 4-94](#).

Return to [Summary Table](#).

Figure 4-87. SRCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-94. SRCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	CRC and Cryptographic Modules Software Reset 0x0 = The CRC, AES, DES, and SHA/MD5 modules are not reset. 0x1 = The CRC, AES, DES, and SHA/MD5 modules are reset.

4.2.82 SRLCD Register (Offset = 0x590) [reset = 0x0]

LCD Controller Software Reset (SRLCD)

The SRLCD register lets software reset the available LCD module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SRLCD register. While the SRLCD bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SRLCD bit.

There may be latency from the clearing of the SRLCD bit to when the peripheral is ready for use. Software should check the corresponding PRLCD bit to verify that the LCD module registers are ready to be accessed.

NOTE: Use this register to reset the LCD controller modules.

SRLCD is shown in [Figure 4-88](#) and described in [Table 4-95](#).

Return to [Summary Table](#).

Figure 4-88. SRLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-95. SRLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	LCD Module 0 Software Reset 0x0 = LCD module 0 is not reset. 0x1 = LCD module 0 is reset.

4.2.83 SROWIRE Register (Offset = 0x598) [reset = 0x0]

1-Wire Software Reset (SROWIRE)

The SROWIRE register lets software reset the available 1-Wire Module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SROWIRE register. While the SROWIRE bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SROWIRE bit.

There may be latency from the clearing of the SROWIRE bit to when the peripheral is ready for use. Software should check the corresponding PROWIRE bit to verify that the 1-Wire module registers are ready to be accessed.

NOTE: Use this register to reset the 1-Wire controller modules.

SROWIRE is shown in [Figure 4-89](#) and described in [Table 4-96](#).

Return to [Summary Table](#).

Figure 4-89. SROWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-96. SROWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	1-Wire Module Software Reset 0x0 = 1-Wire module is not reset. 0x1 = 1-Wire module is reset.

4.2.84 SREMAC Register (Offset = 0x59C) [reset = 0x0]

Ethernet MAC Software Reset (SREMAC)

The SREMAC register lets software reset the available Ethernet MAC module.

A peripheral is reset by software using a simple 2-step process:

1. Software sets a bit (or bits) in the SREMAC register. While the SREMAC bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the SREMAC bit.

There may be latency from the clearing of the SREMAC bit to when the peripheral is ready for use. Software should check the corresponding PREMAC bit to verify that the Ethernet MAC module registers are ready to be accessed.

NOTE: Use this register to reset the Ethernet controller MAC module.

SREMAC is shown in [Figure 4-90](#) and described in [Table 4-97](#).

Return to [Summary Table](#).

Figure 4-90. SREMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-97. SREMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Ethernet Controller MAC Module 0 Software Reset 0x0 = Ethernet Controller MAC module 0 is not reset. 0x1 = Ethernet Controller MAC module 0 is reset.

4.2.85 RCGCWD Register (Offset = 0x600) [reset = 0x0]

Watchdog Timer Run Mode Clock Gating Control (RCGCWD)

The RCGCWD register lets software enable and disable watchdog modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the watchdog modules

RCGCWD is shown in [Figure 4-91](#) and described in [Table 4-98](#).

Return to [Summary Table](#).

Figure 4-91. RCGCWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R/W-0x0	R/W-0x0

Table 4-98. RCGCWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R/W	0x0	Watchdog Timer 1 Run Mode Clock Gating Control 0x0 = Watchdog module 1 is disabled. 0x1 = Enable and provide a clock to Watchdog module 1 in run mode.
0	R0	R/W	0x0	Watchdog Timer 0 Run Mode Clock Gating Control 0x0 = Watchdog module 0 is disabled. 0x1 = Enable and provide a clock to Watchdog module 0 in run mode.

4.2.86 RCGCTIMER Register (Offset = 0x604) [reset = 0x00]

16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)

The RCGCGPT32 register lets software enable and disable 16/32-bit timer modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the timer modules.

RCGCTIMER is shown in [Figure 4-92](#) and described in [Table 4-99](#).

Return to [Summary Table](#).

Figure 4-92. RCGCTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								R7	R6	R5	R4	R3	R2	R1	R0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-99. RCGCTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	R7	R/W	0x0	16/32-Bit General-Purpose Timer 7 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 7 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 7 in run mode.
6	R6	R/W	0x0	16/32-Bit General-Purpose Timer 6 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 6 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 6 in run mode.
5	R5	R/W	0x0	16/32-Bit General-Purpose Timer 5 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 5 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 5 in run mode.
4	R4	R/W	0x0	16/32-Bit General-Purpose Timer 4 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 4 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 4 in run mode.
3	R3	R/W	0x0	16/32-Bit General-Purpose Timer 3 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 3 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 3 in run mode.
2	R2	R/W	0x0	16/32-Bit General-Purpose Timer 2 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 2 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 2 in run mode.
1	R1	R/W	0x0	16/32-Bit General-Purpose Timer 1 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 1 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 1 in run mode.
0	R0	R/W	0x0	16/32-Bit General-Purpose Timer 0 Run Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 0 is disabled. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 0 in run mode.

4.2.87 RCGCGPIO Register (Offset = 0x608) [reset = 0x00]

General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

The RCGCGPIO register lets software enable and disable GPIO modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the GPIO modules.

RCGCGPIO is shown in [Figure 4-93](#) and described in [Table 4-100](#).

Return to [Summary Table](#).

Figure 4-93. RCGCGPIO Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED														R17	R16
R-0x0														R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-100. RCGCGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	R17	R/W	0x0	GPIO Port T Run Mode Clock Gating Control 0x0 = GPIO port T is disabled. 0x1 = Enable and provide a clock to GPIO port T in run mode.
16	R16	R/W	0x0	GPIO Port S Run Mode Clock Gating Control 0x0 = GPIO port S is disabled. 0x1 = Enable and provide a clock to GPIO port S in run mode.
15	R15	R/W	0x0	GPIO Port R Run Mode Clock Gating Control 0x0 = GPIO port R is disabled. 0x1 = Enable and provide a clock to GPIO port R in run mode.
14	R14	R/W	0x0	GPIO Port Q Run Mode Clock Gating Control 0x0 = GPIO port Q is disabled. 0x1 = Enable and provide a clock to GPIO port Q in run mode.
13	R13	R/W	0x0	GPIO Port P Run Mode Clock Gating Control 0x0 = GPIO port P is disabled. 0x1 = Enable and provide a clock to GPIO port P in run mode.
12	R12	R/W	0x0	GPIO Port N Run Mode Clock Gating Control 0x0 = GPIO port N is disabled. 0x1 = Enable and provide a clock to GPIO port N in run mode.
11	R11	R/W	0x0	GPIO Port M Run Mode Clock Gating Control 0x0 = GPIO port M is disabled. 0x1 = Enable and provide a clock to GPIO port M in run mode.
10	R10	R/W	0x0	GPIO Port L Run Mode Clock Gating Control 0x0 = GPIO port L is disabled. 0x1 = Enable and provide a clock to GPIO port L in run mode.
9	R9	R/W	0x0	GPIO Port K Run Mode Clock Gating Control 0x0 = GPIO port K is disabled. 0x1 = Enable and provide a clock to GPIO port K in run mode.

Table 4-100. RCGCGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	R8	R/W	0x0	GPIO Port J Run Mode Clock Gating Control 0x0 = GPIO port J is disabled. 0x1 = Enable and provide a clock to GPIO port J in run mode.
7	R7	R/W	0x0	GPIO Port H Run Mode Clock Gating Control 0x0 = GPIO port H is disabled. 0x1 = Enable and provide a clock to GPIO port H in run mode.
6	R6	R/W	0x0	GPIO Port G Run Mode Clock Gating Control 0x0 = GPIO port G is disabled. 0x1 = Enable and provide a clock to GPIO port G in run mode.
5	R5	R/W	0x0	GPIO Port F Run Mode Clock Gating Control 0x0 = GPIO port F is disabled. 0x1 = Enable and provide a clock to GPIO port F in run mode.
4	R4	R/W	0x0	GPIO Port E Run Mode Clock Gating Control 0x0 = GPIO port E is disabled. 0x1 = Enable and provide a clock to GPIO port E in run mode.
3	R3	R/W	0x0	GPIO Port D Run Mode Clock Gating Control 0x0 = GPIO port D is disabled. 0x1 = Enable and provide a clock to GPIO port D in run mode.
2	R2	R/W	0x0	GPIO Port C Run Mode Clock Gating Control 0x0 = GPIO port C is disabled. 0x1 = Enable and provide a clock to GPIO port C in run mode.
1	R1	R/W	0x0	GPIO Port B Run Mode Clock Gating Control 0x0 = GPIO port B is disabled. 0x1 = Enable and provide a clock to GPIO port B in run mode.
0	R0	R/W	0x0	GPIO Port A Run Mode Clock Gating Control 0x0 = GPIO port A is disabled. 0x1 = Enable and provide a clock to GPIO port A in run mode.

4.2.88 RCGCDMA Register (Offset = 0x60C) [reset = 0x0]

Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA)

The RCGCDMA register lets software enable and disable the μ DMA module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the μ DMA module.

RCGCDMA is shown in [Figure 4-94](#) and described in [Table 4-101](#).

Return to [Summary Table](#).

Figure 4-94. RCGCDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-101. RCGCDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	μ DMA Module Run Mode Clock Gating Control 0x0 = μ DMA module is disabled. 0x1 = Enable and provide a clock to the μ DMA module in run mode.

4.2.89 RCGCEPI Register (Offset = 0x610) [reset = 0x0]

EPI Run Mode Clock Gating Control (RCGCEPI)

The RCGCEPI register lets software enable and disable the EPI module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the EPI module.

RCGCEPI is shown in [Figure 4-95](#) and described in [Table 4-102](#).

Return to [Summary Table](#).

Figure 4-95. RCGCEPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-102. RCGCEPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	EPI Module Run Mode Clock Gating Control 0x0 = EPI module is disabled. 0x1 = Enable and provide a clock to the EPI module in run mode.

4.2.90 RCGCHIB Register (Offset = 0x614) [reset = 0x1]

Hibernation Run Mode Clock Gating Control (RCGCHIB)

The RCGCHIB register lets software enable and disable the Hibernation module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the Hibernation module.

RCGCHIB is shown in [Figure 4-96](#) and described in [Table 4-103](#).

Return to [Summary Table](#).

Figure 4-96. RCGCHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x1

Table 4-103. RCGCHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x1	Hibernation Module Run Mode Clock Gating Control 0x0 = Hibernation module is disabled. 0x1 = Enable and provide a clock to the Hibernation module in run mode.

4.2.91 RCGCUART Register (Offset = 0x618) [reset = 0x00]

Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART)

The RCGCUART register lets software enable and disable the UART modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the UART modules.

RCGCUART is shown in [Figure 4-97](#) and described in [Table 4-104](#).

Return to [Summary Table](#).

Figure 4-97. RCGCUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								R7	R6	R5	R4	R3	R2	R1	R0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-104. RCGCUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	R7	R/W	0x0	UART Module 7 Run Mode Clock Gating Control 0x0 = UART module 7 is disabled. 0x1 = Enable and provide a clock to UART module 7 in run mode.
6	R6	R/W	0x0	UART Module 6 Run Mode Clock Gating Control 0x0 = UART module 6 is disabled. 0x1 = Enable and provide a clock to UART module 6 in run mode.
5	R5	R/W	0x0	UART Module 5 Run Mode Clock Gating Control 0x0 = UART module 5 is disabled. 0x1 = Enable and provide a clock to UART module 5 in run mode.
4	R4	R/W	0x0	UART Module 4 Run Mode Clock Gating Control 0x0 = UART module 4 is disabled. 0x1 = Enable and provide a clock to UART module 4 in run mode.
3	R3	R/W	0x0	UART Module 3 Run Mode Clock Gating Control 0x0 = UART module 3 is disabled. 0x1 = Enable and provide a clock to UART module 3 in run mode.
2	R2	R/W	0x0	UART Module 2 Run Mode Clock Gating Control 0x0 = UART module 2 is disabled. 0x1 = Enable and provide a clock to UART module 2 in run mode.
1	R1	R/W	0x0	UART Module 1 Run Mode Clock Gating Control 0x0 = UART module 1 is disabled. 0x1 = Enable and provide a clock to UART module 1 in run mode.
0	R0	R/W	0x0	UART Module 0 Run Mode Clock Gating Control 0x0 = UART module 0 is disabled. 0x1 = Enable and provide a clock to UART module 0 in run mode.

4.2.92 RCGCSSI Register (Offset = 0x61C) [reset = 0x0]

Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI)

The RCGCSSI register lets software enable and disable the SSI modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the SSI modules.

RCGCSSI is shown in [Figure 4-98](#) and described in [Table 4-105](#).

Return to [Summary Table](#).

Figure 4-98. RCGCSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												R3	R2	R1	R0
R-0x0												R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-105. RCGCSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	R3	R/W	0x0	SSI Module 3 Run Mode Clock Gating Control 0x0 = SSI module 3 is disabled. 0x1 = Enable and provide a clock to SSI module 3 in run mode.
2	R2	R/W	0x0	SSI Module 2 Run Mode Clock Gating Control 0x0 = SSI module 2 is disabled. 0x1 = Enable and provide a clock to SSI module 2 in run mode.
1	R1	R/W	0x0	SSI Module 1 Run Mode Clock Gating Control 0x0 = SSI module 1 is disabled. 0x1 = Enable and provide a clock to SSI module 1 in run mode.
0	R0	R/W	0x0	SSI Module 0 Run Mode Clock Gating Control 0x0 = SSI module 0 is disabled. 0x1 = Enable and provide a clock to SSI module 0 in run mode.

4.2.93 RCGI2C Register (Offset = 0x620) [reset = 0x00]

Inter-Integrated Circuit Run Mode Clock Gating Control (RCGI2C)

The RCGI2C register lets software enable and disable the I2C modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the I2C modules.

RCGI2C is shown in [Figure 4-99](#) and described in [Table 4-106](#).

Return to [Summary Table](#).

Figure 4-99. RCGI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
R-0x0						R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-106. RCGI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	R9	R/W	0x0	I2C Module 9 Run Mode Clock Gating Control 0x0 = I2C module 9 is disabled. 0x1 = Enable and provide a clock to I2C module 9 in run mode.
8	R8	R/W	0x0	I2C Module 8 Run Mode Clock Gating Control 0x0 = I2C module 8 is disabled. 0x1 = Enable and provide a clock to I2C module 8 in run mode.
7	R7	R/W	0x0	I2C Module 7 Run Mode Clock Gating Control 0x0 = I2C module 7 is disabled. 0x1 = Enable and provide a clock to I2C module 7 in run mode.
6	R6	R/W	0x0	I2C Module 6 Run Mode Clock Gating Control 0x0 = I2C module 6 is disabled. 0x1 = Enable and provide a clock to I2C module 6 in run mode.
5	R5	R/W	0x0	I2C Module 5 Run Mode Clock Gating Control 0x0 = I2C module 5 is disabled. 0x1 = Enable and provide a clock to I2C module 5 in run mode.
4	R4	R/W	0x0	I2C Module 4 Run Mode Clock Gating Control 0x0 = I2C module 4 is disabled. 0x1 = Enable and provide a clock to I2C module 4 in run mode.
3	R3	R/W	0x0	I2C Module 3 Run Mode Clock Gating Control 0x0 = I2C module 3 is disabled. 0x1 = Enable and provide a clock to I2C module 3 in run mode.
2	R2	R/W	0x0	I2C Module 2 Run Mode Clock Gating Control 0x0 = I2C module 2 is disabled. 0x1 = Enable and provide a clock to I2C module 2 in run mode.
1	R1	R/W	0x0	I2C Module 1 Run Mode Clock Gating Control 0x0 = I2C module 1 is disabled. 0x1 = Enable and provide a clock to I2C module 1 in run mode.

Table 4-106. RCGCI2C Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	R0	R/W	0x0	I2C Module 0 Run Mode Clock Gating Control 0x0 = I2C module 0 is disabled. 0x1 = Enable and provide a clock to I2C module 0 in run mode.

4.2.94 RCGCUSB Register (Offset = 0x628) [reset = 0x0]

Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB)

The RCGCUSB register lets software enable and disable the USB module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the USB module.

RCGCUSB is shown in [Figure 4-100](#) and described in [Table 4-107](#).

Return to [Summary Table](#).

Figure 4-100. RCGCUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-107. RCGCUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	USB Module Run Mode Clock Gating Control 0x0 = USB module is disabled. 0x1 = Enable and provide a clock to the USB module in run mode.

4.2.95 RCGCEPHY Register (Offset = 0x630) [reset = 0x0]

Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY)

The RCGCEPHY register lets software enable and disable the PHY module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the PHY module.

RCGCEPHY is shown in [Figure 4-101](#) and described in [Table 4-108](#).

Return to [Summary Table](#).

Figure 4-101. RCGCEPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-108. RCGCEPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Ethernet PHY Module Run Mode Clock Gating Control 0x0 = PHY module is disabled. 0x1 = Enable and provide a clock to the PHY module in run mode.

4.2.96 RCGCCAN Register (Offset = 0x634) [reset = 0x0]

Controller Area Network Run Mode Clock Gating Control (RCGCCAN)

The RCGCCAN register lets software enable and disable the CAN modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the CAN modules.

RCGCCAN is shown in [Figure 4-102](#) and described in [Table 4-109](#).

Return to [Summary Table](#).

Figure 4-102. RCGCCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R/W-0x0	R/W-0x0

Table 4-109. RCGCCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R/W	0x0	CAN Module 1 Run Mode Clock Gating Control 0x0 = CAN module 1 is disabled. 0x1 = Enable and provide a clock to CAN module 1 in run mode.
0	R0	R/W	0x0	CAN Module 0 Run Mode Clock Gating Control 0x0 = CAN module 0 is disabled. 0x1 = Enable and provide a clock to CAN module 0 in run mode.

4.2.97 RCGADC Register (Offset = 0x638) [reset = 0x0]

Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)

The RCGADC register lets software enable and disable the ADC modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the ADC modules.

RCGCADC is shown in [Figure 4-103](#) and described in [Table 4-110](#).

Return to [Summary Table](#).

Figure 4-103. RCGADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R/W-0x0	R/W-0x0

Table 4-110. RCGADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R/W	0x0	ADC Module 1 Run Mode Clock Gating Control 0x0 = ADC module 1 is disabled. 0x1 = Enable and provide a clock to ADC module 1 in run mode.
0	R0	R/W	0x0	ADC Module 0 Run Mode Clock Gating Control 0x0 = ADC module 0 is disabled. 0x1 = Enable and provide a clock to ADC module 0 in run mode.

4.2.98 RCGCACMP Register (Offset = 0x63C) [reset = 0x0]

Analog Comparator Run Mode Clock Gating Control (RCGCACMP)

The RCGCACMP register lets software enable and disable the analog comparator module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the analog comparator module.

RCGCACMP is shown in [Figure 4-104](#) and described in [Table 4-111](#).

Return to [Summary Table](#).

Figure 4-104. RCGCACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-111. RCGCACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Analog Comparator Module 0 Run Mode Clock Gating Control 0x0 = Analog comparator module is disabled. 0x1 = Enable and provide a clock to the analog comparator module in run mode.

4.2.99 RCGCPWM Register (Offset = 0x640) [reset = 0x0]

Pulse Width Modulator Run Mode Clock Gating Control (RCGCPWM)

The RCGCPWM register lets software enable and disable the PWM modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the PWM modules.

RCGCPWM is shown in [Figure 4-105](#) and described in [Table 4-112](#).

Return to [Summary Table](#).

Figure 4-105. RCGCPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-112. RCGCPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	PWM Module 0 Run Mode Clock Gating Control 0x0 = PWM module 0 is disabled. 0x1 = Enable and provide a clock to PWM module 0 in run mode.

4.2.100 RCGCQEI Register (Offset = 0x644) [reset = 0x0]

Quadrature Encoder Interface Run Mode Clock Gating Control (RCGCQEI)

The RCGCQEI register lets software enable and disable the QEI modules in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the QEI modules.

RCGCQEI is shown in [Figure 4-106](#) and described in [Table 4-113](#).

Return to [Summary Table](#).

Figure 4-106. RCGCQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-113. RCGCQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	QEI Module 0 Run Mode Clock Gating Control 0x0 = QEI module 0 is disabled. 0x1 = Enable and provide a clock to QEI module 0 in run mode.

4.2.101 RCGCEEPROM Register (Offset = 0x658) [reset = 0x0]

EEPROM Run Mode Clock Gating Control (RCGCEEPROM)

The RCGCEEPROM register lets software enable and disable the EEPROM module in run mode. When enabled, the module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

RCGCEEPROM is shown in [Figure 4-107](#) and described in [Table 4-114](#).

Return to [Summary Table](#).

Figure 4-107. RCGCEEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-114. RCGCEEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	EEPROM Module 0 Run Mode Clock Gating Control 0x0 = EEPROM module is disabled. 0x1 = Enable and provide a clock to the EEPROM module in run mode.

4.2.102 RCGCCCM Register (Offset = 0x674) [reset = 0x0]

CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM)

The RCGCCCM register lets software enable and disable the CRC and Encryption Modules (AES, DES, and SHA/MD5) in run mode. When enabled, the modules are provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the CRC, AES, DES, and SHA/MD5 modules.

RCGCCCM is shown in [Figure 4-108](#) and described in [Table 4-115](#).

Return to [Summary Table](#).

Figure 4-108. RCGCCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-115. RCGCCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	CRC and Cryptographic Modules Run Mode Clock Gating Control 0x0 = The CRC, AES, DES, and SHA/MD 5 modules are disabled. 0x1 = Enable and provide a clock to the CRC, AES, DES, and SHA/MD5 modules in run mode.

4.2.103 RCGLCD Register (Offset = 0x690) [reset = 0x0]

LCD Controller Run Mode Clock Gating Control (RCGLCD)

The RCGLCD register lets software enable and disable the LCD Controller module in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the LCD Controller module.

RCGLCD is shown in [Figure 4-109](#) and described in [Table 4-116](#).

Return to [Summary Table](#).

Figure 4-109. RCGLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-116. RCGLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	LCD Controller Module 0 Run Mode Clock Gating Control 0x0 = LCD Controller module 0 is disabled. 0x1 = Enable and provide a clock to LCD Controller module 0 in run mode.

4.2.104 RCGCOWIRE Register (Offset = 0x698) [reset = 0x0]

1-Wire Run Mode Clock Gating Control (RCGCOWIRE)

The RCGCOWIRE register lets software enable and disable the 1-Wire module in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the 1-Wire module.

RCGCOWIRE is shown in [Figure 4-110](#) and described in [Table 4-117](#).

Return to [Summary Table](#).

Figure 4-110. RCGCOWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-117. RCGCOWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	1-Wire Module 0 Run Mode Clock Gating Control 0x0 = 1-Wire module 0 is disabled. 0x1 = Enable and provide a clock to 1-Wire module 0 in run mode.

4.2.105 RCGCEMAC Register (Offset = 0x69C) [reset = 0x0]

Ethernet MAC Run Mode Clock Gating Control (RCGCEMAC)

The RCGCEMAC register lets software enable and disable the Ethernet MAC module in run mode. When enabled, a module is provided a clock, and accesses to module registers are allowed. When disabled, the clock is disabled to save power, and accesses to module registers generate a bus fault.

NOTE: This register controls the clocking for the Ethernet Controller module.

RCGCEMAC is shown in [Figure 4-111](#) and described in [Table 4-118](#).

Return to [Summary Table](#).

Figure 4-111. RCGCEMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R/W-0x0

Table 4-118. RCGCEMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R/W	0x0	Ethernet MAC Module 0 Run Mode Clock Gating Control 0x0 = Ethernet MAC module 0 is disabled. 0x1 = Enable and provide a clock to Ethernet MAC module 0 in run mode.

4.2.106 SCGCWD Register (Offset = 0x700) [reset = 0x0]

Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD)

The SCGCWD register lets software enable and disable watchdog modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the watchdog modules.

SCGCWD is shown in [Figure 4-112](#) and described in [Table 4-119](#).

Return to [Summary Table](#).

Figure 4-112. SCGCWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														S1	S0
R-0x0														R/W-0x0	R/W-0x0

Table 4-119. SCGCWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	S1	R/W	0x0	Watchdog Timer 1 Sleep Mode Clock Gating Control 0x0 = Watchdog module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to Watchdog module 1 in sleep mode.
0	S0	R/W	0x0	Watchdog Timer 0 Sleep Mode Clock Gating Control 0x0 = Watchdog module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to Watchdog module 0 in sleep mode.

4.2.107 SCGCTIMER Register (Offset = 0x704) [reset = 0x00]

16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER)

The SCGCGPT32 register lets software enable and disable 16/32-bit timer modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the timer modules.

SCGCTIMER is shown in [Figure 4-113](#) and described in [Table 4-120](#).

Return to [Summary Table](#).

Figure 4-113. SCGCTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								S7	S6	S5	S4	S3	S2	S1	S0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-120. SCGCTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	S7	R/W	0x0	16/32-Bit General-Purpose Timer 7 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 7 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 7 in sleep mode.
6	S6	R/W	0x0	16/32-Bit General-Purpose Timer 6 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 6 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 6 in sleep mode.
5	S5	R/W	0x0	16/32-Bit General-Purpose Timer 5 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 5 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 5 in sleep mode.
4	S4	R/W	0x0	16/32-Bit General-Purpose Timer 4 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 4 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 4 in sleep mode.
3	S3	R/W	0x0	16/32-Bit General-Purpose Timer 3 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 3 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 3 in sleep mode.

Table 4-120. SCGCTIMER Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	S2	R/W	0x0	16/32-Bit General-Purpose Timer 2 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 2 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 2 in sleep mode.
1	S1	R/W	0x0	16/32-Bit General-Purpose Timer 1 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 1 in sleep mode.
0	S0	R/W	0x0	16/32-Bit General-Purpose Timer 0 Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 0 in sleep mode.

4.2.108 SCGCGPIO Register (Offset = 0x708) [reset = 0x00]

General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO)

The SCGCGPIO register lets software enable and disable GPIO modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the GPIO modules.

SCGCGPIO is shown in [Figure 4-114](#) and described in [Table 4-121](#).

Return to [Summary Table](#).

Figure 4-114. SCGCGPIO Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						S17	S16
R-0x0						R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
S15	S14	S13	S12	S11	S10	S9	S8
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-121. SCGCGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	S17	R/W	0x0	GPIO Port T Sleep Mode Clock Gating Control 0x0 = GPIO port T is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port T in sleep mode.
16	S16	R/W	0x0	GPIO Port S Sleep Mode Clock Gating Control 0x0 = GPIO port S is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port S in sleep mode.
15	S15	R/W	0x0	GPIO Port R Sleep Mode Clock Gating Control 0x0 = GPIO port R is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port R in sleep mode.
14	S14	R/W	0x0	GPIO Port Q Sleep Mode Clock Gating Control 0x0 = GPIO port Q is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port Q in sleep mode.
13	S13	R/W	0x0	GPIO Port P Sleep Mode Clock Gating Control 0x0 = GPIO port P is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port P in sleep mode.
12	S12	R/W	0x0	GPIO Port N Sleep Mode Clock Gating Control 0x0 = GPIO port N is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port N in sleep mode.
11	S11	R/W	0x0	GPIO Port M Sleep Mode Clock Gating Control 0x0 = GPIO port M is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port M in sleep mode.

Table 4-121. SCGCGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	S10	R/W	0x0	GPIO Port L Sleep Mode Clock Gating Control 0x0 = GPIO port L is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port L in sleep mode.
9	S9	R/W	0x0	GPIO Port K Sleep Mode Clock Gating Control 0x0 = GPIO port K is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port K in sleep mode.
8	S8	R/W	0x0	GPIO Port J Sleep Mode Clock Gating Control 0x0 = GPIO port J is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port J in sleep mode.
7	S7	R/W	0x0	GPIO Port H Sleep Mode Clock Gating Control 0x0 = GPIO port H is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port H in sleep mode.
6	S6	R/W	0x0	GPIO Port G Sleep Mode Clock Gating Control 0x0 = GPIO port G is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port G in sleep mode.
5	S5	R/W	0x0	GPIO Port F Sleep Mode Clock Gating Control 0x0 = GPIO port F is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port F in sleep mode.
4	S4	R/W	0x0	GPIO Port E Sleep Mode Clock Gating Control 0x0 = GPIO port E is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port E in sleep mode.
3	S3	R/W	0x0	GPIO Port D Sleep Mode Clock Gating Control 0x0 = GPIO port D is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port D in sleep mode.
2	S2	R/W	0x0	GPIO Port C Sleep Mode Clock Gating Control 0x0 = GPIO port C is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port C in sleep mode.
1	S1	R/W	0x0	GPIO Port B Sleep Mode Clock Gating Control 0x0 = GPIO port B is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port B in sleep mode.
0	S0	R/W	0x0	GPIO Port A Sleep Mode Clock Gating Control 0x0 = GPIO port A is disabled in sleep mode. 0x1 = Enable and provide a clock to GPIO port A in sleep mode.

4.2.109 SCGCDMA Register (Offset = 0x70C) [reset = 0x0]

Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA)

The SCGCDMA register lets software enable and disable the μ DMA module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the μ DMA module.

SCGCDMA is shown in [Figure 4-115](#) and described in [Table 4-122](#).

Return to [Summary Table](#).

Figure 4-115. SCGCDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-122. SCGCDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	μ DMA Module Sleep Mode Clock Gating Control 0x0 = μ DMA module is disabled in sleep mode. 0x1 = Enable and provide a clock to the μ DMA module in sleep mode.

4.2.110 SCGCEPI Register (Offset = 0x710) [reset = 0x0]

EPI Sleep Mode Clock Gating Control (SCGCEPI)

The SCGCEPI register lets software enable and disable the EPI module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the EPI module.

SCGCEPI is shown in [Figure 4-116](#) and described in [Table 4-123](#).

Return to [Summary Table](#).

Figure 4-116. SCGCEPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-123. SCGCEPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	EPI Module Sleep Mode Clock Gating Control 0x0 = EPI module is disabled in sleep mode. 0x1 = Enable and provide a clock to the EPI module in sleep mode.

4.2.111 SCGCHIB Register (Offset = 0x714) [reset = 0x1]

Hibernation Sleep Mode Clock Gating Control (SCGCHIB)

The SCGCHIB register lets software enable and disable the Hibernation module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the Hibernation module.

SCGCHIB is shown in [Figure 4-117](#) and described in [Table 4-124](#).

Return to [Summary Table](#).

Figure 4-117. SCGCHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x1

Table 4-124. SCGCHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x1	Hibernation Module Sleep Mode Clock Gating Control 0x0 = Hibernation module is disabled in sleep mode. 0x1 = Enable and provide a clock to the Hibernation module in sleep mode.

4.2.112 SCGCUART Register (Offset = 0x718) [reset = 0x00]

Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART)

The SCGCUART register lets software enable and disable the UART modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the UART modules.

SCGCUART is shown in [Figure 4-118](#) and described in [Table 4-125](#).

Return to [Summary Table](#).

Figure 4-118. SCGCUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								S7	S6	S5	S4	S3	S2	S1	S0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-125. SCGCUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	S7	R/W	0x0	UART Module 7 Sleep Mode Clock Gating Control 0x0 = UART module 7 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 7 in sleep mode.
6	S6	R/W	0x0	UART Module 6 Sleep Mode Clock Gating Control 0x0 = UART module 6 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 6 in sleep mode.
5	S5	R/W	0x0	UART Module 5 Sleep Mode Clock Gating Control 0x0 = UART module 5 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 5 in sleep mode.
4	S4	R/W	0x0	UART Module 4 Sleep Mode Clock Gating Control 0x0 = UART module 4 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 4 in sleep mode.
3	S3	R/W	0x0	UART Module 3 Sleep Mode Clock Gating Control 0x0 = UART module 3 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 3 in sleep mode.
2	S2	R/W	0x0	UART Module 2 Sleep Mode Clock Gating Control 0x0 = UART module 2 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 2 in sleep mode.
1	S1	R/W	0x0	UART Module 1 Sleep Mode Clock Gating Control 0x0 = UART module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 1 in sleep mode.
0	S0	R/W	0x0	UART Module 0 Sleep Mode Clock Gating Control 0x0 = UART module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to UART module 0 in sleep mode.

4.2.113 SCGCSSI Register (Offset = 0x71C) [reset = 0x0]

Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI)

The SCGCSSI register lets software enable and disable the SSI modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the SSI modules.

SCGCSSI is shown in [Figure 4-119](#) and described in [Table 4-126](#).

Return to [Summary Table](#).

Figure 4-119. SCGCSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												S3	S2	S1	S0
R-0x0												R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-126. SCGCSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	S3	R/W	0x0	SSI Module 3 Sleep Mode Clock Gating Control 0x0 = SSI module 3 is disabled in sleep mode. 0x1 = Enable and provide a clock to SSI module 3 in sleep mode.
2	S2	R/W	0x0	SSI Module 2 Sleep Mode Clock Gating Control 0x0 = SSI module 2 is disabled in sleep mode. 0x1 = Enable and provide a clock to SSI module 2 in sleep mode.
1	S1	R/W	0x0	SSI Module 1 Sleep Mode Clock Gating Control 0x0 = SSI module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to SSI module 1 in sleep mode.
0	S0	R/W	0x0	SSI Module 0 Sleep Mode Clock Gating Control 0x0 = SSI module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to SSI module 0 in sleep mode.

4.2.114 SCGCI2C Register (Offset = 0x720) [reset = 0x00]

Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C)

The SCGCI2C register lets software enable and disable the I2C modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the I2C modules.

SCGCI2C is shown in [Figure 4-120](#) and described in [Table 4-127](#).

Return to [Summary Table](#).

Figure 4-120. SCGCI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
R-0x0						R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-127. SCGCI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	S9	R/W	0x0	I2C Module 9 Sleep Mode Clock Gating Control 0x0 = I2C module 9 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 9 in sleep mode.
8	S8	R/W	0x0	I2C Module 8 Sleep Mode Clock Gating Control 0x0 = I2C module 8 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 8 in sleep mode.
7	S7	R/W	0x0	I2C Module 7 Sleep Mode Clock Gating Control 0x0 = I2C module 7 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 7 in sleep mode.
6	S6	R/W	0x0	I2C Module 6 Sleep Mode Clock Gating Control 0x0 = I2C module 6 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 6 in sleep mode.
5	S5	R/W	0x0	I2C Module 5 Sleep Mode Clock Gating Control 0x0 = I2C module 5 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 5 in sleep mode.
4	S4	R/W	0x0	I2C Module 4 Sleep Mode Clock Gating Control 0x0 = I2C module 4 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 4 in sleep mode.
3	S3	R/W	0x0	I2C Module 3 Sleep Mode Clock Gating Control 0x0 = I2C module 3 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 3 in sleep mode.
2	S2	R/W	0x0	I2C Module 2 Sleep Mode Clock Gating Control 0x0 = I2C module 2 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 2 in sleep mode.
1	S1	R/W	0x0	I2C Module 1 Sleep Mode Clock Gating Control 0x0 = I2C module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 1 in sleep mode.

Table 4-127. SCGCI2C Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	S0	R/W	0x0	I2C Module 0 Sleep Mode Clock Gating Control 0x0 = I2C module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to I2C module 0 in sleep mode.

4.2.115 SCGCUSB Register (Offset = 0x728) [reset = 0x0]

Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB)

The SCGCUSB register lets software enable and disable the USB module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the USB module.

SCGCUSB is shown in [Figure 4-121](#) and described in [Table 4-128](#).

Return to [Summary Table](#).

Figure 4-121. SCGCUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-128. SCGCUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	USB Module Sleep Mode Clock Gating Control 0x0 = USB module is disabled in sleep mode. 0x1 = Enable and provide a clock to the USB module in sleep mode.

4.2.116 SCGCEPHY Register (Offset = 0x730) [reset = 0x0]

Ethernet PHY Sleep Mode Clock Gating Control (SCGCEPHY)

The SCGCEPHY register lets software enable and disable the PHY module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the PHY module.

SCGCEPHY is shown in [Figure 4-122](#) and described in [Table 4-129](#).

Return to [Summary Table](#).

Figure 4-122. SCGCEPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-129. SCGCEPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	PHY Module Sleep Mode Clock Gating Control 0x0 = PHY module is disabled in sleep mode. 0x1 = Enable and provide a clock to the PHY module in sleep mode.

4.2.117 SCGCCAN Register (Offset = 0x734) [reset = 0x0]

Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN)

The SCGCCAN register lets software enable and disable the CAN modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the CAN modules.

SCGCCAN is shown in [Figure 4-123](#) and described in [Table 4-130](#).

Return to [Summary Table](#).

Figure 4-123. SCGCCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														S1	S0
R-0x0														R/W-0x0	R/W-0x0

Table 4-130. SCGCCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	S1	R/W	0x0	CAN Module 1 Sleep Mode Clock Gating Control 0x0 = CAN module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to CAN module 1 in sleep mode.
0	S0	R/W	0x0	CAN Module 0 Sleep Mode Clock Gating Control 0x0 = CAN module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to CAN module 0 in sleep mode.

4.2.118 SCGCADC Register (Offset = 0x738) [reset = 0x0]

Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC)

The SCGCADC register lets software enable and disable the ADC modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the ADC modules.

SCGCADC is shown in [Figure 4-124](#) and described in [Table 4-131](#).

Return to [Summary Table](#).

Figure 4-124. SCGCADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														S1	S0
R-0x0														R/W-0x0	R/W-0x0

Table 4-131. SCGCADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	S1	R/W	0x0	ADC Module 1 Sleep Mode Clock Gating Control 0x0 = ADC module 1 is disabled in sleep mode. 0x1 = Enable and provide a clock to ADC module 1 in sleep mode.
0	S0	R/W	0x0	ADC Module 0 Sleep Mode Clock Gating Control 0x0 = ADC module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to ADC module 0 in sleep mode.

4.2.119 SCGCACMP Register (Offset = 0x73C) [reset = 0x0]

Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP)

The SCGCACMP register lets software enable and disable the analog comparator module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the analog comparator module.

SCGCACMP is shown in [Figure 4-125](#) and described in [Table 4-132](#).

Return to [Summary Table](#).

Figure 4-125. SCGCACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-132. SCGCACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	Analog Comparator Module 0 Sleep Mode Clock Gating Control 0x0 = Analog comparator module is disabled in sleep mode. 0x1 = Enable and provide a clock to the analog comparator module in sleep mode.

4.2.120 SCGCPWM Register (Offset = 0x740) [reset = 0x0]

Pulse Width Modulator Sleep Mode Clock Gating Control (SCGCPWM)

The SCGCPWM register lets software enable and disable the PWM modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the PWM modules.

SCGCPWM is shown in [Figure 4-126](#) and described in [Table 4-133](#).

Return to [Summary Table](#).

Figure 4-126. SCGCPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-133. SCGCPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	PWM Module 0 Sleep Mode Clock Gating Control 0x0 = PWM module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to PWM module 0 in sleep mode.

4.2.121 SCGCQEI Register (Offset = 0x744) [reset = 0x0]

Quadrature Encoder Interface Sleep Mode Clock Gating Control (SCGCQEI)

The SCGCQEI register lets software enable and disable the QEI modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the QEI modules.

SCGCQEI is shown in [Figure 4-127](#) and described in [Table 4-134](#).

Return to [Summary Table](#).

Figure 4-127. SCGCQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-134. SCGCQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	QEI Module 0 Sleep Mode Clock Gating Control 0x0 = QEI module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to QEI module 0 in sleep mode.

4.2.122 SCGCEEPROM Register (Offset = 0x758) [reset = 0x0]

EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM)

The SCGCEEPROM register lets software enable and disable the EEPROM module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

SCGCEEPROM is shown in [Figure 4-128](#) and described in [Table 4-135](#).

Return to [Summary Table](#).

Figure 4-128. SCGCEEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-135. SCGCEEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	EEPROM Module 0 Sleep Mode Clock Gating Control 0x0 = EEPROM module is disabled in sleep mode. 0x1 = Enable and provide a clock to the EEPROM module in sleep mode.

4.2.123 SCGCCCM Register (Offset = 0x774) [reset = 0x0]

CRC and Cryptographic Modules Sleep Mode Clock Gating Control (SCGCCCM)

The SCGCCCM register lets software enable and disable the CRC and Encryption Control, AES, DES, and SHA/MD5 modules in sleep mode. When enabled, the modules are provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the CRC, AES, DES, and SHA/MD5 modules.

SCGCCCM is shown in [Figure 4-129](#) and described in [Table 4-136](#).

Return to [Summary Table](#).

Figure 4-129. SCGCCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-136. SCGCCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	CRC and Cryptographic Modules Sleep Mode Clock Gating Control 0x0 = The CRC, AES, DES, and SHA/MD5 modules are disabled in sleep mode. 0x1 = Enable and provide a clock to the CRC, AES, DES, and SHA/MD5 modules in sleep mode.

4.2.124 SCGCLCD Register (Offset = 0x790) [reset = 0x0]

LCD Controller Sleep Mode Clock Gating Control (SCGCLCD)

The SCGCLCD register lets software enable and disable the LCD Controller module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the LCD Controller module.

SCGCLCD is shown in [Figure 4-130](#) and described in [Table 4-137](#).

Return to [Summary Table](#).

Figure 4-130. SCGCLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-137. SCGCLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	LCD Controller Module 0 Sleep Mode Clock Gating Control 0x0 = LCD Controller module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to LCD Controller module 0 in sleep mode.

4.2.125 SCGCOWIRE Register (Offset = 0x798) [reset = 0x0]

1-Wire Sleep Mode Clock Gating Control (SCGCOWIRE)

The SCGCOWIRE register lets software enable and disable the 1-Wire module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the 1-Wire module.

SCGCOWIRE is shown in [Figure 4-131](#) and described in [Table 4-138](#).

Return to [Summary Table](#).

Figure 4-131. SCGCOWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-138. SCGCOWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	1-Wire Module 0 Sleep Mode Clock Gating Control 0x0 = 1-Wire module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to 1-Wire module 0 in sleep mode.

4.2.126 SCGCEMAC Register (Offset = 0x79C) [reset = 0x0]

Ethernet MAC Sleep Mode Clock Gating Control (SCGCEMAC)

The SCGCEMAC register lets software enable and disable the Ethernet MAC module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the Ethernet MAC module.

SCGCEMAC is shown in [Figure 4-132](#) and described in [Table 4-139](#).

Return to [Summary Table](#).

Figure 4-132. SCGCEMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															S0
R-0x0															R/W-0x0

Table 4-139. SCGCEMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0	R/W	0x0	Ethernet MAC Module 0 Sleep Mode Clock Gating Control 0x0 = Ethernet MAC module 0 is disabled in sleep mode. 0x1 = Enable and provide a clock to Ethernet MAC module 0 in sleep mode.

4.2.127 DCGCWD Register (Offset = 0x800) [reset = 0x0]

Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD)

The DCGCWD register lets software enable and disable watchdog modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the watchdog modules.

DCGCWD is shown in [Figure 4-133](#) and described in [Table 4-140](#).

Return to [Summary Table](#).

Figure 4-133. DCGCWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														D1	D0
R-0x0														R/W-0x0	R/W-0x0

Table 4-140. DCGCWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	D1	R/W	0x0	Watchdog Timer 1 Deep-Sleep Mode Clock Gating Control 0x0 = Watchdog module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to Watchdog module 1 in deep-sleep mode.
0	D0	R/W	0x0	Watchdog Timer 0 Deep-Sleep Mode Clock Gating Control 0x0 = Watchdog module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to Watchdog module 0 in deep-sleep mode.

4.2.128 DCGCTIMER Register (Offset = 0x804) [reset = 0x00]

16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER)

The DCGCTIMER register lets software enable and disable 16/32-bit timer modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the timer modules.

DCGCTIMER is shown in [Figure 4-134](#) and described in [Table 4-141](#).

Return to [Summary Table](#).

Figure 4-134. DCGCTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								D7	D6	D5	D4	D3	D2	D1	D0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-141. DCGCTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	D7	R/W	0x0	16/32-Bit General-Purpose Timer 7 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 7 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 7 in deep-sleep mode.
6	D6	R/W	0x0	16/32-Bit General-Purpose Timer 6 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 6 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 6 in deep-sleep mode.
5	D5	R/W	0x0	16/32-Bit General-Purpose Timer 5 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 5 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 5 in deep-sleep mode.
4	D4	R/W	0x0	16/32-Bit General-Purpose Timer 4 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 4 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 4 in deep-sleep mode.
3	D3	R/W	0x0	16/32-Bit General-Purpose Timer 3 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 3 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 3 in deep-sleep mode.

Table 4-141. DCGTIMER Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	D2	R/W	0x0	16/32-Bit General-Purpose Timer 2 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 2 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 2 in deep-sleep mode.
1	D1	R/W	0x0	16/32-Bit General-Purpose Timer 1 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 1 in deep-sleep mode.
0	D0	R/W	0x0	16/32-Bit General-Purpose Timer 0 Deep-Sleep Mode Clock Gating Control 0x0 = 16/32-bit general-purpose timer module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 16/32-bit general-purpose timer module 0 in deep-sleep mode.

4.2.129 DCGCGPIO Register (Offset = 0x808) [reset = 0x00]

General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO)

The DCGCGPIO register lets software enable and disable GPIO modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the GPIO modules.

DCGCGPIO is shown in [Figure 4-135](#) and described in [Table 4-142](#).

Return to [Summary Table](#).

Figure 4-135. DCGCGPIO Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED														D17	D16
R-0x0														R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-142. DCGCGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	D17	R/W	0x0	GPIO Port T Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port T is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port T in deep-sleep mode.
16	D16	R/W	0x0	GPIO Port S Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port S is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port S in deep-sleep mode.
15	D15	R/W	0x0	GPIO Port R Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port R is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port R in deep-sleep mode.
14	D14	R/W	0x0	GPIO Port Q Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port Q is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port Q in deep-sleep mode.
13	D13	R/W	0x0	GPIO Port P Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port P is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port P in deep-sleep mode.
12	D12	R/W	0x0	GPIO Port N Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port N is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port N in deep-sleep mode.
11	D11	R/W	0x0	GPIO Port M Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port M is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port M in deep-sleep mode.

Table 4-142. DCGCGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	D10	R/W	0x0	GPIO Port L Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port L is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port L in deep-sleep mode.
9	D9	R/W	0x0	GPIO Port K Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port K is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port K in deep-sleep mode.
8	D8	R/W	0x0	GPIO Port J Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port J is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port J in deep-sleep mode.
7	D7	R/W	0x0	GPIO Port H Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port H is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port H in deep-sleep mode.
6	D6	R/W	0x0	GPIO Port G Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port G is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port G in deep-sleep mode.
5	D5	R/W	0x0	GPIO Port F Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port F is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port F in deep-sleep mode.
4	D4	R/W	0x0	GPIO Port E Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port E is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port E in deep-sleep mode.
3	D3	R/W	0x0	GPIO Port D Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port D is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port D in deep-sleep mode.
2	D2	R/W	0x0	GPIO Port C Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port C is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port C in deep-sleep mode.
1	D1	R/W	0x0	GPIO Port B Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port B is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port B in deep-sleep mode.
0	D0	R/W	0x0	GPIO Port A Deep-Sleep Mode Clock Gating Control 0x0 = GPIO port A is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to GPIO port A in deep-sleep mode.

4.2.130 DCGCDMA Register (Offset = 0x80C) [reset = 0x0]

Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA)

The DCGCDMA register lets software enable and disable the μ DMA module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the μ DMA module.

DCGCDMA is shown in [Figure 4-136](#) and described in [Table 4-143](#).

Return to [Summary Table](#).

Figure 4-136. DCGCDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-143. DCGCDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	μ DMA Module Deep-Sleep Mode Clock Gating Control 0x0 = μ DMA module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the μ DMA module in deep-sleep mode.

4.2.131 DCGCEPI Register (Offset = 0x810) [reset = 0x0]

EPI Deep-Sleep Mode Clock Gating Control (DCGCEPI)

The DCGCEPI register lets software enable and disable the EPI module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the EPI module.

DCGCEPI is shown in [Figure 4-137](#) and described in [Table 4-144](#).

Return to [Summary Table](#).

Figure 4-137. DCGCEPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-144. DCGCEPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	EPI Module Deep-Sleep Mode Clock Gating Control 0x0 = EPI module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the EPI module in deep-sleep mode.

4.2.132 DCGCHIB Register (Offset = 0x814) [reset = 0x1]

Hibernation Deep-Sleep Mode Clock Gating Control (DCGCHIB)

The DCGCHIB register lets software enable and disable the Hibernation module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the Hibernation module.

DCGCHIB is shown in [Figure 4-138](#) and described in [Table 4-145](#).

Return to [Summary Table](#).

Figure 4-138. DCGCHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x1

Table 4-145. DCGCHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x1	Hibernation Module Deep-Sleep Mode Clock Gating Control 0x0 = Hibernation module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the Hibernation module in deep-sleep mode.

4.2.133 DCGCUART Register (Offset = 0x818) [reset = 0x00]

Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART)

The DCGCUART register lets software enable and disable the UART modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the UART modules.

DCGCUART is shown in [Figure 4-139](#) and described in [Table 4-146](#).

Return to [Summary Table](#).

Figure 4-139. DCGCUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								D7	D6	D5	D4	D3	D2	D1	D0
R-0x0								R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-146. DCGCUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	D7	R/W	0x0	UART Module 7 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 7 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 7 in deep-sleep mode.
6	D6	R/W	0x0	UART Module 6 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 6 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 6 in deep-sleep mode.
5	D5	R/W	0x0	UART Module 5 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 5 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 5 in deep-sleep mode.
4	D4	R/W	0x0	UART Module 4 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 4 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 4 in deep-sleep mode.
3	D3	R/W	0x0	UART Module 3 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 3 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 3 in deep-sleep mode.
2	D2	R/W	0x0	UART Module 2 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 2 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 2 in deep-sleep mode.
1	D1	R/W	0x0	UART Module 1 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 1 in deep-sleep mode.

Table 4-146. DCGCUART Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	D0	R/W	0x0	UART Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = UART module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to UART module 0 in deep-sleep mode.

4.2.134 DCGCSSI Register (Offset = 0x81C) [reset = 0x0]

Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI)

The DCGCSSI register lets software enable and disable the SSI modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the SSI modules.

DCGCSSI is shown in [Figure 4-140](#) and described in [Table 4-147](#).

Return to [Summary Table](#).

Figure 4-140. DCGCSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												D3	D2	D1	D0
R-0x0												R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-147. DCGCSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	D3	R/W	0x0	SSI Module 3 Deep-Sleep Mode Clock Gating Control 0x0 = SSI module 3 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to SSI module 3 in deep-sleep mode.
2	D2	R/W	0x0	SSI Module 2 Deep-Sleep Mode Clock Gating Control 0x0 = SSI module 2 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to SSI module 2 in deep-sleep mode.
1	D1	R/W	0x0	SSI Module 1 Deep-Sleep Mode Clock Gating Control 0x0 = SSI module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to SSI module 1 in deep-sleep mode.
0	D0	R/W	0x0	SSI Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = SSI module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to SSI module 0 in deep-sleep mode.

4.2.135 DCGCI2C Register (Offset = 0x820) [reset = 0x00]

Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C)

The DCGCI2C register lets software enable and disable the I2C modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the I2C modules.

DCGCI2C is shown in [Figure 4-141](#) and described in [Table 4-148](#).

Return to [Summary Table](#).

Figure 4-141. DCGCI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R-0x0						R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 4-148. DCGCI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	D9	R/W	0x0	I2C Module 9 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 9 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 9 in deep-sleep mode.
8	D8	R/W	0x0	I2C Module 8 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 8 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 8 in deep-sleep mode.
7	D7	R/W	0x0	I2C Module 7 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 7 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 7 in deep-sleep mode.
6	D6	R/W	0x0	I2C Module 6 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 6 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 6 in deep-sleep mode.
5	D5	R/W	0x0	I2C Module 5 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 5 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 5 in deep-sleep mode.
4	D4	R/W	0x0	I2C Module 4 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 4 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 4 in deep-sleep mode.
3	D3	R/W	0x0	I2C Module 3 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 3 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 3 in deep-sleep mode.

Table 4-148. DCGCI2C Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	D2	R/W	0x0	I2C Module 2 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 2 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 2 in deep-sleep mode.
1	D1	R/W	0x0	I2C Module 1 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 1 in deep-sleep mode.
0	D0	R/W	0x0	I2C Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = I2C module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to I2C module 0 in deep-sleep mode.

4.2.136 DCGCUSB Register (Offset = 0x828) [reset = 0x0]

Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB)

The DCGCUSB register lets software enable and disable the USB module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the USB module.

DCGCUSB is shown in [Figure 4-142](#) and described in [Table 4-149](#).

Return to [Summary Table](#).

Figure 4-142. DCGCUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-149. DCGCUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	USB Module Deep-Sleep Mode Clock Gating Control 0x0 = USB module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the USB module in deep-sleep mode.

4.2.137 DCGCEPHY Register (Offset = 0x830) [reset = 0x0]

Ethernet PHY Deep-Sleep Mode Clock Gating Control (DCGCEPHY)

The DCGCEPHY register lets software enable and disable the PHY module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the PHY module.

DCGCEPHY is shown in [Figure 4-143](#) and described in [Table 4-150](#).

Return to [Summary Table](#).

Figure 4-143. DCGCEPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-150. DCGCEPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	PHY Module Deep-Sleep Mode Clock Gating Control 0x0 = PHY module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the PHY module in deep-sleep mode.

4.2.138 DCGCCAN Register (Offset = 0x834) [reset = 0x0]

Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN)

The DCGCCAN register lets software enable and disable the CAN modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the CAN modules.

DCGCCAN is shown in [Figure 4-144](#) and described in [Table 4-151](#).

Return to [Summary Table](#).

Figure 4-144. DCGCCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														D1	D0
R-0x0														R/W-0x0	R/W-0x0

Table 4-151. DCGCCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	D1	R/W	0x0	CAN Module 1 Deep-Sleep Mode Clock Gating Control 0x0 = CAN module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to CAN module 1 in deep-sleep mode.
0	D0	R/W	0x0	CAN Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = CAN module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to CAN module 0 in deep-sleep mode.

4.2.139 DCGADC Register (Offset = 0x838) [reset = 0x0]

Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC)

The DCGADC register lets software enable and disable the ADC modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the ADC modules.

DCGCADC is shown in [Figure 4-145](#) and described in [Table 4-152](#).

Return to [Summary Table](#).

Figure 4-145. DCGADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														D1	D0
R-0x0														R/W-0x0	R/W-0x0

Table 4-152. DCGADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	D1	R/W	0x0	ADC Module 1 Deep-Sleep Mode Clock Gating Control 0x0 = ADC module 1 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to ADC module 1 in deep-sleep mode.
0	D0	R/W	0x0	ADC Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = ADC module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to ADC module 0 in deep-sleep mode.

4.2.140 DCGCACMP Register (Offset = 0x83C) [reset = 0x0]

Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP)

The DCGCACMP register lets software enable and disable the analog comparator module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the analog comparator module.

DCGCACMP is shown in [Figure 4-146](#) and described in [Table 4-153](#).

Return to [Summary Table](#).

Figure 4-146. DCGCACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-153. DCGCACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	Analog Comparator Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = Analog comparator module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the analog comparator module in deep-sleep mode.

4.2.141 DCGCPWM Register (Offset = 0x840) [reset = 0x0]

Pulse Width Modulator Deep-Sleep Mode Clock Gating Control (DCGCPWM)

The DCGCPWM register lets software enable and disable the PWM modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the PWM modules.

DCGCPWM is shown in [Figure 4-147](#) and described in [Table 4-154](#).

Return to [Summary Table](#).

Figure 4-147. DCGCPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-154. DCGCPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	PWM Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = PWM module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to PWM module 0 in deep-sleep mode.

4.2.142 DCGCQEI Register (Offset = 0x844) [reset = 0x0]

Quadrature Encoder Interface Deep-Sleep Mode Clock Gating Control (DCGCQEI)

The DCGCQEI register lets software enable and disable the QEI modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the QEI modules.

DCGCQEI is shown in [Figure 4-148](#) and described in [Table 4-155](#).

Return to [Summary Table](#).

Figure 4-148. DCGCQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-155. DCGCQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	QEI Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = QEI module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to QEI module 0 in deep-sleep mode.

4.2.143 DCGCEEPROM Register (Offset = 0x858) [reset = 0x0]

EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM)

The DCGCEEPROM register lets software enable and disable the EEPROM module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

DCGCEEPROM is shown in [Figure 4-149](#) and described in [Table 4-156](#).

Return to [Summary Table](#).

Figure 4-149. DCGCEEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-156. DCGCEEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	EEPROM Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = EEPROM module is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to the EEPROM module in deep-sleep mode.

4.2.144 DCGCCCM Register (Offset = 0x874) [reset = 0x0]

CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control (DCGCCCM)

The DCGCCCM register lets software enable and disable the CRC, AES, DES, and SHA/MD5 modules in deep-sleep mode. When enabled, the modules are provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the CRC, AES, DES, and SHA/MD modules.

DCGCCCM is shown in [Figure 4-150](#) and described in [Table 4-157](#).

Return to [Summary Table](#).

Figure 4-150. DCGCCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-157. DCGCCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	<p>CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control</p> <p>0x0 = The CRC, AES, DES, and SHA/MD5 modules are disabled in deep-sleep mode.</p> <p>0x1 = Enable and provide a clock to the CRC, AES, DES, and SHA/MD5 modules in deep-sleep mode.</p>

4.2.145 DCGCLCD Register (Offset = 0x890) [reset = 0x0]

LCD Controller Deep-Sleep Mode Clock Gating Control (DCGCLCD)

The DCGCLCD register lets software enable and disable the LCD Controller module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the LCD Controller module.

DCGCLCD is shown in [Figure 4-151](#) and described in [Table 4-158](#).

Return to [Summary Table](#).

Figure 4-151. DCGCLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-158. DCGCLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	LCD Controller Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = LCD controller module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to LCD controller module 0 in deep-sleep mode.

4.2.146 DCGCOWIRE Register (Offset = 0x898) [reset = 0x0]

1-Wire Deep-Sleep Mode Clock Gating Control (DCGCOWIRE)

The DCGCOWIRE register lets software enable and disable the 1-Wire module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the 1-Wire module.

DCGCOWIRE is shown in [Figure 4-152](#) and described in [Table 4-159](#).

Return to [Summary Table](#).

Figure 4-152. DCGCOWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-159. DCGCOWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	1-Wire Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = 1-Wire module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to 1-Wire module 0 in deep-sleep mode.

4.2.147 DCGCEMAC Register (Offset = 0x89C) [reset = 0x0]

Ethernet MAC Deep-Sleep Mode Clock Gating Control (DCGCEMAC)

The DCGCEMAC register lets software enable and disable the Ethernet MAC module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

NOTE: This register controls the clocking for the Ethernet MAC module.

DCGCEMAC is shown in [Figure 4-153](#) and described in [Table 4-160](#).

Return to [Summary Table](#).

Figure 4-153. DCGCEMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															D0
R-0x0															R/W-0x0

Table 4-160. DCGCEMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	D0	R/W	0x0	Ethernet MAC Module 0 Deep-Sleep Mode Clock Gating Control 0x0 = Ethernet MAC module 0 is disabled in deep-sleep mode. 0x1 = Enable and provide a clock to Ethernet MAC module 0 in deep-sleep mode.

4.2.148 PCWD Register (Offset = 0x900) [reset = 0x3]

Watchdog Timer Power Control (PCWD)

NOTE: The Watchdog Timer modules do not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCWD register controls the power applied to the Watchdog Module module. The function of this bit depends on the current mode of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCWD, SCGCWD, and DCGCWD registers. If the Rn, Sn, or Dn bit of the respective RCGCWD, SCGCWD, and DCGCWD registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of the value of the corresponding Pn bit in the PCWD register.

However, if the Rn, Sn, or Dn bit of the respective RCGCWD, SCGCWD, and DCGCWD registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCWD register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-161](#) details the differences.

Table 4-161. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic or leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCWD is shown in [Figure 4-154](#) and described in [Table 4-162](#).

Return to [Summary Table](#).

Figure 4-154. PCWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														P1	P0
R-0x0														R/W-0x1	R/W-0x1

Table 4-162. PCWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	P1	R/W	0x1	<p>Watchdog Timer 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCWD, SCGCWD, or DCGCWD register is clear.</p> <p>0x0 = Watchdog Timer 1 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = Watchdog Timer 1 module is powered but does not receive a clock. In this case, the module is inactive.</p>
0	P0	R/W	0x1	<p>Watchdog Timer 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCWD, SCGCWD or DCGCWD register is clear.</p> <p>0x0 = Watchdog Timer 0 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = Watchdog Timer 0 module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.149 PCTIMER Register (Offset = 0x904) [reset = 0xFF]

16/32-Bit General-Purpose Timer Power Control (PCTIMER)

NOTE: The Timer module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCTIMER register controls the power applied to the Timer module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCTIMER, SCGCTIMER, and DCGCTIMER registers. If the Rn, Sn, or Dn bit of the respective RCGCTIMER, SCGCTIMER, and DCGCTIMER registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of the corresponding Pn bit in the PCTIMER register.

However, if the Rn, Sn, or Dn bit of the respective RCGCTIMER, SCGCTIMER, and DCGCTIMER registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCTIMER register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-163](#) details the differences.

Table 4-163. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCTIMER is shown in [Figure 4-155](#) and described in [Table 4-164](#).

Return to [Summary Table](#).

Figure 4-155. PCTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								P7	P6	P5	P4	P3	P2	P1	P0
R-0x0								R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1

Table 4-164. PCTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	P7	R/W	0x1	General-Purpose Timer 7 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 7 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 7 module is powered but does not receive a clock. In this case, the module is inactive.
6	P6	R/W	0x1	General-Purpose Timer 6 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 6 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 6 module is powered but does not receive a clock. In this case, the module is inactive.
5	P5	R/W	0x1	General-Purpose Timer 5 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 5 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 5 module is powered but does not receive a clock. In this case, the module is inactive.
4	P4	R/W	0x1	General-Purpose Timer 4 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 4 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 4 module is powered but does not receive a clock. In this case, the module is inactive.
3	P3	R/W	0x1	General-Purpose Timer 3 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 3 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 3 module is powered but does not receive a clock. In this case, the module is inactive.
2	P2	R/W	0x1	General-Purpose Timer 2 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 2 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 2 module is powered but does not receive a clock. In this case, the module is inactive.
1	P1	R/W	0x1	General-Purpose Timer 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear. 0x0 = Timer 1 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Timer 1 module is powered but does not receive a clock. In this case, the module is inactive.

Table 4-164. PCTIMER Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	P0	R/W	0x1	<p>General-Purpose Timer 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCTIMER, SCGCTIMER or DCGCTIMER register is clear.</p> <p>0x0 = Timer 0 module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = Timer 0 module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.150 PCGPIO Register (Offset = 0x908) [reset = 0x3FFFF]

General-Purpose Input/Output Power Control (PCGPIO)

register type RW, reset 0x0003FFFF

NOTE: The GPIO modules do not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCGPIO register controls the power applied to the GPIO module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCGPIO, SCGCGPIO, and DCGCGPIO registers. If the Rn, Sn, or Dn bit of the respective RCGCGPIO, SCGCGPIO, and DCGCGPIO registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCGPIO register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCGPIO, SCGCGPIO, and DCGCGPIO registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCGPIO register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-165](#) lists the differences.

Table 4-165. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCGPIO is shown in [Figure 4-156](#) and described in [Table 4-166](#).

Return to [Summary Table](#).

Figure 4-156. PCGPIO Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED														P17	P16
R-0x0														R/W-0x1	R/W-0x1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1

Table 4-166. PCGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	P17	R/W	0x1	<p>GPIO Port T Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port T is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port T is powered but does not receive a clock. In this case, the module is inactive.</p>
16	P16	R/W	0x1	<p>GPIO Port S Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port S is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port S is powered but does not receive a clock. In this case, the module is inactive.</p>
15	P15	R/W	0x1	<p>GPIO Port R Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port R is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port R is powered but does not receive a clock. In this case, the module is inactive.</p>
14	P14	R/W	0x1	<p>GPIO Port Q Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port Q is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port Q is powered but does not receive a clock. In this case, the module is inactive.</p>
13	P13	R/W	0x1	<p>GPIO Port P Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port P is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port P is powered but does not receive a clock. In this case, the module is inactive.</p>
12	P12	R/W	0x1	<p>GPIO Port N Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port N is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port N is powered but does not receive a clock. In this case, the module is inactive.</p>
11	P11	R/W	0x1	<p>GPIO Port M Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port M is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port M is powered but does not receive a clock. In this case, the module is inactive.</p>

Table 4-166. PCGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	P10	R/W	0x1	<p>GPIO Port L Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port L is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port L is powered but does not receive a clock. In this case, the module is inactive.</p>
9	P9	R/W	0x1	<p>GPIO Port K Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port K is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port K is powered but does not receive a clock. In this case, the module is inactive.</p>
8	P8	R/W	0x1	<p>GPIO Port J Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port J is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port J is powered but does not receive a clock. In this case, the module is inactive.</p>
7	P7	R/W	0x1	<p>GPIO Port H Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port H is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port H is powered but does not receive a clock. In this case, the module is inactive.</p>
6	P6	R/W	0x1	<p>GPIO Port G Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port G is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port G is powered but does not receive a clock. In this case, the module is inactive.</p>
5	P5	R/W	0x1	<p>GPIO Port F Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port F is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port F is powered but does not receive a clock. In this case, the module is inactive.</p>
4	P4	R/W	0x1	<p>GPIO Port E Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port E is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port E is powered but does not receive a clock. In this case, the module is inactive.</p>

Table 4-166. PCGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P3	R/W	0x1	<p>GPIO Port D Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port D is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port D is powered but does not receive a clock. In this case, the module is inactive.</p>
2	P2	R/W	0x1	<p>GPIO Port C Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port C is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port C is powered but does not receive a clock. In this case, the module is inactive.</p>
1	P1	R/W	0x1	<p>GPIO Port B Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port B is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port B is powered but does not receive a clock. In this case, the module is inactive.</p>
0	P0	R/W	0x1	<p>GPIO Port A Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCGPIO, SCGCGPIO, or DCGCGPIO register is clear.</p> <p>0x0 = GPIO port A is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = GPIO port A is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.151 PCDMA Register (Offset = 0x90C) [reset = 0x1]

Micro Direct Memory Access Power Control (PCDMA)

NOTE: The μ DMA module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCDMA register controls the power applied to the DMA module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCDMA, SCGCDMA, and DCGCDMA registers. If the Rn, Sn, or Dn bit of the respective RCGCDMA, SCGCDMA, and DCGCDMA registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCDMA register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCDMA, SCGCDMA, and DCGCDMA registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCDMA register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-167](#) lists the differences.

Table 4-167. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCDMA is shown in [Figure 4-157](#) and described in [Table 4-168](#).

Return to [Summary Table](#).

Figure 4-157. PCDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-168. PCDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>μDMA Module Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCDMA, SCGCDMA, or DCGCDMA register is clear.</p> <p>0x0 = The μDMA module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The μDMA module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.152 PCEPI Register (Offset = 0x910) [reset = 0x1]

External Peripheral Interface Power Control (PCEPI)

register type RW, reset 0x00000001

NOTE: The EPI module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCEPI register controls the power applied to the EPI module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCEPI, SCGCEPI, and DCGCEPI registers. If the Rn, Sn, or Dn bit of the respective RCGCEPI, SCGCEPI, and DCGCEPI registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCEPI register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCEPI, SCGCEPI, and DCGCEPI registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCEPI register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-169](#) lists the differences.

Table 4-169. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCEPI is shown in [Figure 4-158](#) and described in [Table 4-170](#).

Return to [Summary Table](#).

Figure 4-158. PCEPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-170. PCEPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>EPI Module Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCEPI, SCGCEPI or DCGCEPI register is clear.</p> <p>0x0 = The EPI module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The EPI module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.153 PCHIB Register (Offset = 0x914) [reset = 0x1]

Hibernation Power Control (PCHIB)

NOTE: The Hibernation module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCHIB register controls the power applied to the HIB module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCHIB, SCGCHIB, and DCGCHIB registers. If the Rn, Sn, or Dn bit of the respective RCGCHIB, SCGCHIB, and DCGCHIB registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCHIB register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCHIB, SCGCHIB, and DCGCHIB registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCHIB register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-171](#) lists the differences.

Table 4-171. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCHIB is shown in [Figure 4-159](#) and described in [Table 4-172](#).

Return to [Summary Table](#).

Figure 4-159. PCHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-172. PCHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>Hibernation Module Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCHIB, SCGCHIB, or DCGCHIB register is clear.</p> <p>0x0 = The HIB module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The HIB module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.154 PCUART Register (Offset = 0x918) [reset = 0xFF]

Universal Asynchronous Receiver/Transmitter Power Control (PCUART)

NOTE: The UART module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCUART register controls the power applied to the UART module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCUART, SCGCUART, and DCGCUART registers. If the Rn, Sn, or Dn bit of the respective RCGCUART, SCGCUART, and DCGCUART registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCUART register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCUART, SCGCUART, and DCGCUART registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCUART register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-173](#) details the differences.

Table 4-173. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCUART is shown in [Figure 4-160](#) and described in [Table 4-174](#).

Return to [Summary Table](#).

Figure 4-160. PCUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								P7	P6	P5	P4	P3	P2	P1	P0
R-0x0								R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1

Table 4-174. PCUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	P7	R/W	0x1	<p>UART Module 7 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 7 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 7 is powered but does not receive a clock. In this case, the module is inactive.</p>
6	P6	R/W	0x1	<p>UART Module 6 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 6 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 6 is powered but does not receive a clock. In this case, the module is inactive.</p>
5	P5	R/W	0x1	<p>UART Module 5 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 5 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 5 is powered but does not receive a clock. In this case, the module is inactive.</p>
4	P4	R/W	0x1	<p>UART Module 4 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 4 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 4 is powered but does not receive a clock. In this case, the module is inactive.</p>
3	P3	R/W	0x1	<p>UART Module 3 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 3 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 3 is powered but does not receive a clock. In this case, the module is inactive.</p>
2	P2	R/W	0x1	<p>UART Module 2 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 2 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 2 is powered but does not receive a clock. In this case, the module is inactive.</p>
1	P1	R/W	0x1	<p>UART Module 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 1 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 1 is powered but does not receive a clock. In this case, the module is inactive.</p>

Table 4-174. PCUART Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	P0	R/W	0x1	<p>UART Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUART, SCGCUART, or DCGCUART register is clear.</p> <p>0x0 = The UART module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The UART module 0 is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.155 PCSSI Register (Offset = 0x91C) [reset = 0xF]

Synchronous Serial Interface Power Control (PCSSI)

NOTE: The SSI module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCSSI register controls the power applied to the SSI module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCSSI, SCGCSSI, and DCGCSSI registers. If the Rn, Sn, or Dn bit of the respective RCGCSSI, SCGCSSI, and DCGCSSI registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCSSI register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCSSI, SCGCSSI, and DCGCSSI registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCSSI register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-175](#) details the differences.

Table 4-175. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCSSI is shown in [Figure 4-161](#) and described in [Table 4-176](#).

Return to [Summary Table](#).

Figure 4-161. PCSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												P3	P2	P1	P0
R-0x0												R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1

Table 4-176. PCSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	P3	R/W	0x1	SSI Module 3 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCSSI, SCGCSSI, or DCGCSSI register is clear. 0x0 = The SSI module 3 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = The SSI module 3 is powered but does not receive a clock. In this case, the module is inactive.
2	P2	R/W	0x1	SSI Module 2 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCSSI, SCGCSSI, or DCGCSSI register is clear. 0x0 = The SSI module 2 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = The SSI module 2 is powered but does not receive a clock. In this case, the module is inactive.
1	P1	R/W	0x1	SSI Module 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCSSI, SCGCSSI, or DCGCSSI register is clear. 0x0 = The SSI module 1 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = The SSI module 1 is powered but does not receive a clock. In this case, the module is inactive.
0	P0	R/W	0x1	SSI Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCSSI, SCGCSSI, or DCGCSSI register is clear. 0x0 = The SSI module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = The SSI module 0 is powered but does not receive a clock. In this case, the module is inactive.

4.2.156 PCI2C Register (Offset = 0x920) [reset = 0x3FF]

Inter-Integrated Circuit Power Control (PCI2C)

NOTE: The I2C module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCI2C register controls the power applied to the I2C module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCI2C, SCGCI2C, and DCGCI2C registers. If the Rn, Sn, or Dn bit of the respective RCGCI2C, SCGCI2C, and DCGCI2C registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCI2C register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCI2C, SCGCI2C, and DCGCI2C registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCI2C register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-177](#) details the differences.

Table 4-177. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCI2C is shown in [Figure 4-162](#) and described in [Table 4-178](#).

Return to [Summary Table](#).

Figure 4-162. PCI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
R-0x0						R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1

Table 4-178. PCI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	P9	R/W	X	<p>I2C Module 9 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 9 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 9 is powered but does not receive a clock. In this case, the module is inactive.</p>
8	P8	R/W	X	<p>I2C Module 8 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 8 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 8 is powered but does not receive a clock. In this case, the module is inactive.</p>
7	P7	R/W	0x1	<p>I2C Module 7 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 7 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 7 is powered but does not receive a clock. In this case, the module is inactive.</p>
6	P6	R/W	0x1	<p>I2C Module 6 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 6 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 6 is powered but does not receive a clock. In this case, the module is inactive.</p>
5	P5	R/W	0x1	<p>I2C Module 5 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 5 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 5 is powered but does not receive a clock. In this case, the module is inactive.</p>
4	P4	R/W	0x1	<p>I2C Module 4 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 4 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 4 is powered but does not receive a clock. In this case, the module is inactive.</p>
3	P3	R/W	0x1	<p>I2C Module 3 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 3 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 3 is powered but does not receive a clock. In this case, the module is inactive.</p>

Table 4-178. PCI2C Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	P2	R/W	0x1	<p>I2C Module 2 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 2 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 2 is powered but does not receive a clock. In this case, the module is inactive.</p>
1	P1	R/W	0x1	<p>I2C Module 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 1 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 1 is powered but does not receive a clock. In this case, the module is inactive.</p>
0	P0	R/W	0x1	<p>I2C Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCI2C, SCGCI2C, or DCGCI2C register is clear.</p> <p>0x0 = The I2C module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The I2C module 0 is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.157 PCUSB Register (Offset = 0x928) [reset = 0x1]

Universal Serial Bus Power Control (PCUSB)

The PCUSB register controls the power applied to the USB module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCUSB, SCGCUSB, and DCGCUSB registers. If the Rn, Sn, or Dn bit of the respective RCGCUSB, SCGCUSB, and DCGCUSB registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCUSB register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCUSB, SCGCUSB, and DCGCUSB registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCUSB register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-179](#) lists the differences.

Table 4-179. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCUSB is shown in [Figure 4-163](#) and described in [Table 4-180](#).

Return to [Summary Table](#).

Figure 4-163. PCUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-180. PCUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	USB Module Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCUSB, SCGCUSB, or DCGCUSB register is clear. 0x0 = The USB module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = The USB module is powered but does not receive a clock. In this case, the module is inactive.

4.2.158 PCEPHY Register (Offset = 0x930) [reset = 0x0]

Ethernet PHY Power Control (PCEPHY)

The PCEPHY register controls the power applied to the EEPROM module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCEPHY, SCGCEPHY, and DCGCEPHY registers. If the Rn, Sn, or Dn bit of the respective RCGCEPHY, SCGCEPHY, and DCGCEPHY registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCEPHY register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCEPHY, SCGCEPHY, and DCGCEPHY registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCEPHY register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-181](#) lists the differences.

Table 4-181. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

NOTE: The Ethernet PHY module is not powered up at reset to prevent an automatic negotiation on power-up. To properly initialize the PHY, first inhibit the PHY from running on power up by setting the PHYHOLD bit in the Ethernet Peripheral Configuration (EMACPC) register and then set the P0 bit in the PCEPHY register. Once it is determined that the PHY is ready (by polling the R0 bit in the Peripheral Ready (PREPHY) register), the PHY can be programmed with its appropriate values.

NOTE: If the MOSC is chosen as the clock to the Ethernet PHY then software must enable the MOSC before enabling the Ethernet PHY by setting the P0 bit in the PCEPHY.

PCEPHY is shown in [Figure 4-164](#) and described in [Table 4-182](#).

Return to [Summary Table](#).

Figure 4-164. PCEPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x0

Table 4-182. PCEPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x0	<p>Ethernet PHY Module Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCEPHY, SCGCEPHY, or DCGCEPHY register is clear.</p> <p>0x0 = The EPHY module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The EPHY module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.159 PCCAN Register (Offset = 0x934) [reset = 0x3]

Controller Area Network Power Control (PCCAN)

The PCCAN register controls the power applied to the CAN module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCCAN, SCGCCAN, and DCGCCAN registers. If the Rn, Sn, or Dn bit of the respective RCGCCAN, SCGCCAN, and DCGCCAN registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCCAN register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCCAN, SCGCCAN, and DCGCCAN registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCCAN register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-183](#) lists the differences.

Table 4-183. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCCAN is shown in [Figure 4-165](#) and described in [Table 4-184](#).

Return to [Summary Table](#).

Figure 4-165. PCCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														P1	P0
R-0x0														R/W-0x1	R/W-0x1

Table 4-184. PCCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	P1	R/W	0x1	CAN Module 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCCAN, SCGCCAN, or DCGCCAN register is clear. 0x0 = The CAN module 1 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = The CAN module 1 is powered but does not receive a clock. In this case, the module is inactive.

Table 4-184. PCCAN Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	P0	R/W	0x1	<p>CAN Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCCAN, SCGCCAN, or DCGCCAN register is clear.</p> <p>0x0 = The CAN module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The CAN module 0 is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.160 PCADC Register (Offset = 0x938) [reset = 0x3]

Analog-to-Digital Converter Power Control (PCADC)

NOTE: The ADC module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCADC register controls the power applied to the ADC module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCAD, SCGCAD, and DCGCAD registers. If the Rn, Sn, or Dn bit of the respective RCGCAD, SCGCAD, and DCGCAD registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCADC register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCAD, SCGCAD, and DCGCAD registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCADC register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-185](#) lists the differences.

Table 4-185. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCADC is shown in [Figure 4-166](#) and described in [Table 4-186](#).

Return to [Summary Table](#).

Figure 4-166. PCADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														P1	P0
R-0x0														R/W-0x1	R/W-0x1

Table 4-186. PCADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	P1	R/W	0x1	<p>ADC Module 1 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCAD, SCGCAD, or DCGCAD register is clear.</p> <p>0x0 = The ADC module 1 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The ADC module 1 is powered but does not receive a clock. In this case, the module is inactive.</p>
0	P0	R/W	0x1	<p>ADC Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCAD, SCGCAD, or DCGCAD register is clear.</p> <p>0x0 = The ADC module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The ADC module 0 is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.161 PCACMP Register (Offset = 0x93C) [reset = 0x1]

Analog Comparator Power Control (PCACMP)

NOTE: The ACMP module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCACMP register controls the power applied to the ACMP module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCACMP, SCGCACMP, and DCGCACMP registers. If the Rn, Sn, or Dn bit of the respective RCGCACMP, SCGCACMP, and DCGCACMP registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCACMP register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCACMP, SCGCACMP, and DCGCACMP registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCACMP register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-187](#) lists the differences.

Table 4-187. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCACMP is shown in [Figure 4-167](#) and described in [Table 4-188](#).

Return to [Summary Table](#).

Figure 4-167. PCACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-188. PCACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>Analog Comparator Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCACMP, SCGCACMP, or DCGCACMP register is clear.</p> <p>0x0 = The Analog Comparator module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The Analog Comparator module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.162 PCPWM Register (Offset = 0x940) [reset = 0x1]

Pulse Width Modulator Power Control (PCPWM)

register type RW, reset 0x00000001

NOTE: The PWM module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCPWM register controls the power applied to the PWM module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCPWM, SCGCPWM, and DCGCPWM registers. If the Rn, Sn, or Dn bit of the respective RCGCPWM, SCGCPWM, and DCGCPWM registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCPWM register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCPWM, SCGCPWM, and DCGCPWM registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCPWM register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-189](#) lists the differences.

Table 4-189. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCPWM is shown in [Figure 4-168](#) and described in [Table 4-190](#).

Return to [Summary Table](#).

Figure 4-168. PCPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-190. PCPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>PWM Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCPWM, SCGCPWM, or DCGCPWM register is clear.</p> <p>0x0 = The PWM module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The PWM module 0 is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.163 PCQEI Register (Offset = 0x944) [reset = 0x1]

Quadrature Encoder Interface Power Control (PCQEI)

register type RW, reset 0x00000001

NOTE: The QEI module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCQEI register controls the power applied to the QEI module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCQEI, SCGCQEI, and DCGCQEI registers. If the Rn, Sn, or Dn bit of the respective RCGCQEI, SCGCQEI, and DCGCQEI registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCQEI register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCQEI, SCGCQEI, and DCGCQEI registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCQEI register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-191](#) lists the differences.

Table 4-191. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCQEI is shown in [Figure 4-169](#) and described in [Table 4-192](#).

Return to [Summary Table](#).

Figure 4-169. PCQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-192. PCQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>QEI Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCQEI, SCGCQEI, or DCGCQEI register is clear.</p> <p>0x0 = QEI module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = QEI module 0 is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.164 PCEEPROM Register (Offset = 0x958) [reset = 0x1]

EEPROM Power Control (PCEEPROM)

NOTE: The EEPROM module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCEEPROM register controls the power applied to the EEPROM module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCEEPROM, SCGCEEPROM, and DCGCEEPROM registers. If the Rn, Sn, or Dn bit of the respective RCGCEEPROM, SCGCEEPROM, and DCGCEEPROM registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCEEPROM register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCEEPROM, SCGCEEPROM, and DCGCEEPROM registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCEEPROM register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-193](#) lists the differences.

Table 4-193. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCEEPROM is shown in [Figure 4-170](#) and described in [Table 4-194](#).

Return to [Summary Table](#).

Figure 4-170. PCEEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-194. PCEEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>EEPROM Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCEEPROM, SCGCCEEPROM, or DCGCEEPROM register is clear.</p> <p>0x0 = The EEPROM module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The EEPROM module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.165 PCCCM Register (Offset = 0x974) [reset = 0x1]

CRC and Cryptographic Modules Power Control (PCCCM)

The PCCCM register controls the power applied to the CRC and AES, DES, and SHA/MD5 modules. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCCM, SCGCCM, and DCGCCM registers. If the Rn, Sn, or Dn bit of the respective RCGCCM, SCGCCM, and DCGCCM registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCCCM register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCCM, SCGCCM, and DCGCCM registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCCCM register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-195](#) lists the differences.

Table 4-195. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCCCM is shown in [Figure 4-171](#) and described in [Table 4-196](#).

Return to [Summary Table](#).

Figure 4-171. PCCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-196. PCCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>CRC and Cryptographic Modules Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCCCM, SCGCCCM, or DCGCCCM register is clear.</p> <p>0x0 = The CRC, AES, DES, and SHA/MD5 modules are not powered and do not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The CRC, AES, DES, and SHA/MD5 modules are powered, but do not receive a clock. In this case, the modules are inactive.</p>

4.2.166 PCLCD Register (Offset = 0x990) [reset = 0x1]

LCD Controller Power Control (PCLCD)

The PCLCD register controls the power applied to the LCD module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCLCD, SCGCLCD, and DCGCLCD registers. If the Rn, Sn, or Dn bit of the respective RCGCLCD, SCGCLCD, and DCGCLCD registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCLCD register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCLCD, SCGCLCD, and DCGCLCD registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCLCD register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-197](#) lists the differences.

Table 4-197. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCLCD is shown in [Figure 4-172](#) and described in [Table 4-198](#).

Return to [Summary Table](#).

Figure 4-172. PCLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-198. PCLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	LCD Controller Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCLCD, SCGCLCD, or DCGCLCD register is clear. 0x0 = LCD module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = LCD module 0 is powered but does not receive a clock. In this case, the module is inactive.

4.2.167 PCOWIRE Register (Offset = 0x998) [reset = 0x1]

1-Wire Power Control (PCOWIRE)

NOTE: The 1-Wire module does not currently provide the ability to respond to the power-down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The PCOWIRE register controls the power applied to the 1-Wire module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCOWIRE, SCGCOWIRE, and DCGCOWIRE registers. If the Rn, Sn, or Dn bit of the respective RCGCOWIRE, SCGCOWIRE, and DCGCOWIRE registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCOWIRE register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCOWIRE, SCGCOWIRE, and DCGCOWIRE registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCOWIRE register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-199](#) lists the differences.

Table 4-199. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCOWIRE is shown in [Figure 4-173](#) and described in [Table 4-200](#).

Return to [Summary Table](#).

Figure 4-173. PCOWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-200. PCOWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	<p>1-Wire Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCOWIRE, SCGCOWIRE, or DCGCOWIRE register is clear.</p> <p>0x0 = The 1-Wire module is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state.</p> <p>0x1 = The 1-Wire module is powered but does not receive a clock. In this case, the module is inactive.</p>

4.2.168 PCEMAC Register (Offset = 0x99C) [reset = 0x1]

Ethernet MAC Power Control (PCEMAC)

The PCEMAC register controls the power applied to the EMAC module. The function of this bit depends on the current state of the device (run, sleep or deep-sleep mode) and value of the corresponding bits in the RCGCEMAC, SCGCCEMAC, and DCGCEMAC registers. If the Rn, Sn, or Dn bit of the respective RCGCEMAC, SCGCCEMAC, and DCGCEMAC registers is 1 and the device is in that mode, the module is powered and receives a clock regardless of what the corresponding Pn bit in the PCEMAC register is.

However, if the Rn, Sn, or Dn bit of the respective RCGCEMAC, SCGCCEMAC, and DCGCEMAC registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the PCEMAC register. In this case, when the Pn bit is clear, the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. [Table 4-201](#) lists the differences.

Table 4-201. Module Power Control

Rn, Sn, or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register	Pn	Description
0	0	Module is not powered and does not receive a clock. In this case, the state of the peripheral is not retained. This is the lowest power consumption state of any peripheral, because it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the RCGCx, SCGCx, or DCGCx register is 1 or the P0 bit is changed to 1. Software must reinitialize the peripheral when reenabled due to the loss of state.
0	1	Module is powered but does not receive a clock. In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral, because it consumes only leakage current.
1	X	Module is powered and receives a clock.

PCEMAC is shown in [Figure 4-174](#) and described in [Table 4-202](#).

Return to [Summary Table](#).

Figure 4-174. PCEMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															P0
R-0x0															R/W-0x1

Table 4-202. PCEMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	P0	R/W	0x1	Ethernet MAC Module 0 Power Control. The Pn bit encodings do not apply if the corresponding bit in the RCGCEMAC, SCGCCEMAC, or DCGCEMAC register is clear. 0x0 = Ethernet MAC Module 0 is not powered and does not receive a clock. In this case, the state of the module is not retained. This configuration provides the lowest power consumption state. 0x1 = Ethernet MAC module 0 is powered but does not receive a clock. In this case, the module is inactive.

4.2.169 PRWD Register (Offset = 0xA00) [reset = 0x0]

Watchdog Timer Peripheral Ready (PRWD)

The PRWD register indicates whether the watchdog modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCWD bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCWD bit is changed. A reset change is initiated if the corresponding SRWD bit is changed from 0 to 1.

The PRWD bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRWD is shown in [Figure 4-175](#) and described in [Table 4-203](#).

Return to [Summary Table](#).

Figure 4-175. PRWD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R-0x0	R-0x0

Table 4-203. PRWD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R	0x0	Watchdog Timer 1 Peripheral Ready 0x0 = Watchdog module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = Watchdog module 1 is ready for access.
0	R0	R	0x0	Watchdog Timer 0 Peripheral Ready 0x0 = Watchdog module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = Watchdog module 0 is ready for access.

4.2.170 PRTIMER Register (Offset = 0xA04) [reset = 0x00]

16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER)

The PRGPT32 register indicates whether the timer modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCGPT32 bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCGPT32 bit is changed. A reset change is initiated if the corresponding SRGPT32 bit is changed from 0 to 1.

The PRGPT32 bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRTIMER is shown in [Figure 4-176](#) and described in [Table 4-204](#).

Return to [Summary Table](#).

Figure 4-176. PRTIMER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								R7	R6	R5	R4	R3	R2	R1	R0
R-0x0								R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 4-204. PRTIMER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	R7	R	0x0	16/32-Bit General-Purpose Timer 7 Peripheral Ready 0x0 = 16/32-bit timer module 7 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 7 is ready for access.
6	R6	R	0x0	16/32-Bit General-Purpose Timer 6 Peripheral Ready 0x0 = 16/32-bit timer module 6 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 6 is ready for access.
5	R5	R	0x0	16/32-Bit General-Purpose Timer 5 Peripheral Ready 0x0 = 16/32-bit timer module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 5 is ready for access.
4	R4	R	0x0	16/32-Bit General-Purpose Timer 4 Peripheral Ready 0x0 = 16/32-bit timer module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 4 is ready for access.
3	R3	R	0x0	16/32-Bit General-Purpose Timer 3 Peripheral Ready 0x0 = 16/32-bit timer module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 3 is ready for access.
2	R2	R	0x0	16/32-Bit General-Purpose Timer 2 Peripheral Ready 0x0 = 16/32-bit timer module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 2 is ready for access.

Table 4-204. PRTIMER Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	R1	R	0x0	16/32-Bit General-Purpose Timer 1 Peripheral Ready 0x0 = 16/32-bit timer module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 1 is ready for access.
0	R0	R	0x0	16/32-Bit General-Purpose Timer 0 Peripheral Ready 0x0 = 16/32-bit timer module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 16/32-bit timer module 0 is ready for access.

4.2.171 PRGPIO Register (Offset = 0xA08) [reset = 0x00]

General-Purpose Input/Output Peripheral Ready (PRGPIO)

The PRGPIO register indicates whether the GPIO modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCGPIO bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCGPIO bit is changed. A reset change is initiated if the corresponding SRGPIO bit is changed from 0 to 1.

The PRGPIO bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRGPIO is shown in [Figure 4-177](#) and described in [Table 4-205](#).

Return to [Summary Table](#).

Figure 4-177. PRGPIO Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED														R17	R16
R-0x0														R-0x0	R-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 4-205. PRGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	R17	R	0x0	GPIO Port T Peripheral Ready 0x0 = GPIO port T is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port T is ready for access.
16	R16	R	0x0	GPIO Port S Peripheral Ready 0x0 = GPIO port S is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port S is ready for access.
15	R15	R	0x0	GPIO Port R Peripheral Ready 0x0 = GPIO port R is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port R is ready for access.
14	R14	R	0x0	GPIO Port Q Peripheral Ready 0x0 = GPIO port Q is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port Q is ready for access.
13	R13	R	0x0	GPIO Port P Peripheral Ready 0x0 = GPIO port P is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port P is ready for access.
12	R12	R	0x0	GPIO Port N Peripheral Ready 0x0 = GPIO port N is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port N is ready for access.
11	R11	R	0x0	GPIO Port M Peripheral Ready 0x0 = GPIO port M is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port M is ready for access.

Table 4-205. PRGPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	R10	R	0x0	GPIO Port L Peripheral Ready 0x0 = GPIO port L is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port L is ready for access.
9	R9	R	0x0	GPIO Port K Peripheral Ready 0x0 = GPIO port K is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port K is ready for access.
8	R8	R	0x0	GPIO Port J Peripheral Ready 0x0 = GPIO port J is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port J is ready for access.
7	R7	R	0x0	GPIO Port H Peripheral Ready 0x0 = GPIO port H is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port H is ready for access.
6	R6	R	0x0	GPIO Port G Peripheral Ready 0x0 = GPIO port G is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port G is ready for access.
5	R5	R	0x0	GPIO Port F Peripheral Ready 0x0 = GPIO port F is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port F is ready for access.
4	R4	R	0x0	GPIO Port E Peripheral Ready 0x0 = GPIO port E is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port E is ready for access.
3	R3	R	0x0	GPIO Port D Peripheral Ready 0x0 = GPIO port D is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port D is ready for access.
2	R2	R	0x0	GPIO Port C Peripheral Ready 0x0 = GPIO port C is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port C is ready for access.
1	R1	R	0x0	GPIO Port B Peripheral Ready 0x0 = GPIO port B is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port B is ready for access.
0	R0	R	0x0	GPIO Port A Peripheral Ready 0x0 = GPIO port A is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = GPIO port A is ready for access.

4.2.172 PRDMA Register (Offset = 0xA0C) [reset = 0x0]

Micro Direct Memory Access Peripheral Ready (PRDMA)

The PRDMA register indicates whether the μ DMA module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCDMA bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGDMA bit is changed. A reset change is initiated if the corresponding SRDMA bit is changed from 0 to 1.

The PRDMA bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRDMA is shown in [Figure 4-178](#) and described in [Table 4-206](#).

Return to [Summary Table](#).

Figure 4-178. PRDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-206. PRDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	μ DMA Module Peripheral Ready 0x0 = The μ DMA module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = The μ DMA module is ready for access.

4.2.173 PREPI Register (Offset = 0xA10) [reset = 0x0]

EPI Peripheral Ready (PREPI)

The PREPI register indicates whether the EPI module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCEPI bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCEPI bit is changed. A reset change is initiated if the corresponding SREPI bit is changed from 0 to 1.

The PREPI bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PREPI is shown in [Figure 4-179](#) and described in [Table 4-207](#).

Return to [Summary Table](#).

Figure 4-179. PREPI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-207. PREPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	EPI Module Peripheral Ready 0x0 = The EPI module is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = The EPI module is ready for access.

4.2.174 PRHIB Register (Offset = 0xA14) [reset = 0x1]

Hibernation Peripheral Ready (PRHIB)

The PRHIB register indicates whether the Hibernation module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCHIB bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCHIB bit is changed. A reset change is initiated if the corresponding SRHIB bit is changed from 0 to 1.

The PRHIB bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRHIB is shown in [Figure 4-180](#) and described in [Table 4-208](#).

Return to [Summary Table](#).

Figure 4-180. PRHIB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x1

Table 4-208. PRHIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x1	<p>Hibernation Module Peripheral Ready</p> <p>0x0 = The Hibernation module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.</p> <p>0x1 = The Hibernation module is ready for access.</p>

4.2.175 PRUART Register (Offset = 0xA18) [reset = 0x00]

Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART)

The PRUART register indicates whether the UART modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCUART bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCUART bit is changed. A reset change is initiated if the corresponding SRUART bit is changed from 0 to 1.

The PRUART bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRUART is shown in [Figure 4-181](#) and described in [Table 4-209](#).

Return to [Summary Table](#).

Figure 4-181. PRUART Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								R7	R6	R5	R4	R3	R2	R1	R0
R-0x0								R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 4-209. PRUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	R7	R	0x0	UART Module 7 Peripheral Ready 0x0 = UART module 7 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 7 is ready for access.
6	R6	R	0x0	UART Module 6 Peripheral Ready 0x0 = UART module 6 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 6 is ready for access.
5	R5	R	0x0	UART Module 5 Peripheral Ready 0x0 = UART module 5 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 5 is ready for access.
4	R4	R	0x0	UART Module 4 Peripheral Ready 0x0 = UART module 4 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 4 is ready for access.
3	R3	R	0x0	UART Module 3 Peripheral Ready 0x0 = UART module 3 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 3 is ready for access.
2	R2	R	0x0	UART Module 2 Peripheral Ready 0x0 = UART module 2 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 2 is ready for access.
1	R1	R	0x0	UART Module 1 Peripheral Ready 0x0 = UART module 1 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 1 is ready for access.

Table 4-209. PRUART Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	R0	R	0x0	UART Module 0 Peripheral Ready 0x0 = UART module 0 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = UART module 0 is ready for access.

4.2.176 PRSSI Register (Offset = 0xA1C) [reset = 0x0]

Synchronous Serial Interface Peripheral Ready (PRSSI)

The PRSSI register indicates whether the SSI modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCSSI bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCSSI bit is changed. A reset change is initiated if the corresponding SRSSI bit is changed from 0 to 1.

The PRSSI bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRSSI is shown in [Figure 4-182](#) and described in [Table 4-210](#).

Return to [Summary Table](#).

Figure 4-182. PRSSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												R3	R2	R1	R0
R-0x0												R-0x0	R-0x0	R-0x0	R-0x0

Table 4-210. PRSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	R3	R	0x0	SSI Module 3 Peripheral Ready 0x0 = SSI module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = SSI module 3 is ready for access.
2	R2	R	0x0	SSI Module 2 Peripheral Ready 0x0 = SSI module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = SSI module 2 is ready for access.
1	R1	R	0x0	SSI Module 1 Peripheral Ready 0x0 = SSI module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = SSI module 1 is ready for access.
0	R0	R	0x0	SSI Module 0 Peripheral Ready 0x0 = SSI module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = SSI module 0 is ready for access.

4.2.177 PRI2C Register (Offset = 0xA20) [reset = 0x00]

Inter-Integrated Circuit Peripheral Ready (PRI2C)

The PRI2C register indicates whether the I2C modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCI2C bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCI2C bit is changed. A reset change is initiated if the corresponding SRI2C bit is changed from 0 to 1.

The PRI2C bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRI2C is shown in [Figure 4-183](#) and described in [Table 4-211](#).

Return to [Summary Table](#).

Figure 4-183. PRI2C Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
R-0x0						R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 4-211. PRI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	R9	R	0x0	I2C Module 9 Peripheral Ready 0x0 = I2C module 9 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 9 is ready for access.
8	R8	R	0x0	I2C Module 8 Peripheral Ready 0x0 = I2C module 8 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 8 is ready for access.
7	R7	R	0x0	I2C Module 7 Peripheral Ready 0x0 = I2C module 7 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 7 is ready for access.
6	R6	R	0x0	I2C Module 6 Peripheral Ready 0x0 = I2C module 6 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 6 is ready for access.
5	R5	R	0x0	I2C Module 5 Peripheral Ready 0x0 = I2C module 5 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 5 is ready for access.
4	R4	R	0x0	I2C Module 4 Peripheral Ready 0x0 = I2C module 4 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 4 is ready for access.
3	R3	R	0x0	I2C Module 3 Peripheral Ready 0x0 = I2C module 3 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 3 is ready for access.

Table 4-211. PRI2C Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	R2	R	0x0	I2C Module 2 Peripheral Ready 0x0 = I2C module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 2 is ready for access.
1	R1	R	0x0	I2C Module 1 Peripheral Ready 0x0 = I2C module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 1 is ready for access.
0	R0	R	0x0	I2C Module 0 Peripheral Ready 0x0 = I2C module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = I2C module 0 is ready for access.

4.2.178 PRUSB Register (Offset = 0xA28) [reset = 0x0]

Universal Serial Bus Peripheral Ready (PRUSB)

The PRUSB register indicates whether the USB module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCUSB bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCUSB bit is changed. A reset change is initiated if the corresponding SRUSB bit is changed from 0 to 1.

The PRUSB bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRUSB is shown in [Figure 4-184](#) and described in [Table 4-212](#).

Return to [Summary Table](#).

Figure 4-184. PRUSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-212. PRUSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	USB Module Peripheral Ready 0x0 = The USB module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = The USB module is ready for access.

4.2.179 PREPHY Register (Offset = 0xA30) [reset = 0x0]

Ethernet PHY Peripheral Ready (PREPHY)

The PREPHY register indicates whether the Ethernet PHY module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCEPHY bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCEPHY bit is changed. A reset change is initiated if the corresponding SREPHY bit is changed from 0 to 1.

The PREPHY bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PREPHY is shown in [Figure 4-185](#) and described in [Table 4-213](#).

Return to [Summary Table](#).

Figure 4-185. PREPHY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-213. PREPHY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	<p>Ethernet PHY Module Peripheral Ready</p> <p>0x0 = The Ethernet PHY module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.</p> <p>0x1 = The Ethernet PHY module is ready for access.</p>

4.2.180 PRCAN Register (Offset = 0xA34) [reset = 0x0]

Controller Area Network Peripheral Ready (PRCAN)

The PRCAN register indicates whether the CAN modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCCAN bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCCAN bit is changed. A reset change is initiated if the corresponding SRCAN bit is changed from 0 to 1.

The PRCAN bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRCAN is shown in [Figure 4-186](#) and described in [Table 4-214](#).

Return to [Summary Table](#).

Figure 4-186. PRCAN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R-0x0	R-0x0

Table 4-214. PRCAN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R	0x0	CAN Module 1 Peripheral Ready 0x0 = CAN module 1 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = CAN module 1 is ready for access.
0	R0	R	0x0	CAN Module 0 Peripheral Ready 0x0 = CAN module 0 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = CAN module 0 is ready for access.

4.2.181 PRADC Register (Offset = 0xA38) [reset = 0x0]

Analog-to-Digital Converter Peripheral Ready (PRADC)

The PRADC register indicates whether the ADC modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCADC bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGADC bit is changed. A reset change is initiated if the corresponding SRADC bit is changed from 0 to 1.

The PRADC bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRADC is shown in [Figure 4-187](#) and described in [Table 4-215](#).

Return to [Summary Table](#).

Figure 4-187. PRADC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														R1	R0
R-0x0														R-0x0	R-0x0

Table 4-215. PRADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	R1	R	0x0	ADC Module 1 Peripheral Ready 0x0 = ADC module 1 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = ADC module 1 is ready for access.
0	R0	R	0x0	ADC Module 0 Peripheral Ready 0x0 = ADC module 0 is not ready for access. It is unlocked, unpowered, or in the process of completing a reset sequence. 0x1 = ADC module 0 is ready for access.

4.2.182 PRACMP Register (Offset = 0xA3C) [reset = 0x0]

Analog Comparator Peripheral Ready (PRACMP)

The PRACMP register indicates whether the analog comparator module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCACMP bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCACMP bit is changed. A reset change is initiated if the corresponding SRACMP bit is changed from 0 to 1.

The PRACMP bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRACMP is shown in [Figure 4-188](#) and described in [Table 4-216](#).

Return to [Summary Table](#).

Figure 4-188. PRACMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-216. PRACMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	Analog Comparator Module 0 Peripheral Ready 0x0 = The analog comparator module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = The analog comparator module is ready for access.

4.2.183 PRPWM Register (Offset = 0xA40) [reset = 0x0]

Pulse Width Modulator Peripheral Ready (PRPWM)

The PRPWM register indicates whether the PWM modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCPWM bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCPWM bit is changed. A reset change is initiated if the corresponding SRPWM bit is changed from 0 to 1.

The PRPWM bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRPWM is shown in [Figure 4-189](#) and described in [Table 4-217](#).

Return to [Summary Table](#).

Figure 4-189. PRPWM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-217. PRPWM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	PWM Module 0 Peripheral Ready 0x0 = PWM module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = PWM module 0 is ready for access.

4.2.184 PRQEI Register (Offset = 0xA44) [reset = 0x0]

Quadrature Encoder Interface Peripheral Ready (PRQEI)

The PRQEI register indicates whether the QEI modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCQEI bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCQEI bit is changed. A reset change is initiated if the corresponding SRQEI bit is changed from 0 to 1.

The PRQEI bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRQEI is shown in [Figure 4-190](#) and described in [Table 4-218](#).

Return to [Summary Table](#).

Figure 4-190. PRQEI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-218. PRQEI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	QEI Module 0 Peripheral Ready 0x0 = QEI module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = QEI module 0 is ready for access.

4.2.185 PREEPROM Register (Offset = 0xA58) [reset = 0x0]

EEPROM Peripheral Ready (PREEPROM)

The PREEPROM register indicates whether the EEPROM module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCEEPROM bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCEEPROM bit is changed. A reset change is initiated if the corresponding SREEPROM bit is changed from 0 to 1.

The PREEPROM bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PREEPROM is shown in [Figure 4-191](#) and described in [Table 4-219](#).

Return to [Summary Table](#).

Figure 4-191. PREEPROM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-219. PREEPROM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	EEPROM Module 0 Peripheral Ready 0x0 = The EEPROM module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = The EEPROM module is ready for access.

4.2.186 PRCCM Register (Offset = 0xA74) [reset = 0x0]

CRC and Cryptographic Modules Peripheral Ready (PRCCM)

The PRCCM register indicates whether the CRC and Cryptographic Modules(AES, DES, and SHA/MD5) are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCCCM bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCCM bit is changed. A reset change is initiated if the corresponding SRCCM bit is changed from 0 to 1.

The PRCCM bit is cleared on any of the preceding events and is not set again until the modules are completely powered, enabled, and internally reset.

PRCCM is shown in [Figure 4-192](#) and described in [Table 4-220](#).

Return to [Summary Table](#).

Figure 4-192. PRCCM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-220. PRCCM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	<p>CRC and Cryptographic Modules Peripheral Ready</p> <p>0x0 = The CRC, AES, DES, and SHA/MD modules are not ready for access. They are unclocked, unpowered, or in the process of completing a reset sequence.</p> <p>0x1 = The CRC, AES, DES, and SHA/MD modules are ready for access.</p>

4.2.187 PRLCD Register (Offset = 0xA90) [reset = 0x0]

LCD Controller Peripheral Ready (PRLCD)

The PRLCD register indicates whether the LCD Controller modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCLCD bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGLCD bit is changed. A reset change is initiated if the corresponding SRLCD bit is changed from 0 to 1.

The PRLCD bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PRLCD is shown in [Figure 4-193](#) and described in [Table 4-221](#).

Return to [Summary Table](#).

Figure 4-193. PRLCD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-221. PRLCD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	LCD Controller Module 0 Peripheral Ready 0x0 = LCD Controller module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = LCD Controller module 0 is ready for access.

4.2.188 PROWIRE Register (Offset = 0xA98) [reset = 0x0]

1-Wire Peripheral Ready (PROWIRE)

The PROWIRE register indicates whether the 1-Wire modules are ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCOWIRE bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCOWIRE bit is changed. A reset change is initiated if the corresponding SROWIRE bit is changed from 0 to 1.

The PROWIRE bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PROWIRE is shown in [Figure 4-194](#) and described in [Table 4-222](#).

Return to [Summary Table](#).

Figure 4-194. PROWIRE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-222. PROWIRE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	1-Wire Module 0 Peripheral Ready 0x0 = 1-Wire module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 0x1 = 1-Wire module 0 is ready for access.

4.2.189 PREMAC Register (Offset = 0xA9C) [reset = 0x0]

Ethernet MAC Peripheral Ready (PREMAC)

The PREMAC register indicates whether the Ethernet module is ready to be accessed by software following a change in status of power, run mode clocking, or reset. A power change is initiated if the corresponding PCEMAC bit is changed from 0 to 1. A run mode clocking change is initiated if the corresponding RCGCEMAC bit is changed. A reset change is initiated if the corresponding SREMAC bit is changed from 0 to 1.

The PREMAC bit is cleared on any of the preceding events and is not set again until the module is completely powered, enabled, and internally reset.

PREMAC is shown in [Figure 4-195](#) and described in [Table 4-223](#).

Return to [Summary Table](#).

Figure 4-195. PREMAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R0
R-0x0															R-0x0

Table 4-223. PREMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	R0	R	0x0	<p>Ethernet MAC Module 0 Peripheral Ready</p> <p>0x0 = Ethernet MAC module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.</p> <p>0x1 = Ethernet MAC module 0 is ready for access.</p>

4.2.190 UNIQUEID0 to UNIQUEID3 Register (Offset = 0xF20 to 0xF2C) [reset = X]

Unique ID 0 (UNIQUEID0), offset 0xF20

Unique ID 1 (UNIQUEID1), offset 0xF24

Unique ID 2 (UNIQUEID2), offset 0xF28

Unique ID 3 (UNIQUEID3), offset 0xF2C

These registers contain a unique 128-bit identifier that cannot be modified by the user. This value is unique to each individual die but is not a random value. This unique device identifier can be used to initiate secure boot processes or as a serial number for USB or other end applications.

UNIQUEIDn is shown in [Figure 4-196](#) and described in [Table 4-224](#).

Return to [Summary Table](#).

Figure 4-196. UNIQUEIDn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID															
R-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															
R-X															

Table 4-224. UNIQUEIDn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ID	R	X	Unique ID The result of registers 0 to 3 concatenated defines the unique 128-bit device identifier.

4.3 Cryptographic System Control (CCM) Registers

This section lists and describes the system control registers for the CRC and cryptographic (AES, DES, SHA) modules. All address offsets are relative to the CCM base address of 0x44030000.

[Table 4-225](#) lists the memory-mapped registers for the System Control. All register offset addresses not listed in [Table 4-225](#) should be considered as reserved locations and the register contents should not be modified.

Additional flash and ROM registers defined in the System Control register space are described in the [Chapter 7](#).

Table 4-225. CCM Registers

Offset	Acronym	Register Name	Section
0x204	CCMCGREQ	Cryptographic Modules Clock Gating Request	Section 4.3.1

Complex bit access types are encoded to fit into small table cells. [Table 4-226](#) lists the codes that are used for access types in this section.

Table 4-226. CCM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

4.3.1 CCMCGREQ Register (Offset = 0x204) [reset = 0x0]

Cryptographic Modules Clock Gating Request (CCMCGREQ)

The Cryptographic Modules Clock Gating Request (CCMCGREQ) register is written to enable or disable clock gating in the AES, DES, and SHA/MD5 Module.

CCMCGREQ is shown in [Figure 4-197](#) and described in [Table 4-227](#).

Return to [Summary Table](#).

Figure 4-197. CCMCGREQ Register

31	30	29	28	27	26	25	24
RESERVED							
R/W-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0x0							
7	6	5	4	3	2	1	0
RESERVED					DESCFG	AESCFG	SHACFG
R/W-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 4-227. CCMCGREQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R/W	0x0	
2	DESCFG	R/W	0x0	DES Clock Gating Request 0x0 = Request to ungate clock gating 0x1 = Request to gate the clock
1	AESCFG	R/W	0x0	AES Clock Gating Request 0x0 = Request to ungate clock gating 0x1 = Request to gate the clock
0	SHACFG	R/W	0x0	SHA/MD5 Clock Gating Request 0x0 = Request to ungate clock gating 0x1 = Request to gate the clock

Processor Support and Exception Module

This module is an AHB peripheral that handles system-level Cortex-M4 FPU exceptions. For functions with registers mapped into this aperture, if the function is not available on a device, then all writes to the associated registers are ignored and reads return zeros.

Topic	Page
5.1 Functional Description	469
5.2 System Exception Registers	470

5.1 Functional Description

The System Exception module provides control and status of the system-level interrupts. All the interrupt events are ORed together before being sent to the interrupt controller, so the System Exception module can generate only one interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the System Exception Masked Interrupt Status (SYSEXCMI) register. The interrupt events that can trigger a controller-level interrupt are defined in the System Exception Interrupt Mask (SYSEXCIM) register by setting the corresponding interrupt mask bits. If interrupts are not used, the raw interrupt status is always visible through the System Exception Raw Interrupt Status (SYSEXCRI) register. Interrupts are always cleared (for both the SYSEXCMI and SYSEXCRI registers) by writing 1 to the corresponding bit in the System Exception Interrupt Clear (SYSEXCIC) register.

5.2 System Exception Registers

[Table 5-1](#) lists the memory-mapped registers for the processor support and exception module. All register offset addresses not listed in [Table 5-1](#) should be considered as reserved locations and the register contents should not be modified.

Table 5-1. System Exception Registers

Offset	Acronym	Register Name	Section
0x0	SYSEXCRI	System Exception Raw Interrupt Status	Section 5.2.1
0x4	SYSEXCIM	System Exception Interrupt Mask	Section 5.2.2
0x8	SYSEXCMI	System Exception Masked Interrupt Status	Section 5.2.3
0xC	SYSEXCIC	System Exception Interrupt Clear	Section 5.2.4

Complex bit access types are encoded to fit into small table cells. [Table 5-2](#) shows the codes that are used for access types in this section.

Table 5-2. System Exception Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

5.2.1 SYSEXCRI Register (Offset = 0x0) [reset = 0x0]

System Exception Raw Interrupt Status (SYSEXCRI)

The SYSEXCRI register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

SYSEXCRI is shown in [Figure 5-1](#) and described in [Table 5-3](#).

Return to [Summary Table](#).

Figure 5-1. SYSEXCRI Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		FPIXCRIS	FPOFCRIS	FPUFCRIS	FPIOCRIS	FPDZCRIS	FPIDCRIS
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 5-3. SYSEXCRI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5	FPIXCRIS	R	0x0	Floating-Point Inexact Exception Raw Interrupt Status This bit is cleared by writing 1 to the IXCIC bit in the SYSEXCIC register.
4	FPOFCRIS	R	0x0	Floating-Point Overflow Exception Raw Interrupt Status This bit is cleared by writing 1 to the OFCIC bit in the SYSEXCIC register.
3	FPUFCRIS	R	0x0	Floating-Point Underflow Exception Raw Interrupt Status This bit is cleared by writing 1 to the UFCIC bit in the SYSEXCIC register.
2	FPIOCRIS	R	0x0	Floating-Point Invalid Operation Raw Interrupt Status This bit is cleared by writing 1 to the IOCIC bit in the SYSEXCIC register.
1	FPDZCRIS	R	0x0	Floating-Point Divide By 0 Exception Raw Interrupt Status This bit is cleared by writing 1 to the DZCIC bit in the SYSEXCIC register.
0	FPIDCRIS	R	0x0	Floating-Point Input Denormal Exception Raw Interrupt Status This bit is cleared by writing 1 to the IDCIC bit in the SYSEXCIC register.

5.2.2 SYSEXCIM Register (Offset = 0x4) [reset = 0x0]

System Exception Interrupt Mask (SYSEXCIM)

The SYSEXCIM register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

SYSEXCIM is shown in [Figure 5-2](#) and described in [Table 5-4](#).

Return to [Summary Table](#).

Figure 5-2. SYSEXCIM Register

31	30	29	28	27	26	25	24
RESERVED							
R/W-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0x0							
7	6	5	4	3	2	1	0
RESERVED		FPIXCIM	FPOFCIM	FPUFCIM	FPIOCIM	FPDZCIM	FPIDCIM
R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 5-4. SYSEXCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R/W	0x0	
5	FPIXCIM	R/W	0x0	Floating-Point Inexact Exception Interrupt Mask
4	FPOFCIM	R/W	0x0	Floating-Point Overflow Exception Interrupt Mask
3	FPUFCIM	R/W	0x0	Floating-Point Underflow Exception Interrupt Mask
2	FPIOCIM	R/W	0x0	Floating-Point Invalid Operation Interrupt Mask
1	FPDZCIM	R/W	0x0	Floating-Point Divide By 0 Exception Interrupt Mask
0	FPIDCIM	R/W	0x0	Floating-Point Input Denormal Exception Interrupt Mask

5.2.3 SYSEXCMIIS Register (Offset = 0x8) [reset = 0x0]

System Exception Masked Interrupt Status (SYSEXCMIIS)

The SYSEXCMIIS register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

SYSEXCMIIS is shown in [Figure 5-3](#) and described in [Table 5-5](#).

Return to [Summary Table](#).

Figure 5-3. SYSEXCMIIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		FPIXCMIS	FPOFCMIS	FPUFCMIS	FPIOCMIS	FPDZCMIS	FPIDCMIS
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 5-5. SYSEXCMIIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5	FPIXCMIS	R	0x0	Floating-Point Inexact Exception Masked Interrupt Status This bit is cleared by writing 1 to the FPIXICIC bit in the SYSEXCIC register.
4	FPOFCMIS	R	0x0	Floating-Point Overflow Exception Masked Interrupt Status This bit is cleared by writing 1 to the FPOFCIC bit in the SYSEXCIC register.
3	FPUFCMIS	R	0x0	Floating-Point Underflow Exception Masked Interrupt Status This bit is cleared by writing 1 to the FPUFCIC bit in the SYSEXCIC register.
2	FPIOCMIS	R	0x0	Floating-Point Invalid Operation Masked Interrupt Status This bit is cleared by writing 1 to the FPIOCIC bit in the SYSEXCIC register.
1	FPDZCMIS	R	0x0	Floating-Point Divide By 0 Exception Masked Interrupt Status This bit is cleared by writing 1 to the FPDZCIC bit in the SYSEXCIC register.
0	FPIDCMIS	R	0x0	Floating-Point Input Denormal Exception Masked Interrupt Status This bit is cleared by writing 1 to the FPIDCIC bit in the SYSEXCIC register.

5.2.4 SYSEXCIC Register (Offset = 0xC) [reset = 0x0]

System Exception Interrupt Clear (SYSEXCIC)

The SYSEXCIC register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

SYSEXCIC is shown in [Figure 5-4](#) and described in [Table 5-6](#).

Return to [Summary Table](#).

Figure 5-4. SYSEXCIC Register

31	30	29	28	27	26	25	24
RESERVED							
W1C-0x0							
23	22	21	20	19	18	17	16
RESERVED							
W1C-0x0							
15	14	13	12	11	10	9	8
RESERVED							
W1C-0x0							
7	6	5	4	3	2	1	0
RESERVED		FPIXCIC	FPOFCIC	FPUFCIC	FPIOCIC	FPDZCIC	FPIDCIC
W1C-0x0		W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0

Table 5-6. SYSEXCIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	W1C	0x0	
5	FPIXCIC	W1C	0x0	Floating-Point Inexact Exception Interrupt Clear Writing 1 to this bit clears the FPIXCRIS bit in the SYSEXCIC register and the FPIXCMIS bit in the SYSEXCIC register.
4	FPOFCIC	W1C	0x0	Floating-Point Overflow Exception Interrupt Clear Writing 1 to this bit clears the FPOFCRIS bit in the SYSEXCIC register and the FPOFCMIS bit in the SYSEXCIC register.
3	FPUFCIC	W1C	0x0	Floating-Point Underflow Exception Interrupt Clear Writing 1 to this bit clears the FPUFCRIS bit in the SYSEXCIC register and the FPUFCMIS bit in the SYSEXCIC register.
2	FPIOCIC	W1C	0x0	Floating-Point Invalid Operation Interrupt Clear Writing 1 to this bit clears the FPIOCRIS bit in the SYSEXCIC register and the FPIOCMIS bit in the SYSEXCIC register.
1	FPDZCIC	W1C	0x0	Floating-Point Divide By 0 Exception Interrupt Clear Writing 1 to this bit clears the FPDZCRIS bit in the SYSEXCIC register and the FPDZCMIS bit in the SYSEXCIC register.
0	FPIDCIC	W1C	0x0	Floating-Point Input Denormal Exception Interrupt Clear Writing 1 to this bit clears the FPIDCRIS bit in the SYSEXCIC register and the FPIDCMIS bit in the SYSEXCIC register.

Hibernation Module

The Hibernation Module manages removal and restoration of power to provide a means for reducing system power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal or at a certain time using the built-in Real-Time Clock (RTC). The Hibernation module can be independently supplied from an external battery or an auxiliary power supply.

Topic	Page
6.1 Introduction	476
6.2 Block Diagram.....	477
6.3 Functional Description	477
6.4 Initialization and Configuration	490
6.5 HIB Registers	493

6.1 Introduction

The Hibernation also integrates a tamper module which provides mechanisms to detect, respond to, and log system tampering events. The Tamper module is designed to be low power and operate from either a battery or the MCU I/O voltage supply.

The Hibernation module has the following features:

- 32-bit real-time seconds counter (RTC) with 1/32768 second resolution and a 15-bit subseconds counter
 - 32-bit RTC seconds match register and a 15-bit subseconds match for timed wake-up and interrupt generation with 1/32768 second resolution
 - RTC predivider trim for making fine adjustments to the clock rate
- Hardware Calendar Function
 - Year, Month, Day, Day of Week, Hours, Minutes, Seconds
 - Four-year leap compensation
 - 24-hour or AM/PM configuration
- Two mechanisms for power control
 - System power control using discrete external regulator
 - On-chip power control using internal switches under register control
- V_{DD} supplies power when valid, even if $V_{BAT} > V_{DD}$
- Dedicated pin for waking using an external signal
- Capability to configure external reset (RST) pin and/or up to four GPIO port pins as wake source, with programmable wake level
- Tamper Functionality
 - Support for four tamper inputs
 - Configurable level, weak pullup, and glitch filter
 - Configurable tamper event response
 - Logging of up to four tamper events
 - Optional BBRAM erase on tamper detection
 - Tamper wake from hibernate capability
 - Hibernation clock input failure detect with a switch to the internal oscillator on detection
- RTC operational and hibernation memory valid as long as V_{DD} or V_{BAT} is valid
- Low-battery detection, signaling, and interrupt generation, with optional wake on low battery
- GPIO pin state can be retained during hibernation
- Clock source from a n internal low frequency oscillator (HIB LFIOSC) or a 32.768-kHz external crystal or oscillator
- Sixteen 32-bit words of battery-backed memory to save state during hibernation
- Programmable interrupts for:
 - RTC match
 - External wake
 - Low battery

6.2 Block Diagram

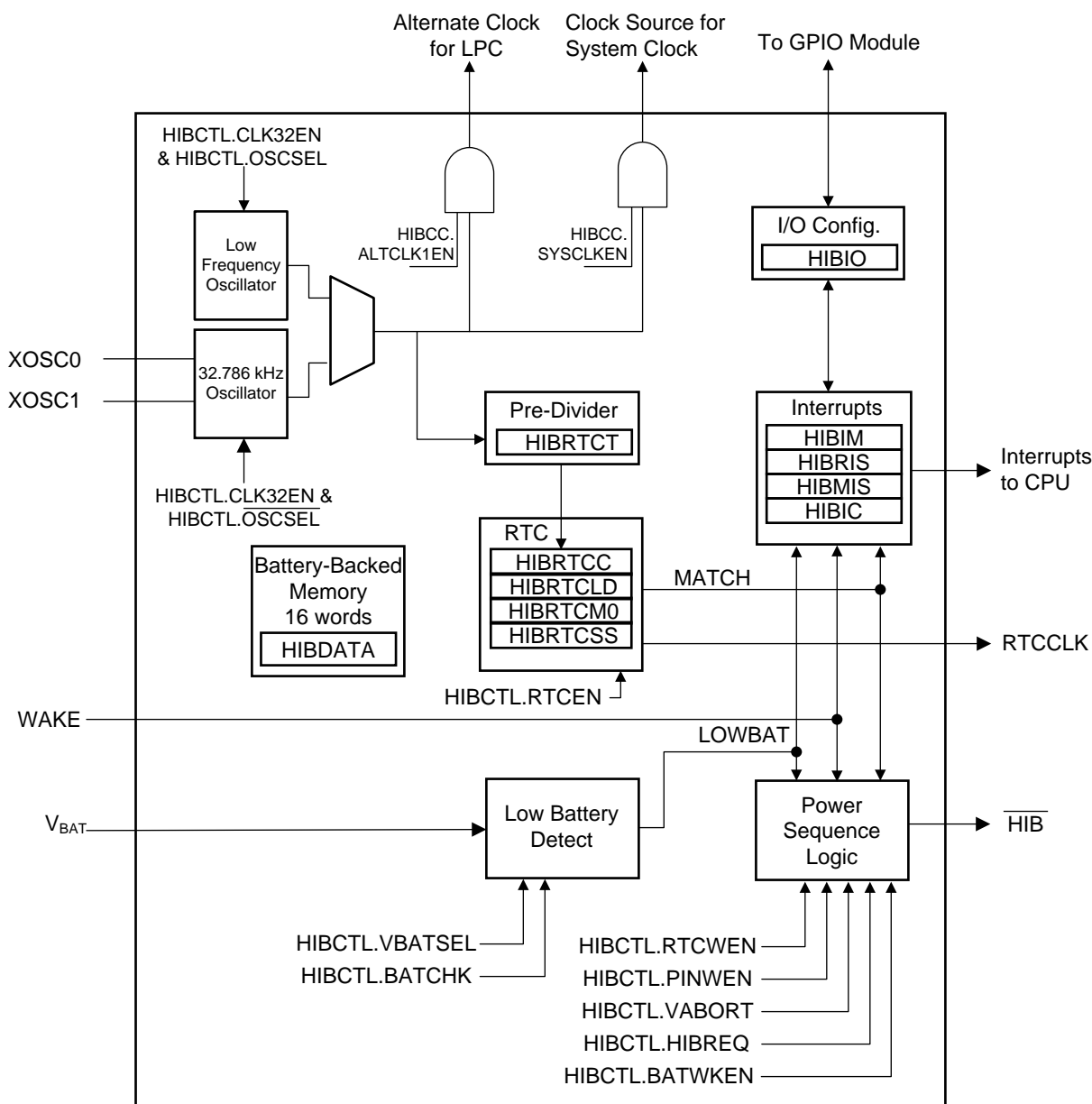


Figure 6-1. Hibernation Module Block Diagram

NOTE: References to alternate clock to LPC apply only to devices that have LPC.

6.3 Functional Description

The Hibernation module provides two mechanisms for power control:

- The first mechanism uses internal switches to control power to the Cortex-M4F as well as to most analog and digital functions while retaining I/O pin power (VDD3ON mode).
- The second mechanism controls the power to the microcontroller with a control signal (HIB) that signals an external voltage regulator to turn on or off.

The Hibernation module power source is supplied by V_{DD} as long as it is within a valid range, even if $V_{BAT} > V_{DD}$. The Hibernation module also has an independent clock source to maintain a real-time clock (RTC) when the system clock is powered down. Hibernate mode can be entered through one of two ways:

- The user initiates hibernation by setting the HIBREQ bit in the Hibernation Control (HIBCTL) register
- Power is arbitrarily removed from V_{DD} while a valid V_{BAT} is applied

When in hibernation, the module signals an external voltage regulator to turn the power back on when an external pin (WAKE, RST or a wake-enabled GPIO pin) is asserted or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low and optionally prevent hibernation or wake from hibernation when the battery voltage falls below a certain threshold. Note that multiple wake sources can be configured at the same time to generate a wake signal such that any of them can wake the module.

When waking from hibernation, the HIB signal is deasserted. The return of V_{DD} causes a POR to be executed. The time from when the WAKE signal is asserted to when code begins execution is equal to the wake-up time ($t_{WAKE_TO_HIB}$) plus the power-on-reset time (t_{POR}).

6.3.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, hibernation registers must be written only with a timing gap between accesses. The delay time is $t_{HIB_REG_ACCESS}$, therefore software must guarantee that this delay is inserted between back-to-back writes to Hibernation registers or between a write followed by a read. The WC interrupt in the HIBMIS register can be used to notify the application when the Hibernation modules registers can be accessed. Alternatively, software may make use of the WRC bit in the Hibernation Control (HIBCTL) register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll HIBCTL for WRC = 1 prior to accessing any hibernation register.

Back-to-back reads from Hibernation module registers have no timing restrictions. Reads are performed at the full peripheral clock rate.

6.3.2 Hibernation Clock Source

The HIB module can be clocked by one of three different clock sources:

- A 32.768-kHz oscillator
- An external 32.768-kHz clock source
- An internal low frequency oscillator (HIB LFIOOSC)

[Table 6-1](#) summarizes the encodings for the bits in the HIBCTL register that are required for each clock source to be enabled. Note that CLK32EN must be set for any Hibernation clock source to be valid. The Hibernation module is not enabled until the CLK32EN bit is set. The HIB clock source is the source of the RTC Oscillator (RTCOSC), which can be selected as the system clock source by programming a 0x4 in the OSCSRC field of the Run and Sleep Mode Configuration (RSCLKCFG) register in the System Control Module. See [Chapter 4](#) for more information.

Table 6-1. HIB Clock Source Configurations

HIB Clock Source	CLK32EN	OSCSEL	OSCBYP
32.768 kHz oscillator	1	0	0
External 32.768-kHz clock source	1	0	1
Low-frequency internal oscillator (HIB LFIOOSC) ⁽¹⁾	1	1	0

⁽¹⁾ The frequency can have wide variations; see the HIB electrical specifications in the device-specific data sheet for more details.

To use an external crystal, a 32.768-kHz crystal is connected to the XOSC0 and XOSC1 pins. Alternatively, a 32.768-kHz oscillator can be connected to the XOSC0 pin, leaving XOSC1 unconnected. Care must be taken that the voltage amplitude of the 32.768-kHz oscillator is less than V_{BAT} , otherwise, the Hibernation module may draw power from the oscillator and not V_{BAT} during hibernation. See [Figure 6-2](#) and [Figure 6-3](#).

Alternatively, a low frequency oscillator source (HIB LFIOSC) present in the Hibernation module can be a clock source. (The frequency can have wide variations; see the HIB clock specifications in the device-specific data sheet for more details.) The intent of this source is to provide an internal low-power clock source to enable the use of the asynchronous pin wakes and memory storage without the requirement of an external crystal. To enable the HIB LFIOSC to be the clock source for the Hibernation module, both the OSCSEL bit and the CLK32EN bit in the Hibernation Control (HIBCTL) register must be set.

NOTE: The HIB low-frequency oscillator (HIB LFIOSC) has a wide frequency variation, therefore the RTC is not accurate when using this clock source. It is not recommended to use the HIB LFIOSC as an RTC clock source.

The Hibernation module is enabled by setting the CLK32EN bit of the HIBCTL register. The CLK32EN bit must be set before accessing any other Hibernation module register. The type of clock source used for the HIB module is selected by setting the OSCSEL and OSCBYP bit of the HIBCTL register. If the internal low frequency precision oscillator is used as the clock source, the OSCSEL bit should be set to a 1 at the same time the CLK32EN bit is set. If a crystal is used for the clock source, the software must leave a delay of $t_{\text{HIBOSC_START}}$ after writing to the CLK32EN bit and before any other accesses to the Hibernation module registers. The delay allows the crystal to power up and stabilize. If an external oscillator is used for the clock source, no delay is needed. When using an external clock source, the OSCBYP bit in the HIBCTL register should be set. When using a crystal clock source, the GNDX pin should be connected to digital ground along with the crystal load capacitors, as shown in Figure 6-2. When using an external clock source, the GNDX pin should be connected to digital ground.

NOTE: In the following figures, the parameters R_{BAT} and C_{BAT} have recommended values of $51\Omega \pm 5\%$ and $0.1\mu\text{F} \pm 5\%$, respectively. See the HIB electrical specifications in the device-specific data sheet for more information.

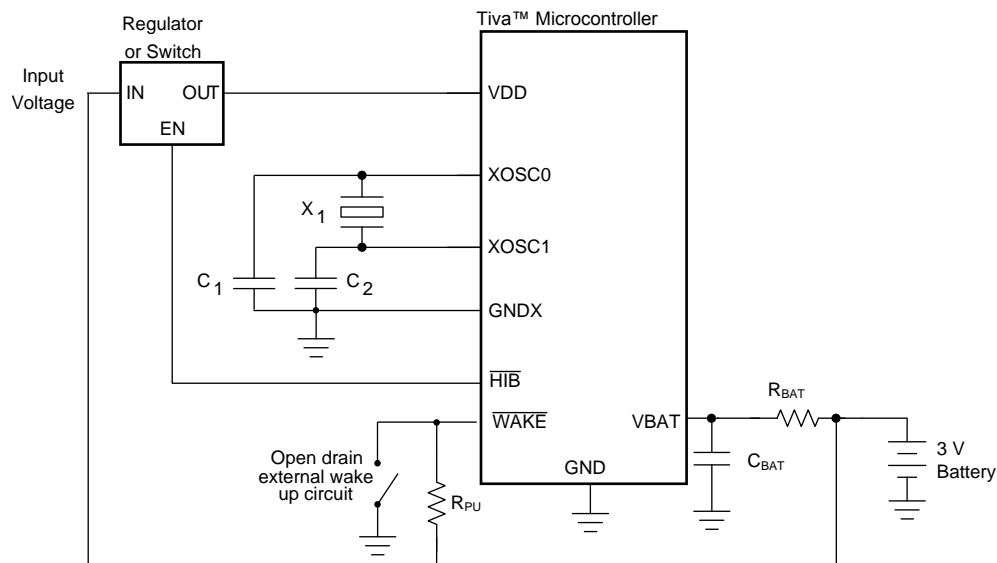


Figure 6-2. Using a Crystal as the Hibernation Clock Source with a Single Battery Source

NOTE: Some devices may not supply the GNDX signal. If GNDX is absent, the crystal load capacitors can be tied to GND externally. See for pins specific to your device.

X_1 = Crystal frequency is f_{XOSC_XTAL} .

$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

R_{PU} = Pullup resistor is 200 k Ω

R_{BAT} = 51 Ω $\pm 5\%$

C_{BAT} = 0.1 μF $\pm 20\%$

See the HIB electrical specifications in the device-specific data sheet for specific parameter values.

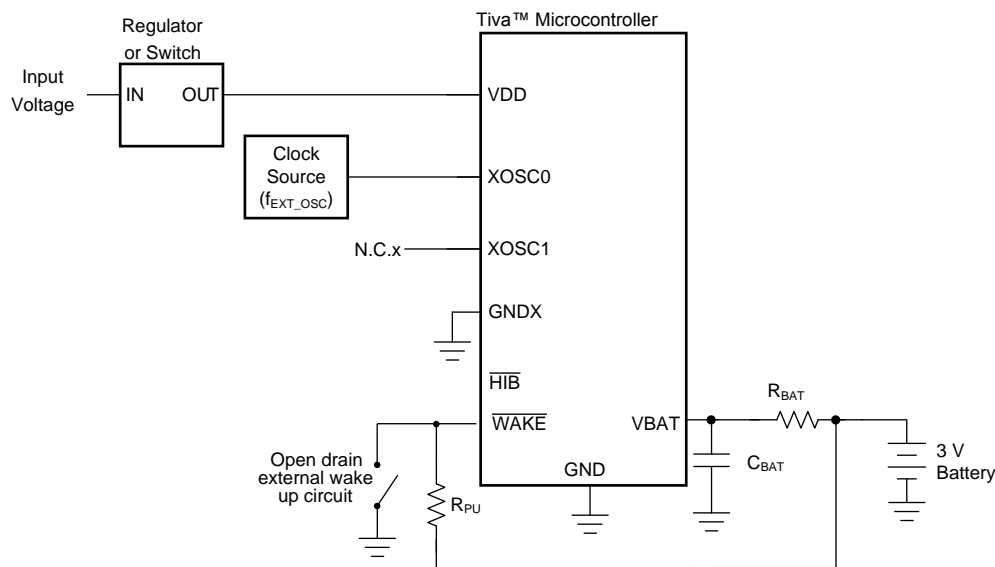


Figure 6-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode

NOTE: Some devices may not supply a GNDX signal. See for pins specific to your device.

R_{PU} = Pullup resistor is 1 M Ω

R_{BAT} = 51 Ω $\pm 5\%$

C_{BAT} = 0.1 μF $\pm 20\%$

6.3.2.1 Hibernate Clock Output RTCOSC

The clock source that is configured as the HIB clock has the option of becoming an internal output, RTCOSC, and being selected as the clock source for the system clock. To enable RTCOSC as a system clock source, the SYSCLKEN bit must be set in the Hibernate Clock Control (HIBCC) register.

6.3.3 System Implementation

Several different system configurations are possible when using the Hibernation module:

- Using a single battery source, where the battery provides both V_{DD} and V_{BAT} , as shown in [Figure 6-2](#).
- Using the VDD3ON mode, where V_{DD} continues to be powered in hibernation, allowing the GPIO pins to retain their states, as shown in [Figure 6-3](#). In this mode, V_{DDC} is powered off internally. In VDD3ON mode, the RETCLR bit in the HIBCTL register must be set so that after power is reapplied, GPIO retention is held until software clears the bit. GPIO retention is released when software writes a 0 to the RETCLR bit.

- Using separate sources for V_{DD} and V_{BAT} . In this mode, additional circuitry is required for system start-up without a battery or with a depleted battery.
- Using a regulator to provide both V_{DD} and V_{BAT} with a switch enabled by HIB to remove V_{DD} during hibernation as shown in Figure 6-4.

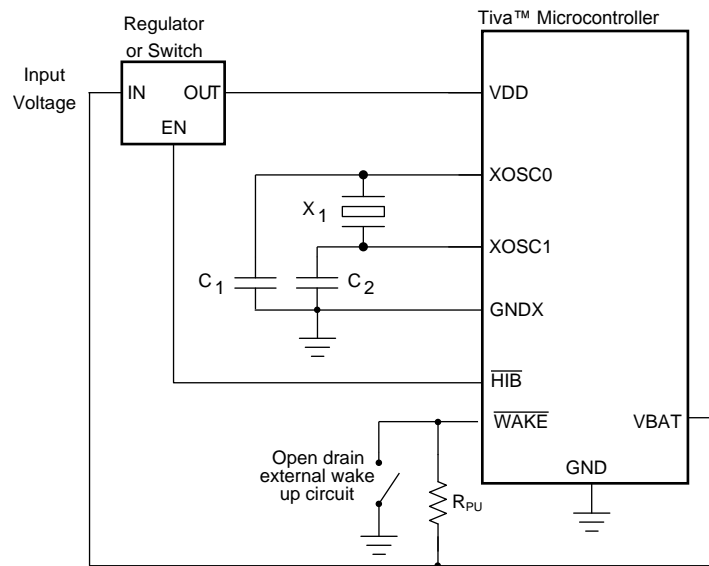


Figure 6-4. Using a Regulator for Both V_{DD} and V_{BAT}

NOTE: Some devices may not supply a GNDX signal. See for pins specific to your device.

Adding external capacitance to the V_{BAT} supply reduces the accuracy of the low-battery measurement and should be avoided if possible. The diagrams referenced in this section only show the connection to the Hibernation pins and not to the full system.

If the application does not require the use of the Hibernation module, see . In this situation, the HIB bit in the Hibernation Run Mode Clock Gating Control (RCGCHIB) register must be cleared, disabling the system clock to the Hibernation module and Hibernation module registers are not accessible.

6.3.4 Battery Management

NOTE: System-level factors may affect the accuracy of the low-battery detect circuit. The designer should consider battery type, discharge characteristics, and a test load during battery voltage measurements.

The Hibernation module can be independently powered by a battery or an auxiliary power source using the VBAT pin. The module can monitor the voltage level of the battery and detect when the voltage drops below V_{LOWBAT} . The voltage threshold can be between 1.9 V and 2.5 V and is configured using the VBATSEL field in the HIBCTL register. The module can also be configured so that it does not go into Hibernation mode if the battery voltage drops below this threshold. In addition, battery voltage is monitored while in hibernation, and the microcontroller can be configured to wake from hibernation if the battery voltage goes below the threshold using the BATWKEN bit in the HIBCTL register.

The Hibernation module is designed to detect a low-battery condition and set the LOWBAT bit of the Hibernation Raw Interrupt Status (HIBRIS) register when this occurs. If the VABORT bit in the HIBCTL register is also set, then the module is prevented from entering Hibernation mode when a low-battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see Section 6.3.13).

6.3.5 Real-Time Clock

The RTC module is designed to keep wall time. The RTC can operate in seconds counter mode or calendar mode. A 32.768 kHz clock source along with a 15-bit predivider reduces the clock to 1 Hz. The 1-Hz clock is used to increment the 32-bit counter and keep track of seconds. In calendar mode, registers are provided which support the tracking of date, month, year and day-of-week. A match register can be configured to interrupt or wake the system from hibernate. In addition, a software trim register is implemented to allow the user to compensate for oscillator inaccuracies using software.

6.3.5.1 RTC Counter - Seconds/Subseconds Mode

The clock signal to the RTC is provided by either of the 32.768-kHz clock sources available to the Hibernation module. The Hibernation RTC Counter (HIBRTCC) register displays the seconds value. The Hibernation RTC Sub Seconds register (HIBRTCSS) is provided for additional time resolution of an application requiring less than one-second divisions.

The RTC is enabled by setting the RTCEN bit of the HIBCTL register. The RTCEN bit is also used along with the CALEN bit in the Hibernation Calendar Control (HIBCALCTL) register to enable the calendar. Thus, if the calendar is enabled, the RTC registers, HIBRTCC, HIBRTCSS, HIBRTCM0 and HIBRTCLD, cannot be used. The RTC counter and subseconds counters begin counting immediately when RTCEN is set. Both counters count up. The RTC continues counting as long as the RTC is enabled and a valid V_{BAT} is present, regardless of whether V_{DD} is present or if the device is in hibernation.

The HIBRTCC register is set by writing the Hibernation RTC Load (HIBRTCLD) register. A write to the HIBRTCLD register clears the 15-bit subseconds counter field, RTCSSC, in the HIBRTCSS register. To ensure a valid read of the RTC value, the HIBRTCC register should be read first, followed by a read of the RTCSSC field in the HIBRTCSS register and then a re-read of the HIBRTCC register. If the two values for the HIBRTCC are equal, the read is valid. By following this procedure, errors in the application caused by the HIBRTCC register rolling over by a count of 1 during a read of the RTCSSC field are prevented. The RTC can be configured to generate an alarm by setting the RTCAL0 bit in the HIBIM register. When an RTC match occurs, an interrupt is generated and displayed in the HIBRIS register. Refer to [Section 6.3.5.2](#) for more information.

If the RTC is enabled, only a cold POR, where both V_{BAT} and V_{DD} are removed, resets the RTC registers. If any other reset occurs while the RTC is enabled, such as an external RST assertion or BOR reset, the RTC is not reset. The RTC registers can be reset under any type of system reset as long as the RTC, external wake pins and tamper pins are not enabled.

A buffered version of the 32.768-kHz signal Hibernate clock source is available on the RTCCLK signal output, which is muxed with a GPIO pin. The RTCCLK signal can be the external 32.786-kHz clock source or the HIB LFIOSC depending on the value of the OSCSEL bit in the HIBCTL register. See or pin mux information and [Chapter 17](#) for additional details on initialization and configuration of this signal. The pin does not output RTCCLK when Hibernate mode is active or before the RTCCLK GPIO digital function has been selected through the GPIO Digital Enable (GPIODEN) register in the GPIO module. This includes selecting the RTCCLK signal as an output source in the GPIO Port Control (GPIOPCTL) register and setting the SYSCLKEN bit within the Hibernate Clock Control (HIBCC) register.

NOTE: The HIB low-frequency oscillator (HIB LFIOSC) has a wide frequency variation, therefore the RTC is not accurate when using this clock source. In addition, the RTCCLK signal may not meet the specification shown in .

6.3.5.2 RTC Match - Seconds/Subseconds Mode

The Hibernation module includes a 32-bit match register, HIBRTCM0, which is compared to the value of the RTC 32-bit counter, HIBRTCC. The match functionality also extends to the subseconds counter. The 15-bit field (RTCSSM) in the HIBRTCSS register is compared to the value of the 15-bit subseconds counter. When a match occurs, the RTCALTO bit is set in the HIBRIS register. For applications using Hibernate mode, the processor can be programmed to wake from Hibernate mode by setting the RTCWEN bit in the HIBCTL register. The processor can also be programmed to generate an interrupt to the interrupt controller by setting the RTCALTO bit in the HIBIM register.

The match interrupt generation takes priority over an interrupt clear. Therefore, writes to the RTCALT0 bit in the Hibernation Interrupt Clear (HIBIC) register do not clear the RTCALT0 bit if the HIBRTCC value and the HIBRTCM0 value are equal. There are several methodologies to avoid this occurrence, such as writing a new value to the HIBRTCLD register prior to writing the HIBIC to clear the RTCALT0. Another example, would be to disable the RTC and re-enable the RTC by clearing and setting the RTCEN bit in the HIBCTL register.

NOTE: A Hibernate request made while a match event is valid causes the module to immediately wake up. This occurs when the RTCWEN bit is set and the RTCALT0 bit in the HIBRIS register is set at the same time the HIBREQ bit in the HIBCTL register is written to a 1. This can be avoided by clearing the RTCAL0 bit in the HIBRIS register by writing a 1 to the corresponding bit in the HIBIC register before setting the HIBREQ bit. Another example would be to disable the RTC and re-enable the RTC by clearing and setting the RTCEN bit in the HIBCTL register.

6.3.5.3 RTC Calendar

The RTC Calendar function is selected by setting the CALEN bit in the HIB Calendar Control (HIBCALCTL) register. In this mode, six 32-bit registers provide the read (HIBCAL0/1), match (HIBCALM0/1), and load (HIBCALLD0/1) interface. The standard RTC registers: HIBRTCC, HIBRTCLD, HIBRTCSS, and HIBRTCM0 are disabled when the calendar function is enabled and read back as all 0s in this mode. In addition, writes have no effect on these registers when the calendar function is enabled.

The Hibernation Calendar n (HIBCALn), Hibernation Calendar Match (HIBCALMn) and Hibernation Calendar Load (HIBCALLDn) register fields are written or stored in hexadecimal. When reading the Hibernation Calendar n (HIBCALn) registers, the status of the VALID bit in the HIBCAL0/1 register must be checked to ensure the registers are in sync before reading.

The calendar function will keep track of the following:

- Seconds (0-59 seconds)
- Minutes (0-59 minutes)
- Hours (0-23 or 0-11 hours with an AM/PM option)
- Day of the week (0-6)
- Day of the month (1-31 days)
- Month (1-12 months)
- Year (00-99 years)

The hours may be reported with AM/PM or 24-hour based on the CAL24 bit in the HIBCALCTL register. The leap year compensation is handled within the calendar function. The number of days in February are adjusted to 29 whenever the year is divisible by four.

6.3.5.3.1 RTC Calendar Match

The HIB Calendar Match function can be used to generate an interrupt on a match of seconds, minutes, hours, and day of month. The day of the week, year and month are not included in the match function. To ignore a match function for the hours, minutes, or seconds, set each of the upper two bits to 1 in the respective fields of the HIBCALMn register. To ignore the day of the month, set the DOM field to all zeros in the HIBCALM1 register. If a match occurs in any field, the RTCALT0 bit is set in the HIBRIS register.

6.3.5.4 RTC Trim

The RTC counting rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, HIBRTCT. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds in RTC counter mode, when bits [5:0] in the HIBRTCC register change from 0x00 to 0x01, to divide the input clock. Trim is applied every 60 seconds in calendar mode. This configuration allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate and down from 0x7FFF in order to speed up the RTC rate.

Care must be taken when using trim values that are near to the subseconds match value in the HIBRTCSS register. It is possible when using trim values above 0x7FFF to receive two match interrupts for the same counter value. In addition, it is possible when using trim values below 0x7FFF to miss a match interrupt.

In the case of a trim value above 0x7FFF, when the RTCSSC value in the HIBRTCSS register reaches 0x7FFF, the RTCC value increments from 0x0 to 0x1 while the RTCSSC value is decreased by the trim amount. The RTCSSC value is counted up again to 0x7FFF before rolling over to 0x0 to begin counting up again. If the match value is within this range, the match interrupt is triggered twice. For example, as shown in [Figure 6-5](#), if the match interrupt was configured with RTCM0 = 0x1 and RTCSSM = 0x7FFD, two interrupts would be triggered.

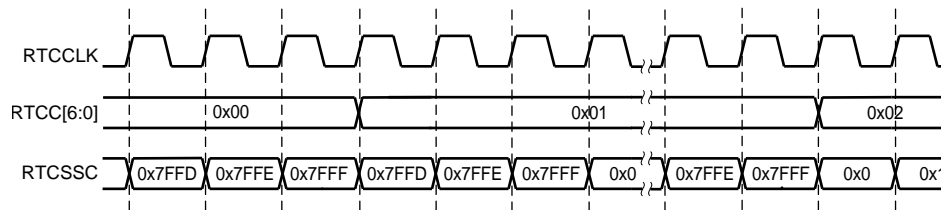


Figure 6-5. Counter Behavior with a TRIM Value of 0x8002

In the case of a trim value below 0x7FFF, the RTCSSC value is advanced from 0x7FFF to the trim value while the RTCC value is incremented from 0x0 to 0x1. If the match value is within that range, the match interrupt is not triggered. For example, as shown in [Figure 6-6](#), if the match interrupt was configured with RTCM0 = 0x1 and RTCSSM = 0x2, an interrupt would never be triggered.

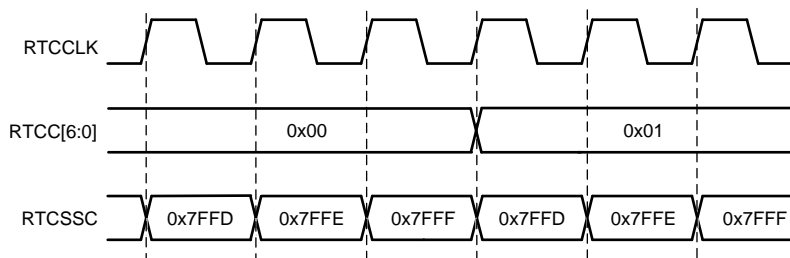


Figure 6-6. Counter Behavior with a TRIM Value of 0x7FFC

6.3.6 Tamper

The Tamper module provides a user with mechanisms to detect, respond to, and log system tampering events. The Tamper module is designed to be low power and operate either from a battery or the MCU I/O voltage supply. This module is a submodule of the Hibernate module.

6.3.6.1 Tamper Block Diagram

[Figure 6-7](#) shows the Tamper block diagram.

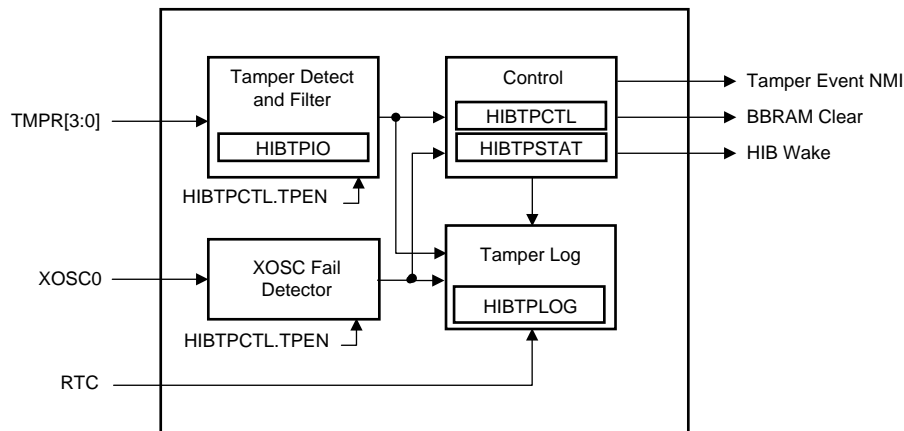


Figure 6-7. Tamper Block Diagram

6.3.6.2 Functional Description

The Tamper module provides mechanisms to detect, respond, and log system tamper events. A tamper event is detected by state transitions on up to four GPIOs. The module may respond to a tamper event by clearing all or part of the hibernate module memory, generating a tamper event signal to the System Control module. The event will also be logged with a RTC timestamp to allow for tamper investigation.

6.3.6.2.1 Tamper Detection

Qualified tamper events are detected through an XOSC_n pin failure or through tamper I/O level matches which pass through a glitch filter. Tamper I/O pad events are detected by comparing the level on a tamper I/O pad with an expected value. The tamper I/O is sampled using the hibernate clock source and when the glitch filtering is enabled, must be stable for approximately 100 ms. This provides debounce filtering of a breakaway switch as a results of a drop impact. The tamper module contains one long glitch filter and one short glitch filter which uses an OR of the inputs as shown in [Figure 6-8](#). This implies if two Tamper inputs are asserted and one deasserts, the glitch filter runs to time-out or until the second Tamper input is deasserted. The glitch filter or tamper logging logic does not trigger again if the tamper event match continues. The glitch filter resets on the deassertion of the tamper conditions or when a qualified tamper event is logged.

If the XOSC_n pins are enabled for use with the Hibernation module and subsequently fail, a tamper event is detected and is indicated by the STATE field in the HIB Tamper Status (HIBTPSTAT) register. In addition, the XOSCST and XOSCFAIL bits can be read for further details on the external oscillator source state.

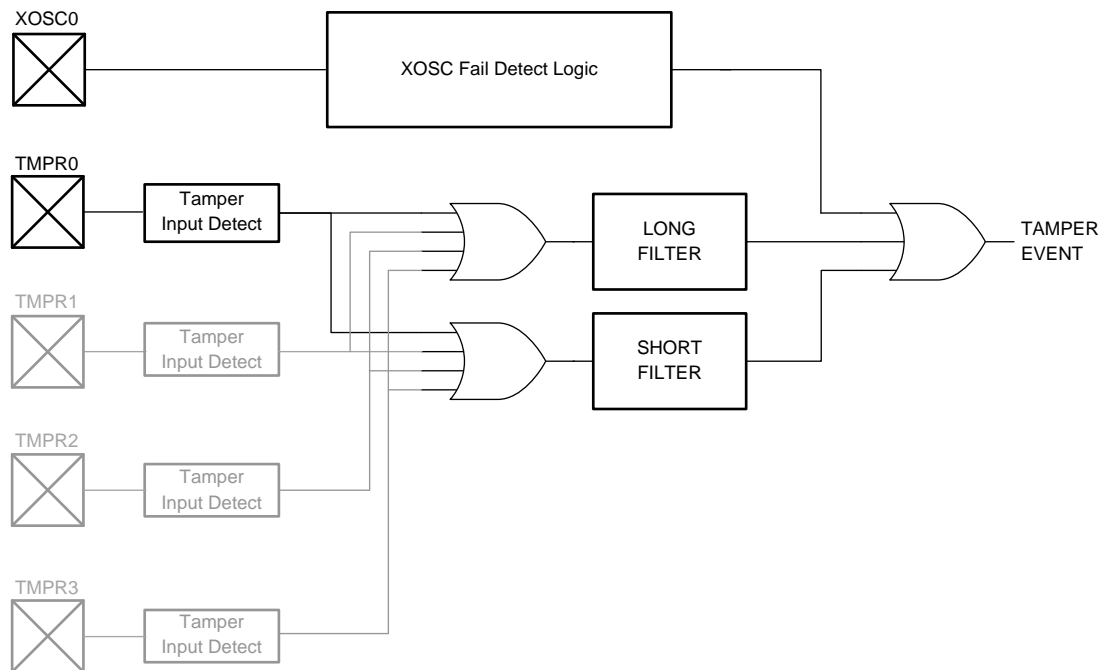


Figure 6-8. Tamper Pad With Glitch Filtering

6.3.6.2.2 Tamper Event Responses

There are many responses to a tamper event including clearing some or all of Hibernate memory and generating a tamper signal to the System Control Module. The descriptions of the possible event responses follows.

- **Tamper Register Status**

The tamper status is indicated by the STATE bit field of the HIB Tamper Status (HIBTPSTAT) register. The register bits are reset to 0x0 on cold POR. When the tamper I/O is enabled/configured, the STATE field shows 0x1. The STATE field is set to 0x2 when a tamper event is detected. The software may reset the trigger source and the STATE field by writing to the TPCLR bit in the HIBTPCTL register.

- **System Event Response**

When a tamper event is detected, an NMI is generated. The NMI handler is responsible for performing any other system responses, including a simulate POR. If the tamper event was an XOSC fail condition, the part switches to the HIB LFIOSC. Once XOSC is stable, the XOSC may be enabled as the clock source once again.

- **Hibernate Memory Clearing**

On a tamper event, software has the option to clear all, the upper half, lower half, or none of the Hibernate memory. The feature is controlled through the MEMCLR field of the HIBTPCTL register.

- **Wake from Hibernate**

A tamper event will assert a wake event to the MCU if the WAKE bit in the HIBTPCTL register is set.

6.3.6.2.3 Tamper Event Logging

Up to four tamper events are stored in HIB Tamper Log n (HIBTPLOGn) registers within the Hibernate module. When a tamper event occurs the following status is logged:

- The RTC seconds or calendar values of year, minutes, day of month, hours and seconds in the HIBTPLOG0, HIBTPLOG2, HIBTPLOG4, and HIBTPLOG6 registers

NOTE: 24-hour mode must be used if RTC calendar mode is enabled. This mode is selected by setting the CAL24 bit in HIB Calendar Control (HIBCALCTL) register.

- The tamper status of the TMPRn pins and the XOSCn pins in the HIBTPLOG1, HIBTPLOG3, and HIBTPLOG5 registers. The HIBTPLOG7 register captures the OR of all events occurring after the 3rd event is logged in the HIBTPLOG5 register.

On the assertion of a qualified tamper event (rising edge) on any of the TMPRn pins or an XOSC failure signal, the current status of all tamper inputs are logged in the HIBTPLOGn register.

6.3.6.2.4 Clearing a Tamper Event

After a tamper event, the HIB Tamper Log (HIBTPLOGn) registers and the NMI to the processor may be cleared by writing a 1 to the TPCLR bit in the HIBTPCTL register. This clear status is reflected by the STATE bit in the HIBSTPSTAT register changing from 0x2 back to a 0x1. If the source of the tamper event comes from an XOSC failure, the clearing of a tamper event is delayed while the clock is switched to LFIOSC. The NMI interrupt handler may access the module immediately, but should read the HIBTPLOGn registers before issuing a tamper clear in the HIBTPCTL register.

NOTE: The HIBTPLOG7 register is sticky and is only cleared by a Hibernate module reset.

6.3.6.2.5 Tamper I/O Control

Up to four tamper I/Os are available. These signals are individually enabled and the detection level can be configured per pin. Enabling the tamper IO will override all settings made in the GPIO module. Each tamper IO has a weak pullup.

6.3.6.2.6 Tamper Clocking

The Hibernate clock is the clock source for the Tamper module. When an external oscillator is used and tamper is enabled, the external oscillator is monitored by the Tamper module. If the external oscillator stops for any reason, the XOSCFAIL bit is set in the HIBTPSTAT register and the Hibernate clock source is switched to the HIB LFIOSC immediately. When the XOSCST bit in the HIBTPSTAT register is 0, indicating the external oscillator is active, a 1 can be written to the XOSCFAIL bit to clear it and re-enable the external 32.768-kHz oscillator.

NOTE: Because the HIB LFIOSC has a wide frequency variation, it should not be configured as the HIB clock source when accurate monitoring of the tamper logs are important.

6.3.6.2.7 Tamper Resets

The Tamper module uses the resets from the Hibernate module.

NOTE: The Hibernation module registers are reset under two conditions:

1. Any type of system reset (if the RTCEN and the PINWEN bits in the HIBCTL register are clear and the TPEN bit in the HIBTPCTL register is clear).
2. A cold POR occurs when both the V_{DD} and V_{BAT} supplies are removed.

Any other reset condition is ignored by the Hibernation module.

6.3.7 Battery-Backed Memory

The Hibernation module contains 16 32-bit words of memory that are powered from the battery or an auxiliary power supply and therefore retained during hibernation. The processor software can save state information in this memory prior to hibernation and recover the state upon waking. To access the upper eight words of memory, the processor must be in privilege mode. Refer to [Section 1.4.1](#) for more information about processor privilege mode. The battery-backed memory can be accessed through the HIBDATA registers. If both V_{DD} and V_{BAT} are removed, the contents of the HIBDATA registers are not retained.

6.3.8 Power Control Using HIB

NOTE: The Hibernation Module requires special system implementation considerations when using HIB to control power, as it is intended to power-down all other sections of the microcontroller. All system signals and power supplies that connect to the chip must be driven to 0 V or powered down with the same regulator controlled by HIB.

The Hibernation module controls power to the microcontroller through the use of the HIB pin which is intended to be connected to the enable signal of the external regulators providing 3.3 V to the microcontroller and other circuits. When the HIB signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the microcontroller and any parts of the system that are powered by the regulator. The Hibernation module remains powered from the V_{BAT} supply until a Wake event. Power to the microcontroller is restored by deasserting the HIB signal, which causes the external regulator to turn power back on to the chip.

6.3.9 Power Control Using VDD3ON Mode

The Hibernation module may also be configured to cut power to all internal modules during Hibernate mode. While in this state, if VDD3ON is set in the HIBCTL register, all pins are held in the state they were in prior to entering hibernation. For example, inputs remain inputs; outputs driven high remain driven high, and so on. There are important procedural and functional items to note when in VDD3ON mode:

- JTAG Ports C[0] - C[3] do not retain their state in Hibernate VDD3ON mode.
- If GPIO pins K[7:4] are not used as a wake source, they should not be left floating. An internal pullup resistor may be configured by the application before entering Hibernate mode by programming the GPIO Pull-Up Select (GPIOPUR) register in the GPIO module.
- In the VDD3ON mode, the regulator should maintain 3.3 V power to the microcontroller during Hibernate. GPIO retention is disabled when the RETCLR bit is cleared in the HIBCTL register.
- When entering hibernation in VDD3ON mode, the supply rails to the Ethernet resistors R1, R2, R3, R4 found in [Figure 15-15](#) must be switched off.

6.3.10 Initiating Hibernate

Hibernate mode is initiated when the HIBREQ bit of the HIBCTL register is set. If a wake-up condition has not been configured using the PINWEN or RTCWEN bits in the HIBCTL register, the hibernation request is ignored. In addition, if the battery voltage is below the threshold voltage defined by the VBATSEL field in the HIBCTL register, the hibernation request is ignored.

6.3.11 Waking from Hibernate

The Hibernation module can be configured to wake from Hibernate mode if any of the following are enabled:

- External WAKE
- External RST
- GPIO K[7:4]
- Tamper TMPR[3:0]
- Tamper XOSC failure

The Hibernation module can also be configured to wake from hibernate when the following events occur:

- RTC match wake event
- Low Battery wake event

The external WAKE pin is enabled by setting the PINWEN bit in the HIBCTL register. The external WAKE pin can generate an interrupt by programming the EXTWEN bit in the Hibernation Interrupt Mask (HIBIM) register.

NOTE: If an external WAKE signal is asserted, the application is responsible for clearing the signal source once the EXTWEN bit has been registered in the Hibernation Raw Interrupt Status (HIBRIS) register.

To use the RST pin as a wake source, the WURSTEN bit must be set in the Hibernate I/O Configuration (HIBIO) register and the WUUNLK bit must be set in the same register.

To enable any of the assigned GPIO pins as a wake source, the WUUNLK bit must be set in the HIBIO register and the wake configuration must be programmed through the GPIOWAKEPEN and GPIOWAKELVL registers in the GPIO module. See [Chapter 17](#) for more information on programming the GPIOs.

NOTE: The RST pin and GPIO wake sources are cleared by a write to either or both the RSTWK and PADIOWK bits. This clears the source of interrupts for RSTWK, PADIOWK and the GPIOWAKESTAT register.

TMPR[3:0] are enabled by setting the appropriate ENn bits the Tamper IO Control and Status (HIBTPIO) register. The HIBTPIO register overrides the GPIO port configuration registers. By setting the WAKE bit in the Tamper Control (HIBTPCTL) register, a tamper event can cause a wake from Hibernate. If a tamper event occurs, the time of the event and the status of the tamper pins are logged in the Tamper Log (HIBTPLOG) register.

By setting the RTCWEN bit in the HIBCTL register a wake from hibernate can occur when the value of the HIBRTCC register matches the value of the HIBRTCM0 register and the value of the RTCSSC field matches the RTCSSM field in the HIBRTCSS register.

To allow a wake from Hibernate on a low battery event, the BATWKEN bit in the HIBCTL register must be set. In this configuration, the battery voltage is checked every 512 seconds while in hibernation. If the voltage is below the level specified by the VBATSEL field, the LOWBAT interrupt is set in the HIBRIS register.

Upon external wake-up, external reset, tamper event, or RTC match, the Hibernation module delays coming out of hibernation until V_{DD} is above the minimum specified voltage, see .

When the Hibernation module wakes, the microcontroller performs a normal power-on reset. The normal power-on reset does not reset the Hibernation module or Tamper module, but does reset the rest of the microcontroller. Software can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see [Section 6.3.13](#)) and by looking for state data in the battery-backed memory (see [Section 6.3.7](#)).

6.3.12 Arbitrary Power Removal

The microcontroller goes into hibernation if V_{DD} is arbitrarily removed when the CLK32EN bit is set and any of the following bits are set:

- TPEN bit in the HIBTPCTL register
- PINWEN bit in the HIBCTL register
- RTCEN bit in the HIBCTL register

The microcontroller wakes from hibernation when power is reapplied.

If the CLK32EN bit is set but the TPEN, PINWEN, and RTCEN bits are all clear, the microcontroller still goes into hibernation if power is removed; however, when V_{DD} is reapplied, the MCU executes a cold POR and the Hibernation module is reset. If the CLK32EN bit is not set and V_{DD} is arbitrarily removed, the part is simply powered off and executes a cold POR when power is reapplied.

If V_{DD} is arbitrarily removed while a Flash memory or HIBDATA register write operation is in progress, the write operation must be retried after V_{DD} is reapplied.

6.3.13 Interrupts and Status

The Hibernation module can generate interrupts when the following conditions occur:

- Assertion of WAKE pin
- RTC match
- Low battery detected
- Write complete/capable
- Assertion of an external RESET pin
- Assertion of an external wake-enabled GPIO pin (port K[7:4])

All of the interrupts except for the tamper signals are ORed together before being sent to the interrupt controller, so the Hibernation module can only generate a single interrupt request to the controller at any given time. The software interrupt handler can service multiple interrupt events by reading the Hibernation Masked Interrupt Status (HIBMIS) register. Software can also read the status of the Hibernation module at any time by reading the HIBRIS register which shows all of the pending events. This register can be used after waking from hibernation to see if a wake condition was caused by one of the events above or by a power loss.

The WAKE pin can generate interrupts in Run, Sleep and Deep Sleep Mode. The events that can trigger an interrupt are configured by setting the appropriate bits in the Hibernation Interrupt Mask (HIBIM) register. Pending interrupts can be cleared by writing the corresponding bit in the Hibernation Interrupt Clear (HIBIC) register.

6.4 Initialization and Configuration

The Hibernation module has several different configurations. The following sections show the recommended programming sequence for various scenarios. Because the Hibernation module runs at a low frequency and is asynchronous to the rest of the microcontroller, which is run off the system clock, software must allow a delay of $t_{HIB_REG_ACCESS}$ after writes to registers (see [Section 6.3.1](#)). The WC interrupt in the HIBMIS register can be used to notify the application when the Hibernation modules registers can be accessed.

6.4.1 Initialization

The Hibernation module comes out of reset with the system clock enabled to the module, but if the system clock to the module has been disabled, then it must be re-enabled, even if the RTC feature is not used. See [Section 4.2.90](#).

If a 32.768-kHz crystal is used as the Hibernation module clock source, perform the following steps:

1. Write 0x0000.0010 to the HIBIM register to enable the WC interrupt.
2. Write 0x40 to the HIBCTL register at offset 0x10 to enable the oscillator input.
3. Wait until the WC interrupt in the HIBMIS register has been triggered before performing any other operations with the Hibernation module.

If a 32.768-kHz single-ended oscillator is used as the Hibernation module clock source, then perform the following steps:

1. Write 0x0000.0010 to the HIBIM register to enable the WC interrupt.
2. Write 0x0001.0040 to the HIBCTL register at offset 0x10 to enable the oscillator input and bypass the on-chip oscillator.
3. Wait until the WC interrupt in the HIBMIS register has been triggered before performing any other operations with the Hibernation module.

If the internal low frequency oscillator is used as the Hibernation module clock source, then perform the following steps:

1. Write 0x0000.0010 to the HIBIM register to enable the WC interrupt.
2. Write 0x0008.0040 to the HIBCTL register at offset 0x10 to enable the internal low frequency oscillator.
3. Wait until the WC interrupt in the HIBMIS register has been triggered before performing any other operations with the Hibernation module.

The above steps are only necessary when the entire system is initialized for the first time. If the microcontroller has been in hibernation, then the Hibernation module has already been powered up and the above steps are not necessary. The software can detect that the Hibernation module and clock are already powered by examining the CLK32EN bit of the HIBCTL register.

6.4.2 RTC Match Functionality (No Hibernation)

Use the following steps to implement the RTC match functionality of the Hibernation module:

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write the required RTC match value to the HIBRTCM0 register at offset 0x004 and the RTCSSM field in the HIBRTCSS register at offset 0x028.
3. Write the required RTC load value to the HIBRTCLD register at offset 0x00C.
4. Set the required RTC match interrupt mask in the RTCALT0 in the HIBIM register at offset 0x014.
5. Write 0x0000.0041 to the HIBCTL register at offset 0x010 to enable the RTC to begin counting.

6.4.3 RTC Match and Wake From Hibernation

Use the following steps to implement the RTC match and wake-up functionality of the Hibernation module:

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write the required RTC match value to the HIBRTCM0 register at offset 0x004 and the RTCSSM field in the HIBRTCSS register at offset 0x028.
3. Write the required RTC load value to the HIBRTCLD register at offset 0x00C. This write causes the 15-bit subseconds counter to be cleared.
4. Write any data to be retained during hibernation to the HIBDATA register at offsets 0x030-0x06F.
5. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004B to the HIBCTL register at offset 0x010.

6.4.4 External Wake From Hibernation

Use the following steps to implement the Hibernation module with the external WAKE pin as the wake-up source for the microcontroller:

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write any data to be retained during hibernation to the HIBDATA register at offsets 0x030-0x06F.
3. Enable the external wake and start the hibernation sequence by writing 0x0000.0052 to the HIBCTL register at offset 0x010.

Use the following steps to program the external RESET pin as the wake source for the microcontroller:

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write any data to be retained during hibernation to the HIBDATA register at offsets 0x030-0x06F.
3. Enable the external RESET pin as a wake source by writing a 0x0000.0011 to the HIBIO register at offset 0x02C.
4. When the IOWRC bit in the HIBIO register is read as 1, clear the WUUNLK bit in the HIBIO register to lock the current pad configuration so that any other writes to the WURSTEN bit in the HIBIO register will be ignored.
5. The hibernation sequence may be initiated by writing 0x4000.0152 to the HIBCTL register. Note that when using RESET, the user must enable VDD3ON mode and set the RETCLR bit in the HIBCTL register.

Use the following steps to program GPIO port K pins K[7:4] as the wake source for the microcontroller:

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write any data to be retained during hibernation to the HIBDATA register at offsets 0x030-0x06F.
3. Configure the GPIOWAKEPEN and GPIOWAKELVL registers at offsets 0x540 and 0x544 in the GPIO module. Enable the I/O wake pad configuration by writing 0x0000.0001 to the HIBIO register at offset 0x010.
4. When the IOWRC bit in the HIBIO register is read as 1, write 0x0000.0000 to the HIBIO register to lock the current pad configuration so that any other writes to the GPIOWAKEPEN and GPIOWAKELVL register will be ignored.
5. Clear any pending interrupts by writing a 1 to the PADIOWK bit in the HIBIC register.
6. The hibernation sequence may be initiated by writing 0x4000.0152 to the HIBCTL register. Note for Port M external wake, the user must enable VDD3ON mode and set the RETCLR bit in the HIBCTL register.

6.4.5 RTC or External Wake From Hibernation

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write the required RTC match value to the HIBRTCM0 register at offset 0x004 and the RTCSSM field in the HIBRTCSS register at offset 0x028.
3. Write the required RTC load value to the HIBRTCLD register at offset 0x00C. This write causes the 15-bit subseconds counter to be cleared.
4. Write any data to be retained during hibernation to the HIBDATA register at offsets 0x030-0x06F.
5. Set the RTC Match/External Wake-Up and start the hibernation sequence by writing 0x0000.005B to the HIBCTL register at offset 0x010.

6.4.6 Tamper Initialization

Use the following steps to configure the Tamper module to interrupt the processor when a Tmpr signal has triggered:

NOTE: Unlike other functions, the Tamper pins do not need to be configured for the GPIO in the GPIOAFSEL register. The Tamper IO Control and Status (HIBTPIO) register overrides configurations made to the GPIO module.

1. Write 0x0000.0041 to the HIBCTL register at offset 0x010 to enable the 32.768-kHz Hiberate oscillator and enable the RTC.
2. Enable the four Tamper I/O to trigger on the a high state on any of the pins by writing 0x0F0F.0F0F to the HIBTPIO register at offset 0x410.
3. Write 0x0000.0001 to the HIBTPCTL register to enable the tamper.

NOTE: When tamper is enabled, the following HIBCTL register bits are locked and cannot be modified:

- OSCSEL
 - OSCDRV
 - OSCBYP
 - VDD3ON
 - CLK32EN
 - RTCEN
-

6.5 HIB Registers

[Table 6-2](#) lists the memory-mapped registers for the HIB. All register offset addresses not listed in [Table 6-2](#) should be considered as RESERVED locations and the register contents should not be modified.

All addresses given are relative to the Hibernation Module base address at 0x400FC000 (ending address of 0x400FCFFF). The system clock to the Hibernation module must be enabled before the registers can be programmed (see [Section 4.2.90](#)). There must be a delay of 3 system clocks after the Hibernation module clock is enabled before any Hibernation module registers are accessed. In addition, the CLK32EN bit in the HIBCTL register must be set before accessing any other Hibernation module register.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored (see [Section 6.3.1](#)). The HIBIO register and bits RSTWK, PADIOWK, and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers and bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

NOTE: The Hibernation module registers are reset under two conditions:

1. Any type of system reset (if the RTCEN and the PINWEN bits in the HIBCTL register are clear and the TPEN bit in the HIBTPCTL register is clear).
2. A cold POR occurs when both the V_{DD} and V_{BAT} supplies are removed.

Any other reset condition is ignored by the Hibernation module.

The following registers can be accessed only through privileged mode (see [Chapter 4](#) for more details) :

- HIBTPCTL
- HIBPTSTAT
- HIBTPIO
- HIBTPLOG
- Upper eight words of memory (HIBDATA register 0x50 to 0x6F)

Table 6-2. HIB Registers

Offset	Acronym	Register Name	Section
0x0	HIBRTCC	Hibernation RTC Counter	Section 6.5.1
0x4	HIBRTCM0	Hibernation RTC Match 0	Section 6.5.2
0xC	HIBRTCLD	Hibernation RTC Load	Section 6.5.3
0x10	HIBCTL	Hibernation Control	Section 6.5.4
0x14	HIBIM	Hibernation Interrupt Mask	Section 6.5.5
0x18	HIBRIS	Hibernation Raw Interrupt Status	Section 6.5.6
0x1C	HIBMIS	Hibernation Masked Interrupt Status	Section 6.5.7
0x20	HIBIC	Hibernation Interrupt Clear	Section 6.5.8
0x24	HIBRTCT	Hibernation RTC Trim	Section 6.5.9
0x28	HIBRTCSS	Hibernation RTC Sub Seconds	Section 6.5.10
0x2C	HIBIO	Hibernation IO Configuration	Section 6.5.11
0x30 to 0x6F	HIBDATA	Hibernation Data	Section 6.5.12
0x300	HIBCALCTL	Hibernation Calendar Control	Section 6.5.13
0x310	HIBCAL0	Hibernation Calendar 0	Section 6.5.14
0x314	HIBCAL1	Hibernation Calendar 1	Section 6.5.15

Table 6-2. HIB Registers (continued)

Offset	Acronym	Register Name	Section
0x320	HIBCALLD0	Hibernation Calendar Load 0	Section 6.5.16
0x324	HIBCALLD1	Hibernation Calendar Load 1	Section 6.5.17
0x330	HIBCALM0	Hibernation Calendar Match 0	Section 6.5.18
0x334	HIBCALM1	Hibernation Calendar Match 1	Section 6.5.19
0x360	HIBLOCK	Hibernation Lock	Section 6.5.20
0x400	HIBTPCTL	HIB Tamper Control	Section 6.5.21
0x404	HIBTPSTAT	HIB Tamper Status	Section 6.5.22
0x410	HIBTPIO	HIB Tamper I/O Control	Section 6.5.23
0x4E0	HIBTPLOG0	HIB Tamper Log 0	Section 6.5.24
0x4E4	HIBTPLOG1	HIB Tamper Log 1	Section 6.5.25
0x4E8	HIBTPLOG2	HIB Tamper Log 2	Section 6.5.24
0x4EC	HIBTPLOG3	HIB Tamper Log 3	Section 6.5.25
0x4F0	HIBTPLOG4	HIB Tamper Log 4	Section 6.5.24
0x4F4	HIBTPLOG5	HIB Tamper Log 5	Section 6.5.25
0x4F8	HIBTPLOG6	HIB Tamper Log 6	Section 6.5.24
0x4FC	HIBTPLOG7	HIB Tamper Log 7	Section 6.5.25
0xFC0	HIBPP	Hibernation Peripheral Properties	Section 6.5.26
0xFC8	HIBCC	Hibernation Clock Control	Section 6.5.27

Complex bit access types are encoded to fit into small table cells. [Table 6-3](#) shows the codes that are used for access types in this section.

Table 6-3. HIB Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

6.5.1 HIBRTCC Register (Offset = 0x0) [reset = 0x0]

Hibernation RTC Counter (HIBRTCC)

This register is the current 32-bit value of the RTC counter.

The RTC counter consists of a 32-bit seconds counter and a 15-bit sub seconds counter. The RTC counters are reset by the Hibernation module reset. The RTC 32-bit seconds counter can be set by the user using the HIBRTCLD register. When the 32-bit seconds counter is set, the 15-bit sub second counter is cleared.

The RTC value can be read by first reading the HIBRTCC register, reading the RTCSSC field in the HIBRTCSS register, and then rereading the HIBRTCC register. If the two values for HIBRTCC are equal, the read is valid.

NOTE: There is a minimum system clock rate of three times the HIB clock rate to properly read the HIBRTCC register.

HIBRTCC is shown in [Figure 6-9](#) and described in [Table 6-4](#).

Return to [Summary Table](#).

Figure 6-9. HIBRTCC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCC																															
R-0x0																															

Table 6-4. HIBRTCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RTCC	R	0x0	RTC Counter A read returns the 32-bit counter value, which represents the seconds elapsed since the RTC was enabled. This register is read-only. To change the value, use the HIBRTCLD register.

6.5.2 HIBRTCM0 Register (Offset = 0x4) [reset = 0xFFFFFFFF]

Hibernation RTC Match 0 (HIBRTCM0)

This register is the 32-bit seconds match register for the RTC counter. The 15-bit sub second match value is stored in the reading the RTCSSC field in the HIBRTCSS register and can be used in conjunction with this register for a more precise time match.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

HIBRTCM0 is shown in [Figure 6-10](#) and described in [Table 6-5](#).

Return to [Summary Table](#).

Figure 6-10. HIBRTCM0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCM0																															
R/W-0xFFFFFFFF																															

Table 6-5. HIBRTCM0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RTCM0	R/W	0xFFFFFFFF	RTC Match 0 A write loads the value into the RTC match register. A read returns the current match value.

6.5.3 HIBRTCLD Register (Offset = 0xC) [reset = 0x0]

Hibernation RTC Load (HIBRTCLD)

This register is used to load a 32-bit value loaded into the RTC counter. The load occurs immediately upon this register being written. When this register is written, the 15-bit sub seconds counter is also cleared.

NOTE: This register is protected from errant code by using the HIBLOCK register. This register is write-only; any reads to this register read back as zeros.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

HIBRTCLD is shown in [Figure 6-11](#) and described in [Table 6-6](#).

Return to [Summary Table](#).

Figure 6-11. HIBRTCLD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCLD																															
W-0x0																															

Table 6-6. HIBRTCLD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RTCLD	W	0x0	RTC Load A write loads the current value into the RTC counter (RTCC). A read returns the 32-bit load value.

6.5.4 HIBCTL Register (Offset = 0x10) [reset = 0x80002000]

Hibernation Control (HIBCTL)

This register is the control register for the Hibernation module. This register must be written last before a hibernate event is issued. Writes to other registers after the HIBREQ bit is set are not guaranteed to complete before hibernation is entered.

NOTE: Writes to this register have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required synchronization has elapsed. While the WRC bit is clear, any attempts to write this register are ignored. Reads may occur at any time.

Note that once tamper is enabled, the following HIBCTL clock configuration bits and bus write stall bit are locked and cannot be modified:

- OSCSEL
- OSCDRV
- OSCBYP
- VDD3ON
- CLK32EN
- RTCEN

HIBCTL is shown in [Figure 6-12](#) and described in [Table 6-7](#).

Return to [Summary Table](#).

Figure 6-12. HIBCTL Register

31	30	29	28	27	26	25	24
WRC	RETCLR	RESERVED					
R-0x1	R/W-0x0	R-0x0					
23	22	21	20	19	18	17	16
RESERVED				OSCSEL	RESERVED	OSCDRV	OSCBYP
R-0x0				R/W-0x0	R-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED	VBATSEL		RESERVED		BATCHK	BATWKEN	VDD3ON
R-0x0	R/W-0x1		R-0x0		R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
VABORT	CLK32EN	RESERVED	PINWEN	RTCWEN	RESERVED	HIBREQ	RTCEN
R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0

Table 6-7. HIBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	WRC	R	0x1	<p>Write Complete/Capable</p> <p>Software must poll this bit between write requests and defer writes until WRC = 1 to ensure proper operation.</p> <p>An interrupt can be configured to indicate the WRC has completed. The bit name WRC means "Write Complete," which is the normal use of the bit (between write accesses).</p> <p>However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software.</p> <p>This difference may be exploited by software at reset time to detect which method of programming is appropriate:</p> <p>0 = software delay loops required</p> <p>1 = WRC paced available.</p> <p>0x0 = The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior.</p> <p>0x1 = The interface is ready to accept a write.</p>
30	RETCLR	R/W	0x0	<p>GPIO Retention/Clear</p> <p>This bit is used when the VDD3ON bit is set.</p> <p>This bit must be set when entering the hibernate state when the VDD3ON bit is set.</p> <p>This does not affect behavior when VDD3ON is clear.</p> <p>This bit must be set when enabling VDD3ON mode.</p> <p>0x0 = GPIO retention is released when power is reapplied. The GPIOs are initialized to default values.</p> <p>0x1 = GPIO retention set until software clears this bit.</p>
29-20	RESERVED	R	0x0	
19	OSCSEL	R/W	0x0	<p>Oscillator Select</p> <p>This bit is used to select between the use of an external 32.768-kHz source or the HIB internal low frequency oscillator (HIB LFIOSC).</p> <p>To enable the HIB LFIOSC, CLK32EN must be programmed to 1 at the same time the OSCSEL bit is set.</p> <p>Thus the HIBCTL register should be written with 0x00080040. The HIB low-frequency oscillator has a wide frequency variation, therefore the RTC is not accurate when using this clock source.</p> <p>0x0 = External 32.768-kHz clock source is enabled.</p> <p>0x1 = HIB Low frequency oscillator (HIB LFIOSC) is enabled.</p>
18	RESERVED	R	0x0	
17	OSCDRV	R/W	0x0	<p>Oscillator Drive Capability</p> <p>This bit is used to compensate for larger or smaller filtering capacitors.</p> <p>This bit is not meant to be changed once the Hibernation oscillator has started.</p> <p>Oscillator stability is not guaranteed if the user changes this value after the oscillator is running.</p> <p>0x0 = Low drive strength is enabled, 12 pF.</p> <p>0x1 = High drive strength is enabled, 24 pF.</p>
16	OSCBYP	R/W	0x0	<p>Oscillator Bypass</p> <p>0x0 = The internal 32.768-kHz Hibernation oscillator is enabled. This bit should be cleared when using an external 32.768-kHz crystal.</p> <p>0x1 = The internal 32.768-kHz Hibernation oscillator is disabled and powered down. This bit should be set when using a single-ended oscillator attached to XOSC0.</p>
15	RESERVED	R	0x0	

Table 6-7. HIBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14-13	VBATSEL	R/W	0x1	<p>Select for Low-Battery Comparator</p> <p>This field selects the battery level that is used when checking the battery status.</p> <p>If the battery voltage is below the specified level, the LOWBAT interrupt bit in the HIBRIS register is set.</p> <p>0x0 = 1.9 Volts</p> <p>0x1 = 2.1 Volts (default)</p> <p>0x2 = 2.3 Volts</p> <p>0x3 = 2.5 Volts</p>
12-11	RESERVED	R	0x0	
10	BATCHK	R/W	0x0	<p>Check Battery Status</p> <p>0x0 = When read, indicates that the low-battery comparator cycle is not active. Writing a 0 has no effect.</p> <p>0x1 = When read, indicates the low-battery comparator cycle has not completed. Setting this bit initiates a low-battery comparator cycle. If the battery voltage is below the level specified by VBATSEL field, the LOWBAT interrupt bit in the HIBRIS register is set. A hibernation request is held off if a battery check is in progress.</p>
9	BATWKEN	R/W	0x0	<p>Wake on Low Battery</p> <p>0x0 = The battery voltage level is not automatically checked. Low battery voltage does not cause the microcontroller to wake from hibernation.</p> <p>0x1 = In RTC mode, when this bit is set, the battery voltage level is checked every 512 seconds while in hibernation. In calendar mode, the battery voltage is checked on minutes divisible by 8 while in hibernation. If the voltage is below the level specified by VBATSEL field, the microcontroller wakes from hibernation and the LOWBAT interrupt bit in the HIBRIS register is set.</p>
8	VDD3ON	R/W	0x0	<p>VDD Powered</p> <p>Regardless of the status of the VDD3ON bit, the HIB signal is asserted during Hibernate mode.</p> <p>Thus, when VDD3ON is set, the HIB signal should not be connected to the 3.3V regulator, and the 3.3V power source should remain connected.</p> <p>When this bit is set while in hibernation, all pins are held in the state they were in prior to entering hibernation.</p> <p>For example, inputs remain inputs</p> <p>outputs driven high remain driven high, and so on.</p> <p>Ports retain their state in VDD3ON mode until the RETCLR bit is cleared.</p> <p>The RETCLR bit must be set when the VDD3ON bit is set.</p> <p>0x0 = The internal switches are not used. The HIB signal should be used to control an external switch or regulator.</p> <p>0x1 = The internal switches control the power to the on-chip modules (VDD3ON mode).</p>
7	VABORT	R/W	0x0	<p>Power Cut Abort Enable</p> <p>0x0 = The microcontroller goes into hibernation regardless of the voltage level of the battery.</p> <p>0x1 = When this bit is set, the battery voltage level is checked before entering hibernation. If VBAT is less than the voltage specified by VBATSEL, the microcontroller does not go into hibernation.</p>
6	CLK32EN	R/W	0x0	<p>Clocking Enable</p> <p>This bit must be enabled to use the Hibernation module.</p> <p>0x0 = The Hibernation module clock source is disabled.</p> <p>0x1 = The Hibernation module clock source is enabled.</p>
5	RESERVED	R	0x0	

Table 6-7. HIBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	PINWEN	R/W	0x0	<p>External Wake and Interrupt Pin Enable</p> <p>The external I/O wake pad interrupt is set if the WAKE pin is asserted in Run, Sleep, or Deep Sleep mode regardless of whether the PINWEN bit is 0x0 or 0x1.</p> <p>The interrupt may be forwarded to the processor by setting the EXTW bit in the HIBIM register.</p> <p>0x0 = The status of the WAKE or an external I/O wake pad source pin has no effect on hibernation.</p> <p>0x1 = An assertion of the WAKE pin or an external I/O wake pad source takes the microcontroller out of hibernation. An external I/O wake pad interrupt may be generated in active mode.</p>
3	RTCWEN	R/W	0x0	<p>RTC Wake-up Enable</p> <p>0x0 = An RTC match event has no effect on hibernation.</p> <p>0x1 = An RTC match event (the value the HIBRTCC register matches the value of the HIBRTCM0 register and the value of the RTCSSC field matches the RTCSSM field in the HIBRTCSS register) takes the microcontroller out of hibernation.</p>
2	RESERVED	R	0x0	
1	HIBREQ	R/W	0x0	<p>Hibernation Request</p> <p>After a wake-up event, this bit is automatically cleared by hardware. A hibernation request is ignored if both the PINWEN and RTCWEN bits are clear.</p> <p>0x0 = No hibernation request.</p> <p>0x1 = Set this bit to initiate hibernation.</p>
0	RTCEN	R/W	0x0	<p>RTC Timer /Calendar Enable</p> <p>This bit must be set to enable RTC or calendar mode.</p> <p>For calendar mode enable, the CALEN bit in the HIBCALCTL register must also be set.</p> <p>The low-frequency oscillator has a wide frequency variation, therefore the RTC is not accurate when using this clock source.</p> <p>0x0 = The Hibernation module RTC and calendar mode are disabled.</p> <p>0x1 = The Hibernation module RTC and calendar mode are enabled.</p>

6.5.5 HIBIM Register (Offset = 0x14) [reset = 0x0]

Hibernation Interrupt Mask (HIBIM)

This register is the interrupt mask register for the Hibernation module interrupt sources. Each bit in this register masks the corresponding bit in the Hibernation Raw Interrupt Status (HIBRIS) register. If a bit is unmasked, the interrupt is sent to the interrupt controller. If the bit is masked, the interrupt is not sent to the interrupt controller. The WC bit of the HIBIM register may be set before the CLK32EN bit of the HIBCTL register is set. This allows software to use the WC interrupt trigger to detect when the RTCOSC clock is stable, which may be in excess of one second. If the WC bit is set before the CLK32EN has been set, the mask value is not preserved over a hibernate cycle unless the bit is written a second time.

NOTE: The WC bit of this register is in the system clock domain such that a write to this bit is immediate and may be done before the CLK32EN bit is set in the HIBCTL register.

HIBIM is shown in [Figure 6-13](#) and described in [Table 6-8](#).

Return to [Summary Table](#).

Figure 6-13. HIBIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
VDDFAIL	RSTWK	PADIOWK	WC	EXTW	LOWBAT	RESERVED	RTCALTO
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0

Table 6-8. HIBIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	VDDFAIL	R/W	0x0	VDD Fail Interrupt Mask 0x0 = The VDDFAIL interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the VDDFAIL bit in the HIBRIS register is set.
6	RSTWK	R/W	0x0	Reset Pad I/O Wake-Up Interrupt Mask 0x0 = The RSTWK interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RSTWK bit in the HIBRIS register is set.
5	PADIOWK	R/W	0x0	Pad I/O Wake-Up Interrupt Mask 0x0 = The PADIOWK interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PADIOWK bit in the HIBRIS register is set.
4	WC	R/W	0x0	External Write Complete/Capable Interrupt Mask 0x0 = The WC interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the WC bit in the HIBRIS register is set.

Table 6-8. HIBIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	EXTW	R/W	0x0	External Wake-Up Interrupt Mask 0x0 = The EXTW interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the EXTW bit in the HIBRIS register is set.
2	LOWBAT	R/W	0x0	Low Battery Voltage Interrupt Mask 0x0 = The LOWBAT interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the LOWBAT bit in the HIBRIS register is set.
1	RESERVED	R	0x0	
0	RTCALTO	R/W	0x0	RTC Alert 0 Interrupt Mask 0x0 = The RTCALTO interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RTCALTO bit in the HIBRIS register is set.

6.5.6 HIBRIS Register (Offset = 0x18) [reset = 0x0]

Hibernation Raw Interrupt Status (HIBRIS)

This register is the raw interrupt status for the Hibernation module interrupt sources. Each bit can be masked by clearing the corresponding bit in the HIBIM register. When a bit is masked, the interrupt is not sent to the interrupt controller. Bits in this register are cleared by writing a 1 to the corresponding bit in the Hibernation Interrupt Clear (HIBIC) register or by entering hibernation.

HIBRIS is shown in [Figure 6-14](#) and described in [Table 6-9](#).

Return to [Summary Table](#).

Figure 6-14. HIBRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
VDDFAIL	RSTWK	PADIOWK	WC	EXTW	LOWBAT	RESERVED	RTCALTO
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 6-9. HIBRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	VDDFAIL	R	0x0	VDD Fail Raw Interrupt Status 0x0 = No VDDFAIL interrupt condition exists. 0x1 = An interrupt is sent to the interrupt controller because of arbitrary power removal or because one or more of the supplies (VDD, VDDA or VDDC) has dropped below the defined operating range.
6	RSTWK	R	0x0	Reset Pad I/O Wake-Up Raw Interrupt Status 0x0 = The RESET pin has not been asserted or has not been enabled to wake the device from hibernation. 0x1 = An interrupt is sent to the interrupt controller because the RESET pin has been programmed to wake the device from hibernation.
5	PADIOWK	R	0x0	Pad I/O Wake-Up Raw Interrupt Status 0x0 = One of the wake-enabled GPIO pins or the external RESET pin has not been asserted or has not been enabled to wake the device from hibernation. 0x1 = An interrupt is sent to the interrupt controller because one of the wake-enabled GPIO pins or the external RESET pin has been asserted.
4	WC	R	0x0	Write Complete/Capable Raw Interrupt Status This bit is cleared by writing a 1 to the WC bit in the HIBIC register. 0x0 = The WRC bit in the HIBCTL has not been set. 0x1 = The WRC bit in the HIBCTL has been set.

Table 6-9. HIBRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	EXTW	R	0x0	<p>External Wake-Up Raw Interrupt Status</p> <p>A wake signal source must be cleared by the application after the interrupt has been registered.</p> <p>This bit is cleared by writing a 1 to the EXTW bit in the HIBIC register.</p> <p>The EXTW bit is set if the WAKE pin is asserted in any mode of operation (Run, Sleep, Deep Sleep) regardless of whether the PINWEN bit is set in the HIBCTL register.</p> <p>0x0 = The WAKE pin has not been asserted.</p> <p>0x1 = The WAKE pin has been asserted.</p>
2	LOWBAT	R	0x0	<p>Low Battery Voltage Raw Interrupt Status</p> <p>This bit is cleared by writing a 1 to the LOWBAT bit in the HIBIC register.</p> <p>0x0 = The battery voltage has not dropped below VLOWBAT.</p> <p>0x1 = The battery voltage dropped below VLOWBAT.</p>
1	RESERVED	R	0x0	
0	RTCALTO	R	0x0	<p>RTC Alert 0 Raw Interrupt Status</p> <p>This bit is cleared by writing a 1 to the RTCALTO bit in the HIBIC register.</p> <p>0x0 = No match</p> <p>0x1 = If the RTC is enabled, the value of the HIBRTCC register matches the value in the HIBRTCM0 register and the value of the RTCSSC field matches the RTCSSM field in the HIBRTCSS register. If the Calendar function is enabled, this interrupt status indicates that one or more of the allowed fields in the HIBCAL0/1 register matches in the HIBCALM0/1 register..</p>

6.5.7 HIBMIS Register (Offset = 0x1C) [reset = 0x0]

Hibernation Masked Interrupt Status (HIBMIS)

This register is the masked interrupt status for the Hibernation module interrupt sources. Bits in this register are the AND of the corresponding bits in the HIBRIS and HIBIM registers. When both corresponding bits are set, the bit in this register is set, and the interrupt is sent to the interrupt controller.

HIBMIS is shown in [Figure 6-15](#) and described in [Table 6-10](#).

Return to [Summary Table](#).

Figure 6-15. HIBMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
VDDFAIL	RSTWK	PADIOWK	WC	EXTW	LOWBAT	RESERVED	RTCALTO
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 6-10. HIBMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	VDDFAIL	R	0x0	VDD Fail Interrupt Mask 0x0 = An VDDFAIL interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a an arbitrary loss of power or because on or more of the voltage supplies (VDD, VDDA or VDDC) has dropped below the defined operating range.
6	RSTWK	R	0x0	Reset Pad I/O Wake-Up Interrupt Mask 0x0 = An external reset interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a RESET pin assertion.
5	PADIOWK	R	0x0	Pad I/O Wake-Up Interrupt Mask 0x0 = An external GPIO or reset interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a wake-enabled GPIO or RESET pin assertion.
4	WC	R	0x0	Write Complete/Capable Masked Interrupt Status This bit is cleared by writing a 1 to the WC bit in the HIBIC register. 0x0 = The WRC bit has not been set or the interrupt is masked. 0x1 = An unmasked interrupt was signaled due to the WRC bit being set.
3	EXTW	R	0x0	External Wake-Up Masked Interrupt Status This bit is cleared by writing a 1 to the EXTW bit in the HIBIC register. 0x0 = An external wake-up interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a WAKE pin assertion.

Table 6-10. HIBMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	LOWBAT	R	0x0	<p>Low Battery Voltage Masked Interrupt Status</p> <p>This bit is cleared by writing a 1 to the LOWBAT bit in the HIBIC register.</p> <p>0x0 = A low-battery voltage interrupt has not occurred or is masked.</p> <p>0x1 = An unmasked interrupt was signaled due to a low-battery voltage condition.</p>
1	RESERVED	R	0x0	
0	RTCALTO	R	0x0	<p>RTC Alert 0 Masked Interrupt Status</p> <p>The MIS may apply to either the RTC or calendar block depending on which is enabled.</p> <p>This bit is cleared by writing a 1 to the RTCALTO bit in the HIBIC register.</p> <p>0x0 = An RTC or calendar match interrupt has not occurred or is masked.</p> <p>0x1 = An unmasked interrupt was signaled due to an RTC or calendar match.</p>

6.5.8 HIBIC Register (Offset = 0x20) [reset = 0x0]

Hibernation Interrupt Clear (HIBIC)

This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources. Writing a 1 to a bit clears the corresponding interrupt in the HIBRIS register.

NOTE: Writes to the RSTWK, PADIOWK and WC bits of this register are immediate and the status may be read from the HIBRIS and HIBMIS registers without monitoring the WRC bit of the HIBCTL register.

NOTE: All I/O wake sources are cleared by a write to either or both the RSTWK and PADIOWK bits. This clears the source of interrupts for RSTWK, PADIOWK and the GPIOWAKESTAT register.

HIBIC is shown in [Figure 6-16](#) and described in [Table 6-11](#).

Return to [Summary Table](#).

Figure 6-16. HIBIC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
VDDFAIL	RSTWK	PADIOWK	WC	EXTW	LOWBAT	RESERVED	RTCALTO
R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R-0x0	R/W1C-0x0

Table 6-11. HIBIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	VDDFAIL	R/W1C	0x0	VDD Fail Interrupt Clear Writing a 1 to this bit clears the VDDFAIL bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status.
6	RSTWK	R/W1C	0x0	Reset Pad I/O Wake-Up Interrupt Clear Writing a 1 to this bit clears the RSTWK bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status.
5	PADIOWK	R/W1C	0x0	Pad I/O Wake-Up Interrupt Clear Writing a 1 to this bit clears the PADIOWK bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status.
4	WC	R/W1C	0x0	Write Complete/Capable Interrupt Clear Writing a 1 to this bit clears the WC bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status.
3	EXTW	R/W1C	0x0	External Wake-Up Interrupt Clear Writing a 1 to this bit clears the EXTW bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status.

Table 6-11. HIBIC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	LOWBAT	R/W1C	0x0	Low Battery Voltage Interrupt Clear Writing a 1 to this bit clears the LOWBAT bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status.
1	RESERVED	R	0x0	
0	RTCALTO	R/W1C	0x0	RTC Alert0 Masked Interrupt Clear Writing a 1 to this bit clears the RTCALTO bit in the HIBRIS and HIBMIS registers. Reads return the raw interrupt status. The timer interrupt source cannot be cleared if the RTC value and the HIBRTCM0 register / RTCMSS field values are equal. The match interrupt takes priority over the interrupt clear.

6.5.9 HIBRTCT Register (Offset = 0x24) [reset = 0x7FFF]

Hibernation RTC Trim (HIBRTCT)

This register contains the value that is used to trim the RTC clock predivider. It represents the computed underflow value that is used during the trim cycle. It is represented as $0x7FFF \pm N$ clock cycles, where N is the number of clock cycles to add or subtract every 64 seconds in RTC mode or 60 seconds in calendar mode.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

HIBRTCT is shown in [Figure 6-17](#) and described in [Table 6-12](#).

Return to [Summary Table](#).

Figure 6-17. HIBRTCT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TRIM															
R-0x0																R/W-0x7FFF															

Table 6-12. HIBRTCT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	TRIM	R/W	0x7FFF	RTC Trim Value This value is loaded into the RTC predivider every 64 seconds in RTC counter mode. In calendar mode, the value is loaded every 60 seconds. It is used to adjust the RTC rate to account for drift and inaccuracy in the clock source. Compensation can be adjusted by software by moving the default value of 0x7FFF up or down. Moving the value up slows down the RTC and moving the value down speeds up the RTC.

6.5.10 HIBRTCSS Register (Offset = 0x28) [reset = 0x0]

Hibernation RTC Sub Seconds (HIBRTCSS)

This register contains the RTC sub seconds counter and match values. The RTC value can be read by first reading the HIBRTCC register, reading the RTCSSC field in the HIBRTCSS register, and then rereading the HIBRTCC register. If the two values for HIBRTCC are equal, the read is valid.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

NOTE: There is a minimum system clock rate of three times the HIB clock rate to properly read the HIBRTCSS register.

HIBRTCSS is shown in [Figure 6-18](#) and described in [Table 6-13](#).

Return to [Summary Table](#).

Figure 6-18. HIBRTCSS Register

31	30	29	28	27	26	25	24
RESERVED	RTCSSM						
R-0x0	R/W-0x0						
23	22	21	20	19	18	17	16
RTCSSM							
R/W-0x0							
15	14	13	12	11	10	9	8
RESERVED	RTCSSC						
R-0x0	R-0x0						
7	6	5	4	3	2	1	0
RTCSSC							
R-0x0							

Table 6-13. HIBRTCSS Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	
30-16	RTCSSM	R/W	0x0	RTC Sub Seconds Match The match value is contained in this field in one RTCOSC clock increments. A read returns the current seconds match value.
15	RESERVED	R	0x0	
14-0	RTCSSC	R	0x0	RTC Sub Seconds Count This field contains the sub second RTC count and is read as RTCOSC clock units. For the 32.768-kHz clock source, this would be in units of 1/32,768 seconds.

6.5.11 HIBIO Register (Offset = 0x2C) [reset = 0x80000000]

Hibernation IO Configuration (HIBIO)

This register is used to lock and unlock the external wake pin levels and enable the external RST pin and/or GPIO pins, Port K[7:4], as valid external WAKE sources.

NOTE: This register is in the system clock domain and does not require monitoring the WRC bit of the HIBCTL register before issuing a read or write of this register. Writes to this register are immediate.

NOTE: This register is in the core voltage domain and will not retain values over a hibernate cycle

HIBIO is shown in [Figure 6-19](#) and described in [Table 6-14](#).

Return to [Summary Table](#).

Figure 6-19. HIBIO Register

31	30	29	28	27	26	25	24
IOWRC	RESERVED						
R-0x1	R-0x0						
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			WURSTEN	RESERVED			WUUNLK
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 6-14. HIBIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOWRC	R	0x1	I/O Write Complete Indicates whether or not the configuration that was programmed by the WURSTEN bit or GPIOWAKEPEN and GPIOWAKELVL registers have propagated through the pad ring. 0x0 = The changes programmed in the external pad I/O wake source registers have not propagated through the pad I/O. 0x1 = The changes programmed in the external pad I/O wake source registers have propagated through the pad I/O.
30-5	RESERVED	R	0x0	
4	WURSTEN	R/W	0x0	Reset Wake Source Enable This register bit programming takes affect after WUUNLK has been set. 0x0 = The RST signal is not enabled as a wake source. 0x1 = The RST signal is enabled as a wake source.
3-1	RESERVED	R	0x0	
0	WUUNLK	R/W	0x0	I/O Wake Pad Configuration Enable 0x0 = The I/O WAKE configuration set by the WURSTEN bit or in the GPIO module registers GPIOWAKEPEN and GPIOWAKELVL is ignored. 0x1 = Implement the I/O WAKE configuration, level and enables for the external RST pin and/or GPIO wake-enabled pins. This bit must be cleared before issuing a hibernate request by setting the HIBREQ bit in the HIBCTL register.

6.5.12 HIBDATA Register (Offset = 0x030 to 0x06F) [reset = X]

Hibernation Data (HIBDATA)

This address space is implemented as a 16x32-bit memory (64 bytes). It can be loaded by the system processor in order to store state information and retains its state during a power cut operation as long as a battery is present. HIBDATA registers 0x050 to 0x064 (upper eight words) may only be accessed using the processor privileged mode (default).

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

NOTE: If V_{DD} is arbitrarily removed while a HIBDATA register write operation is in progress, the write operation must be retried after V_{DD} is reapplied.

HIBDATA is shown in [Figure 6-20](#) and described in [Table 6-15](#).

Return to [Summary Table](#).

Figure 6-20. HIBDATA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTD																															
R/W-X																															

Table 6-15. HIBDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RTD	R/W	X	Hibernation Module NV Data

6.5.13 HIBCALCTL Register (Offset = 0x300) [reset = 0x0]

Hibernation Calendar Control (HIBCALCTL)

The Hibernate calendar is enabled by setting the CALEN bit in the HIBCALCTL register. If the BCD bit is set, the fields are reported in BCD format.

HIBCALCTL is shown in [Figure 6-21](#) and described in [Table 6-16](#).

Return to [Summary Table](#).

Figure 6-21. HIBCALCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					CAL24	RESERVED	CALEN
R-0x0					R/W-0x0	R-0x0	R/W-0x0

Table 6-16. HIBCALCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	CAL24	R/W	0x0	Calendar Mode 0x0 = 12 hour, AM/PM Mode 0x1 = 24 hour mode
1	RESERVED	R	0x0	
0	CALEN	R/W	0x0	RTC Calendar/Counter Mode Select The RTC must be enabled by setting the RTCEN bit in the HIBCTL register to use this mode select. 0x0 = RTC Counter mode enabled. 0x1 = Calendar mode enabled

6.5.14 HIBCAL0 Register (Offset = 0x310) [reset = 0x0]

Hibernation Calendar 0 (HIBCAL0)

The Hibernation Calendar 0 (HIBCAL0) register is used when the CALEN bit is set in the HIBCALCTL register.

HIBCAL0 is shown in [Figure 6-22](#) and described in [Table 6-17](#).

[Return to Summary Table.](#)

Figure 6-22. HIBCAL0 Register

31	30	29	28	27	26	25	24
VALID	RESERVED						
R-0x0	R-0x0						
23	22	21	20	19	18	17	16
RESERVED	AMPM	RESERVED	HR				
R-0x0	R-0x0	R-0x0	R-0x0				
15	14	13	12	11	10	9	8
RESERVED		MIN					
R-0x0		R-0x0					
7	6	5	4	3	2	1	0
RESERVED		SEC					
R-0x0		R-0x0					

Table 6-17. HIBCAL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	VALID	R	0x0	Valid Calendar Load The calendar may take several cycles to update as the values roll over. This bit indicates whether the HIBCAL0 register contents are valid. 0x0 = Register currently updating or initializing 0x1 = HIBCAL0 register valid and ready.
30-23	RESERVED	R	0x0	
22	AMPM	R	0x0	AM/PM Designation This bit is used when CAL 24 =0 in the HIBCALCTL register. 0x0 = AM 0x1 = PM
21	RESERVED	R	0x0	
20-16	HR	R	0x0	Hours This field holds the hour information in hexadecimal. For military time, bits 20:16 range from 0x0 to 0x17 (0 to 23 hours). For standard time (AM/PM mode) bits 20:16 range from 0x0 to 0x11, with 0x0 representing 12AM or 12 PM.
15-14	RESERVED	R	0x0	
13-8	MIN	R	0x0	Minutes This field holds the minute information in hexadecimal. Bits 13:8 correspond to hex values from 0x0 to 0x3b (0 to 59 minutes).
7-6	RESERVED	R	0x0	
5-0	SEC	R	0x0	Seconds This field holds the seconds value in hexadecimal. Bits 5:0 correspond to hex values from 0x0 to 0x3b (0 to 59 seconds).

6.5.15 HIBCAL1 Register (Offset = 0x314) [reset = 0x0]

Hibernation Calendar 1 (HIBCAL1)

The Hibernation Calendar 1 (HIBCAL1) register is used when the CALEN bit is set in the HIBCALCTL register.

HIBCAL1 is shown in [Figure 6-23](#) and described in [Table 6-18](#).

[Return to Summary Table.](#)

Figure 6-23. HIBCAL1 Register

31	30	29	28	27	26	25	24
VALID	RESERVED				DOW		
R-0x0	R-0x0				R-0x0		
23	22	21	20	19	18	17	16
RESERVED				YEAR			
R-0x0				R-0x0			
15	14	13	12	11	10	9	8
RESERVED				MON			
R-0x0				R-0x0			
7	6	5	4	3	2	1	0
RESERVED			DOM				
R-0x0			R-0x0				

Table 6-18. HIBCAL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	VALID	R	0x0	Valid Calendar Load The calendar may take several cycles to update as the values roll over. This bit indicates whether the HIBCAL1 register contents are valid. 0x0 = Register currently updating or initializing 0x1 = HIBCAL1 register valid and ready.
30-27	RESERVED	R	0x0	
26-24	DOW	R	0x0	Day of Week This field displays the day of the week in the encodings 0x0 to 0x6. The application defines which days are assigned to each encoding.
23	RESERVED	R	0x0	
22-16	YEAR	R	0x0	Year The last two digits of the year are stored in hexadecimal in this field. Bits 22:16 correspond to hex values from 0x0 to 0x63 (0 to 99 years).
15-12	RESERVED	R	0x0	
11-8	MON	R	0x0	Month This field holds the month value in hexadecimal. Bits 11:8 correspond to hex values from 0x1 to 0xC (1 to 12 months).
7-5	RESERVED	R	0x0	
4-0	DOM	R	0x0	Day of Month This field holds the day of the month value in hexadecimal. Bits 4:0 correspond to hex values from 0x1 to 1F (1 to 31 days). The value 0 is used to show an ignore match.

6.5.16 HIBCALLD0 Register (Offset = 0x320) [reset = 0x0]

Hibernation Calendar Load 0 (HIBCALLD0)

The Hibernation Calendar Load (HIBCALLD0) register is used when the CALEN bit is set in the HIBCALCTL register.

NOTE: This register is write-only; any reads to this register read back as zeros. Errant writes to the HIBCALLD0/1 registers are protected by the Hibernate HIBLOCK register.

HIBCALLD0 is shown in [Figure 6-24](#) and described in [Table 6-19](#).

Return to [Summary Table](#).

Figure 6-24. HIBCALLD0 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED	AMPM	RESERVED	HR				
R-0x0	W-0x0	R-0x0	W-0x0				
15	14	13	12	11	10	9	8
RESERVED		MIN					
R-0x0		W-0x0					
7	6	5	4	3	2	1	0
RESERVED		SEC					
R-0x0		W-0x0					

Table 6-19. HIBCALLD0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0x0	
22	AMPM	W	0x0	AM/PM Designation This bit is used when CAL24 = 0 in the HIBCALCTL register. 0x0 = AM 0x1 = PM
21	RESERVED	R	0x0	
20-16	HR	W	0x0	Hours This field holds the hour information in hexadecimal. Bits 20:16 correspond to hex values from 0x0 to 0x17 (0 to 23 hours).
15-14	RESERVED	R	0x0	
13-8	MIN	W	0x0	Minutes This field holds the minute information in hexadecimal. Bits 13:8 correspond to hex values from 0x0 to 0x3B (0 to 59 minutes).
7-6	RESERVED	R	0x0	
5-0	SEC	W	0x0	Seconds This field holds the seconds value in hexadecimal. Bits 5:0 correspond to hex values from 0x0 to 0x3B (0 to 59 seconds).

6.5.17 HIBCALLD1 Register (Offset = 0x324) [reset = 0x0]

Hibernation Calendar Load (HIBCALLD1)

The Hibernation Calendar Load 1 (HIBCALLD1) register is used when the CALEN bit is set in the HIBCALCTL register.

NOTE: This register is write-only; any reads to this register read back as zeros. Errant writes to the HIBCALLD0/1 registers are protected by the Hibernate HIBLOCK register.

HIBCALLD1 is shown in [Figure 6-25](#) and described in [Table 6-20](#).

Return to [Summary Table](#).

Figure 6-25. HIBCALLD1 Register

31	30	29	28	27	26	25	24
RESERVED					DOW		
R-0x0					W-0x0		
23	22	21	20	19	18	17	16
RESERVED	YEAR						
R-0x0				W-0x0			
15	14	13	12	11	10	9	8
RESERVED				MON			
R-0x0				W-0x0			
7	6	5	4	3	2	1	0
RESERVED			DOM				
R-0x0			W-0x0				

Table 6-20. HIBCALLD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0x0	
26-24	DOW	W	0x0	Day of Week This field is written with the day of the week in the encodings 0x0 to 0x6. The application defines which days are assigned to each encoding.
23	RESERVED	R	0x0	
22-16	YEAR	W	0x0	Year The last two digits of the year are written in this field in hexadecimal. For example, "12" would be programmed into this field for 2012. Bits 22:16 correspond to hex values from 0x0 to 0x63 (0 to 99 years).
15-12	RESERVED	R	0x0	
11-8	MON	W	0x0	Month The month value is written in this field in hexadecimal. Bits 11:8 correspond to hex values from 0x1 to 0xC (1 to 12 months).
7-5	RESERVED	R	0x0	
4-0	DOM	W	0x0	Day of Month The day of the month value is written in this field in hexadecimal. Bits 4:0 correspond to hex values from 0x1 to 1F (1 to 31 days). The encoding 0x0 is RESERVED for the ignore match function.

6.5.18 HIBCALM0 Register (Offset = 0x330) [reset = 0x0]

Hibernation Calendar Match 0 (HIBCALM0)

The Hibernation Calendar Match 0 (HIBCALM0) register is used when the CALEN bit is set in the HIBCALCTL register. This register is loaded with desired match values for calendar mode. Once the HIBCAL0/1 register values equal the HIBCALM0/1 register values, the RTCALT0 bit is set in the HIBRIS register.

NOTE: The day of week, month and year are not included in the match functionality.

HIBCALM0 is shown in [Figure 6-26](#) and described in [Table 6-21](#).

Return to [Summary Table](#).

Figure 6-26. HIBCALM0 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED	AMPM	RESERVED	HR				
R-0x0	R/W-0x0	R-0x0	R/W-0x0				
15	14	13	12	11	10	9	8
RESERVED		MIN					
R-0x0		R/W-0x0					
7	6	5	4	3	2	1	0
RESERVED		SEC					
R-0x0		R/W-0x0					

Table 6-21. HIBCALM0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0x0	
22	AMPM	R/W	0x0	AM/PM Designation This bit is used when CAL 24 =0 in the HIBCALCTL register. 0x0 = AM 0x1 = PM
21	RESERVED	R	0x0	
20-16	HR	R/W	0x0	Hours This field match value for the hours in hexadecimal units. Bits 20:16 correspond to hex values from 0x0 to 0x17 (0 to 23 hours). To ignore the hours match, write this field to all 1s.
15-14	RESERVED	R	0x0	
13-8	MIN	R/W	0x0	Minutes This field holds the match value for minutes in hexadecimal units. Bits 13:8 correspond to hex values from 0x0 to 0x3B (0 to 59 minutes). To ignore the hours match, write this field to all 1s.
7-6	RESERVED	R	0x0	
5-0	SEC	R/W	0x0	Seconds This field holds the match value for seconds. The value is represented in hexadecimal. Bits 5:0 correspond to hex values from 0x0 to 0x3b (0 to 59 seconds). To ignore the hours match, write this field to all 1s.

6.5.19 HIBCALM1 Register (Offset = 0x334) [reset = 0x0]

Hibernation Calendar Match 1 (HIBCALM1)

The Hibernation Calendar Match 1 (HIBCALM1) register is used when the CALEN bit is set in the HIBCALCTL register. This register is loaded with desired match values for calendar mode. Once the HIBCAL0/1 register values equal the HIBCALM0/1 register values, the RTCALT0 bit is set in the HIBRIS register.

NOTE: The day of week, month and year are not included in the match functionality.

HIBCALM1 is shown in [Figure 6-27](#) and described in [Table 6-22](#).

Return to [Summary Table](#).

Figure 6-27. HIBCALM1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																									DOM						
R-0x0																									R/W-0x0						

Table 6-22. HIBCALM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4-0	DOM	R/W	0x0	Day of Month This field holds the match value for the day of the month in hexadecimal. Bits 4:0 correspond to hex values from 0x1 to 1F (1 to 31 days). To disable match for the day of the month, the value 0x0 is used.

6.5.20 HIBLOCK Register (Offset = 0x360) [reset = 0x0]

Hibernation Lock (HIBLOCK)

Writing 0xA3359554 to the HIBLOCK register enables write access to the HIBRTCLD, HIBCALLD0, HIBCALLD1 and Tamper registers. Writing any other value to the HIBLOCK register re-enables the locked state for register writes to all the other registers. Reading the HIBLOCK register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the HIBLOCK register returns 0x00000001 when locked; otherwise, the returned value is 0x00000000 (unlocked).

HIBLOCK is shown in [Figure 6-28](#) and described in [Table 6-23](#).

Return to [Summary Table](#).

Figure 6-28. HIBLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIBLOCK																															
R/W-0x0																															

Table 6-23. HIBLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HIBLOCK	R/W	0x0	Hibernate Lock A write of 0xA3359554 unlocks the HIBRCTL and Tamper registers.

6.5.21 HIBTPCTL Register (Offset = 0x400) [reset = 0x0]

HIB Tamper Control (HIBTPCTL)

The Tamper Control (HIBTPCTL) register provides control of the module.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

NOTE: Errant writes to the Tamper registers are protected by the Hibernate HIBLOCK register.

HIBTPCTL is shown in [Figure 6-29](#) and described in [Table 6-24](#).

Return to [Summary Table](#).

Figure 6-29. HIBTPCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				WAKE	RESERVED	MEMCLR	
R-0x0				R/W-0x0	R-0x0	R/W-0x0	
7	6	5	4	3	2	1	0
RESERVED			TPCLR	RESERVED			TPEN
R-0x0			W1C-0x0	R-0x0			R/W-0x0

Table 6-24. HIBTPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	WAKE	R/W	0x0	Wake from Hibernate on a Tamper Event 0x0 = Do not wake from hibernate on a tamper event. 0x1 = Wake from hibernate on a tamper event.
10	RESERVED	R	0x0	
9-8	MEMCLR	R/W	0x0	HIB Memory Clear on Tamper Event 0x0 = Do not Clear HIB memory on tamper event. 0x1 = Clear Lower 32 Bytes of HIB memory on tamper event 0x2 = Clear upper 32 Bytes of HIB memory on tamper event 0x3 = Clear all HIB memory on tamper event
7-5	RESERVED	R	0x0	
4	TPCLR	W1C	0x0	Tamper Event Clear Writing a 1 to this bit clears the tamper event. The status of the clear is reflected in the STATE bit field.
3-1	RESERVED	R	0x0	

Table 6-24. HIBTPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	TPEN	R/W	0x0	<p>Tamper Module Enable</p> <p>This bit enables the Tamper module.</p> <p>Once tamper is enabled, the following HIBCTL register bits are locked and cannot be modified: OSCSEL OSCDRV OSCBYP VDD3ON CLK32EN RTCEN</p> <p>0x0 = Tamper module disabled.</p> <p>0x1 = Tamper module Enabled.</p>

6.5.22 HIBTPSTAT Register (Offset = 0x404) [reset = 0x0]

HIB Tamper Status (HIBTPSTAT)

The HIB Tamper Status (HIBTPCTL) register provides status of the module.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

NOTE: Errant writes to the Tamper registers are protected by the Hibernate HIBLOCK register.

HIBTPSTAT is shown in [Figure 6-30](#) and described in [Table 6-25](#).

Return to [Summary Table](#).

Figure 6-30. HIBTPSTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				STATE		XOSCST	XOSCFAIL
R-0x0				R-0x0		R-0x0	R/W1C-0x0

Table 6-25. HIBTPSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-2	STATE	R	0x0	Tamper Module Status Tamper is defined as being configured when the tamper I/Os have been enabled (setting the ENx bits in the HIBTPIO register). 0x0 = Tamper disabled. 0x1 = Tamper configured. 0x2 = Tamper pin event occurred.
1	XOSCST	R	0x0	External Oscillator Status 0x0 = Active 0x1 = Stopped
0	XOSCFAIL	R/W1C	0x0	External Oscillator Failure Write a 1 to this bit to clear it. 0x0 = External oscillator is valid. 0x1 = External oscillator has failed

6.5.23 HIBTPIO Register (Offset = 0x410) [reset = 0x0]

HIB Tamper I/O Control (HIBTPIO)

The HIB Tamper I/O Control (HIBTPIO) register provides control of the Tamper I/O.

NOTE: Except for the HIBIO and a portion of the HIBIC register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the HIBCTL register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See [Section 6.3.1](#). The HIBIO register and bits RSTWK, PADIOWK and WC of the HIBIC register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the HIBCTL and HIBIM before the CLK32EN bit in the HIBCTL register has been set may produce unexpected results.

NOTE: Errant writes to the Tamper registers are protected by the Hibernate HIBLOCK register.

HIBTPIO is shown in [Figure 6-31](#) and described in [Table 6-26](#).

Return to [Summary Table](#).

Figure 6-31. HIBTPIO Register

31	30	29	28	27	26	25	24
RESERVED				GFLTR3	PUEN3	LEV3	EN3
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
RESERVED				GFLTR2	PUEN2	LEV2	EN2
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED				GFLTR1	PUEN1	LEV1	EN1
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED				GFLTR0	PUEN0	LEV0	EN0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 6-26. HIBTPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0x0	
27	GFLTR3	R/W	0x0	TMPR3 Glitch Filtering 0x0 = A trigger match level is ignored until the TMPR3 signal is stable for two hibernate clocks. 0x1 = A trigger match level is ignored until the TMPR3 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz).
26	PUEN3	R/W	0x0	TMPR3 Internal Weak Pullup Enable 0x0 = Pullup disabled 0x1 = Pullup enabled
25	LEV3	R/W	0x0	TMPR3 Trigger Level 0x0 = Trigger on level low 0x1 = Trigger on level high
24	EN3	R/W	0x0	TMPR3 Enable 0x0 = Detect disabled 0x1 = Detect enabled
23-20	RESERVED	R	0x0	

Table 6-26. HIBTPIO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	GFLTR2	R/W	0x0	TMPR2 Glitch Filtering 0x0 = A trigger match level is ignored until the TMPR2 signal is stable for two hibernate clocks. 0x1 = A trigger match level is ignored until the TMPR 2 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz).
18	PUEN2	R/W	0x0	TMPR2 Internal Weak Pullup Enable 0x0 = Pullup disabled 0x1 = Pullup enabled
17	LEV2	R/W	0x0	TMPR2 Trigger Level 0x0 = Trigger on level low 0x1 = Trigger on level high
16	EN2	R/W	0x0	TMPR2 Enable 0x0 = Detect disabled 0x1 = Detect enabled
15-12	RESERVED	R	0x0	
11	GFLTR1	R/W	0x0	TMPR1 Glitch Filtering 0x0 = A trigger match level is ignored until the TMPR1 signal is stable for two hibernate clocks. 0x1 = A trigger match level is ignored until the TMPR1 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz).
10	PUEN1	R/W	0x0	TMPR1 Internal Weak Pullup Enable 0x0 = Pullup disabled 0x1 = Pullup enabled
9	LEV1	R/W	0x0	TMPR1 Trigger Level 0x0 = Trigger on level low 0x1 = Trigger on level high
8	EN1	R/W	0x0	TMPR1 Enable 0x0 = Detect disabled 0x1 = Detect enabled
7-4	RESERVED	R	0x0	
3	GFLTR0	R/W	0x0	TMPR0 Glitch Filtering 0x0 = A trigger match level is ignored until the TMPR0 signal is stable for two hibernate clocks. 0x1 = A trigger match level is ignored until the TMPR0 signal is stable for 3071 Hibernate Clocks (93.7 ms using 32.768 kHz).
2	PUEN0	R/W	0x0	TMPR0 Internal Weak Pullup Enable 0x0 = Pullup disabled 0x1 = Pullup enabled
1	LEV0	R/W	0x0	TMPR0 Trigger Level 0x0 = Trigger on level low 0x1 = Trigger on level high
0	EN0	R/W	0x0	TMPR0 Enable 0x0 = Detect disabled 0x1 = Detect enabled

6.5.24 HIBTPLOG0, HIBTPLOG2, HIBTPLOG4, HIBTPLOG6 Registers (Offset = 0x4E0 to 0x4F8) [reset = 0x0]

HIB Tamper Log 0 (HIBTPLOG0), offset 0x4E0

HIB Tamper Log 2 (HIBTPLOG2), offset 0x4E8

HIB Tamper Log 4 (HIBTPLOG4), offset 0x4F0

HIB Tamper Log 6 (HIBTPLOG6), offset 0x4F8

The HIB Tamper Log (HIBTPLOG) even registers capture the time information during a tamper event. Up to four tamper logs can be stored. The HIBTPLOG registers are cleared when the TPCLR bit is written in the HIBTPCTL register.

NOTE: It is recommended that an external oscillator is used if accurate time stamps on the tamper log are critical.

HIBTPLOG0 is shown in [Figure 6-32](#) and described in [Table 6-27](#).

Return to [Summary Table](#).

Figure 6-32. HIBTPLOG0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIME																															
R-0x0																															

Table 6-27. HIBTPLOG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TIME	R	0x0	<p>Tamper Log Calendar Information.</p> <p>When the hibernate module is configured for RTC count mode, the time from the RTCCC register is captured on a tamper event.</p> <p>If the calendar function is enabled, the captured time is configured as the hex values for year, month, day, hour, minute and seconds. 24 hour mode should be used by setting the CAL24 bit in HIBCALCTL register.</p> <p>The format of the calendar information is as follows:</p> <p>TIME[31:26]: Year (0-64)</p> <p>TIME[25:22]: Month</p> <p>TIME[21:17]: Day of month</p> <p>TIME[16:12]: Hours</p> <p>TIME[11:6]: Minutes</p> <p>TIME[5:0]: Seconds</p>

6.5.25 HIBTPLOG1, HIBTPLOG3, HIBTPLOG5, HIBTPLOG7 Registers (Offset = 0x4E4 to 0x4FC) [reset = 0x0]

HIB Tamper Log 1 (HIBTPLOG1), offset 0x4E4

HIB Tamper Log 3 (HIBTPLOG3), offset 0x4EC

HIB Tamper Log 5 (HIBTPLOG5), offset 0x4F4

HIB Tamper Log 7 (HIBTPLOG7), offset 0x4FC

The HIB Tamper Log (HIBTPLOGn) odd registers capture the trigger information during a tamper event. Up to four tamper logs can be stored. The HIBTPLOG registers are cleared when the TPCLR bit is set to 1 in the HIBTPCTL register. The HIBTPLOG7 register contains to OR of all events after the 3rd event is logged in HIBTPLOG5. The HIBTPLOG7 register is cleared on a Hibernation module reset.

HIBTPLOG1 is shown in [Figure 6-33](#) and described in [Table 6-28](#).

Return to [Summary Table](#).

Figure 6-33. HIBTPLOG1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							XOSC
R-0x0							R-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				TRIG3	TRIG2	TRIG1	TRIG0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 6-28. HIBTPLOG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	XOSC	R	0x0	Status of external 32.768-kHz oscillator 0x0 = Default 0x1 = 32.768-kHz oscillator has failed
15-4	RESERVED	R	0x0	
3	TRIG3	R	0x0	Status of TMPR[3] Trigger 0x0 = Default 0x1 = A tamper event has been detected on TMPR[3]
2	TRIG2	R	0x0	Status of TMPR[2] Trigger 0x0 = Default 0x1 = A tamper event has been detected on TMPR[2]
1	TRIG1	R	0x0	Status of TMPR[1] Trigger 0x0 = Default 0x1 = A tamper event has been detected on TMPR[1]
0	TRIG0	R	0x0	Status of TMPR[0] Trigger 0x0 = Default 0x1 = A tamper event has been detected on TMPR[0]

6.5.26 HIBPP Register (Offset = 0xFC0) [reset = 0x2]

Hibernation Peripheral Properties (HIBPP)

This register describes the features available within the Hibernation Module.

HIBPP is shown in [Figure 6-34](#) and described in [Table 6-29](#).

Return to [Summary Table](#).

Figure 6-34. HIBPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						TAMPER	WAKENC
R-0x0						R-0x1	R-0x0

Table 6-29. HIBPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	TAMPER	R	0x1	Tamper Pin Presence 0x0 = Tamper module is not present. 0x1 = Tamper module is present.
0	WAKENC	R	0x0	Wake Pin Presence 0x0 = WAKE pin is present. 0x1 = WAKE pin is not part of the package pinout.

6.5.27 HIBCC Register (Offset = 0xFC8) [reset = 0x0]

Hibernation Clock Control (HIBCC)

This register enables alternate clock sources.

NOTE: This register is in the system clock domain. Writes to this register do not require waiting for the WRC bit of the HIBCTL register to be set.

HIBCC is shown in [Figure 6-35](#) and described in [Table 6-30](#).

Return to [Summary Table](#).

Figure 6-35. HIBCC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							SYSCLOCKEN
R-0x0							R/W-0x0

Table 6-30. HIBCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	SYSCLOCKEN	R/W	0x0	RTCOSC to System Clock Enable This bit RTCOSC clock to be sent to the system control for selection as a possible system clock source. Default mode is disabled to support low power modes. 0x0 = RTCOSC is not available as a system clock source. 0x1 = RTCOSC is available for use as a system clock source.

Internal Memory

The MSP432E4 microcontrollers support 256KB of bit-banded SRAM, internal ROM, 1024KB of flash memory, and 6KB of EEPROM.

The MSP432E4 microcontrollers provide 1024KB of on-chip flash memory. The flash memory is configured as four banks of 16K × 128 bits (4 × 256KB total) that are two-way interleaved. Memory blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

The MSP432E4 microcontrollers provide enhanced performance and power savings by implementation of two sets of instruction prefetch buffers. Each prefetch buffer is 2 × 256 bits and can be combined as a 4 × 256-bit prefetch buffer.

The EEPROM module provides a well-defined register interface to support accesses to the EEPROM with both a random access style of read and write as well as a rolling or sequential access scheme. A password model allows the application to lock one or more EEPROM blocks to control access on 16-word boundaries.

Topic	Page
7.1 Block Diagram.....	532
7.2 Functional Description	532
7.3 Flash Registers	550
7.4 EEPROM Registers	571
7.5 System Control Memory Registers	589

7.1 Block Diagram

Figure 7-1 shows the internal memory and control structure. The dashed box in the figure indicate registers residing in the System Control module.

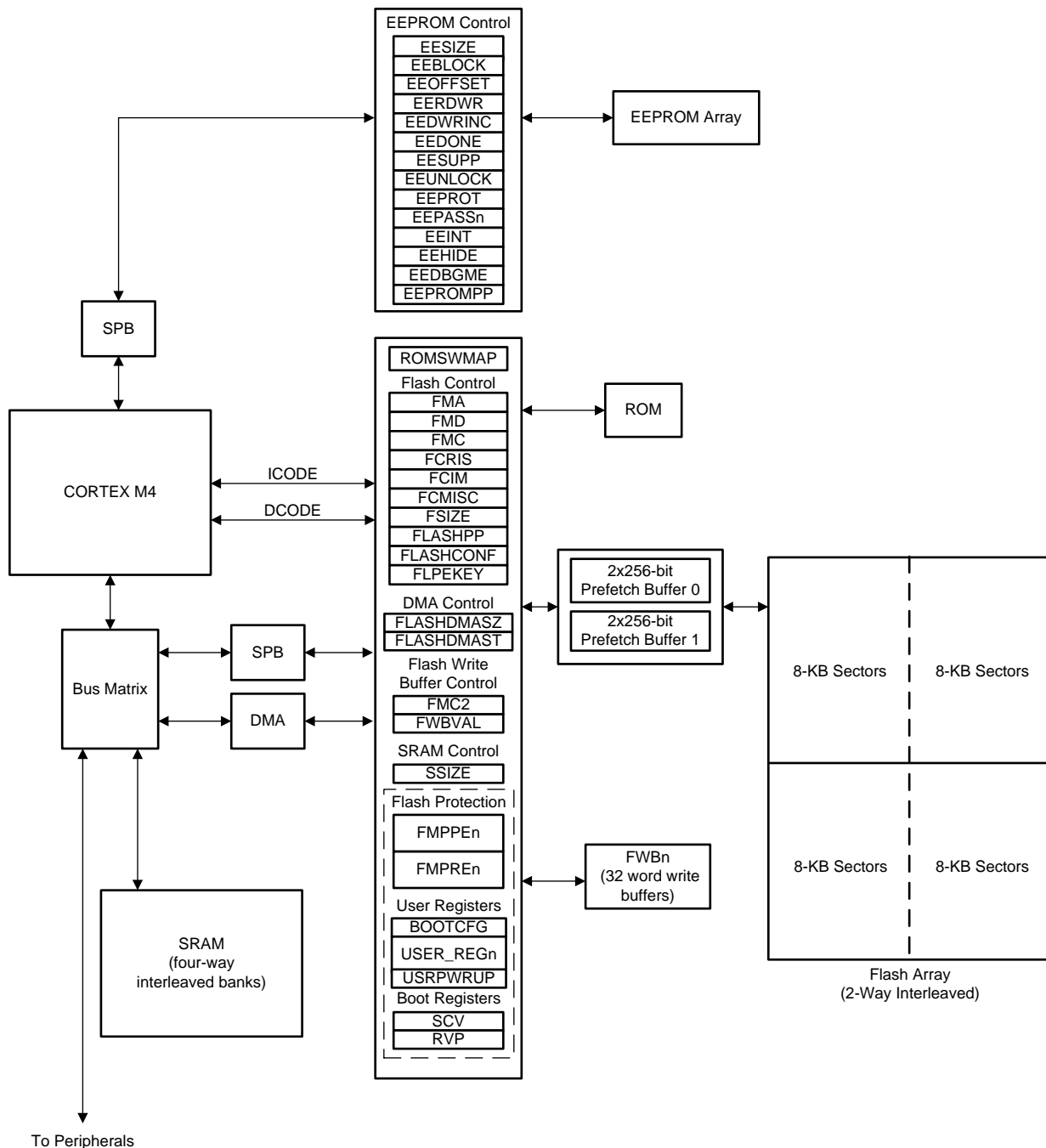


Figure 7-1. Internal Memory Block Diagram

7.2 Functional Description

This section describes the functionality of the SRAM, ROM, flash, and EEPROM memories.

NOTE: The μ DMA has read-only access to flash (in run mode only).

7.2.1 SRAM

The internal system SRAM of the MSP432E4 devices is located at address 0x2000.0000 of the device memory map. To reduce the number of time consuming read-modify-write (RMW) operations, Arm provides bit-banding technology in the processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation. The bit-band base is located at address 0x2200.0000.

The bit-band alias is calculated by using [Equation 4](#).

$$\text{bit-band alias} = \text{bit-band base} + (\text{byte offset} \times 32) + (\text{bit number} \times 4) \quad (4)$$

For example, if bit 3 at address 0x2000.1000 is to be modified, the bit-band alias is calculated as:

$$0x2200.0000 + (0x1000 \times 32) + (3 \times 4) = 0x2202.000C$$

With the alias address calculated, an instruction performing a read/write to address 0x2202.000C allows direct access to only bit 3 of the byte at address 0x2000.1000.

For details about bit-banding, see [Section 1.5.5](#).

NOTE: The SRAM is implemented using four-way, 32-bit wide interleaved SRAM banks (separate SRAM arrays). These SRAM banks allow for increased speed between memory accesses. When using interleaving, a write to one bank followed by a read of another bank can occur in successive clock cycles without incurring any delay. However, a write access that is followed immediately by a read access to the same bank incurs a stall of a single clock cycle.

The SRAM layout allows for multiple masters to access different SRAM banks simultaneously. If two masters attempt to access the same SRAM bank, the master with the higher priority gains access to the memory bus and the master with the lower priority is stalled by one wait state. If four masters attempt to access the same SRAM bank, access by the master with the lowest priority is delayed by three wait states. The CPU core always has the highest priority for SRAM accesses.

7.2.2 ROM

The internal ROM is at address 0x0100.0000 of the device memory map.

The ROM contains the following components:

- Bootloader and vector table
- Peripheral Driver Library for product-specific peripherals and interfaces
- AES cryptography tables
- CRC error detection functionality

The bootloader is used as an initial program loader (when the flash location 0x0000.0004, the reset vector location is 1s [that is, erased state of flash]) and as a firmware upgrade mechanism that is application-initiated (by calling back to the bootloader).

The Peripheral Driver Library APIs in the ROM can be called by applications, reducing flash memory requirements and freeing the flash memory to be used for other purposes (such as additional features in the application).

AES is a publicly defined encryption standard used by the U.S. government.

CRC is a technique to validate whether a block of data has the same contents as when previously checked.

NOTE: CRC and AES software programs are available for backward compatibility. A device that has enhanced CRC and AES integrated modules should use this hardware for best performance. See [Chapter 13](#) and [Chapter 9](#) for more information.

7.2.2.1 Boot Configuration

After POR and device initialization occurs, the hardware loads the stack pointer from flash or ROM based on the presence of an application in flash and the state of the EN bit in the BOOTCFG register. If the flash address 0x0000.0004 contains an erased word (value 0xFFFF.FFFF) or the EN bit is of the BOOTCFG register is clear, the stack pointer and reset vector pointer are loaded from ROM at address 0x0100.0000 and 0x0100.0004, respectively. The bootloader executes and configures the available boot slave interfaces and waits for an external memory to load its software. The bootloader uses a simple packet interface to provide synchronous communication with the device. The speed of the bootloader is determined by the internal oscillator (PIOSC) frequency. The following serial interfaces can be used:

- UART0
- SSI0
- I2C0
- USB

If the check of the flash at address 0x0000.0004 contains a valid reset vector value and the EN bit in the BOOTCFG register is set, the stack pointer and reset vector values are fetched from the beginning of flash. This application stack pointer and reset vector are loaded and the processor executes the application directly. Otherwise, the stack pointer and reset vector values are fetched from the beginning of ROM.

Figure 7-2 shows the bootloader selection sequence.

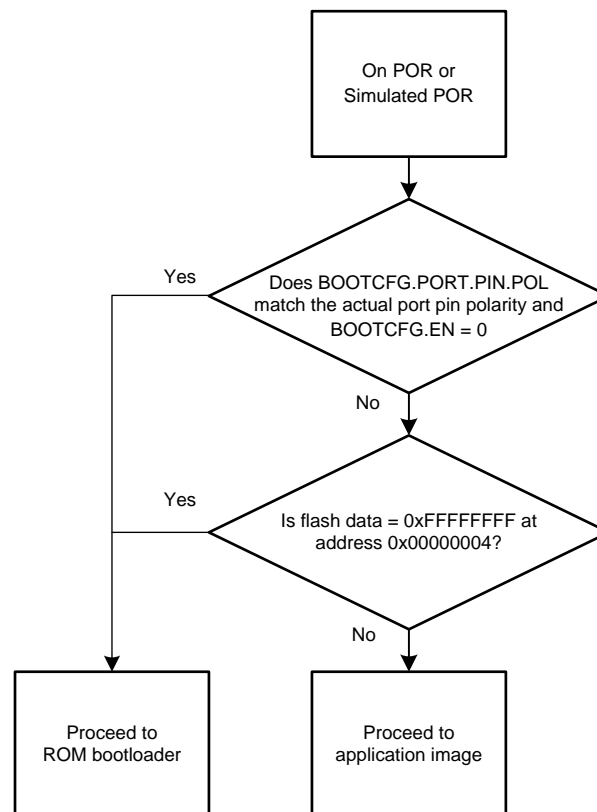


Figure 7-2. Boot Configuration Flow

7.2.2.2 Peripheral Driver Library

A table at the beginning of the ROM points to the entry points for the APIs that are provided in the ROM. Accessing the API through these tables provides scalability; while the API locations can change in future versions of the ROM, the API tables do not. The tables are split into two levels; the main table contains one pointer per peripheral which points to a secondary table that contains one pointer per API that is associated with that peripheral. The main table is located at 0x0100.0010, after the Cortex-M4F vector table in the ROM.

Additional APIs are available for graphics and USB functions but are not preloaded into ROM. The graphics library provides a set of graphics primitives and a widget set for creating graphical user interfaces on MSP432E4 microcontroller-based boards that have a graphical display. The USB library is a set of data types and functions for creating USB Device, Host, or On-The-Go (OTG) applications on MSP432E4 microcontroller-based boards.

7.2.2.3 Advanced Encryption Standard (AES) Cryptography

AES is a strong encryption method with reasonable performance and size. AES is fast in both hardware and software, is fairly easy to implement, and requires little memory. AES is ideal for applications that can use prearranged keys, such as setup during manufacturing or configuration.

7.2.2.4 Cyclic Redundancy Check (CRC) Error Detection

The CRC technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (for example, XOR all bits) because it catches changes more readily. When device initialization is executing from ROM, a CRC-32 validates the data being transferred into registers and memory. The CRC ensures no instructions were skipped in a sequence or no data was corrupted during transfer.

7.2.3 Flash Memory

The flash memory is configured in groups of four banks of 16K × 128 bits (4 × 256KB total) that are two-way interleaved (see [Figure 7-3](#)).

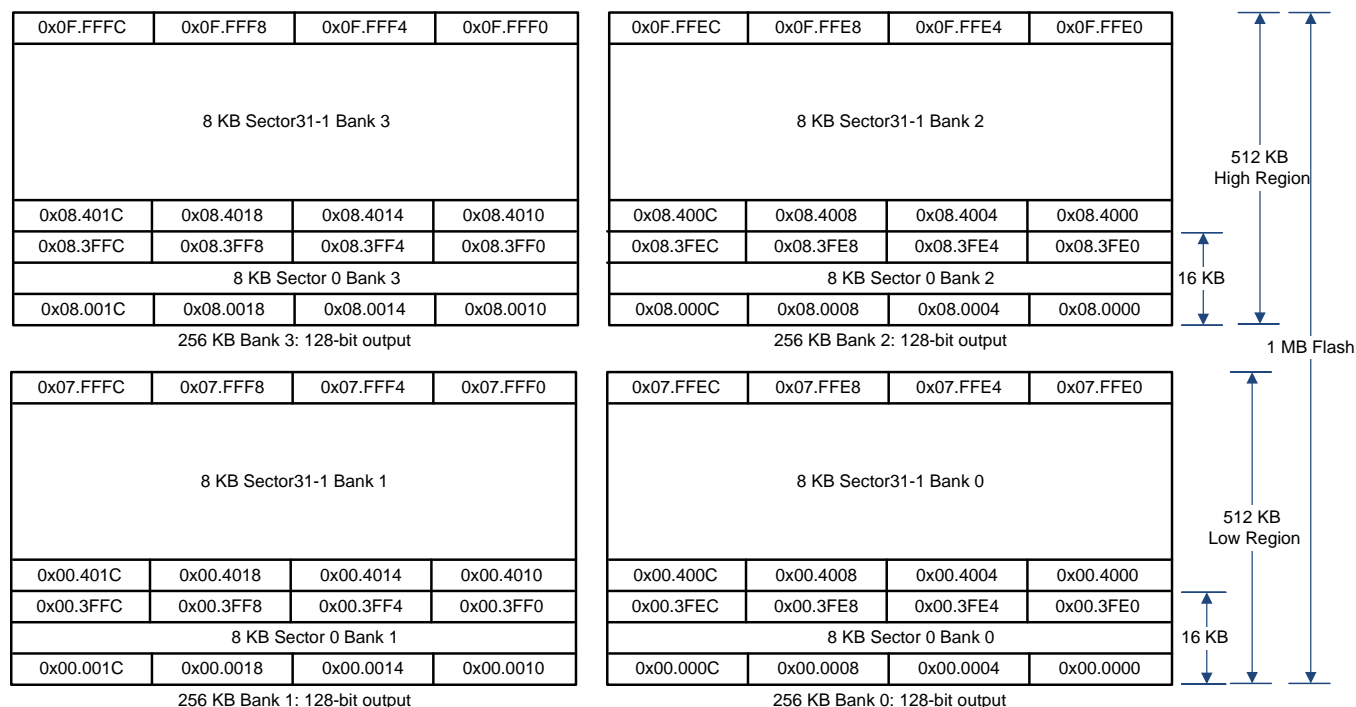


Figure 7-3. Flash Memory Configuration

The interleaved memory prefetches 256 bits at a time. The prefetch buffers allow the maximum performance of a 120-MHz CPU speed to be maintained with linear code or loops that fit within the prefetch buffer. It is recommended that code be compiled with switches set to eliminate literals as much as possible, as a literal causes a flash access for that word and a stall for the wait states. Most compilers support transforming literals into in-line code, which executes faster in a system in which the memory subsystem is slower than the CPU.

Because the memory is two-way interleaved and each bank individually is an 8KB sector, when the user erases a sector, using the ERASE bits in the Flash Memory Control (FMC) register, it is a 16KB erase. Erasing a block causes the entire contents of the block to be reset to all 1s.

7.2.3.1 Flash Configuration

Depending on the CPU frequency, the application must program the flash clock high time (FBCHT), flash bank clock edge (FBCE) and flash wait states (FWS) in the Memory Timing Parameter Register 0 for main flash and EEPROM (MEMTIM0) at System Control Module offset 0x0C0. [Table 7-1](#) lists details the bit field values that are required for the given CPU frequency ranges.

Table 7-1. MEMTIM0 Register Configuration and Frequency

CPU Frequency Range (f) in MHz	Time Period Range (t) in ns	Flash Bank Clock High Time (FBCHT)	Flash Bank Clock Edge (FBCE)	Flash Wait States (FWS)
16	62.5	0x0	1	0x0
16 < f ≤ 40	62.5 > t ≥ 25	0x2	0	0x1
40 < f ≤ 60	25 > t ≥ 16.67	0x3	0	0x2
60 < f ≤ 80	16.67 > t ≥ 12.5	0x4	0	0x3
80 < f ≤ 100	12.5 > t ≥ 10	0x5	0	0x4
100 < f ≤ 120	10 > t ≥ 8.33	0x6	0	0x5

To update the MEMTIM0 register with the new flash configuration values, the MEMTIMU bit must be set in the Run and Sleep Mode Configuration (RSCLKCFG) register at System Control offset 0x0B0.

NOTE: The associated flash and EEPROM fields in the MEMTIM0 register must be programmed to the same values. For example, the FWS field must be programmed to the same value as the EWS field.

7.2.3.2 Prefetch Buffers

The prefetch buffers can exist as a single set of 2 × 256-bit buffers or 4 × 256-bit buffers, depending on the SPFE bit programmed in the Flash Configuration (FLASHCONF) register at offset 0xFC8. At reset, all four buffers are enabled. The buffers are filled using a least-recently-used (LRU) method. When operating in a single set buffer configuration, the two, 256-bit buffers create a deterministic configuration as each *next* write is sent to the previous buffer that was written. [Figure 7-4](#) depicts the single 256-bit buffer set. The single prefetch buffer set should only be used when the code execution must be purely deterministic for the number of clock cycles it takes to execute. Using the four prefetch buffer configuration is the preferred method of configuration.

		255	224	223	192	191	160	159	128	127	96	95	64	63	32	31	0
Prefetch Buffer 0	TAG	WORD 7	WORD 6	WORD 5	WORD 4	WORD 3	WORD 2	WORD 1	WORD 0								
Prefetch Buffer 1	TAG	WORD 7	WORD 6	WORD 5	WORD 4	WORD 3	WORD 2	WORD 1	WORD 0								

Figure 7-4. Single 256-Bit Prefetch Buffer Set

When the buffers are configured as four, 256-bit buffers, they function as one set with one of the four buffers tagged as the LRU and the next to be used when an autofill or miss occurs.

		255	224	223	192	191	160	159	128	127	96	95	64	63	32	31	0
Prefetch Buffer 0	TAG	WORD 7	WORD 6	WORD 5	WORD 4	WORD 3	WORD 2	WORD 1	WORD 0								
Prefetch Buffer 1	TAG	WORD 7	WORD 6	WORD 5	WORD 4	WORD 3	WORD 2	WORD 1	WORD 0								
Prefetch Buffer 2	TAG	WORD 7	WORD 6	WORD 5	WORD 4	WORD 3	WORD 2	WORD 1	WORD 0								
Prefetch Buffer 3	TAG	WORD 7	WORD 6	WORD 5	WORD 4	WORD 3	WORD 2	WORD 1	WORD 0								

Figure 7-5. Four 256-Bit Prefetch Buffer Configuration

The address of the autofill is stored in this tag register so that address violations can be identified immediately and miss processing can begin directly. Every ICODE access is checked against valid tags to see if the target word is already in the buffers.

If there is a hit, the target word is immediately sent to the CPU with no wait states. If there is a miss, the prefetch buffer is invalidated and the miss is processed as a 256-bit read from the flash subsystem to fill the next, least-recently used prefetch buffer. Two memory banks are read in parallel to retrieve 256-bits worth of data.

If an autofill has been started and a miss occurs, the autofill completes before the miss is processed. If an autofill occurs that hits the prefetch buffer being processed for the autofill, then the ICODE bus is stalled until the autofill is complete and new entry can be accessed. For an instruction miss, access to the flash bank starts immediately after the address is available, provided the flash subsystem is not already processing a DCODE bus access or a program or erase operation in the same banks. The target word is passed to the CPU one cycle after it is written to the prefetch buffer.

Figure 7-6 shows the timing diagram for a hit in the prefetch buffer.

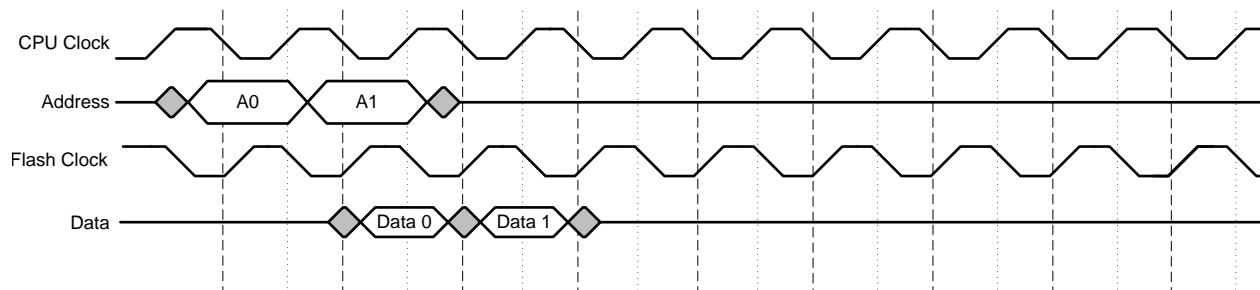


Figure 7-6. Single Cycle Access, 0 Wait States

The flash memory can operate at the CPU clock speed with zero-wait-state accesses when data is resident in the prefetch buffers. When an access does not hit in the prefetch buffer, there is a delay that is incurred while the data is transferred from the flash. This delay is dependent on the programmed CPU frequency. See Table 7-1 for required CPU frequency versus programmed wait-state delay information. Figure 7-7 shows the events that occur as the CPU steps through the words in the prefetch buffer that has just been loaded until it reaches the end of the current prefetch line. The notable events follow (see Figure 7-7):

- **EVENT A:** When the CPU has a miss in the prefetch buffer, a line is fetched from flash. The target word is written to the prefetch buffer and sent to the CPU one cycle after.
- **EVENT B:** When the CPU reaches Word 3, the next 256-bit buffer line is fetched, resulting in a zero-wait-state access of word 0 of the next line.
- **EVENT C:** After this word, if the CPU is still executing sequentially, word 0 of the next buffer line that was fetched is sent to the CPU with zero-wait-state delay.
- **EVENT D:** Word 0 from the second fetch that occurred is sent to the CPU.

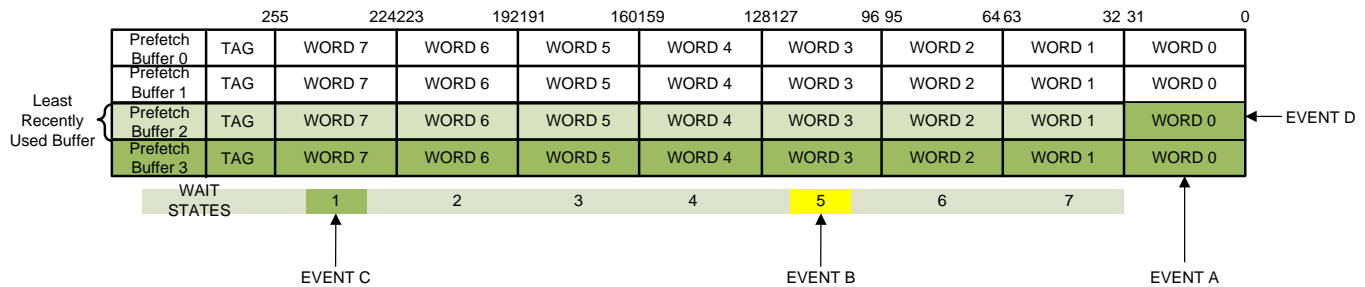


Figure 7-7. Prefetch Fills From Flash

If the CPU target word is beyond word 2 (word 3 through word 7) then the next prefetch fill begins immediately, and, depending on the CPU frequency, a delay is incurred between CPU access of word 7 and word 0 of the next line.

NOTE: For optimal prefetch buffer performance, align application code and branches on 8-word boundaries.

NOTE: Because the prefetch buffers and flash memory can effectively be used at 20 MHz and higher, an application can have an improvement in current consumption from 16 MHz to 20 MHz.

The prefetch buffers can be forced ON and OFF by setting the FPFON and FPFOFF bits in the Flash Configuration (FLASHCONF) register at 0xFC8. If the application sets the FPFON or FPFOFF bit while the CPU is currently reading or writing to flash, the prefetch buffer action of turning on or off happens only after the flash operation has completed. This feature can be used in test modes when determining optimum memory configuration for code.

Prefetch buffer valid tags can be cleared in the following ways:

- Any Flash Configuration (FLASHCONF) register changes, such as:
 - Disabling the prefetch buffer by setting the FPFOFF bit
 - Setting the CLRTV bit to clear the prefetch buffer tags
- A system reset
- ROM accesses
- Error during ICODE accesses
- System aborts
- Mirror mode changes

NOTE: If the prefetch buffers are enabled and application code branches to a location other than flash memory which then modifies the flash memory, the prefetch tags must be cleared before returning to flash code execution. Prefetch buffer valid tags can be cleared by setting the CLRTV bit in the FLASHCONF register.

7.2.3.3 Flash Mirror Mode

Flash mirroring allows multiple copies of software to exist in flash simultaneously. The software can run from the lower banks at the same time software is updating a mirrored copy on the upper bank. In addition to the data, the bootloader in both the lower and upper banks must be mirrored while programming the flash contents. If data needs to be recovered, a hot swap can be done by setting the FMME bit in the FLASHCONF register to ensure the flash banks are idle during the swap. The prefetch buffers must be invalidated during the execution of a hot swap. Next, the address translation logic decodes up to 512KB from the upper banks to the lower banks. Once the banks are swapped, the mirrored flash image is then used. The address translation logic translates the address to the upper banks until the next swap. [Figure 7-8](#) depicts the necessary configuration when executing flash mirroring.

NOTE: After a mirror mode has been executed and the code locations have been swapped from the upper memory banks to the lower memory banks, the application can continue to read from the lower memory bank address locations. However, when erasing or programming the swapped memory, the application must use the real upper memory address of the code before it was swapped. For example, in [Figure 7-8](#), when the yellow highlighted location 0x00.3FE8 is swapped with 0x08.3FE8, the next read location of the application is 0x00.3FEC. However, if the application were to program or erase the next location, it would need to write or erase location 0x08.3FEC

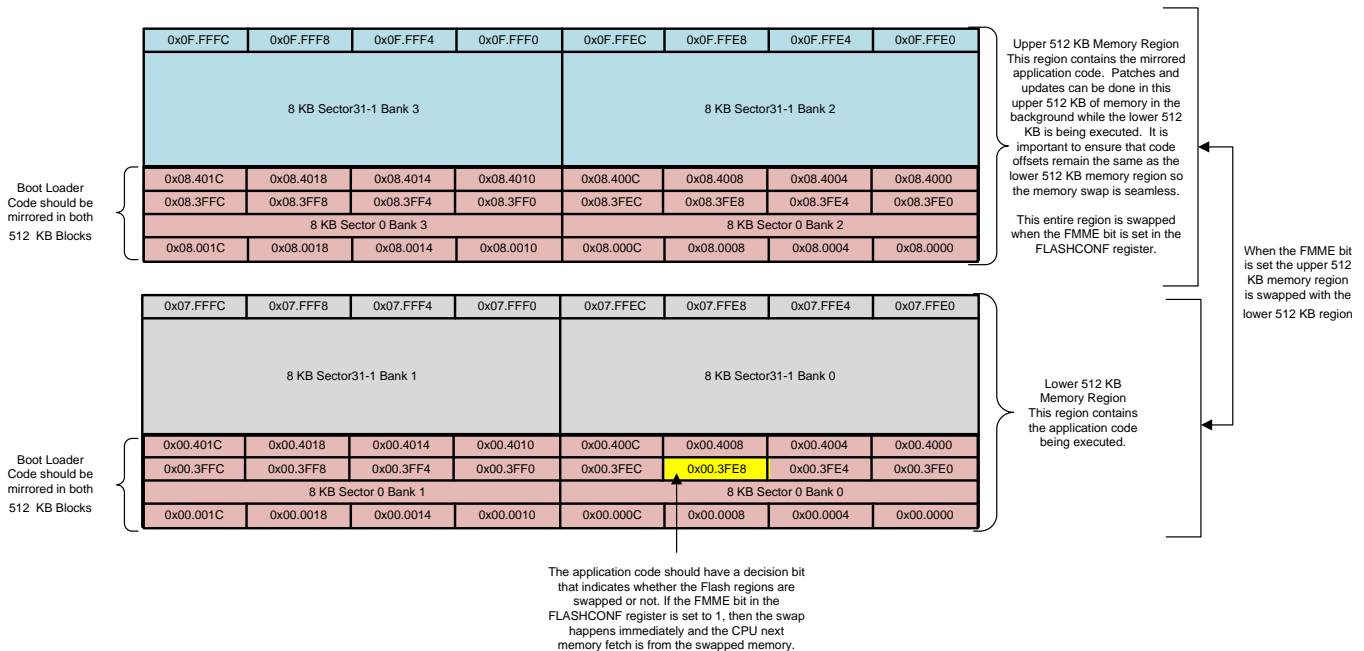


Figure 7-8. Mirror Mode Function

7.2.3.4 Protected Flash Memory Registers

The user is provided execution protection through 16 pairs of 32-bit wide registers. The policy for each protection form is controlled by individual bits (per policy per block) in the FMPPEn and FMPREN registers.

- Flash Memory Protection Program Enable (FMPPEn) : In the flash, 16KB blocks can be individually protected from being programmed or erased. Because each bit of the FMPPE register represents a 2KB block, the application must clear all the bits in one byte to protect one 16KB block. Execute-only protection can only be programmed in 16KB increments. For example, to protect the first 16KB block, bits [7:0] must be set to 0s. When bits in the FMPPEn register are set, the corresponding block can be programmed (written) or erased. When bits are cleared, the corresponding block can not be changed. When a block is protected by clearing bits in both FMPPEn and FMPREN registers, execute-only protection can be achieved.

- Flash Memory Protection Read Enable (FMPREn) : If a bit is set in this register, the corresponding block can be executed or read by software or debuggers. If a bits in this register are cleared and the same block in the FMPREn register is cleared, the corresponding block can only be executed, and contents of the memory block are prohibited from being read as data. FMPREn protection can be programmed in 2KB increments, unlike the FMPPEn, which must be programmed in 16KB increments. However, if an application does read-protect a 16KB block, eight bits must be written from 1 to 0.

The policies can be combined as listed in [Table 7-2](#).

Table 7-2. Flash Memory Protection Policy Combinations

FMPPEn	FMPREn	Protection
0	0	Execute-only protection. The block can only be executed and cannot be written or erased. This mode is used to protect code.
1	0	The block can be written, erased, or executed, but cannot be read. This combination is unlikely to be used.
0	1	Read-only protection. The block can be read or executed but cannot be written or erased. This mode is used to lock the block from further modification while allowing any read or execute access.
1	1	No protection. The block can be written, erased, executed, or read.

A flash memory access that attempts to read a read-protected block (FMPREn bit is clear) is prohibited and generates a bus fault. A flash memory access that attempts to program or erase a program-protected block (FMPPEn bit is clear) is prohibited and can optionally generate an interrupt (by setting the AMASK bit in the Flash Controller Interrupt Mask (FCIM) register) to alert software developers of poorly behaving software during the development and debug phases.

The factory settings for the FMPREn and FMPPEn registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits can be changed by clearing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from 1 to 0 and not committed, it can be restored by executing a simulated power-on-reset (SIM_POR) event. The changes are committed using the Flash Memory Control (FMC) register. For details on programming these bits, see [Section 7.2.3.12](#).

7.2.3.5 Execute-Only Protection

An application designer can use the execute-only protection to help protect the integrity and confidentiality of software stored in these protected regions, helping prevent, for example, unauthorized software updates or software reverse-engineering. If, however, the attacker has debug access or has malicious code running, the confidentiality of the code in the execute-only region during runtime could be compromised, for example by observing the effect of each instruction on CPU registers. Therefore, an application designer should consider implementing additional security measures such as debug lock, carefully written software update mechanisms, use of the memory protection unit (MPU), and other appropriate security measures to protect valuable software.

Literal data introduces a complication to the protection mechanism. When C code is compiled and linked, literal data (constants, and so on) is typically placed in the text section, between functions, by the compiler. The literal data is accessed at run time through the use of the LDR instruction, which loads the data from memory using a PC-relative memory address. The execution of the LDR instruction generates a read transaction across the DCode bus of the Cortex-M4, which is subject to the execute-only protection mechanism. If the accessed block is marked as execute only, the transaction is blocked, and the processor is prevented from loading the constant data and, therefore, inhibiting correct execution. Therefore, using execute-only protection requires that literal data be handled differently. There are three ways to address this:

1. Use a compiler that allows literal data to be collected into a separate section that is put into one or more read-enabled flash blocks. Note that the LDR instruction can use a PC-relative address, in which case the literal pool cannot be located outside the span of the offset, or the software can reserve a register to point to the base address of the literal pool and the LDR offset is relative to the beginning of the pool.
2. Use a compiler that generates literal data from arithmetic instruction immediate data and subsequent computation.

3. Use method 1 or 2 (in assembly language) if the compiler does not support either method.

7.2.3.6 Read-Only Protection

Read-only protection prevents the contents of the flash block from being reprogrammed, while still allowing the content to be read by processor or the debug interface. If a FMPREn bit is cleared, all read accesses to the flash memory block are disallowed, including any data accesses. Care must be taken not to store required data in a flash memory block that has the associated FMPREN bit cleared.

The read-only mode does not prevent read access to the stored program, but it does provide protection against accidental (or malicious) erasure or programming. Read-only is especially useful for utilities like the bootloader when the debug interface is permanently disabled. In such combinations, the bootloader, which provides access control to the flash memory, is protected from being erased or modified.

7.2.3.7 Permanently Disabling Debug

For sensitive applications, the debug interface to the processor and peripherals can be permanently disabled, blocking all accesses to the device through the JTAG or SWD interfaces. With the debug interface disabled, it is still possible to perform standard IEEE instructions (such as boundary scan operations), but access to the processor and peripherals is blocked.

The DBG0 and DBG1 bits of the Boot Configuration (BOOTCFG) register control whether the debug interface is turned on or off.

The debug interface should not be permanently disabled without providing some mechanism, such as the bootloader, to provide customer-installable updates or bug fixes. Disabling the debug interface is permanent and cannot be reversed.

7.2.3.8 Interrupts

The flash memory controller can generate interrupts when the following conditions are observed:

- Programming Interrupt: Signals when a program or erase action is complete (PRIS).
- Access Interrupt: Signals when a program or erase action has been attempted on a 16-kB block of memory that is protected by its corresponding FMPPEn bit (ARIS).
- EEPROM Interrupt
- Pump Voltage Interrupt: Indicates if the regulated voltage of the pump went out of specification during a flash operation and the operation was terminated (VOLTRIS).
- Invalid Data Interrupt: Signals when a bit in flash that was previously programmed as a 0 is now requested to be programmed as a 1 (INVDRIS).
- ERASE Operation Interrupt: Indicates an ERASE operation failed (ERRIS).

The interrupt events that can trigger a controller-level interrupt are defined in the Flash Controller Masked Interrupt Status (FCMIS) register (see [Section 7.3.5](#)) by setting the corresponding MASK bits. If interrupts are not used, the raw interrupt status is always visible through the Flash Controller Raw Interrupt Status (FCRIS) register (see [Section 7.3.4](#)).

Interrupts are always cleared (for both the FCMIS and FCRIS registers) by writing 1 to the corresponding bit in the Flash Controller Masked Interrupt Status and Clear (FCMISC) register (see [Section 7.3.6](#)).

7.2.3.9 μ DMA

The μ DMA can be programmed to read from flash. The Flash DMA Address Size (FLASHDMASZ) register configures 2KB regions of flash that can be accessed by the μ DMA. The starting address for this μ DMA-accessible region is defined in the Flash DMA Starting Address (FLASHDMAST) register. When the DFA bit is set in the FLASHPP register, the μ DMA can access the enabled region configured by the FLASHDMASZ and FLASHDMAST registers. The μ DMA checks the Flash Protection Program Enable n (FMPPEn) registers for masked 2KB flash regions before initiating the transfer. If the access is out of range, then a bus fault is generated.

NOTE: The μ DMA can only access flash in run mode (not available in low power modes).

7.2.3.10 Flash Memory Programming

The MSP432E4 devices provide a user-friendly interface for flash memory programming. All erase/program operations are handled by three registers: Flash Memory Address (FMA), Flash Memory Data (FMD), and Flash Memory Control (FMC). If the debug capabilities of the microcontroller have been deactivated, resulting in a locked state, a recovery sequence must be performed in order to reactivate the debug module (see [Section 3.3.4.3](#)).

When a flash memory operation write, page erase, or mass erase is executed in a flash bank, access to that particular bank pair is inhibited. As a result, instruction and literal fetches to the bank pair are held off until the flash memory operation is complete. If instruction execution is required during a flash memory operation, the code that is executing must be placed in SRAM and executed from there while the flash operation is in progress.

NOTE: When programming flash memory, the following characteristics of the memory must be considered:

- Only an erase can change bits from 0 to 1.
 - A write can only change bits from 1 to 0. If the write attempts to change a 0 to a 1, the write fails and no bits are changed.
 - All flash operations are completed before entering sleep or deep-sleep mode.
-

7.2.3.10.1 To Program a 32-Bit Word

1. Write source data to the FMD register.
2. Write the target address to the FMA register.
3. Write the flash memory write key and the WRITE bit (a value of 0xA442.0001) to the FMC register. The write key can be 0xA442 or the value programmed into the FLPEKEY register depending on the KEY value in the BOOTCFG register. See [Section 7.5.2](#) and [Section 7.3.9](#) for more information.
4. Poll the FMC register until the WRITE bit is cleared.

7.2.3.10.2 To Perform an Erase of a 16KB Sector

1. Write the 16KB aligned address to the FMA register.
2. Write the flash memory write key and the ERASE bit to the FMC register.
3. Poll the FMC register until the ERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the FCIM register.

7.2.3.10.3 To Perform a Mass Erase of the Flash Memory

1. Write the flash memory write key and the MERASE bit to the FMC register.
2. Poll the FMC register until the MERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the FCIM register.

7.2.3.11 32-Word Flash Memory Write Buffer

A 32-word write buffer provides the capability to perform faster write accesses to the flash memory by programming two 32-bit words at a time, allowing 32 words to be programmed in the same time as 16 would take using the method described above. The data for the buffered write is written to the Flash Write Buffer (FWBn) registers.

The registers are 32-word aligned with flash memory, and therefore the register FWB0 corresponds with the address in FMA where bits [6:0] of FMA are all 0. FWB1 corresponds with the address in FMA + 0x4 and so on. Only the FWBn registers that have been updated since the previous buffered flash memory write operation are written. The Flash Write Buffer Valid (FWBVAL) register shows which registers have been written since the last buffered flash memory write operation. This register contains a bit for each of the 32 FWBn registers, where bit[n] of FWBVAL corresponds to FWBn. The FWBn register has been updated if the corresponding bit in the FWBVAL register is set.

7.2.3.11.1 To Program 32 Words With a Single Buffered Flash Memory Write Operation

1. Write the source data to the FWBn registers.
2. Write the target address to the FMA register. This must be a 32-word aligned address (that is, bits [6:0] in FMA must be 0s).
3. Write the flash memory write key and the WRBUF bit to the FMC2 register.
4. Poll the FMC2 register until the WRBUF bit is cleared or wait for the PMIS interrupt to be signaled.

7.2.3.12 Nonvolatile Register Programming – Flash Memory Resident Registers

NOTE: The Boot Configuration (BOOTCFG) register requires a POR before the committed changes take effect.

This section details how to update the registers in [Table 7-3](#), which are resident within the flash memory. These registers exist in a separate space from the main flash memory array and are not affected by an ERASE or MASS ERASE operation. The bits in these registers can be changed from 1 to 0 with a commit operation. The register contents are unaffected by any reset condition except power-on reset, which returns the register contents to 0xFFFF.FFFE for the BOOT Configuration (BOOTCFG) register and 0xFFFF.FFFF for all others.

NOTE: Do not change the values of the reserved bits in the BOOTCFG register. The reserved bits must remain at their default value of all 1s.

By committing the register values using the COMT bit in the Flash Memory Control (FMC) register, the register contents become nonvolatile and are therefore retained following power cycling. After the register contents are committed, the only way to restore the factory default values is to perform the sequence described in [Section 3.3.4.3](#).

All of the FMPREN, FMPPEN and USER_REGn registers, in addition to the BOOTCFG register, can be committed in nonvolatile memory. The FMPREN, FMPPEN, and USER_REGn registers can be tested before being committed; the BOOTCFG register cannot. To program the BOOTCFG register, the value must be written into the Flash Memory Data (FMD) register before it is committed. The BOOTCFG configuration cannot be tried and verified before committing to nonvolatile memory.

NOTE: All flash memory resident registers can only have bits changed from 1 to 0 by user programming. The FMPREN, FMPPEN, and BOOTCFG registers can be committed multiple times, but the USER_REGn registers can only be committed once, after the entire register has been set to 1s. After being committed, the USER_REGn registers can only be returned to their factory default values of all 1s by performing the sequence described in [Section 3.3.4.3](#). The mass erase of the main flash memory array caused by the sequence is performed before restoring these registers.

[Table 7-3](#) lists the FMA address required for commitment of each of the registers and the source of the data to be written when the FMC register is written with a key value of 0xA442 or the PEKEY value of the FLPEKEY register. The key value used is determined by the KEY bit in the BOOTCFG register at reset. If the KEY value is 0x0, the PEKEY value in the FLPEKEY register is used for commits in the FMC/FMC2 register. If the KEY value is 0x1, the value 0xA442 is used as the WRKEY in the FMC and FMC2 registers. If the After writing the COMT bit, the user can poll the FMC register to wait for the commit operation to complete.

NOTE: To ensure nonvolatile register data integrity, nonvolatile register commits should not be interrupted with a power loss. If data integrity is compromised during a commit because of a power loss, a toggle mass erase function can be performed to clear these registers. See [Table 7-3](#) for the list of nonvolatile registers.

Table 7-3. User-Programmable Flash Memory Resident Registers

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPRE2	0x0000.0004	FMPRE2
FMPRE3	0x0000.0006	FMPRE3
FMPRE4	0x0000.0008	FMPRE4
FMPRE5	0x0000.000A	FMPRE5
FMPRE6	0x0000.000C	FMPRE6
FMPRE7	0x0000.000E	FMPRE7
FMPRE8	0x0000.0010	FMPRE8
FMPRE9	0x0000.0012	FMPRE9
FMPRE10	0x0000.0014	FMPRE10
FMPRE11	0x0000.0016	FMPRE11
FMPRE12	0x0000.0018	FMPRE12
FMPRE13	0x0000.001A	FMPRE13
FMPRE14	0x0000.001C	FMPRE14
FMPRE15	0x0000.001E	FMPRE15
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
FMPPE4	0x0000.0009	FMPPE4
FMPPE5	0x0000.000B	FMPPE5
FMPPE6	0x0000.000D	FMPPE6
FMPPE7	0x0000.000F	FMPPE7
FMPPE8	0x0000.00011	FMPPE8
FMPPE9	0x0000.00013	FMPPE9
FMPPE10	0x0000.00015	FMPPE10
FMPPE11	0x0000.00017	FMPPE11
FMPPE12	0x0000.00019	FMPPE12
FMPPE13	0x0000.0001B	FMPPE13
FMPPE14	0x0000.0001D	FMPPE14
FMPPE15	0x0000.0001F	FMPPE15
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_REG2	0x8000.0002	USER_REG2
USER_REG3	0x8000.0003	USER_REG3
BOOTCFG	0x7510.0000	FMD

7.2.4 EEPROM

The EEPROM includes the following features:

- 6K bytes of memory accessible as 1536 32-bit words
- 96 blocks of 16 words (64 bytes) each
- Built-in wear leveling
- Access protection per block
- Lock protection option for the whole peripheral as well as per block using 32-bit to 96-bit unlock codes

(application selectable)

- Interrupt support for write completion to avoid polling
- Endurance of 500-K writes (when writing at fixed offset in every alternate page in circular fashion) to 15M operations (when cycling through two pages) per each two-page block.

7.2.4.1 Functional Description

The EEPROM module provides a well-defined register interface to support accesses to the EEPROM with both a random access style of read and write as well as a rolling or sequential access scheme.

A protection mechanism allows locking EEPROM blocks to prevent writes under a set of circumstances as well as reads under the same or different circumstances. The password model allows the application to lock one or more EEPROM blocks to control access on 16-word boundaries.

7.2.4.1.1 Blocks

There are 96 blocks of 16 words each in the EEPROM. These are readable and writable as words. Bytes and half-words can be read, and these accesses do not have to occur on a word boundary. The entire word is read and any unneeded data is simply ignored. The EEPROM blocks are writable only on a word basis. To write a byte, it is necessary to read the word value, modify the appropriate byte, and write the word back.

Each block is addressable as an offset within the EEPROM, using a block select register. Each word is offset addressable within the selected block.

The current block is selected by the EEPROM Current Block (EEBLOCK) register. The current offset is selected and checked for validity by the EEPROM Current Offset (EEOFFSET) register. The application can write the EEOFFSET register any time, and it is also automatically incremented when the EEPROM Read-Write with Increment (EERDWRINC) register is accessed. However, the EERDWRINC register does not increment the block number, but instead wraps within the block.

Blocks are individually protectable. Attempts to read from a block for which the application does not have permission return 0xFFFF.FFFF. Attempts to write into a block for which the application does not have permission results in an error in the EEPROM Done Status (EEDONE) register.

7.2.4.1.2 Timing Considerations

After enabling or resetting the EEPROM module, software must wait until the WORKING bit in the EEDONE register is clear before accessing any EEPROM registers.

NOTE: Software must ensure there are no flash memory writes or erases pending before performing an EEPROM operation. When the FMC register reads as 0x0000.00000 and the WRBUF bit of the FMC2 register is clear, there are no flash memory writes or erases pending.

EEPROM operations must be completed before entering Sleep or Deep-Sleep mode. Ensure the EEPROM operations have completed by checking the EEPROM Done Status (EEDONE) register before issuing a WFI instruction to enter Sleep or Deep-Sleep.

Writes to words within a block are delayed by a variable amount of time. The application can use an interrupt to be notified when the write is done, or alternatively poll for the done status in the EEDONE register. The variability ranges from the write timing of the EEPROM to the erase timing of EEPROM, where the erase timing is less than the write timing of most external EEPROMs.

Depending on the CPU frequency, the application must program the EEPROM Clock High Time (EBCHT), EEPROM Bank Clock Edge (EBCE) and the EEPROM Wait States (EWS) in the Memory Timing Parameter Register 0 for main flash and EEPROM (MENTIM0) register at System Control Module offset 0x0C0.

Table 7-4. MEMENTIM0 Register Configuration and Frequency

CPU Frequency range (f) in MHz	Time Period Range (t) in ns	EEPROM Bank Clock High Time (EBCHT)	EEPROM Bank Clock Edge (EBCE)	EEPROM Wait States (EWS)
16	62.5	0x0	1	0x0
16 < f ≤ 40	62.5 > t ≥ 25	0x2	0	0x1
40 < f ≤ 60	25 > t ≥ 16.67	0x3	0	0x2
60 < f ≤ 80	16.67 > t ≥ 12.5	0x4	0	0x3
80 < f ≤ 100	12.5 > t ≥ 10	0x5	0	0x4
100 < f ≤ 120	10 > t ≥ 8.33	0x6	0	0x5

NOTE: The associated flash and EEPROM fields in the MEMENTIM0 register must be programmed to the same values. For example, the FWS field must be programmed to the same value as the EWS field.

7.2.4.1.3 Locking and Passwords

The EEPROM can be locked at both the module level and the block level. The lock is controlled by a password that is stored in the EEPROM Password (EEPASSn) registers and can be any 32-bit to 96-bit value other than all 1s. Block 0 is the master block, the password for block 0 protects the control registers as well as all other blocks. Each block can be further protected with a password for that block.

If a password is registered for block 0, then the whole module is locked at reset. As a result, the EEBLOCK register cannot be changed from 0 until block 0 is unlocked.

A password registered with any block, including block 0, allows for protection rules that control access of that block based on whether it is locked or unlocked. Generally, the lock can be used to prevent write accesses when locked or can prevent read and write accesses when locked.

All password protected blocks are locked at reset. To unlock a block, the correct password value must be written to the EEPROM Unlock (EEUNLOCK) register by writing to it once, twice, or three times, depending on the size of the password. A block or the module can be re-locked by writing 0xFFFF.FFFF to the EEUNLOCK register because 0xFFFF.FFFF is not a valid password.

7.2.4.1.4 Protection and Access Control

The PROT protection field in the EEPROM Protection (EEPROT) register provides discrete control of read and write access for each block which allows various protection models per block. The protection configurations allowed are as follows:

- PROT = 0x0
 - Without password: Readable and writable at any time. This mode is the default when there is no password.
 - With password: Readable, but only writable when unlocked by the password. This mode is the default when there is a password.
- PROT = 0x1
 - With password: Readable or writable only when unlocked.
 - This value has no meaning when there is no password.
- PROT = 0x2
 - Without password: Readable but not writable.
 - With password: Readable only when unlocked, not writable under any conditions.

Additionally, access protection can be applied based on the processor mode. This configuration allows for supervisor-only access or supervisor and user access, which is the default. Supervisor-only access mode also prevents access by the μ DMA and Debugger.

Additionally, the master block can be used to control access protection for the protection mechanism itself. If access control for block 0 is for supervisor only, then the whole module can only be accessed in supervisor mode.

7.2.4.1.5 Hidden Blocks

Hiding provides a temporary form of protection. Every block except block 0 can be hidden, which prevents all accesses until the next reset.

This mechanism can allow a boot or initialization routine to access some data which is then made inaccessible to all further accesses. Because boot and initialization routines control the capabilities of the application, hidden blocks provide a powerful isolation of the data when debug is disabled.

A typical use model would be to have the initialization code store passwords, keys, and/or hashes to use for verification of the rest of the application. Once performed, the block is then hidden and made inaccessible until the next reset which then re-enters the initialization code.

7.2.4.1.6 Power and Reset Safety

Once the EEDONE register indicates that a location has been successfully written, the data is retained until that location is written again. There is no power or reset race after the EEDONE register indicates a write has completed.

7.2.4.1.7 Interrupt Control

The EEPROM module allows for an interrupt when a write completes to prevent the use of polling. The interrupt can be used to drive an application ISR which can then write more words or verify completion. The interrupt mechanism is used any time the EEDONE register goes from working to done, whether because of an error or the successful completion of a program or erase operation. This interrupt mechanism works for data writes, writes to password and protection registers, and mass erase using the EEPROM Debug Mass Erase (EEDGBME) register. The EEPROM interrupt is signaled to the core using the flash memory interrupt vector. Software can determine that the source of the interrupt was the EEPROM by examining bit 2 of the Flash Controller Masked Interrupt Status and Clear (FCMISC) register.

7.2.4.1.8 Theory of Operation

The EEPROM operates using a traditional bank model which implements EEPROM-type cells, but uses sector erase. Additionally, words are replicated in the blocks to allow 500K or more erase cycles when needed, which means that each word has a latest version. As a result, a write creates a new version of the word in a new location, making the previous value obsolete. When a block runs out of room to store the latest version of a word, a copy buffer is used. The copy buffer copies the latest words of each block. The original block is then erased. Finally, the copy buffer contents are copied back to the block.

The EEPROM module includes functionality to prevent data corruption due to power-loss or a brown-out event during programming or erase operations. These conditions prevent corruption of non-targeted memory areas but cannot guarantee that the operation is completed successfully. See for important timing information on EEPROM protection. The EEPROM mechanism properly tracks all state information to provide complete safety and protection. Although it should not normally be possible, errors during programming can occur in certain circumstances, for example, the voltage rail dropping during programming. In these cases, the EESUPP register can be used to know if a program or an erase had failed.

7.2.4.1.9 Debug Mass Erase

The EEPROM debug mass erase allows the developer to mass erase the EEPROM. For the mass erase to occur correctly, there can be no active EEPROM operations. After the last EEPROM operation, the application must ensure that no EEPROM registers are updated, including modifying the EEBLOCK and the EEOFFSET registers without doing an actual read or write operation. To hold off these operations, the application should reset the EEPROM module by setting the R0 bit in the EEPROM Software Reset (SREEPROM) register, wait until WORKING bit in the EEPROM Done Status (EEDONE) register is clear, and then enable the debug mass erase by setting the ME bit in the EEPROM Debug Mass Erase (EEDGBME) register.

7.2.4.1.10 Error During Programming

Operations such as data-write, password set, protection set, and copy buffer erase can perform multiple operations. For example, a normal write performs two underlying writes: the control word write and the data write. If the control word writes but the data fails (for example, due to a voltage drop), the overall write fails with indication provided in the EEDONE register. Failure and the corrective action is broken down by the type of operation:

- If a normal write fails such that the control word is written but the data fails to write, the safe course of action is to retry the operation once the system is otherwise stable, for example, when the voltage is stabilized. After the retry, the control word and write data are advanced to the next location.
- If a password or protection write fails, the safe course of action is to retry the operation once the system is otherwise stable. In the event that multi-word passwords can be written outside of a manufacturing or bring-up mode, care must be taken to ensure all words are written in immediate succession. If not, then partial password unlock would need to be supported to recover.
- If the word write requires the block to be written to the copy buffer, then it is possible to fail or lose power during the subsequent operations. A control word mechanism is used to track what step the EEPROM was in if a failure occurs. If not completed, the EESUPP register indicates the partial completion.

After a reset and prior to writing any data to the EEPROM, software must read the EESUPP register and check for the presence of any error condition which can indicate that a write or erase was in progress when the system was reset due to a voltage drop. If either the PRETRY or ERETRY bits are set, the peripheral should be reset by setting and then clearing the R0 bit in the EEPROM Software Reset (SREEPROM) register and waiting for the WORKING bit in the EEDONE register to clear before again checking the EESUPP register for error indicators. This procedure should allow the EEPROM to recover from the write or erase error. In very isolated cases, the EESUPP register can continue to register an error after this operation, in which case the reset should be repeated. After recovery, the application should rewrite the data which was being programmed when the initial failure occurred.

7.2.4.1.11 Soft Reset Handling

The following soft resets should not be asserted during an EEPROM program or erase operation:

- Software reset (SYSRESREQ)
- Software peripheral reset
- Watchdog reset (if configured as a system reset in the RESBEHAVCTL register)
- MOSC failure reset
- BOR reset (if configured as a system reset in the RESBEHAVCTL register)
- External reset (if configured as a system reset in the RESBEHAVCTL register)
- Writes to the HSSR register

The WORKING bit of the EEDONE register can be checked before the reset is asserted to see if an EEPROM program or erase operation is occurring. Soft resets can occur when using a debugger and should be avoided during an EEPROM operation. A reset such as the Watchdog reset can be mapped to an external reset using a GPIO, or Hibernate can be entered, if time is not a concern.

7.2.4.1.12 Endurance

Endurance is per meta-block, which is 8 blocks. Endurance is measured in two ways:

1. To the application, it is the number of writes that can be performed.
2. To the microcontroller, it is the number of erases that can be performed on the meta-block.

Because of the second measure, the number of writes depends on how the writes are performed. For example:

- One word can be written more than 500000 times, but, these writes affect the meta-block that the word is within. As a result, writing one word 500000 times, then trying to write a nearby word 500000 times is not assured to work. To ensure success, the words should be written more in parallel.
- All words can be written in a sweep with a total of more than 500000 sweeps which updates all words more than 500000 times.

- Different words can be written such that any or all words can be written more than 500000 times when write counts per word stay about the same. For example, offset 0 could be written three times, then offset 1 could be written two times, then offset 2 is written four times, then offset 1 is written twice, then offset 0 is written again. As a result, all three offsets have four writes at the end of the sequence. This kind of balancing within seven writes maximizes the endurance of different words within the same meta-block.

7.2.4.2 EEPROM Initialization and Configuration

Before any writes to EEPROM registers, enable the clock to the EEPROM module with the EEPROM Run Mode Clock Gating Control (RCGCEEPROM) register (see [Section 4.2.101](#)) and execute the following initialization steps:

1. Insert a delay of 6 cycles plus function call overhead.
2. Poll the WORKING bit in the EEPROM Done Status (EEDONE) register until it is clear, indicating that the EEPROM has completed its power-on initialization. When WORKING = 0, continue.
3. Read the PRETRY and ERETRY bits in the EEPROM Support Control and Status (EESUPP) register. If either of the bits are set, return an error, else continue.
4. Reset the EEPROM module using the EEPROM Software Reset (SREEPROM) register at offset 0x558 in the System Control register space.
5. Insert a delay of six cycles plus function call overhead.
6. Poll the WORKING bit in the EEPROM Done Status (EEDONE) register to determine when it is clear. When WORKING = 0, continue.
7. Read the PRETRY and ERETRY bits in the EESUPP register. If either of the bits are set, return an error, else the EEPROM initialization is complete and software can use the peripheral as normal.

NOTE: Failure to perform these initialization steps after a reset can lead to incorrect operation or permanent data loss if the EEPROM is later written.

If the PRETRY or ERETRY bits are set in the ESUPP register, the EEPROM was unable to recover its state. If power is stable when this occurs, this indicates a fatal error and is likely an indication that the EEPROM memory has exceeded its specified lifetime write and erase specification. If the supply voltage is unstable when this return code is observed, retrying the operation once the voltage is stabilized can clear the error.

The EEPROM initialization function code is named EEPROMInit() in the SimpleLink SDK.

7.2.5 Bus Matrix Memory Accesses

[Table 7-5](#) identifies the bus masters and their access to the various memories on the bus matrix.

Table 7-5. Master Memory Access Availability

Master	Flash Access	ROM Access	SRAM Access	EEPROM Access	External Memory Access (Through EPI)
CPU Instruction Bus	Yes	Yes (read only)	Yes	Yes	Yes
CPU Data Bus	Yes	Yes (read only)	–	Yes	Yes
μDMA	Yes (read only, run-mode only)	–	Yes	Yes	Yes
Ethernet Module	–	–	Yes	–	–
USB	–	–	Yes	–	–
LCD	–	–	Yes	–	Yes

7.3 Flash Registers

[Table 7-6](#) lists the memory-mapped registers for the FLASH. All register offset addresses not listed in [Table 7-6](#) should be considered as reserved locations and the register contents should not be modified. Registers in this section are relative to the memory control base address of 0x400FD000.

Table 7-6. Flash Registers

Offset	Acronym	Register Name	Section
0x0	FMA	Flash Memory Address	Section 7.3.1
0x4	FMD	Flash Memory Data	Section 7.3.2
0x8	FMC	Flash Memory Control	Section 7.3.3
0xC	FCRIS	Flash Controller Raw Interrupt Status	Section 7.3.4
0x10	FCIM	Flash Controller Interrupt Mask	Section 7.3.5
0x14	FCMISC	Flash Controller Masked Interrupt Status and Clear	Section 7.3.6
0x20	FMC2	Flash Memory Control 2	Section 7.3.7
0x30	FWBVAL	Flash Write Buffer Valid	Section 7.3.8
0x3C	FLPEKEY	Flash Program/Erase Key	Section 7.3.9
0x100 to 0x17C	FWB0 to FWB31	Flash Write Buffer 0 to Flash Write Buffer 32	Section 7.3.10
0xFC0	FLASHPP	Flash Peripheral Properties	Section 7.3.11
0xFC4	SSIZE	SRAM Size	Section 7.3.12
0xFC8	FLASHCONF	Flash Configuration Register	Section 7.3.13
0xFCC	ROMSWMAP	ROM Third-Party Software	Section 7.3.14
0xFD0	FLASHDMASZ	Flash DMA Address Size	Section 7.3.15
0xFD4	FLASHDMAST	Flash DMA Starting Address	Section 7.3.16

Complex bit access types are encoded to fit into small table cells. [Table 7-7](#) shows the codes that are used for access types in this section.

Table 7-7. FLASH Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

7.3.1 FMA Register (Offset = 0x0) [reset = 0x0]

Flash Memory Address (FMA)

During a write operation, this register contains a 4-byte-aligned address and specifies where the data is written. During erase operations for flash space that is not user configurable (that is, FMPREn, FMPPEn, USER_REGn, BOOTCFG), this register contains a 16KB-aligned CPU byte address and specifies which block is erased. Note that the alignment requirements must be met by software or the results of the operation are unpredictable.

FMA is shown in [Figure 7-9](#) and described in [Table 7-8](#).

Return to [Summary Table](#).

Figure 7-9. FMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OFFSET																			
R-0x0												R/W-0x0																			

Table 7-8. FMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19-0	OFFSET	R/W	0x0	Address Offset Address offset in Flash memory where operation is performed, except for non-volatile registers.

7.3.2 FMD Register (Offset = 0x4) [reset = 0x0]

Flash Memory Data (FMD)

This register contains the data to be written during the programming cycle. Note that the contents of this register are undefined for a read access of an execute-only block. This register is not used during erase cycles.

FMD is shown in [Figure 7-10](#) and described in [Table 7-9](#).

Return to [Summary Table](#).

Figure 7-10. FMD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0x0																															

Table 7-9. FMD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0x0	Data Value Data value for write operation.

7.3.3 FMC Register (Offset = 0x8) [reset = 0x0]

Flash Memory Control (FMC)

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the Flash Memory Address (FMA) register (see [Section 7.3.1](#)). If the access is a write access, the data contained in the Flash Memory Data (FMD) register (see [Section 7.3.2](#)) is written to the specified address.

For non-volatile registers, FMPREn, FMPPEn, USER_REGn, and USER_REGn, the respective register is programmed with the value to be written rather than the FMD register.

This register must be the final register written and initiates the memory operation. The four control bits in the lower byte of this register are used to initiate memory operations.

Care must be taken not to set multiple control bits as the results of such an operation are unpredictable.

FMC is shown in [Figure 7-11](#) and described in [Table 7-10](#).

Return to [Summary Table](#).

Figure 7-11. FMC Register

31	30	29	28	27	26	25	24
WRKEY							
W-0x0							
23	22	21	20	19	18	17	16
WRKEY							
W-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				COMT	MERASE	ERASE	WRITE
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 7-10. FMC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	WRKEY	W	0x0	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 or the PEKEY value in the FLPEKEY register must be written into this field for a Flash memory write to occur. The use of 0xA442 or PEKEY is dependent on the value of the KEY bit in the BOOTCFG register at 0x1D0. Writes to the FMC register without this WRKEY value are ignored. A read of this field returns the value 0.
15-4	RESERVED	R	0x0	
3	COMT	R/W	0x0	Commit Register Value This bit is used to commit writes to Flash-memory-resident registers and to monitor the progress of that process. 0x0 = A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous commit access is complete. 0x1 = Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete.

Table 7-10. FMC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	MERASE	R/W	0x0	<p>Mass Erase Flash Memory</p> <p>This bit is used to mass erase the Flash main memory and to monitor the progress of that process.</p> <p>For information on erase time, see the device-specific data sheet.</p> <p>0x0 = A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous mass erase access is complete.</p> <p>0x1 = Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase access is not complete.</p>
1	ERASE	R/W	0x0	<p>Erase a Page of Flash Memory</p> <p>This bit is used to erase a page of Flash memory and to monitor the progress of that process.</p> <p>For information on erase time, see the device-specific data sheet.</p> <p>0x0 = A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous page erase access is complete.</p> <p>0x1 = Set this bit to erase the Flash memory page specified by the contents of the FMA register. When read, a 1 indicates that the previous page erase access is not complete.</p>
0	WRITE	R/W	0x0	<p>Write a Word into Flash Memory</p> <p>This bit is used to write a word into Flash memory and to monitor the progress of that process.</p> <p>For information on programming time, see the device-specific data sheet.</p> <p>0x0 = A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous write update access is complete.</p> <p>0x1 = Set this bit to write the data stored in the FMD register into the Flash memory location specified by the contents of the FMA register. When read, a 1 indicates that the write update access is not complete.</p>

7.3.4 FCRIS Register (Offset = 0xC) [reset = 0x0]

Flash Controller Raw Interrupt Status (FCRIS)

This register indicates that the Flash memory controller has an interrupt condition. An interrupt is sent to the interrupt controller only if the corresponding FCIM register bit is set.

FCRIS is shown in [Figure 7-12](#) and described in [Table 7-11](#).

Return to [Summary Table](#).

Figure 7-12. FCRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		PROGRIS	RESERVED	ERRIS	INVDRIS	VOLTRIS	RESERVED
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED					ERIS	PRIS	ARIS
R-0x0					R-0x0	R-0x0	R-0x0

Table 7-11. FCRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	PROGRIS	R	0x0	Program Verify Error Raw Interrupt Status This bit is cleared by writing a 1 to the PROGMISC bit in the FCMISC register. 0x0 = An interrupt has not occurred. 0x1 = An interrupt is pending because the verify of a PROGRAM operation failed. If this error occurs when using the Flash write buffer, software must inspect the affected words to determine where the error occurred.
12	RESERVED	R	0x0	
11	ERRIS	R	0x0	Erase Verify Error Raw Interrupt Status This bit is cleared by writing a 1 to the ERMISC bit in the FCMISC register. 0x0 = An interrupt has not occurred. 0x1 = An interrupt is pending because the verify of an ERASE operation failed. If this error occurs when using the Flash write buffer, software must inspect the affected words to determine where the error occurred.
10	INVDRIS	R	0x0	Invalid Data Raw Interrupt Status This bit is cleared by writing a 1 to the INVMISC bit in the FCMISC register. 0x0 = An interrupt has not occurred. 0x1 = An interrupt is pending because a bit that was previously programmed as a 0 is now being requested to be programmed as a 1.

Table 7-11. FCRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	VOLTRIS	R	0x0	<p>Pump Voltage Raw Interrupt Status</p> <p>This bit is cleared by writing a 1 to the VOLTMISC bit in the FCMISC register.</p> <p>0x0 = An interrupt has not occurred.</p> <p>0x1 = An interrupt is pending because the regulated voltage of the pump went out of spec during the Flash operation and the operation was terminated.</p>
8-3	RESERVED	R	0x0	
2	ERIS	R	0x0	<p>EEPROM Raw Interrupt Status</p> <p>This bit provides status EEPROM operation.</p> <p>This bit is cleared by writing a 1 to the EMISC bit in the FCMISC register.</p> <p>0x0 = An EEPROM interrupt has not occurred.</p> <p>0x1 = An EEPROM interrupt has occurred.</p>
1	PRIS	R	0x0	<p>Programming Raw Interrupt Status</p> <p>This bit provides status on programming cycles which are write or erase actions generated through the FMC or FMC2 register bits (see and).</p> <p>This status is sent to the interrupt controller when the PMASK bit in the FCIM register is set.</p> <p>This bit is cleared by writing a 1 to the PMISC bit in the FCMISC register.</p> <p>0x0 = The programming or erase cycle has not completed.</p> <p>0x1 = The programming or erase cycle has completed.</p>
0	ARIS	R	0x0	<p>Access Raw Interrupt Status</p> <p>This status is sent to the interrupt controller when the AMASK bit in the FCIM register is set.</p> <p>This bit is cleared by writing a 1 to the AMISC bit in the FCMISC register.</p> <p>0x0 = No access has tried to improperly program or erase the Flash memory.</p> <p>0x1 = A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the FMPPEn registers.</p>

7.3.5 FCIM Register (Offset = 0x10) [reset = 0x0]

Flash Controller Interrupt Mask (FCIM)

This register controls whether the Flash memory controller generates interrupts to the controller.

FCIM is shown in [Figure 7-13](#) and described in [Table 7-12](#).

Return to [Summary Table](#).

Figure 7-13. FCIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		PROGMASK	RESERVED	ERMASK	INVDMASK	VOLTMASK	RESERVED
R-0x0		R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED					EMASK	PMASK	AMASK
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 7-12. FCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	PROGMASK	R/W	0x0	Program Verify Error Interrupt Mask 0x0 = The PROGRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PROGRIS bit is set.
12	RESERVED	R	0x0	
11	ERMASK	R/W	0x0	Erase Verify Error Interrupt Mask 0x0 = The ERRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the ERRIS bit is set.
10	INVDMASK	R/W	0x0	Invalid Data Interrupt Mask 0x0 = The INVDRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the INVDRIS bit is set.
9	VOLTMASK	R/W	0x0	Pump Voltage Interrupt Mask 0x0 = The VOLTRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the VOLTRIS bit is set.
8-3	RESERVED	R	0x0	
2	EMASK	R/W	0x0	EEPROM Interrupt Mask 0x0 = The ERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the ERIS bit is set.

Table 7-12. FCIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	PMASK	R/W	0x0	<p>Programming Interrupt Mask</p> <p>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.</p> <p>0x0 = The PRIS interrupt is suppressed and not sent to the interrupt controller.</p> <p>0x1 = An interrupt is sent to the interrupt controller when the PRIS bit is set.</p>
0	AMASK	R/W	0x0	<p>Access Interrupt Mask</p> <p>This bit controls the reporting of the access raw interrupt status to the interrupt controller.</p> <p>0x0 = The ARIS interrupt is suppressed and not sent to the interrupt controller.</p> <p>0x1 = An interrupt is sent to the interrupt controller when the ARIS bit is set.</p>

7.3.6 FCMISC Register (Offset = 0x14) [reset = 0x0]

Flash Controller Masked Interrupt Status and Clear (FCMISC)

This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

FCMISC is shown in [Figure 7-14](#) and described in [Table 7-13](#).

Return to [Summary Table](#).

Figure 7-14. FCMISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		PROGMISC	RESERVED	ERMISC	INVMISC	VOLTMISC	RESERVED
R-0x0		R/W1C-0x0	R-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED					EMISC	PMISC	AMISC
R-0x0					R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 7-13. FCMISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	PROGMISC	R/W1C	0x0	PROGVER Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that an interrupt has not occurred.A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled.Writing a 1 to this bit clears PROGMISC and also the PROGRIS bit in the FCRIS register (see).
12	RESERVED	R	0x0	
11	ERMISC	R/W1C	0x0	ERVER Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that an interrupt has not occurred.A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled.Writing a 1 to this bit clears ERMISC and also the ERRIS bit in the FCRIS register (see).
10	INVMISC	R/W1C	0x0	Invalid Data Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that an interrupt has not occurred.A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled.Writing a 1 to this bit clears INVMISC and also the INVDRIS bit in the FCRIS register (see).
9	VOLTMISC	R/W1C	0x0	VOLT Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that an interrupt has not occurred.A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled.Writing a 1 to this bit clears VOLTMISC and also the VOLTRIS bit in the FCRIS register (see).
8-3	RESERVED	R	0x0	

Table 7-13. FCMISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	EMISC	R/W1C	0x0	EEPROM Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that an interrupt has not occurred. A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled. Writing a 1 to this bit clears EMISC and also the ERIS bit in the FCRIS register (see).
1	PMISC	R/W1C	0x0	Programming Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears PMISC and also the PRIS bit in the FCRIS register (see).
0	AMISC	R/W1C	0x0	Access Masked Interrupt Status and Clear 0x0 = When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit. 0x1 = When read, a 1 indicates that an unmasked interrupt was signaled because a program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the FMPPEn registers. Writing a 1 to this bit clears AMISC and also the ARIS bit in the FCRIS register (see).

7.3.7 FMC2 Register (Offset = 0x20) [reset = 0x0]

Flash Memory Control 2 (FMC2)

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the Flash Memory Address (FMA) register (see [Section 7.3.1](#)). If the access is a write access, the data contained in the Flash Write Buffer (FWB) registers is written.

This register must be the final register written as it initiates the memory operation.

FMC2 is shown in [Figure 7-15](#) and described in [Table 7-14](#).

Return to [Summary Table](#).

Figure 7-15. FMC2 Register

31	30	29	28	27	26	25	24
WRKEY							
W-0x0							
23	22	21	20	19	18	17	16
WRKEY							
W-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							WRBUF
R-0x0							R/W-0x0

Table 7-14. FMC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	WRKEY	W	0x0	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. There are two options for the WRKEY value: If the KEY value in the BOOTCFG register is 0x1 at reset, the value 0xA442 is used as a key enable to initiate the appropriate access cycle for the location specified by the address in the FMA register. If the KEY value in the BOOTCFG register is 0x0 at reset, the value programmed in the FLPEKEY register is used as a key enable to initiate the appropriate access cycle for the location specified by the address in the FMA register. Writes to the FMC2 register without this WRKEY value are ignored. A read of this field returns the value 0.
15-1	RESERVED	R	0x0	
0	WRBUF	R/W	0x0	Buffered Flash Memory Write This bit is used to start a buffered write to Flash memory. For information on programming time, see . 0x0 = A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous buffered Flash memory write access is complete. 0x1 = Set this bit to write the data stored in the FWBn registers to the location specified by the contents of the FMA register. When read, a 1 indicates that the previous buffered Flash memory write access is not complete.

7.3.8 FWBVAL Register (Offset = 0x30) [reset = 0x0]

Flash Write Buffer Valid (FWBVAL)

This register provides a bitwise status of which FWBn registers have been written by the processor since the last write of the Flash memory write buffer. The entries with a 1 are written on the next write of the Flash memory write buffer. This register is cleared after the write operation by hardware. A protection violation on the write operation also clears this status.

Software can program the same 32 words to various Flash memory locations by setting the FWB[n] bits after they are cleared by the write operation. The next write operation then uses the same data as the previous one. In addition, if a FWBn register change should not be written to Flash memory, software can clear the corresponding FWB[n] bit to preserve the existing data when the next write operation occurs.

FWBVAL is shown in [Figure 7-16](#) and described in [Table 7-15](#).

Return to [Summary Table](#).

Figure 7-16. FWBVAL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FWB[n]																															
R/W-0x0																															

Table 7-15. FWBVAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	FWB[n]	R/W	0x0	Flash Memory Write Buffer Bit 0 corresponds to FWB0, offset 0x100, and bit 31 corresponds to FWB31, offset 0x13C. 0x0 = The corresponding FWBn register has no new data to be written. 0x1 = The corresponding FWBn register has been updated since the last buffer write operation and is ready to be written to Flash memory.

7.3.9 FLPEKEY Register (Offset = 0x3C) [reset = 0xFFFF]

Flash Program/Erase Key (FLPEKEY)

This register provides a mechanism for protection from inadvertent writes to flash by supplying a 16-bit key. If the KEY value in the BOOTCFG register is 0, then this value is used as the 16-bit key in place of 0xA442 in the FMC/FMC2 registers for committed flash writes.

This can be used for cases where a new image is downloaded and the first word of the new image has the 16-bit key value to be used for that product. This 16-bit key is used to allow the write to FMC or FMC2 to take place.

FLPEKEY is shown in [Figure 7-17](#) and described in [Table 7-16](#).

Return to [Summary Table](#).

Figure 7-17. FLPEKEY Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PEKEY															
R-0x0																R-0xFFFF															

Table 7-16. FLPEKEY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	PEKEY	R	0xFFFF	Key Value When a value other than all 1s or all 0s, this 16-bit value is used as the "match" for the upper 16-bits of the register FMC and FMC2 keys.

7.3.10 FWB0 to FWB31 Registers (Offset = 0x100 to 0x17C) [reset = 0x0]

Flash Write Buffer n (FWBn)

These 32 registers hold the contents of the data to be written into the Flash memory on a buffered Flash memory write operation. The offset selects one of the 32-bit registers. Only FWBn registers that have been updated since the preceding buffered Flash memory write operation are written into the Flash memory, so it is not necessary to write the entire bank of registers in order to write 1 or 2 words. The FWBn registers are written into the Flash memory with the FWB0 register corresponding to the address contained in FMA. FWB1 is written to the address FMA +0x4 etc. Note that only data bits that are 0 result in the Flash memory being modified. A data bit that is 1 leaves the content of the Flash memory bit at its previous value.

FWBn is shown in [Figure 7-18](#) and described in [Table 7-17](#).

Return to [Summary Table](#).

Figure 7-18. FWBn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0x0																															

Table 7-17. FWBn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0x0	Data Data to be written into the Flash memory.

7.3.11 FLASHPP Register (Offset = 0xFC0) [reset = 0x701401FF]

Flash Peripheral Properties (FLASHPP)

FLASHPP is shown in [Figure 7-19](#) and described in [Table 7-18](#).

Return to [Summary Table](#).

Figure 7-19. FLASHPP Register

31	30	29	28	27	26	25	24
RESERVED	PFC	FMM	DFA	RESERVED			
R-0x0	R-0x1	R-0x1	R-0x1	R-0x0			
23	22	21	20	19	18	17	16
RESERVED	EESS				MAINSS		
R-0x0	R-0x2				R-0x4		
15	14	13	12	11	10	9	8
SIZE							
R-0x1FF							
7	6	5	4	3	2	1	0
SIZE							
R-0x1FF							

Table 7-18. FLASHPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	
30	PFC	R	0x1	Prefetch Buffer Mode 0x0 = Single set of 2x256-bit buffers used. 0x1 = Two sets of 2x256-bit prefetch buffers are available to use and may be enabled through the FLASHCONF register.
29	FMM	R	0x1	Flash Mirror Mode 0x0 = Mirror Mode not available. 0x1 = Flash Mirror Mode is available to be enabled or disabled by user through FLASHCONF register.
28	DFA	R	0x1	DMA Flash Access uDMA can only access flash in Run Mode (not available in low power modes). 0x0 = DMA cannot be used to access Flash 0x1 = DMA may access the Flash memory range specified by the FLASHDMAST and FLASHDMASZ registers
27-23	RESERVED	R	0x0	
22-19	EESS	R	0x2	EEPROM Sector Size of the physical bank 0x0 = RESERVED 0x1 = 2KB 0x2 = 4KB 0x3 = 8KB
18-16	MAINSS	R	0x4	Flash Sector Size of the physical bank 0x0 = RESERVED 0x1 = 2KB 0x2 = 4KB 0x3 = 8KB 0x4 = 16KB
15-0	SIZE	R	0x1FF	Flash Size Indicates the size of the on-chip Flash memory 0x01FF = 1024KB of Flash

7.3.12 SSIZE Register (Offset = 0xFC4) [reset = 0x3FF]

SRAM Size (SSIZE)

This register indicates the size of the on-chip SRAM.

NOTE: This register should be used to determine the size of the SRAM that is implemented on this microcontroller.

SSIZE is shown in [Figure 7-20](#) and described in [Table 7-19](#).

Return to [Summary Table](#).

Figure 7-20. SSIZE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																SIZE															
R-0x0																R-0x3FF															

Table 7-19. SSIZE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	SIZE	R	0x3FF	SRAM Size Indicates the size of the on-chip SRAM. 0x03FF = 256KB of SRAM

7.3.13 FLASHCONF Register (Offset = 0xFC8) [reset = 0x0]

Flash Configuration Register (FLASHCONF)

The FLASHCONF register allows the user to enable or disable various properties of the Flash. The force bits, FBFON and FBFOFF, can be used to test code performance and execution by turning the prefetch buffers on and subsequently forcing them off.

FLASHCONF is shown in [Figure 7-21](#) and described in [Table 7-20](#).

Return to [Summary Table](#).

Figure 7-21. FLASHCONF Register

31	30	29	28	27	26	25	24
RESERVED	FMME	SPFE	RESERVED				
R-0x0	R/W-0x0	R/W-0x0	R-0x0				
23	22	21	20	19	18	17	16
RESERVED			CLRTV	RESERVED		FPFON	FPFOFF
R-0x0			R/W-0x0	R-0x0		R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							
R-0x0							

Table 7-20. FLASHCONF Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	
30	FMME	R/W	0x0	Flash Mirror Mode Enable 0x0 = Flash mirror mode is disabled. 0x1 = Flash mirror mode feature is enabled. Access to the lower banks is translated to upper.
29	SPFE	R/W	0x0	Single Prefetch Mode Enable 0x0 = A 4x256-bit prefetch buffer is enabled and used. 0x1 = A single 2x256-bit prefetch buffer is enabled and used.
28-21	RESERVED	R	0x0	
20	CLRTV	R/W	0x0	Clear Valid Tags This is a self-clearing bit. 0x0 = No effect. 0x1 = Clear valid tags in the prefetch buffer.
19-18	RESERVED	R	0x0	
17	FPFON	R/W	0x0	Force Prefetch On 0x0 = No effect 0x1 = Force prefetch buffers to be enabled.
16	FPFOFF	R/W	0x0	Force Prefetch Off 0x0 = No effect 0x1 = Force prefetch buffers to be disabled.
15-0	RESERVED	R	0x0	

7.3.14 ROMSWMAP Register (Offset = 0xFCC) [reset = 0x0]

ROM Third-Party Software (ROMSWMAP)

This register indicates the presence of third-party software in the on-chip ROM. ROMSWMAP enables the ROM apertures that are available.

ROMSWMAP is shown in [Figure 7-22](#) and described in [Table 7-21](#).

Return to [Summary Table](#).

Figure 7-22. ROMSWMAP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW7EN		SW6EN		SW5EN		SW4EN		SW3EN		SW2EN		SW1EN		SW0EN	
R-0x0		R-0x0		R-0x0		R-0x0		R-0x0		R-0x0		R-0x0		R-0x0	

Table 7-21. ROMSWMAP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-14	SW7EN	R	0x0	ROM SW Region 7 Availability 0x0 = RESERVED 0x1 = Region available to core
13-12	SW6EN	R	0x0	ROM SW Region 6 Availability 0x0 = RESERVED 0x1 = Region available to core
11-10	SW5EN	R	0x0	ROM SW Region 5 Availability 0x0 = RESERVED 0x1 = Region available to core
9-8	SW4EN	R	0x0	ROM SW Region 4 Availability 0x0 = RESERVED 0x1 = Region available to core
7-6	SW3EN	R	0x0	ROM SW Region 3 Availability 0x0 = RESERVED 0x1 = Region available to core
5-4	SW2EN	R	0x0	ROM SW Region 2 Availability 0x0 = RESERVED 0x1 = Region available to core
3-2	SW1EN	R	0x0	ROM SW Region 1 Availability 0x0 = RESERVED 0x1 = Region available to core
1-0	SW0EN	R	0x0	ROM SW Region 0 Availability 0x0 = RESERVED 0x1 = Region available to core

7.3.15 FLASHDMASZ Register (Offset = 0xFD0) [reset = 0x0]

Flash DMA Address Size (FLASHDMASZ)

The FLASHDMASZ register contains the area of Flash that the μ DMA can access.

NOTE: The μ DMA can access Flash in Run Mode only (not available in low power modes).

FLASHDMASZ is shown in [Figure 7-23](#) and described in [Table 7-22](#).

Return to [Summary Table](#).

Figure 7-23. FLASHDMASZ Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														SIZE																	
R-0x0														R/W-0x0																	

Table 7-22. FLASHDMASZ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17-0	SIZE	R/W	0x0	μ DMA-accessible Memory Size The size of the region addressable by the μ DMA. Note that the DFA bit must be set in the FLASHPP register before this value can be programmed. Size of region is defined as $2 \times (\text{SIZE} + 1)$ KB.

7.3.16 FLASHDMAST Register (Offset = 0xFD4) [reset = 0x0]

Flash DMA Starting Address (FLASHDMAST)

The starting address for the Flash region accessible by the μ DMA is programmed in the FLASHDMAST register.

NOTE: The μ DMA can access Flash in Run Mode only (not available in low power modes).

FLASHDMAST is shown in [Figure 7-24](#) and described in [Table 7-23](#).

Return to [Summary Table](#).

Figure 7-24. FLASHDMAST Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				ADDR											
R-0x0				R/W-0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR					RESERVED										
R/W-0x0					R-0x0										

Table 7-23. FLASHDMAST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0x0	
28-11	ADDR	R/W	0x0	Contains the starting address of the flash region accessible by μ DMA if the FLASHPP register DFA bit is set
10-0	RESERVED	R	0x0	

7.4 EEPROM Registers

[Table 7-24](#) lists the memory-mapped registers for the EEPROM. All register offset addresses not listed in [Table 7-24](#) should be considered as reserved locations and the register contents should not be modified.

Registers in this section are relative to the EEPROM base address of 0x400AF000.

The EEPROM module clock must be enabled before the registers can be programmed (see [Section 4.2.101](#)). There must be a delay of 3 system clock cycles after the EEPROM module clock is enabled before any EEPROM module registers are accessed. In addition, after enabling or resetting the EEPROM module, software must wait until the WORKING bit in the EEDONE register is clear before accessing any EEPROM registers.

Table 7-24. EEPROM Registers

Offset	Acronym	Register Name	Section
0x0	EESIZE	EEPROM Size Information	Section 7.4.1
0x4	EEBLOCK	EEPROM Current Block	Section 7.4.2
0x8	EEOFFSET	EEPROM Current Offset	Section 7.4.3
0x10	EERDWR	EEPROM Read-Write	Section 7.4.4
0x14	EERDWRINC	EEPROM Read-Write with Increment	Section 7.4.5
0x18	EEDONE	EEPROM Done Status	Section 7.4.6
0x1C	EESUPP	EEPROM Support Control and Status	Section 7.4.7
0x20	EEUNLOCK	EEPROM Unlock	Section 7.4.8
0x30	EEPROT	EEPROM Protection	Section 7.4.9
0x34 to 0x3C	EEPASS0 to EEPASS2	EEPROM Password 0 to EEPROM Password 2	Section 7.4.10
0x40	EEINT	EEPROM Interrupt	Section 7.4.11
0x50	EEHIDE0	EEPROM Block Hide 0	Section 7.4.12
0x54	EEHIDE1	EEPROM Block Hide 1	Section 7.4.13
0x58	EEHIDE2	EEPROM Block Hide 2	Section 7.4.14
0x80	EEDBGME	EEPROM Debug Mass Erase	Section 7.4.15
0xFC0	EEPROMPP	EEPROM Peripheral Properties	Section 7.4.16

Complex bit access types are encoded to fit into small table cells. [Table 7-25](#) shows the codes that are used for access types in this section.

Table 7-25. EEPROM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

7.4.1 EESIZE Register (Offset = 0x0) [reset = 0x00600600]

EEPROM Size Information (EESIZE)

The EESIZE register indicates the number of 16-word blocks and 32-bit words in the EEPROM.

EESIZE is shown in [Figure 7-25](#) and described in [Table 7-26](#).

Return to [Summary Table](#).

Figure 7-25. EESIZE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED						BLKCNT									
R-0x0						R-0x60									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORDCNT															
R-0x600															

Table 7-26. EESIZE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0x0	
26-16	BLKCNT	R	0x60	Number of 16-Word Blocks This value encoded in this field describes the number of 16-word blocks in the EEPROM.
15-0	WORDCNT	R	0x600	Number of 32-Bit Words This value encoded in this field describes the number of 32-bit words in the EEPROM.

7.4.2 EEBLOCK Register (Offset = 0x4) [reset = 0x0]

EEPROM Current Block (EEBLOCK)

The EEBLOCK register is used to select the EEPROM block for subsequent reads, writes, and protection control. The value is a page offset into the EEPROM, such that the first block is 0, then second block is 1, etc. Each block contains 16 words. Attempts to set an invalid block causes the BLOCK field to be configured to 0. To verify that the intended block is being accessed, software can read the BLOCK field after it has been written. An invalid block can be either a non-existent block or a block that has been hidden using the EEHIDE register. Note that block 0 cannot be hidden.

EEBLOCK is shown in [Figure 7-26](#) and described in [Table 7-27](#).

Return to [Summary Table](#).

Figure 7-26. EEBLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BLOCK															
R-0x0																R/W-0x0															

Table 7-27. EEBLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	BLOCK	R/W	0x0	<p>Current Block This field specifies the block in the EEPROM that is selected for subsequent accesses. Once this field is configured, the read-write registers operate against the specified block, using the EEOFFSET register to select the word within the block. Additionally, the protection and unlock registers are used for the selected block. The maximum value that can be written into this register is determined by the block count, as indicated by the EESIZE register. Attempts to write this field larger than the maximum number of blocks or to a locked block causes this field to be configured to 0.</p>

7.4.3 EEOFFSET Register (Offset = 0x8) [reset = 0x0]

EEPROM Current Offset (EEOFFSET)

The EEOFFSET register is used to select the EEPROM word to read or write within the block selected by the EEBLOCK register. The value is a word offset into the block. Because accesses to the EERDWRINC register change the offset, software can read the contents of this register to determine the current offset.

EEOFFSET is shown in [Figure 7-27](#) and described in [Table 7-28](#).

Return to [Summary Table](#).

Figure 7-27. EEOFFSET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OFFSET			
R-0x0												R/W-0x0			

Table 7-28. EEOFFSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	OFFSET	R/W	0x0	<p>Current Address Offset</p> <p>This value is the current address specified as an offset into the block selected by the EEBLOCK register.</p> <p>Once configured, the read-write registers, EERDRWR and EERDWRINC, operate against that address.</p> <p>The offset is automatically incremented by the EERDWRINC register, with wrap around within the block, which means the offset is incremented from 15 back to 0.</p>

7.4.4 EERDWR Register (Offset = 0x10) [reset = X]

EEPROM Read-Write (EERDWR)

The EERDWR register is used to read or write the EEPROM word at the address pointed to by the EEBLOCK and EEOFFSET registers. If the protection or access rules do not permit access, the operation is handled as follows: if reading is not allowed, the value 0xFFFFFFFF is returned in all cases; if writing is not allowed, the EEDONE register is configured to indicate an error.

NOTE: A read of the EERDWR register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in EEDONE register.

EERDWR is shown in [Figure 7-28](#) and described in [Table 7-29](#).

Return to [Summary Table](#).

Figure 7-28. EERDWR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															
R/W-X																															

Table 7-29. EERDWR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	X	EEPROM Read or Write Data On a read, this field contains the value at the word pointed to by EEOFFSET. On a write, this field contains the data to be stored at the word pointed to by EEOFFSET. For writes, configuring this field starts the write process. If protection and access rules do not permit reads, all 1s are returned. If protection and access rules do not permit writes, the write fails and the EEDONE register indicates failure.

7.4.5 EERDWRINC Register (Offset = 0x14) [reset = X]

EEPROM Read-Write with Increment (EERDWRINC)

The EERDWRINC register is used to read or write the EEPROM word at the address pointed to by the EEBLOCK and EEOFFSET registers, and then increment the OFFSET field in the EEOFFSET register. If the protection or access rules do not permit access, the operation is handled as follows: if reading is not allowed, the value 0xFFFFFFFF is returned in all cases; if writing is not allowed, the EEDONE register is configured to indicate an error. In any case, the OFFSET field is incremented. If the last value is reached, OFFSET wraps around to 0 and points to the first word.

NOTE: A read of the EERDWRINC register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in EEDONE register:

EERDWRINC is shown in [Figure 7-29](#) and described in [Table 7-30](#).

Return to [Summary Table](#).

Figure 7-29. EERDWRINC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															
R/W-X																															

Table 7-30. EERDWRINC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	X	<p>EEPROM Read or Write Data with Increment</p> <p>On a read, this field contains the value at the word pointed to by EEOFFSET.</p> <p>On a write, this field contains the data to be stored at the word pointed to by EEOFFSET.</p> <p>For writes, configuring this field starts the write process.</p> <p>If protection and access rules do not permit reads, all 1s are returned.</p> <p>If protection and access rules do not permit writes, the write fails and the EEDONE register indicates failure.</p> <p>Regardless of error, the OFFSET field in the EEOFFSET register is incremented by 1, and the value wraps around if the last word is reached.</p>

7.4.6 EEDONE Register (Offset = 0x18) [reset = 0x0]

EEPROM Done Status (EEDONE)

The EEDONE register indicates completion status of a write to the following registers:

- EERDWR or EERDWRINC register (for writes to the EEPROM memory)
- EEPROT register (for setting read and protection of the current block)
- EEPASSn registers (for configuring a password for a block)
- EEDBGME register (for mass erase of an EEPROM block)

This register can indicate if the write ended in an error or not. The EEDONE register can be used in conjunction with the EEINT register to indicate completion. The register can be EEDONE polled or read after an EEINT register interrupt fires. If any of the bit values in the EEDONE register are 1 after completion, then an error has occurred for that register write. If all of the bits are clear then the writes completed with success.

NOTE: Reads of the following registers during the EEPROM initialization sequence are only valid when the WORKING bit is 0 in EEDONE register:

- EERDWR or EERDWRINC
- EEPROT
- EEPASSn

EEDONE is shown in [Figure 7-30](#) and described in [Table 7-31](#).

Return to [Summary Table](#).

Figure 7-30. EEDONE Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		WRBUSY	NOPERM	WKCOPY	WKERASE	RESERVED	WORKING
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 7-31. EEDONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5	WRBUSY	R	0x0	Write Busy 0x0 = No error 0x1 = An attempt to access the EEPROM was made while a write was in progress.
4	NOPERM	R	0x0	Write Without Permission 0x0 = No error 0x1 = An attempt was made to write without permission. This error can result because the block is locked, the write violates the programmed access protection, or when an attempt is made to write a password when the password has already been written.

Table 7-31. EEDONE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	WKCOPY	R	0x0	Working on a Copy 0x0 = The EEPROM is not copying. 0x1 = A write is in progress and is waiting for the EEPROM to copy to or from the copy buffer.
2	WKERASE	R	0x0	Working on an Erase 0x0 = The EEPROM is not erasing. 0x1 = A write is in progress and the original block is being erased after being copied.
1	RESERVED	R	0x0	
0	WORKING	R	0x0	EEPROM Working 0x0 = The EEPROM is not working. 0x1 = The EEPROM is performing the requested operation.

7.4.7 EESUPP Register (Offset = 0x1C) [reset = X]

EEPROM Support Control and Status (EESUPP)

The EESUPP register indicates if internal operations are required because an internal copy buffer must be erased or a programming failure has occurred and the operation must be completed. These conditions are explained below as well as in more detail in [Section 7.2.4.1.10](#).

- If either PRETRY or ERETRY is set indicating that an operation must be completed, setting the START bit causes the operation to be performed again
- The PRETRY and ERETRY bits are cleared automatically after the failed operation has been successfully completed.

These bits are not changed by reset, so any condition that occurred before a reset is still indicated after a reset.

EESUPP is shown in [Figure 7-31](#) and described in [Table 7-32](#).

Return to [Summary Table](#).

Figure 7-31. EESUPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				PRETRY	ERETRY	RESERVED	
R-0x0				R-X	R-X	R-0x0	

Table 7-32. EESUPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	PRETRY	R	X	Programming Must Be Retried 0x0 = Programming has not failed. 0x1 = Programming from a copy in either direction failed to complete.
2	ERETRY	R	X	Erase Must Be Retried 0x0 = Erasing has not failed. 0x1 = Erasing failed to complete. If the failed erase is due to the erase of a main buffer, the copy is performed after the erase completes successfully.
1-0	RESERVED	R	0x0	

7.4.8 EEUNLOCK Register (Offset = 0x20) [reset = X]

EEPROM Unlock (EEUNLOCK)

The EEUNLOCK register can be used to unlock the whole EEPROM or a single block using a password. Unlocking is only required if a password is registered using the EEPASSn registers for the block that is selected by the EEBLOCK register. If block 0 has a password, it locks the remaining blocks from any type of access, but uses its own protection mechanism, for example readable, but not writable when locked. In addition, if block 0 has a password, it must be unlocked before unlocking any other block.

The EEUNLOCK register is written between 1 and 3 times to form the 32-bit, 64-bit, or 96-bit password registered using the EEPASSn registers. The value used to configure the EEPASS0 register must always be written last. For example, for a 96-bit password, the value used to configure the EEPASS2 register must be written first followed by the EEPASS1 and EEPASS0 register values. The block or the whole EEPROM can be re-locked by writing 0xFFFFFFFF to this register.

In the event that an invalid value is written to this register, the block remains locked. The state of the EEPROM lock can be determined by reading back the EEUNLOCK register. If a multi-word password is set and the number of words written is incorrect, writing 0xFFFFFFFF to this register reverts the EEPROM lock to the locked state, and the proper unlock sequence can be retried.

Note that the internal logic is balanced to prevent any electrical or time-based attack being used to find the correct password or its length.

NOTE: A read of the EEUNLOCK register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in EEDONE register:

EEUNLOCK is shown in [Figure 7-32](#) and described in [Table 7-33](#).

Return to [Summary Table](#).

Figure 7-32. EEUNLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNLOCK																															
R/W-X																															

Table 7-33. EEUNLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	UNLOCK	R/W	X	<p>EEPROM Unlock</p> <p>The EEPROM is locked if the block referenced by the EEBLOCK register has a password registered, or if the master block (block 0) has a password.</p> <p>Unlocking is performed by writing the password to this register. The block or the EEPROM stays unlocked until it is locked again or until the next reset.</p> <p>It can be locked again by writing 0xFFFFFFFF to this register.</p> <p>0x0 = The EEPROM is locked.</p> <p>0x1 = The EEPROM is unlocked.</p>

7.4.9 EEPROT Register (Offset = 0x30) [reset = 0x0]

EEPROM Protection (EEPROT)

The EEPROT register is used to set or read the protection for the current block, as selected by the EEBLOCK register. Protection and access control is used to determine when a block's contents can be read or written.

NOTE: A read of the EEPROT register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in EEDONE register.

EEPROT is shown in [Figure 7-33](#) and described in [Table 7-34](#).

Return to [Summary Table](#).

Figure 7-33. EEPROT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												ACC	PROT		
R-0x0												R/W-0x0	R/W-0x0		

Table 7-34. EEPROT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	ACC	R/W	0x0	Access Control If this bit is set for block 0, then the whole EEPROM may only be accessed by supervisor code. 0x0 = Both user and supervisor code may access this block of the EEPROM. 0x1 = Only supervisor code may access this block of the EEPROM. I ² C DMA and Debug are also prevented from accessing the EEPROM.
2-0	PROT	R/W	0x0	Protection Control The Protection bits control what context is needed for reading and writing the block selected by the EEBLOCK register, or if block 0 is selected, all blocks. The following values are allowed: 0x0 = This setting is the default. Without password: the block is not protected and is readable and writable at any time. With password: the block is readable, but only writable when unlocked. 0x1 = With password: the block is readable or writable only when unlocked. This value has no meaning when there is no password. 0x2 = Without password: the block is readable, not writable. With password: the block is readable only when unlocked, but is not writable under any conditions. 0x3 = Reserved

7.4.10 EEPASS0 to EEPASS2 Registers (Offset = 0x34 to 0x3C) [reset = X]

EEPROM Password (EEPASS0), offset 0x034

EEPROM Password (EEPASS1), offset 0x038

EEPROM Password (EEPASS2), offset 0x03C

The EEPASSn registers are used to configure a password for a block. A password may only be set once and cannot be changed. The password may be 32-bits, 64-bits, or 96-bits. Each word of the password can be any 32-bit value other than 0xFFFFFFFF (all 1s). To set a password, the EEPASS0 register is written to with a value other than 0xFFFFFFFF. When the write completes, as indicated in the EEDONE register, the application may choose to write to the EEPASS1 register with a value other than 0xFFFFFFFF. When that write completes, the application may choose to write to the **EEPASS2** register with a value other than 0xFFFFFFFF to create a 96-bit password. The registers do not have to be written consecutively, and the EEPASS1 and EEPASS2 registers may be written at a later date. Based on whether 1, 2, or all 3 registers have been written, the unlock code also requires the same number of words to unlock.

NOTE: Once the password is written, the block is not actually locked until either a reset occurs or 0xFFFFFFFF is written to EEUNLOCK.

NOTE: A read of the EEPASSn register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in EEDONE register:

EEPASSn is shown in [Figure 7-34](#) and described in [Table 7-35](#).

Return to [Summary Table](#).

Figure 7-34. EEPASS0 to EEPASS2 Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PASS																															
R/W-X																															

Table 7-35. EEPASS0 to EEPASS2 Registers Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PASS	R/W	X	Password This register reads as 0x1 if a password is registered for this block and 0x0 if no password is registered. A write to this register if it reads as 0x0 sets the password. If an attempt is made to write to this register when it reads as 0x1, the write is ignored and the NOPERM bit in the EEDONE register is set.

7.4.11 EEINT Register (Offset = 0x40) [reset = 0x0]

EEPROM Interrupt (EEINT)

The EEINT register is used to control whether an interrupt should be generated when a write to EEPROM completes as indicated by the EEDONE register value changing from 0x1 to any other value. If the INT bit in this register is set, the ERIS bit in the Flash Controller Raw Interrupt Status (FCRIS) register is set whenever the EEDONE register value changes from 0x1 as the Flash memory and the EEPROM share an interrupt vector.

EEINT is shown in [Figure 7-35](#) and described in [Table 7-36](#).

Return to [Summary Table](#).

Figure 7-35. EEINT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															INT
R-0x0															R/W-0x0

Table 7-36. EEINT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	INT	R/W	0x0	Interrupt Enable 0x0 = No interrupt is generated. 0x1 = An interrupt is generated when the EEDONE register transitions from 1 to 0 or an error occurs. The EEDONE register provides status after a write to an offset location as well as a write to the password and protection bits.

7.4.12 EEHIDE0 Register (Offset = 0x50) [reset = 0x0]

EEPROM Block Hide 0 (EEHIDE0)

The EEHIDE0 register is used to hide one or more blocks other than EEPROM block 0. Bits 1 through 31 of this register correspond to EEPROM blocks 1 through 31. Once hidden, the block is not accessible until the next reset. This model allows initialization code to have access to data which is not visible to the rest of the application. This register also provides for additional security in that there is no password to search for in the code or data.

EEHIDE0 is shown in [Figure 7-36](#) and described in [Table 7-37](#).

Return to [Summary Table](#).

Figure 7-36. EEHIDE0 Register

31	30	29	28	27	26	25	24
H0							
R/W-0x0							
23	22	21	20	19	18	17	16
H0							
R/W-0x0							
15	14	13	12	11	10	9	8
H0							
R/W-0x0							
7	6	5	4	3	2	1	0
H0							RESERVED
R/W-0x0							R-0x0

Table 7-37. EEHIDE0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	H0	R/W	0x0	Hide Block 0 0x0 = The corresponding block is not hidden. 0x1 = The block number that corresponds to the bit number is hidden. A hidden block cannot be accessed, and the OFFSET value in the EEBLOCK register cannot be set to that block number. If an attempt is made to configure the OFFSET field to a hidden block, the EEBLOCK register is cleared. Any attempt to clear a bit in this register that is set is ignored.
0	RESERVED	R	0x0	

7.4.13 EEHIDE1 Register (Offset = 0x54) [reset = 0x0]

EEPROM Block Hide 1 (EEHIDE1)

The EEHIDE register is used to hide one or more blocks. Bits 0 through 31 of the EEHIDE1 register correspond to EEPROM blocks 32 through 63. Bits 0 through 31 of the EEHIDE2 register correspond to EEPROM blocks 64 through 95. Once hidden, the block is not accessible until the next reset. This model allows initialization code to have access to data which is not visible to the rest of the application. This register also provides for additional security in that there is no password to search for in the code or data.

EEHIDE1 is shown in [Figure 7-37](#) and described in [Table 7-38](#).

Return to [Summary Table](#).

Figure 7-37. EEHIDE1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H1																															
R/W-0x0																															

Table 7-38. EEHIDE1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	H1	R/W	0x0	Hide Block 1 0x0 = The corresponding block is not hidden. 0x1 = The block number that corresponds to the bit number is hidden. A hidden block cannot be accessed, and the OFFSET value in the EEBLOCK register cannot be set to that block number. If an attempt is made to configure the OFFSET field to a hidden block, the EEBLOCK register is cleared. Any attempt to clear a bit in this register that is set is ignored.

7.4.14 EEHIDE2 Register (Offset = 0x58) [reset = 0x0]

EEPROM Block Hide 2 (EEHIDE2)

The EEHIDE register is used to hide one or more blocks. Bits 0 through 31 of the EEHIDE1 register correspond to EEPROM blocks 32 through 63. Bits 0 through 31 of the EEHIDE2 register correspond to EEPROM blocks 64 through 95. Once hidden, the block is not accessible until the next reset. This model allows initialization code to have access to data which is not visible to the rest of the application. This register also provides for additional security in that there is no password to search for in the code or data.

EEHIDE2 is shown in [Figure 7-38](#) and described in [Table 7-39](#).

Return to [Summary Table](#).

Figure 7-38. EEHIDE2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H2																															
R/W-0x0																															

Table 7-39. EEHIDE2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	H2	R/W	0x0	<p>Hide Block 2</p> <p>0x0 = The corresponding block is not hidden.</p> <p>0x1 = The block number that corresponds to the bit number is hidden. A hidden block cannot be accessed, and the OFFSET value in the EEBLOCK register cannot be set to that block number. If an attempt is made to configure the OFFSET field to a hidden block, the EEBLOCK register is cleared. Any attempt to clear a bit in this register that is set is ignored.</p>

7.4.15 EEDBGME Register (Offset = 0x80) [reset = 0x0]

EEPROM Debug Mass Erase (EEDBGME)

The EEDBGME register is used to mass erase the EEPROM block back to its default state from the factory. This register is intended to be used only for debug and test purposes, not in production environments. The erase takes place in such a way as to be secure. It first erases all data and then erases the protection mechanism. This register can only be written from supervisor mode by the core, and can also be written by the debug controller when enabled. A key is used to avoid accidental use of this mechanism. Note that if a power down takes place while erasing, the mechanism should be used again to complete the operation. Powering off prematurely does not expose secured data.

To start a mass erase, the whole register must be written as 0xE37B0001. The register reads back as 0x1 until the erase is fully completed at which time it reads as 0x0. The EEDONE register is set to 0x1 when the erase is started and changes to 0x0 or an error when the mass erase is complete.

EEDBGME is shown in [Figure 7-39](#) and described in [Table 7-40](#).

Return to [Summary Table](#).

Figure 7-39. EEDBGME Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY															
W-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															ME
R-0x0															R/W-0x0

Table 7-40. EEDBGME Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	KEY	W	0x0	Erase Key This field must be written with 0xE37B for the ME field to be effective.
15-1	RESERVED	R	0x0	
0	ME	R/W	0x0	Mass Erase 0x0 = No action. 0x1 = When written as a 1, the EEPROM is mass erased. This bit continues to read as 1 until the EEPROM is fully erased.

7.4.16 EEPROMPP Register (Offset = 0xFC0) [reset = 0x1FF]

EEPROM Peripheral Properties (EEPROMPP)

The EEPROMPP register indicates the size of the EEPROM for this part.

EEPROMPP is shown in [Figure 7-40](#) and described in [Table 7-41](#).

Return to [Summary Table](#).

Figure 7-40. EEPROMPP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																SIZE															
R-0x0																R-0x1FF															

Table 7-41. EEPROMPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	SIZE	R	0x1FF	<p>EEPROM Size Indicates the size of the on-chip EEPROM. Any values not shown are reserved.</p> <p>0x0000 = 64 bytes of EEPROM 0x0001 = 128 bytes of EEPROM 0x0003 = 256 bytes of EEPROM 0x0007 = 512 bytes of EEPROM 0x000F = 1KB of EEPROM 0x001F = 2KB of EEPROM 0x003F = 3KB of EEPROM 0x007F = 4KB of EEPROM 0x00FF = 5KB of EEPROM 0x01FF = 6KB of EEPROM</p>

7.5 System Control Memory Registers

Table 7-42 lists the memory-mapped registers for the system control memory. All register offset addresses not listed in Table 7-42 should be considered as reserved locations and the register contents should not be modified.

Offsets are relative to the System Control base address of 0x400FE000.

Table 7-42. System Control Memory Registers

Offset	Acronym	Register Name	Section
0xD4	RVP	Reset Vector Pointer	Section 7.5.1
0x1D0	BOOTCFG	Boot Configuration	Section 7.5.2
0x1E0 to 0x1EC	USER_REG_0 to USER_REG_3	User Register 0 to User Register 3	Section 7.5.3
0x200 to 0x23C	FMPRE_0 to FMPRE_15	Flash Memory Protection Read Enable 0 to Flash Memory Protection Read Enable 15	Section 7.5.4
0x400 to 0x43C	FMPPE_0 to FMPPE_15	Flash Memory Protection Program Enable 0 to Flash Memory Protection Program Enable 15	Section 7.5.5

Complex bit access types are encoded to fit into small table cells. Table 7-43 shows the codes that are used for access types in this section.

Table 7-43. SYSTEM_CONTROL_MEMORY Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

7.5.1 RVP Register (Offset = 0xD4) [reset = 0x0101FFF0]

Reset Vector Pointer (RVP)

The Reset Vector Pointer (RVP) register contains the address of the reset vector of the software module that is to be executed after boot loader execution. The RVP register is initialized by a power-on reset.

RVP is shown in [Figure 7-41](#) and described in [Table 7-44](#).

Return to [Summary Table](#).

Figure 7-41. RVP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RV																															
R-0x0101FFF0																															

Table 7-44. RVP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RV	R	0x0101FFF0	Reset Vector Pointer Address

7.5.2 BOOTCFG Register (Offset = 0x1D0) [reset = 0xFFFFFFFFE]

Boot Configuration (BOOTCFG)

NOTE: The Boot Configuration (BOOTCFG) register requires a POR before the committed changes take effect.

This register is not written directly, but instead uses the FMD register as explained in [Section 7.2.3.12](#). When this register is committed, the new value cannot be read back until after the power cycle. This register provides configuration of a GPIO pin to enable the ROM Bootloader as well as a write-once mechanism to disable external debugger access to the device. At reset, the user has the opportunity to direct the core to execute the ROM Bootloader or the application in flash memory by using a GPIO signal from Ports A through H as configured by the bits in this register (do not select PC0 to PC3, PD7, or PE7, because they are locked by default at reset). At reset, the following sequence is performed:

1. The BOOTCFG register is read. If the EN bit is clear, the ROM Bootloader is executed.
2. In the ROM Bootloader, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x00000000 and execution continues out of the ROM Bootloader.
3. If the EN bit is set or the status does not match the specified polarity, the data at address 0x00000004 is read, and if the data at this address is 0xFFFFFFFF, the ROM is mapped to address 0x00000000 and execution continues out of the ROM Bootloader.
4. If there is data at address 0x00000004 that is not 0xFFFFFFFF, the stack pointer (SP) is loaded from flash memory at address 0x00000000 and the program counter (PC) is loaded from address 0x00000004. The user application begins executing.

Figure 7-42 shows the bootloader selection sequence.

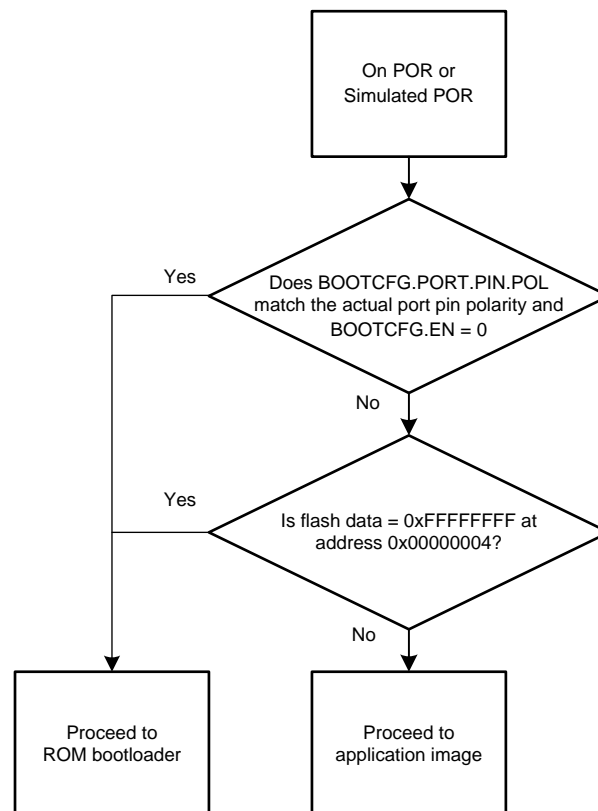


Figure 7-42. Boot Configuration Flow

The DBG0 bit is cleared by the factory and the DBG1 bit is set, which enables external debuggers. Clearing the DBG1 bit disables any external debugger access to the device, starting with the next power-up cycle of the device. The NW bit indicates that bits in the register can be changed from 1 to 0.

By committing the register values using the COMT bit in the FMC register, the register contents become nonvolatile and are therefore retained following power cycling. Before being committed, bits can only be changed from 1 to 0. All Reserved bits must remain as 1s. The reset value shown applies only to power-on reset when the register is not yet committed; any other type of reset does not affect this register. After it is committed, the register retains its value through power-on reset. When committed, the only way to restore the factory default value of this register is to perform the sequence detailed in [Section 3.3.4.3](#).

BOOTCFG is shown in [Figure 7-43](#) and described in [Table 7-45](#).

Return to [Summary Table](#).

Figure 7-43. BOOTCFG Register

31	30	29	28	27	26	25	24
NW	RESERVED						
R-0x1	R-0x7FFF						
23	22	21	20	19	18	17	16
RESERVED							
R-0x7FFF							
15	14	13	12	11	10	9	8
PORT			PIN			POL	EN
R-0x7			R-0x7			R-0x1	R-0x1
7	6	5	4	3	2	1	0
RESERVED			KEY	RESERVED		DBG1	DBG0
R-0x7			R-0x1	R-0x3		R-0x1	R-0x0

Table 7-45. BOOTCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NW	R	0x1	Not Written. When set, this bit indicates that the values in this register can be changed from 1 to 0. When clear, this bit specifies that the contents of this register cannot be changed.
30-16	RESERVED	R	0x7FFF	Ensure that the Reserved bits of this register are not changed from the default of all 1s.
15-13	PORT	R	0x7	Boot GPIO Port. This field selects the port of the GPIO port pin that enables the ROM boot loader at reset. The selected port can be reprogrammed for a different function after reset. Note: Do not select PC0-3, PD7, or PE7, because they are locked by default at reset. 0x0 = Port A 0x1 = Port B 0x2 = Port C 0x3 = Port D 0x4 = Port E 0x5 = Port F 0x6 = Port G 0x7 = Port H

Table 7-45. BOOTCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12-10	PIN	R	0x7	<p>Boot GPIO Pin</p> <p>This field selects the pin number of the GPIO port pin that enables the ROM boot loader at reset.</p> <p>0x0 = Pin 0 0x1 = Pin 1 0x2 = Pin 2 0x3 = Pin 3 0x4 = Pin 4 0x5 = Pin 5 0x6 = Pin 6 0x7 = Pin 7</p>
9	POL	R	0x1	<p>Boot GPIO Polarity</p> <p>When set, this bit selects a high level for the GPIO port pin to enable the ROM boot loader at reset.</p> <p>When clear, this bit selects a low level for the GPIO port pin.</p>
8	EN	R	0x1	<p>Boot GPIO</p> <p>Enable Clearing this bit enables the use of a GPIO pin to enable the ROM Bootloader at reset.</p> <p>When this bit is set, the contents of address 0x00000004 are checked to see if the flash memory has been programmed.</p> <p>If the contents are not 0xFFFFFFFF, the core executes out of flash memory.</p> <p>If the Flash has not been programmed, the core executes out of ROM.</p>
7-5	RESERVED	R	0x7	<p>Ensure that the Reserved bits of this register are not changed from the default of all 1s.</p>
4	KEY	R	0x1	<p>KEY Select</p> <p>This bit chooses between using the value 0xA442 or the PEKEY value in the FLPEKEY register as the WRKEY value in the FMC/FMC2 register.</p> <p>0x0 = The PEKEY value in the FLPEKEY register is committed by user and used as the WRKEY in the FMC/FMC2 register. Writes to FMC/FMC2 register with a 0xA442 key are ignored.</p> <p>0x1 = 0xA442 is used as key</p>
3-2	RESERVED	R	0x3	<p>Ensure that the Reserved bits of this register are not changed from the default of all 1s.</p>
1	DBG1	R	0x1	<p>Debug Control 1</p> <p>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.</p>
0	DBG0	R	0x0	<p>Debug Control 0</p> <p>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.</p>

7.5.3 USER_REG0 to USER_REG3 Registers (Offset = 0x1E0 to 0x1EC) [reset = 0xFFFFFFFF]

User Register 0 (USER_REG0), offset 0x1E0

User Register 1 (USER_REG1), offset 0x1E4

User Register 2 (USER_REG2), offset 0x1E8

User Register 3 (USER_REG3), offset 0x1EC

These registers each provide 32 bits of user-defined data that is nonvolatile. Bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset when the register is not yet committed; any other type of reset does not affect this register. Once committed, the register retains its value through power-on reset. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG section.

USER_REG is shown in [Figure 7-44](#) and described in [Table 7-46](#).

Return to [Summary Table](#).

Figure 7-44. USER_REGn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0xFFFFFFFF																															

Table 7-46. USER_REGn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0xFFFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and once committed, retains its value through power-on reset.

7.5.4 FMPRE0 to FMPRE15 Registers (Offset = 0x200 to 0x23C) [reset = 0xFFFFFFFF]

Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x200

Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

Flash Memory Protection Read Enable 4 (FMPRE4), offset 0x210

Flash Memory Protection Read Enable 5 (FMPRE5), offset 0x214

Flash Memory Protection Read Enable 6 (FMPRE6), offset 0x218

Flash Memory Protection Read Enable 7 (FMPRE7), offset 0x21C

Flash Memory Protection Read Enable 8 (FMPRE8), offset 0x220

Flash Memory Protection Read Enable 9 (FMPRE9), offset 0x224

Flash Memory Protection Read Enable 10 (FMPRE10), offset 0x228

Flash Memory Protection Read Enable 11 (FMPRE11), offset 0x22C

Flash Memory Protection Read Enable 12 (FMPRE12), offset 0x230

Flash Memory Protection Read Enable 13 (FMPRE13), offset 0x234

Flash Memory Protection Read Enable 14 (FMPRE14), offset 0x238

Flash Memory Protection Read Enable 15 (FMPRE15), offset 0x23C

This register stores the read-only protection bits for each 2KB flash block (FMPREn stores the execute-only bits). Note that for protecting sectors, eight bits need to be cleared to create a 16KB read-protected sector.

This register is loaded during the power-on reset sequence. The factory settings for the FMPREn and FMPPEn registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is RW0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter.

NOTE: Do not clear bits [7:0] of the FMPRE0 register to read protect the first 16KB of the flash memory. If this part of memory is read protected, the device cannot boot up, because the ROM reads address 0x4 to determine if a valid application resides in flash memory.

Each FMPREn register controls a 64K block. For additional information, see [Section 7.2.3.4](#).

- FMPRE0 : 0 to 64KB
- FMPRE1 : 65 to 128KB
- FMPRE2 : 129 to 192KB
- FMPRE3 : 193 to 256KB
- FMPRE4 : 257 to 320KB
- FMPRE5 : 321 to 384KB
- FMPRE6 : 385 to 448KB
- FMPRE7 : 449 to 512KB
- FMPRE8 : 513 to 576KB
- FMPRE9 : 577 to 640KB
- FMPRE10 : 641 to 704KB

- FMPRE11 : 705 to 768KB
- FMPRE12 : 769 to 832KB
- FMPRE13 : 833 to 896KB
- FMPRE14 : 897 to 960KB
- FMPRE15 : 961 to 1024KB

FMPRE is shown in [Figure 7-45](#) and described in [Table 7-47](#).

Return to [Summary Table](#).

Figure 7-45. FMPREn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READ_ENABLE																															
R/W-0xFFFFFFFF																															

Table 7-47. FMPREn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Each bit configures a 2KB flash block to be read only. Note that for read-protection of sectors, eight bits need to be cleared to create a 16KB read-protected sector. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

7.5.5 FMPPE0 to FMPPE15 Registers (Offset = 0x400 to 0x43C) [reset = 0xFFFFFFFF]

Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x400
Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404
Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408
Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C
Flash Memory Protection Program Enable 4 (FMPPE4), offset 0x410
Flash Memory Protection Program Enable 5 (FMPPE5), offset 0x414
Flash Memory Protection Program Enable 6 (FMPPE6), offset 0x418
Flash Memory Protection Program Enable 7 (FMPPE7), offset 0x41C
Flash Memory Protection Program Enable 8 (FMPPE8), offset 0x420
Flash Memory Protection Program Enable 9 (FMPPE9), offset 0x424
Flash Memory Protection Program Enable 10 (FMPPE10), offset 0x428
Flash Memory Protection Program Enable 11 (FMPPE11), offset 0x42C
Flash Memory Protection Program Enable 12 (FMPPE12), offset 0x430
Flash Memory Protection Program Enable 13 (FMPPE13), offset 0x434
Flash Memory Protection Program Enable 14 (FMPPE14), offset 0x438
Flash Memory Protection Program Enable 15 (FMPPE15), offset 0x43C

This register stores the execute-only protection bits for each 2KB flash block (FMPPE_n stores the read-only protection bits). Since the memory is two-way interleaved and each bank individually is an 8KB sector, read-only protection must occur across a block size of 16KB. No smaller block size is supported. Note that the Flash Memory Protection Read (FMPRE_n) registers do allow read-protection of a block as small as 2KB, unlike the FMPPE_n registers.

Thus, in order to execute-only protect a 16KB block, a user must program the entire eight bits of the byte to the same value. For example, to protect the first 16KB block, bits [7:0] of the FMPPE0 register need to be cleared to all 0s.

This register is loaded during the power-on reset sequence. The factory settings for the FMPRE_n and FMPPE_n registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. This register is RW0; the user can only change the protection byte from all 1s to all 0s (and may NOT change from all 0 to all 1). The changes are not permanent until the register is committed (saved), at which point the byte change is permanent. If a byte is changed from all 1s to all 0s and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. For additional information, see [Section 7.2.3.4](#).

Each FMPPE_n register controls a 64K block. For additional information, see [Section 7.2.3.4](#).

- FMPPE0 : 0 to 64KB
- FMPPE1 : 65 to 128KB
- FMPPE2 : 129 to 192KB
- FMPPE3 : 193 to 256KB
- FMPPE4 : 257 to 320KB
- FMPPE5 : 321 to 384KB
- FMPPE6 : 385 to 448KB
- FMPPE7 : 449 to 512KB
- FMPPE8 : 513 to 576KB
- FMPPE9 : 577 to 640KB
- FMPPE10 : 641 to 704KB

- FMPPE11 : 705 to 768KB
- FMPPE12 : 769 to 832KB
- FMPPE13 : 833 to 896KB
- FMPPE14 : 897 to 960KB
- FMPPE15 : 961 to 1024KB

FMPPE is shown in [Figure 7-46](#) and described in [Table 7-48](#).

Return to [Summary Table](#).

Figure 7-46. FMPPEn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PROG_ENABLE																															
R/W-0xFFFFFFFF																															

Table 7-48. FMPPEn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Every eighth bit programs an 16KB flash sector to be execute only. The policies may be combined as shown in .

Micro Direct Memory Access (μDMA)

The MSP432E4 microcontrollers include a Direct Memory Access (DMA) controller, known as micro-DMA (μDMA). The μDMA controller provides a way to offload data transfer tasks from the Cortex-M4F processor, allowing for more efficient use of the processor and the available bus bandwidth. The μDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory when the peripheral is ready to transfer more data.

Topic	Page
8.1 Introduction	600
8.2 Block Diagram.....	600
8.3 Functional Description	601
8.4 Initialization and Configuration	613
8.5 μDMA Channel Control Structure Registers	620
8.6 μDMA Registers	626

8.1 Introduction

The μ DMA controller provides the following features:

- Arm® PrimeCell® 32-channel configurable μ DMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
 - Basic for simple transfer scenarios
 - Ping-pong for continuous data flow
 - Scatter-gather for a programmable list of up to 256 arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
 - Independently configured and operated channels
 - Dedicated channels for supported on-chip modules
 - Flexible channel assignments
 - One channel each for receive and transmit path for bidirectional modules
 - Dedicated channel for software-initiated transfers
 - Per-channel configurable priority scheme
 - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between μ DMA controller and the processor core
 - μ DMA controller access that is subordinate to core access
 - RAM striping
 - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Programmable transfer size in binary steps from 1 to 1024
- Source and destination address increment size of byte, halfword, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion with a separate interrupt per channel

8.2 Block Diagram

[Figure 8-1](#) shows the μ DMA block diagram.

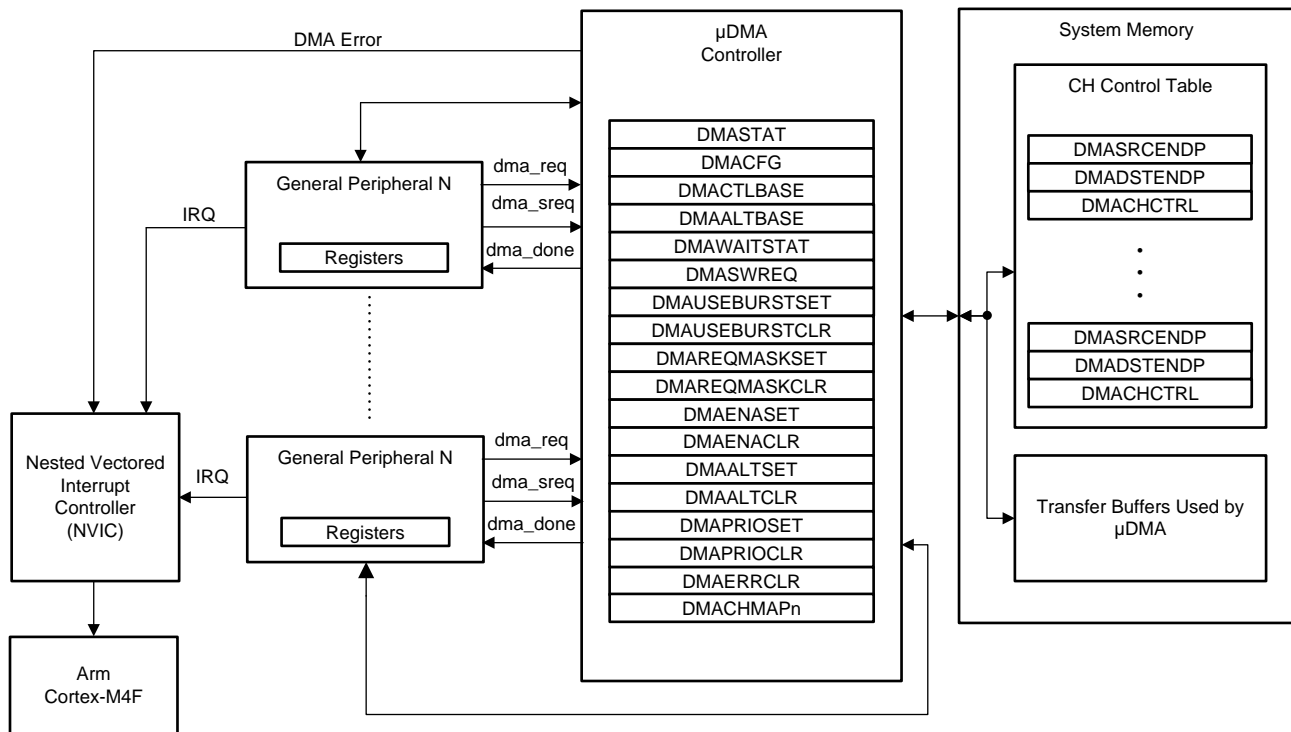


Figure 8-1. μDMA Block Diagram

8.3 Functional Description

The μDMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the Cortex-M4F processor core of the microcontroller. The μDMA controller supports multiple data sizes and address increment schemes, multiple levels of priority among DMA channels, and several transfer modes to allow for sophisticated programmed data transfers. Bus use by the μDMA controller is always subordinate to the processor core, so it never delays a bus transaction by the processor. Because the μDMA controller is only using otherwise idle bus cycles, the data transfer bandwidth it provides is essentially free, with no effect on the rest of the system. The bus architecture has been optimized to greatly enhance the ability of the processor core and the μDMA controller to efficiently share the on-chip bus, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases let both the processor core and the μDMA controller access the bus and perform simultaneous data transfers.

Each peripheral function that is supported has a dedicated channel on the μDMA controller that can be configured independently. The μDMA controller implements a unique configuration method using channel control structures that are maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated "task" lists in memory that allow the μDMA controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The μDMA controller also supports the use of ping-pong buffering to accommodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the μDMA controller rearbiterates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time the peripheral makes a μDMA service request.

8.3.1 Priority

The μ DMA controller assigns priority to each channel based on the channel number and the priority level bit for the channel. Channel number 0 has the highest priority and as the channel number increases, the priority of a channel decreases. Each channel has a priority level bit to provide two levels of priority: default priority and high priority. If the priority level bit is set, then that channel has higher priority than all other channels at default priority. If multiple channels are set for high priority, then the channel number is used to determine relative priority among all the high-priority channels.

The priority bit for a channel can be set using the DMA Channel Priority Set (DMAPRIOSET) register and cleared with the DMA Channel Priority Clear (DMAPRIOLCLR) register.

NOTE: If one peripheral is mapped to two different channels, then the application should either use the default mapping for that peripheral or change the default mapping to another source. For example, if UART1 channels 8 and 9 are enabled for use, then even if channels 22 and 23 are disabled, they must be mapped to software or another peripheral (if available).

8.3.2 Arbitration Size

When a μ DMA channel requests a transfer, the μ DMA controller arbitrates among all the channels making a request and services the μ DMA channel with the highest priority. When a transfer begins, it continues for a selectable number of transfers before re-arbitrating among the requesting channels again. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the μ DMA controller transfers the number of items specified by the arbitration size, it then checks among all the channels making a request and services the channel with the highest priority.

If a lower priority μ DMA channel uses a large arbitration size, the latency for higher priority channels is increased because the μ DMA controller completes the lower priority burst before checking for higher priority requests. Therefore, lower priority channels should not use a large arbitration size for best response on high priority channels.

The arbitration size can also be thought of as a burst size. It is the maximum number of items that are transferred at any one time in a burst. Here, the term arbitration refers to determination of μ DMA channel priority, not arbitration for the bus. When the μ DMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the μ DMA controller is held off whenever the processor must perform a bus transaction on the same bus, even in the middle of a burst transfer.

8.3.3 Request Types

The μ DMA controller responds to two types of requests from a peripheral: single or burst. Each peripheral may support either or both types of requests. A single request means that the peripheral is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The μ DMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the μ DMA channel has been set up for a burst transfer, then the burst request takes precedence. [Table 8-1](#) shows how each peripheral supports the two request types.

Table 8-1. Request Type Support

Peripheral	Event That Generates Single Request	Event That Generates Burst Request
ADC	FIFO not empty	FIFO half full
EPI WFIFO	None	WFIFO level (configurable)
EPI NBRFIFO	None	NBRFIFO level (configurable)
General-Purpose Timer	None	Trigger event
GPIO	None	Trigger event
I ² C TX	TX buffer not full	TX FIFO level (configurable)
I ² C RX	RX buffer not empty	RX FIFO level (configurable)
SSI TX	TX FIFO not full	TX FIFO level (fixed at 4)
SSI RX	RX FIFO not empty	RX FIFO level (fixed at 4)

Table 8-1. Request Type Support (continued)

Peripheral	Event That Generates Single Request	Event That Generates Burst Request
UART TX	TX FIFO not full	TX FIFO level (configurable)
UART RX	RX FIFO not empty	RX FIFO level (configurable)
SHA/MD5	None	Context in DMA request (SHA/MD5 0 Cin) Context out DMA request (SHA/MD5 0 Cout) Data In DMA request (SHA/MD5 0 Din)
AES	None	Context in DMA request (AES0 Cin) Context out DMA request (AES0 Cout) Data in DMA request (AES0 Din) Data out DMA request (AES0 Dout)
DES	None	Context in DMA request (DES0 Cin) Data in DMA request (DES0 Din) Data out DMA request (DES0 Dout)

8.3.3.1 Single Request

When a single request is detected, and not a burst request, the μ DMA controller transfers one item and then stops to wait for another request.

8.3.3.2 Burst Request

When a burst request is detected, the μ DMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

It may be desirable to use only burst transfers and not allow single transfers. For example, perhaps the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time. The single request can be disabled by using the DMA Channel Useburst Set (DMAUSEBURSTSET) register. By setting the bit for a channel in this register, the μ DMA controller only responds to burst requests for that channel.

8.3.4 Channel Configuration

The μ DMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each μ DMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

[Table 8-2](#) lists the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table, and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

Any unused memory in the control table may be used by the application. This includes the control structures for any channels that are unused by the application as well as the unused control word for each channel.

Table 8-2. Control Structure Memory Map

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 8-3 summarizes an individual control structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

Table 8-3. Channel Control Structure

Offset	Description
0x000	Source End Pointer
0x004	Destination End Pointer
0x008	Control Word
0x00C	Unused

The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control word and each field are described in detail in [Section 8.5.3](#). The μ DMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates "stopped." Because the control word is modified by the μ DMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

Before starting a transfer, a μ DMA channel must be enabled by setting the appropriate bit in the DMA Channel Enable Set (DMAENASET) register. A channel can be disabled by setting the channel bit in the DMA Channel Enable Clear (DMAENACLR) register. At the end of a complete μ DMA transfer, the controller automatically disables the channel.

8.3.5 Transfer Modes

The μ DMA controller supports several transfer modes. Two of the modes support simple one-time transfers. Several complex modes support a continuous flow of data.

8.3.5.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the μ DMA controller does not perform any transfers and disables the channel if it is enabled. At the end of a transfer, the μ DMA controller updates the control word to set the mode to Stop.

8.3.5.2 Basic Mode

In Basic mode, the μ DMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a μ DMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary even though the entire transfer should be completed. For example, a software-initiated transfer creates a momentary request, and in Basic mode, only the number of transfers specified by the ARBSIZE field in the DMA Channel Control Word (DMACHCTL) register is transferred on a software request, even if there is more data to transfer.

When all of the items have been transferred using Basic mode, the μ DMA controller sets the mode for that channel to Stop.

Basic mode can be programmed to ignore when XFERSIZE reaches 0x000 and continue copying on request until the channel is stopped manually. If the NXTUSEBURST bit in the μ DMA Channel Control Word (DMACHCTL) register is set while in Basic mode and the XFERSIZE reaches 0x000 and is not written back, transfers continue until the request is deasserted by the peripheral.

8.3.5.3 Auto Mode

Auto mode is similar to Basic mode, except that when a transfer request is received, the transfer runs to completion, even if the μ DMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

When all the items have been transferred using Auto mode, the μ DMA controller sets the mode for that channel to Stop.

8.3.5.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures must be implemented. Both structures are set up by the processor for data transfer between memory and a peripheral. The transfer is started using the primary control structure. When the transfer using the primary control structure is complete, the μ DMA controller reads the alternate control structure for that channel to continue the transfer. Each time this happens, an interrupt is generated, and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch back and forth between buffers as the data flows to or from the peripheral.

For an example showing operation in Ping-Pong mode, see [Figure 8-2](#).

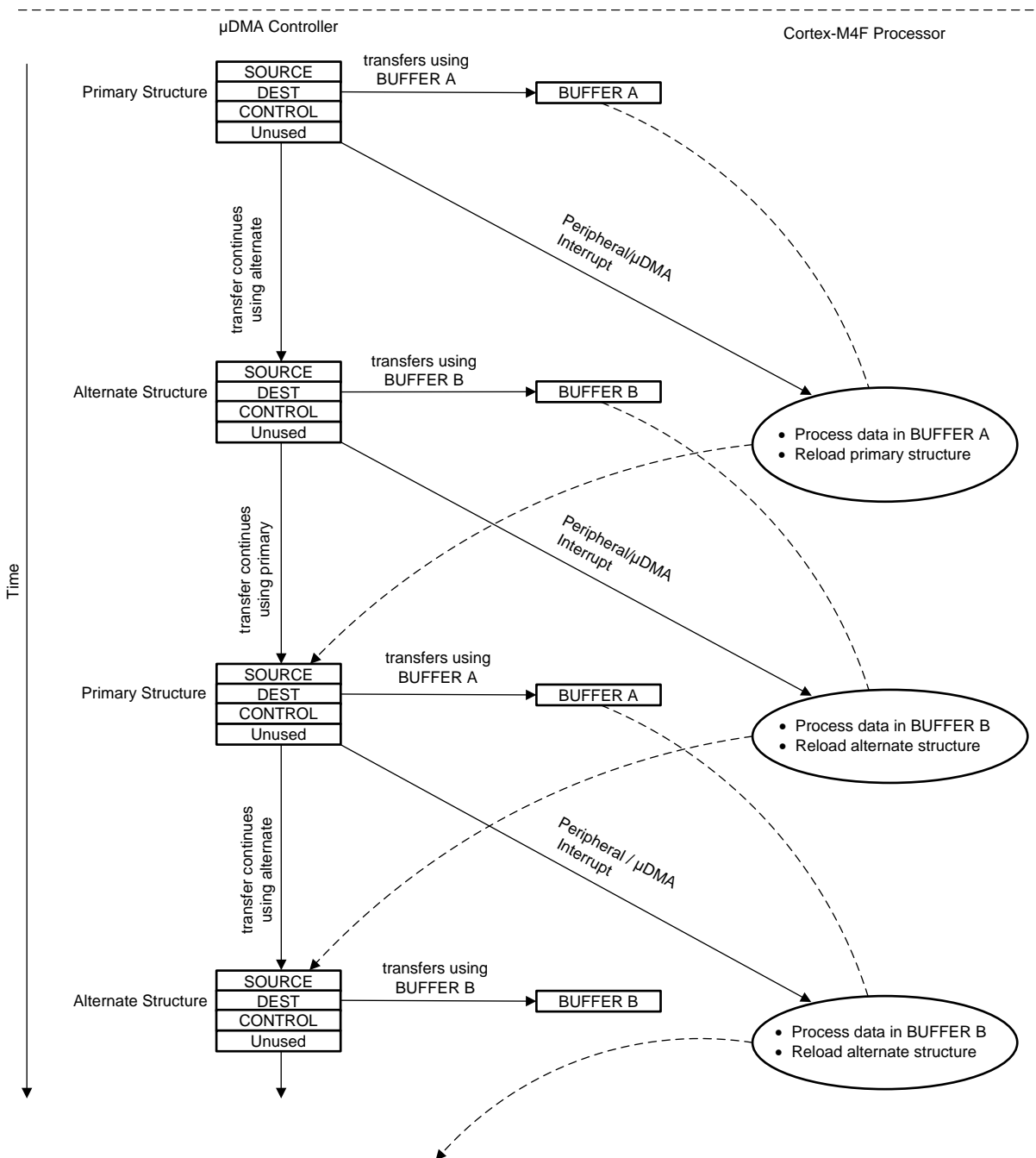


Figure 8-2. Example of Ping-Pong μ DMA Transaction

8.3.5.5 Memory Scatter-Gather

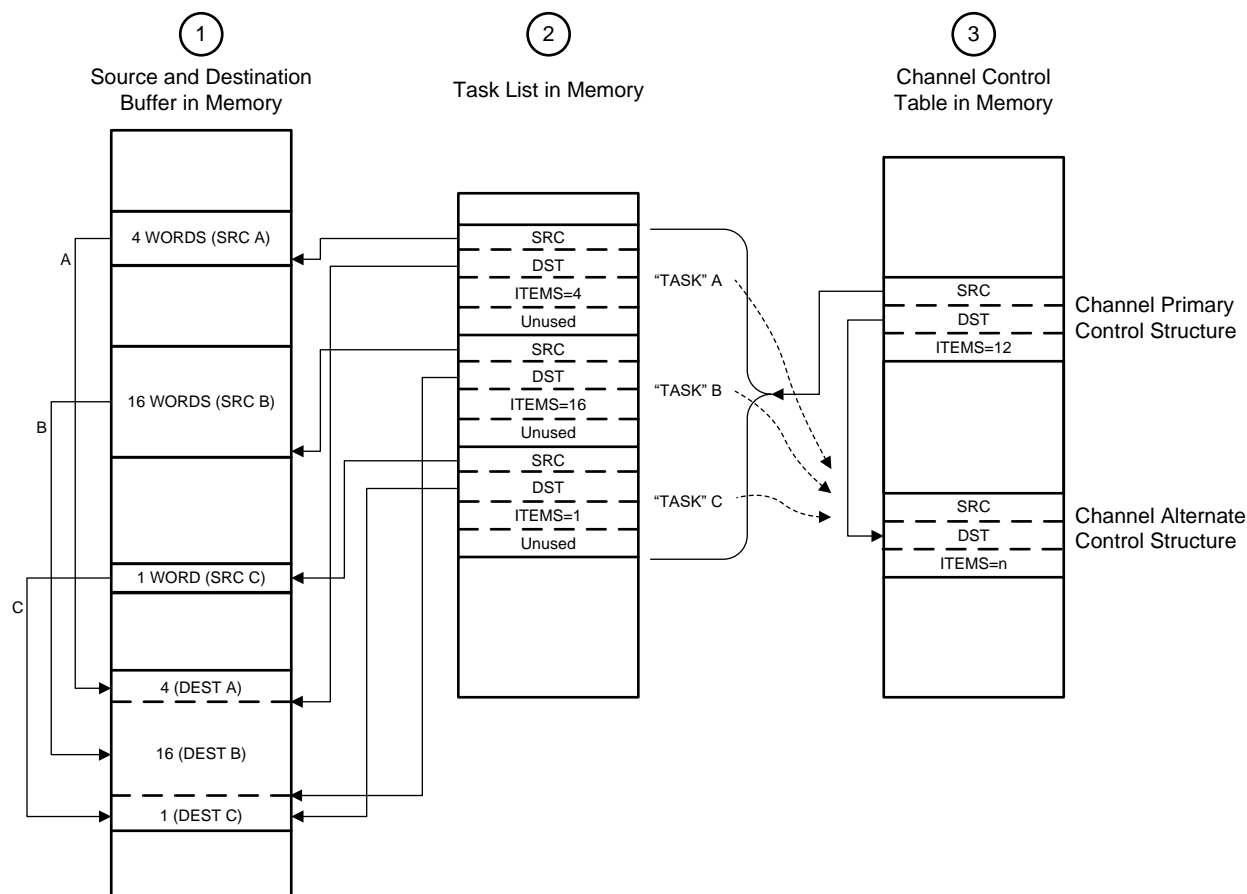
Memory Scatter-Gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather μ DMA operation could be used to selectively read the payload of several stored packets of a communication protocol and store them together in sequence in a memory buffer.

In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to Scatter-Gather mode. Each entry in the table is copied in turn to the alternate structure where it is then executed. The μ DMA controller alternates between using the primary control structure to copy the next transfer instruction from the list and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use Auto transfer mode. Once the last transfer is performed using Auto mode, the μ DMA controller stops. A completion interrupt is generated only after the last transfer. It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a μ DMA request.

By programming the μ DMA controller using this method, a set of up to 256 arbitrary transfers can be performed based on a single μ DMA request.

For an example of operation in Memory Scatter-Gather mode, see [Figure 8-3](#) and [Figure 8-4](#). This example shows **gather** operation, where data in three separate buffers in memory is copied together into one buffer. [Figure 8-3](#) shows how the application sets up a μ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

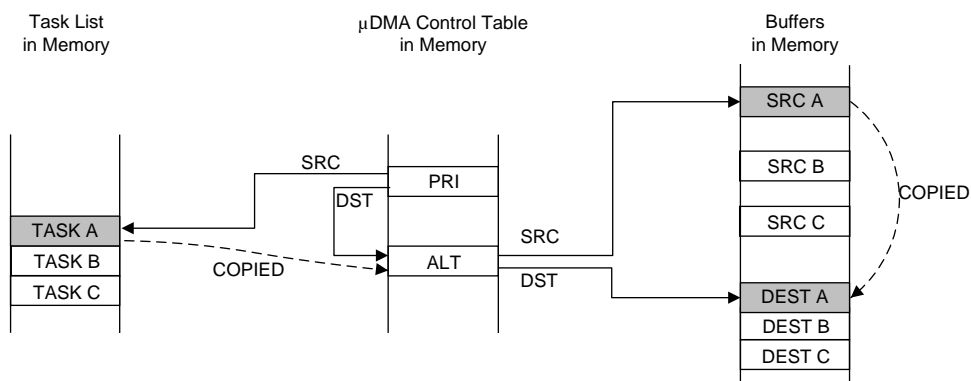
[Figure 8-4](#) shows the sequence as the μ DMA controller performs the three sets of copy operations. First, using the primary control structure, the μ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the μ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.



NOTES:

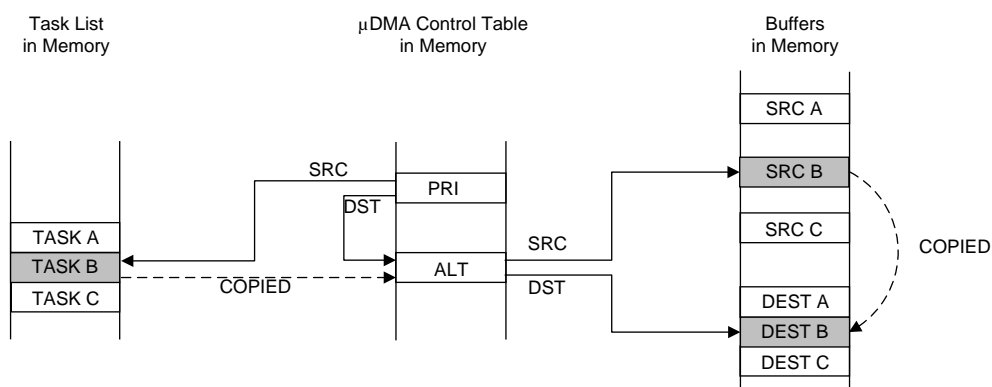
1. Application needs to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up μ DMA task list in memory, which contains the pointers and control configuration for three μ DMA copy tasks.
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the μ DMA controller.
4. The SRC and DST pointers in the task list must point to the last location in the corresponding buffer.

Figure 8-3. Memory Scatter-Gather, Setup and Configuration



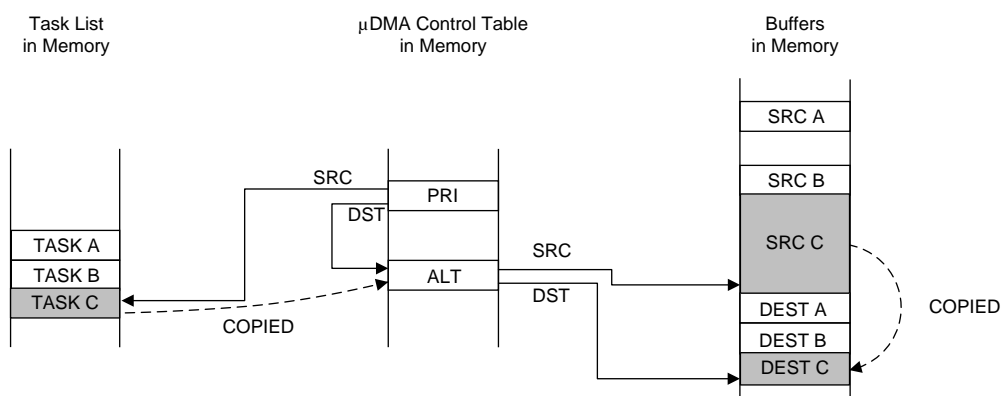
Using the channel's primary control structure, the μ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer A to the destination buffer.



Using the channel's primary control structure, the μ DMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer B to the destination buffer.



Using the channel's primary control structure, the μ DMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer C to the destination buffer.

Figure 8-4. Memory Scatter-Gather, μ DMA Copy Sequence

8.3.5.6 Peripheral Scatter-Gather

Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a μ DMA request. Upon detecting a request from the peripheral, the μ DMA controller uses the primary control structure to copy one entry from the list to the alternate control structure and then performs the transfer. At the end of this transfer, the primary control structure copies the next task to the alternate control structure. If the next task is a memory-to-memory transfer, execution starts immediately and runs to completion; if the next task is a peripheral-type transfer, the μ DMA waits for a peripheral request to begin.

By using this method, the μ DMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

For an example of operation in Peripheral Scatter-Gather mode, see [Figure 8-5](#) and [Figure 8-6](#). This example shows a gather operation, where data from three separate buffers in memory is copied to a single peripheral data register. [Figure 8-5](#) shows how the application sets up a μ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

[Figure 8-6](#) shows the sequence as the μ DMA controller performs the three sets of copy operations. First, using the primary control structure, the μ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the μ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

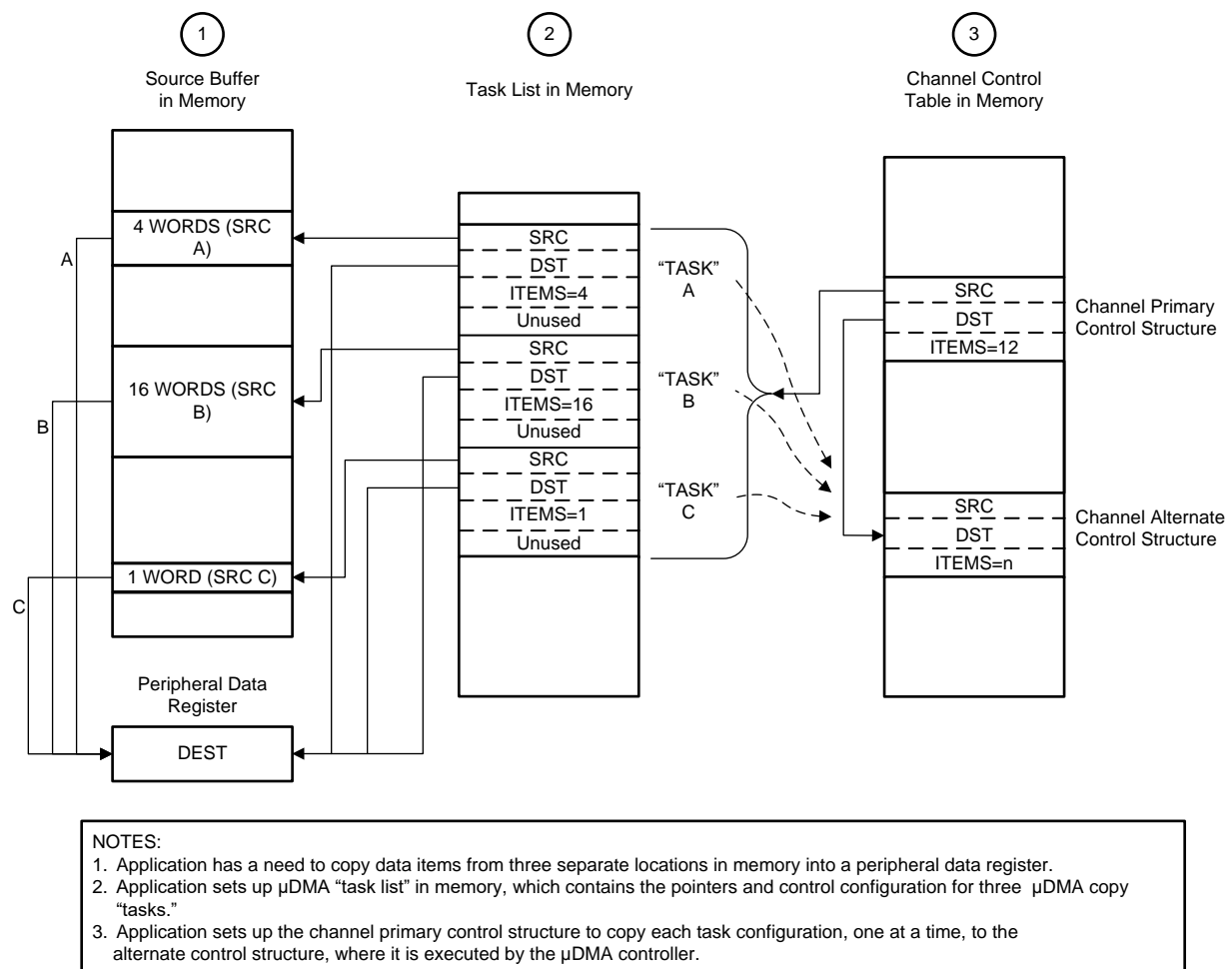


Figure 8-5. Peripheral Scatter-Gather, Setup and Configuration

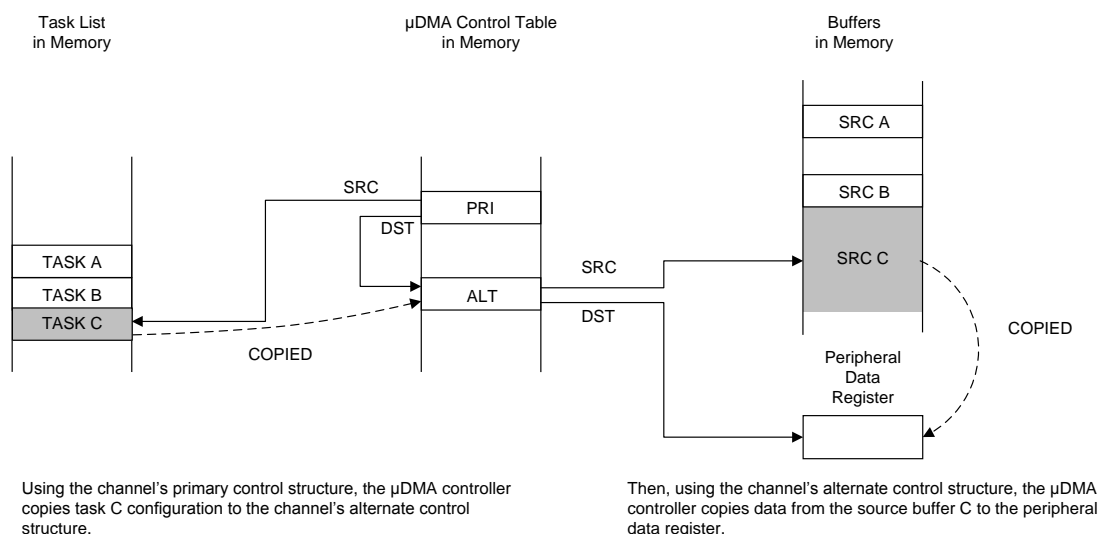
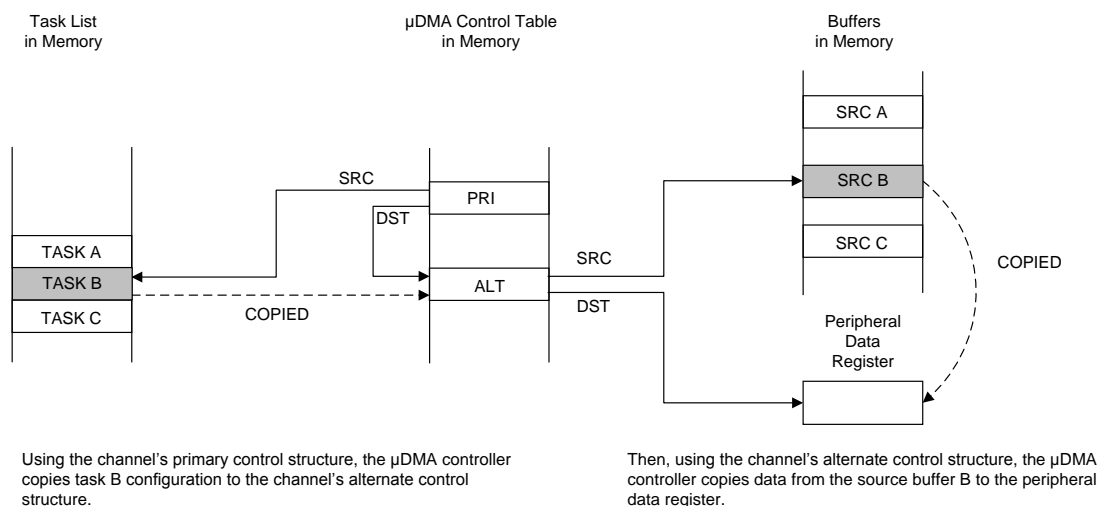
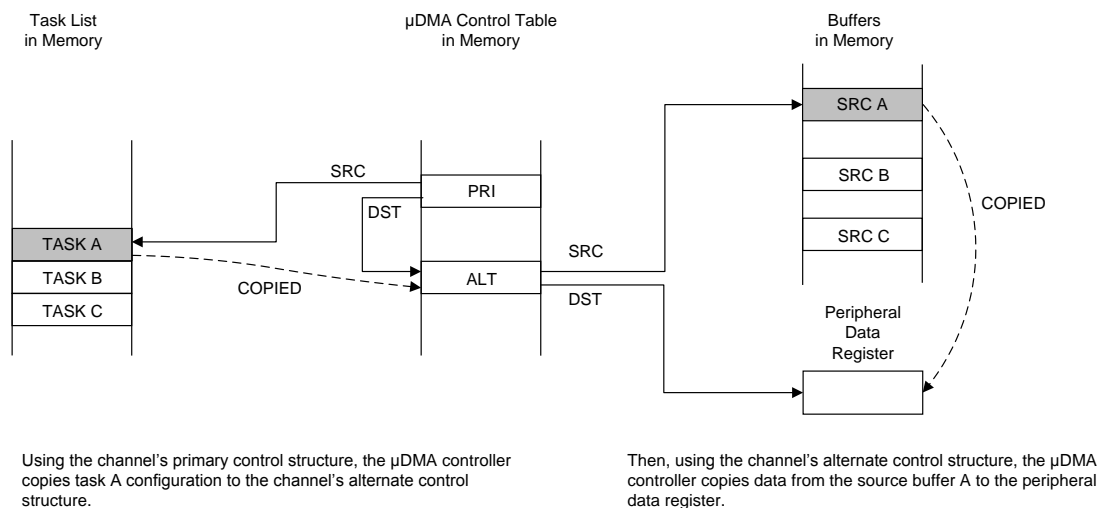


Figure 8-6. Peripheral Scatter-Gather, μDMA Copy Sequence

8.3.6 Transfer Size and Increment

The μ DMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be auto-incremented by bytes, halfwords, or words, or can be set to no increment. The source and destination address increment values can be set independently, and it is not necessary for the address increment to match the data size as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size, but using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 8-4 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 8-4. μ DMA Read Example: 8-Bit Peripheral

Field	Configuration
Source data size	8 bits
Destination data size	8 bits
Source address increment	No increment
Destination address increment	Byte
Source end pointer	Peripheral read FIFO register
Destination end pointer	End of the data buffer in memory

8.3.7 Peripheral Interface

There are two main classes of μ DMA-connected peripherals:

- Peripherals with FIFOs serviced by the μ DMA to transmit or receive data.
- Peripherals that provide trigger inputs to the μ DMA

8.3.7.1 FIFO Peripherals

FIFO peripherals contain a FIFO of data to be sent and a FIFO of data that has been received. The μ DMA controller is used to transfer data between these FIFOs and system memory. For example, when a UART FIFO contains one or more entries, a single transfer request is sent to the μ DMA for processing. If this request has not been processed and the UART FIFO reaches the interrupt FIFO level set in the UART Interrupt FIFO Level Select (UARTIFLS) register, another interrupt is sent to the μ DMA which is higher priority than the single-transfer request. In this instance, an ARBSIZ transfer is performed as configured in the DMACHCTL register. After the transfer is complete, the DMA sends a receive or transmit complete interrupt to the UART Raw Interrupt Status (UARTRIS) register.

If the FIFO peripheral's SETn bit is set in the DMA Channel Useburst Set (DMAUSEBURSTSET) register, then the μ DMA only performs transfers defined by the ARBSIZ bit field in the DMACHCTL register for better bus utilization. For peripherals that tend to transmit and receive in bursts, such as the UART, TI recommends against the use of this configuration, because it could cause the end of transmissions to stick in the FIFO.

8.3.7.2 Trigger Peripherals

Certain peripherals, such as the general-purpose timer, trigger an interrupt to the μ DMA controller when a programmed event occurs. When a trigger event occurs, the μ DMA executes a transfer defined by the ARBSIZ bit field in the DMACHCTL register. If only a single transfer is needed for a μ DMA trigger, then the ARBSIZ field is set to 0x1.

If the trigger peripheral generates another μ DMA request while the prior one is being serviced and that particular channel is the highest priority asserted channel, the second request is processed as soon as the handling of the first is complete. If two additional trigger peripheral μ DMA requests are generated before the completion of the first, the third request is lost.

8.3.8 Software Request

A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the DMA Channel Software Request (DMASWREQ) register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any available software channel using the DMASWREQ register. If a request is initiated by software using a peripheral μ DMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any peripheral channel may be used for software requests as long as the corresponding peripheral is not using μ DMA for data transfer.

NOTE: Channels designated in the table as only "Software" are dedicated software channels. When only one software request is required in an application, dedicated software channels can be used. If multiple software requests in code are required, then peripheral channel software requests should be used for proper μ DMA completion acknowledgment.

8.3.9 Interrupts and Errors

Depending on the peripheral, the μ DMA can indicate transfer completion at the end of an entire transfer or when a FIFO or buffer reaches a certain level ([Table 8-1](#) and the individual peripheral chapters). When a μ DMA transfer is complete, a dma_done signal is sent to the peripheral that initiated the μ DMA event. Interrupts can be enabled within the peripheral to trigger on μ DMA transfer completion. For more information on peripheral μ DMA interrupts, see the individual peripheral chapters. If the transfer uses the software μ DMA channel, then the completion interrupt occurs on the dedicated software μ DMA interrupt vector (see [Table 8-5](#)).

If the μ DMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it disables the μ DMA channel that caused the error and generates an interrupt on the μ DMA error interrupt vector. The processor can read the DMA Bus Error Clear (DMAERRCLR) register to determine if an error is pending. The ERRCLR bit is set if an error occurred. The error can be cleared by writing a 1 to the ERRCLR bit.

[Table 8-5](#) shows the dedicated interrupt assignments for the μ DMA controller.

Table 8-5. μ DMA Interrupt Assignments

Interrupt	Assignment
44	μ DMA software channel transfer
45	μ DMA error

8.4 Initialization and Configuration

8.4.1 Module Initialization

Before the μ DMA controller can be used, it must be enabled in the System Control block and in the peripheral. The location of the channel control structure must also be programmed.

The following steps should be performed one time during system initialization:

1. Enable the μ DMA clock using the RCGCDMA register (see [Section 4.2.88](#)).
2. Enable the μ DMA controller by setting the MASTEREN bit of the DMA Configuration (DMACFG) register.
3. Program the location of the channel control table by writing the base address of the table to the DMA Channel Control Base Pointer (DMACTLBASE) register. The base address must be aligned on a 1024-byte boundary.

8.4.2 Configuring a Memory-to-Memory Transfer

μDMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

8.4.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the DMA Channel Priority Set (DMAPRIOSET) or DMA Channel Priority Clear (DMAPRIOCLR) registers to set the channel to high priority or default priority.
2. Set bit 30 of the DMA Channel Primary Alternate Clear (DMAALTCLR) register to select the primary channel control structure for this transfer.
3. Set bit 30 of the DMA Channel Useburst Clear (DMAUSEBURSTCLR) register to allow the μDMA controller to respond to single and burst requests.
4. Set bit 30 of the DMA Channel Request Mask Clear (DMAREQMASKCLR) register to allow the μDMA controller to recognize requests for this channel.

8.4.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example transfers 256 words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in [Table 8-6](#).

Table 8-6. Channel Control Structure Offsets for Channel 30

Offset	Description
Control Table Base + 0x1E0	Channel 30 source end pointer
Control Table Base + 0x1E4	Channel 30 destination end pointer
Control Table Base + 0x1E8	Channel 30 control word

8.4.2.2.1 Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.
2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to [Table 8-7](#).

Table 8-7. Channel Control Word Configuration for Memory Transfer Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
Reserved	23:22	0	Reserved
DSTPROTO ⁽¹⁾	21	0	Privileged access protection for destination data writes
Reserved	20:19	0	Reserved
SRCPROTO ⁽¹⁾	18	0	Privileged access protection for source data reads
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

⁽¹⁾ The value of this bit must be 1 (privileged) for AES, DES, or SHA accesses.

8.4.2.2.2 Configure Peripheral Interrupts

For memory-to-memory transfers, the peripheral involved must be configured to generate an interrupt when the μ DMA has completed its transfer. Upon completion, the μ DMA sends a dma_done signal to the peripheral.

8.4.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the DMA Channel Enable Set (DMAENASET) register.
2. Issue a transfer request by setting bit 30 of the DMA Channel Software Request (DMASWREQ) register.

The μ DMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the DMAENASET register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the XFERMODE field of the channel control word at offset 0x1E8. This field is automatically cleared at the end of the transfer.

8.4.3 Configuring a Peripheral for Simple Transmit

This example configures the μ DMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses μ DMA channel 7.

8.4.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 7 of the DMA Channel Priority Set (DMAPRIOSET) or DMA Channel Priority Clear (DMAPRIOCLR) registers to set the channel to high priority or default priority.
2. Set bit 7 of the DMA Channel Primary Alternate Clear (DMAALTCLR) register to select the primary channel control structure for this transfer.
3. Set bit 7 of the DMA Channel Useburst Clear (DMAUSEBURSTCLR) register to allow the μ DMA controller to respond to single and burst requests.
4. Set bit 7 of the DMA Channel Request Mask Clear (DMAREQMASKCLR) register to allow the μ DMA controller to recognize requests for this channel.

8.4.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using μ DMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in [Table 8-8](#).

Table 8-8. Channel Control Structure Offsets for Channel 7

Offset	Description
Control Table Base + 0x070	Channel 7 source end pointer
Control Table Base + 0x074	Channel 7 destination end pointer
Control Table Base + 0x078	Channel 7 control word

8.4.3.2.1 Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
2. Program the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to [Table 8-9](#).

Table 8-9. Channel Control Word Configuration for Peripheral Transmit Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	3	Destination address does not increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	0	8-bit source address increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:22	0	Reserved
DSTPROTO ⁽¹⁾	21	0	Privileged access protection for destination data writes
reserved	20:19	0	Reserved
SRCPROTO ⁽¹⁾	18	0	Privileged access protection for source data reads
ARBSIZE	17:14	2	Arbitrates after 4 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	1	Use Basic transfer mode

⁽¹⁾ The value of this bit must be 1 (privileged) for AES, DES, or SHA accesses.

NOTE: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst SET[7] bit should be set in the DMA Channel Useburst Set (DMAUSEBURSTSET) register.

8.4.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the DMA Channel Enable Set (DMAENASET) register.

The μ DMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a μ DMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the μ DMA controller disables the channel and sets the XFERMODE field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the DMA Channel Enable Set (DMAENASET) register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the XFERMODE field of the channel control word at offset 0x078. This field is automatically cleared at the end of the transfer.

If peripheral interrupts are enabled, then the peripheral generates an interrupt when the entire transfer is complete.

8.4.4 Configuring a Peripheral for Ping-Pong Receive

This example configures the μ DMA controller to continuously receive 8-bit data from a peripheral into a pair of 64-byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral uses μ DMA channel 8.

8.4.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the DMA Channel Priority Set (DMAPRIOSET) or DMA Channel Priority Clear (DMAPRIOCLR) registers to set the channel to high priority or default priority.
2. Set bit 8 of the DMA Channel Primary Alternate Clear (DMAALTCLR) register to select the primary channel control structure for this transfer.
3. Set bit 8 of the DMA Channel Useburst Clear (DMAUSEBURSTCLR) register to allow the μ DMA

controller to respond to single and burst requests.

4. Set bit 8 of the DMA Channel Request Mask Clear (DMAREQMASKCLR) register to allow the μ DMA controller to recognize requests for this channel.

8.4.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the μ DMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in [Table 8-10](#).

Table 8-10. Primary and Alternate Channel Control Structure Offsets for Channel 8

Offset	Description
Control Table Base + 0x080	Channel 8 primary source end pointer
Control Table Base + 0x084	Channel 8 primary destination end pointer
Control Table Base + 0x088	Channel 8 primary control word
Control Table Base + 0x280	Channel 8 alternate source end pointer
Control Table Base + 0x284	Channel 8 alternate destination end pointer
Control Table Base + 0x288	Channel 8 alternate control word

8.4.4.2.1 Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.
2. Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
3. Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
4. Program the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088 and the alternate control word at offset 0x288 are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to [Table 8-11](#).
2. Program the alternate channel control word at offset 0x288 according to [Table 8-11](#).

Table 8-11. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	0	8-bit destination address increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	3	Source address does not increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:22	0	Reserved
DSTPROT0	21	0	Privileged access protection for destination data writes
reserved	20:19	0	Reserved
SRCPROT0	18	0	Privileged access protection for source data reads
ARBSIZE	17:14	3	Arbitrates after 8 transfers

Table 8-11. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example (continued)

Field in DMACHCTL	Bits	Value	Description
XFSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	3	Use Ping-Pong transfer mode

NOTE: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst SET[8] bit should be set in the DMA Channel Useburst Set (DMAUSEBURSTSET) register.

8.4.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using ping-pong mode. However, ping-pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

8.4.4.4 Enable the μ DMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the DMA Channel Enable Set (DMAENASET) register.

8.4.4.5 Process Interrupts

The μ DMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the μ DMA request signal, the μ DMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is generated in the peripheral's raw interrupt status register.

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the XFERMODE field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:
 1. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.
 2. Reprogram the primary channel control word at offset 0x88 according to [Table 8-11](#).
2. Read the alternate channel control word at offset 0x288 and check the XFERMODE field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:
 1. Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.
 2. Reprogram the alternate channel control word at offset 0x288 according to [Table 8-11](#).

8.4.5 Configuring Channel Assignments

Channel assignments for each μ DMA channel can be changed using the DMACHMAPn registers. Each 4-bit field represents a μ DMA channel. For channel assignments, see the device-specific data sheet.

For example, to use UART1 RX on channel 8, configure the CH8SEL bit in the DMACHMAP1 register to be 0x1. If a peripheral is enabled on two different channels, the μ DMA channel that has the highest priority for that peripheral takes precedence. Thus, if UART 1 RX is enabled on both channel 8 and channel 22, the UART1 RX channel 22 priority needs to be lowered before channel 8 UART1 RX can be accessed by the μ DMA.

8.5 μDMA Channel Control Structure Registers

The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, thus the base address is n/a (not applicable). The offset for the channel control structures is the offset from the entry in the channel control table.

The μDMA Channel Control Structure holds the transfer settings for a μDMA channel. Each channel has two control structures, which are located in a table in system memory. See [Section 8.3.4](#) for an explanation of the Channel Control Table and the Channel Control Structure.

The channel control structure is one entry in the channel control table. Each channel has a primary and alternate structure. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

[Table 8-18](#) lists the memory-mapped registers for the μDMA Channel Control Structure.

Table 8-12. μDMA Channel Control Structure Registers

Offset	Acronym	Register Name	Section
0x0	DMASRCENDP	DMA Channel Source Address End Pointer	Section 8.5.1
0x4	DMADSTENDP	DMA Channel Destination Address End Pointer	Section 8.5.2
0x8	DMACHCTL	DMA Channel Control Word	Section 8.5.3

Complex bit access types are encoded to fit into small table cells. [Table 8-13](#) lists the codes that are used for access types in this section.

Table 8-13. μDMA Channel Control Structure Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

8.5.1 DMASRCENDP Register (Offset = 0x0) [reset = X]

DMA Channel Source Address End Pointer (DMASRCENDP)

DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a μDMA transfer.

NOTE: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMASRCENDP is shown in [Figure 8-7](#) and described in [Table 8-14](#).

Return to [Summary Table](#).

Figure 8-7. DMASRCENDP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
R/W-X																															

Table 8-14. DMASRCENDP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	X	Source Address End Pointer This field points to the last address of the μDMA transfer source (inclusive). If the source address is not incrementing (the SRCINC field in the DMACHCTL register is 0x3), then this field points at the source location itself (such as a peripheral data register).

8.5.2 DMADSTENDP Register (Offset = 0x4) [reset = X]

DMA Channel Destination Address End Pointer (DMADSTENDP)

DMA Channel Destination Address End Pointer (DMADSTENDP) is part of the Channel Control Structure and is used to specify the destination address for a μDMA transfer.

NOTE: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMADSTENDP is shown in [Figure 8-8](#) and described in [Table 8-15](#).

Return to [Summary Table](#).

Figure 8-8. DMADSTENDP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
																ADDR																					
																R/W-X																					

Table 8-15. DMADSTENDP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	X	Destination Address End Pointer This field points to the last address of the μDMA transfer destination (inclusive). If the destination address is not incrementing (the DSTINC field in the DMACHCTL register is 0x3), then this field points at the destination location itself (such as a peripheral data register).

8.5.3 DMACHCTL Register (Offset = 0x8) [reset = X]

DMA Channel Control Word (DMACHCTL)

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure and is used to specify parameters of a μDMA transfer.

NOTE: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMACHCTL is shown in [Figure 8-9](#) and described in [Table 8-16](#).

Return to [Summary Table](#).

Figure 8-9. DMACHCTL Register

31	30	29	28	27	26	25	24
DSTINC		DSTSIZE		SRCINC		SRCSIZE	
R/W-X		R/W-X		R/W-X		R/W-X	
23	22	21	20	19	18	17	16
RESERVED		DSTPROTO	RESERVED		SRCPROTO	ARBSIZE	
R-0h		R/W-0h	R-X		R/W-0h	R/W-X	
15	14	13	12	11	10	9	8
ARBSIZE		XFERSIZE					
R/W-X		R/W-X					
7	6	5	4	3	2	1	0
XFERSIZE				NXTUSEBURST	XFERMODE		
R/W-X				R/W-X	R/W-X		

Table 8-16. DMACHCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	DSTINC	R/W	X	Destination Address Increment This field configures the destination address increment. The address increment value must be equal or greater than the value of the destination size (DSTSIZE). 0x0 = Byte; Increment by 8-bit locations 0x1 = Half-word; Increment by 16-bit locations 0x2 = Word; Increment by 32-bit locations 0x3 = No increment. Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel.
29-28	DSTSIZE	R/W	X	Destination Data Size This field configures the destination item data size. DSTSIZE must be the same as SRCSIZE. 0x0 = Byte; 8-bit data size 0x1 = Half-word; 16-bit data size 0x2 = Word; 32-bit data size 0x3 = Reserved
27-26	SRCINC	R/W	X	Source Address Increment This field configures the source address increment. The address increment value must be equal or greater than the value of the source size (SRCSIZE). 0x0 = Byte; Increment by 8-bit locations 0x1 = Half-word; Increment by 16-bit locations 0x2 = Word; Increment by 32-bit locations 0x3 = No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDP) for the channel

Table 8-16. DMACHCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
25-24	SRCSIZE	R/W	X	Source Data Size This field configures the source item data size. DSTSIZE must be the same as SRCSIZE. 0x0 = Byte; 8-bit data size. 0x1 = Half-word; 16-bit data size. 0x2 = Word; 32-bit data size. 0x3 = Reserved
23-22	RESERVED	R	0x0	
21	DSTPROTO	R/W	0x0	Destination Privilege Access This bit controls the privilege access protection for destination data writes. For AES, DES, or SHA accesses, this bit must be set to 1. 0 = The access is nonprivileged. 1 = The access is privileged.
20-19	RESERVED	R	X	
18	SRCPROTO	R/W	0x0	Source Privilege Access This bit controls the privilege access protection for source data reads. For AES, DES, or SHA accesses, this bit must be set to 1. 0 = The access is nonprivileged. 1 = The access is privileged.
17-14	ARBSIZE	R/W	X	Arbitration Size This field configures the number of transfers that can occur before the μDMA controller re-arbitrates. The possible arbitration rate configurations represent powers of 2 and are shown below. 0x0 = 1 TransferArbitrates after each μDMA transfer 0x1 = 2 Transfers 0x2 = 4 Transfers 0x3 = 8 Transfers 0x4 = 16 Transfers 0x5 = 32 Transfers 0x6 = 64 Transfers 0x7 = 128 Transfers 0x8 = 256 Transfers 0x9 = 512 Transfers 0xA-0xF = 1024 Transfers. In this configuration, no arbitration occurs during the μDMA transfer because the maximum transfer size is 1024.
13-4	XFERSIZE	R/W	X	Transfer Size (minus 1) This field configures the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items. The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer. The μDMA controller updates this field immediately before entering the arbitration process, so it contains the number of outstanding items that is necessary to complete the μDMA cycle.
3	NXTUSEBURST	R/W	X	Next Useburst This field controls whether the Useburst SET[n] bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the μDMA controller uses single transfers to complete the transaction. If this bit is set, then the controller uses a burst transfer to complete the last transfer.

Table 8-16. DMACHCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2-0	XFERMODE	R/W	X	<p>μDMA Transfer Mode</p> <p>This field configures the operating mode of the μDMA cycle. See Table 8-17 for a detailed explanation of transfer modes. Because this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled.</p> <p>0x0 = Stop 0x1 = Basic 0x2 = Auto-Request 0x3 = Ping-Pong 0x4 = Memory Scatter-Gather 0x5 = Alternate Memory Scatter-Gather 0x6 = Peripheral Scatter-Gather 0x7 = Alternate Peripheral Scatter-Gather</p>

Table 8-17. XFERMODE Bit Field Values

Mode	Description
Stop	Channel is stopped or configuration data is invalid. No more transfers can occur.
Basic	For each trigger (whether from a peripheral or a software request), the μDMA controller performs the number of transfers specified by the ARBSIZE field.
Auto-Request	The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of XFERSIZE items without any further requests.
Ping-Pong	This mode uses both the primary and alternate control structures for this channel. When the number of transfers specified by the XFERSIZE field have completed for the current control structure (primary or alternate), the μDMA controller switches to the other one. These switches continue until one of the control structures is not set to ping-pong mode. At that point, the μDMA controller stops. An interrupt is generated on completion of the transfers configured by each control structure. See Section 8.3.5.4 .
Memory Scatter-Gather	When using this mode, the primary control structure for the channel is configured to allow a list of operations (tasks) to be performed. The source address pointer specifies the start of a table of tasks to be copied to the alternate control structure for this channel. The XFERMODE field for the alternate control structure should be configured to 0x5 (Alternate memory scatter-gather) to perform the task. When the task completes, the μDMA switches back to the primary channel control structure, which then copies the next task to the alternate control structure. This process continues until the table of tasks is empty. The last task must have an XFERMODE value other than 0x5. Note that for continuous operation, the last task can update the primary channel control structure back to the start of the list or to another list. See Section 8.3.5.5 .
Alternate Memory Scatter-Gather	This value must be used in the alternate channel control data structure when the μDMA controller operates in Memory Scatter-Gather mode.
Peripheral Scatter-Gather	This value must be used in the primary channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode. In this mode, the μDMA controller operates exactly the same as in Memory Scatter-Gather mode, except that instead of performing the number of transfers specified by the XFERSIZE field in the alternate control structure at one time, the μDMA controller only performs the number of transfers specified by the ARBSIZE field per trigger; see Basic mode for details. See Section 8.3.5.6 .
Alternate Peripheral Scatter-Gather	This value must be used in the alternate channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode.

8.6 μDMA Registers

Table 8-18 lists the memory-mapped registers for the μDMA. All register offset addresses not listed in Table 8-18 should be considered as reserved locations and the register contents should not be modified.

Table 8-18. μDMA Registers

Offset	Acronym	Register Name	Section
0x0	DMASTAT	DMA Status	Section 8.6.1
0x4	DMACFG	DMA Configuration	Section 8.6.2
0x8	DMACTLBASE	DMA Channel Control Base Pointer	Section 8.6.3
0xC	DMAALTBASE	DMA Alternate Channel Control Base Pointer	Section 8.6.4
0x10	DMAWAITSTAT	DMA Channel Wait-on-Request Status	Section 8.6.5
0x14	DMAWREQ	DMA Channel Software Request	Section 8.6.6
0x18	DMAUSEBURSTSET	DMA Channel Useburst Set	Section 8.6.7
0x1C	DMAUSEBURSTCLR	DMA Channel Useburst Clear	Section 8.6.8
0x20	DMAREQMASKSET	DMA Channel Request Mask Set	Section 8.6.9
0x24	DMAREQMASKCLR	DMA Channel Request Mask Clear	Section 8.6.10
0x28	DMAENASET	DMA Channel Enable Set	Section 8.6.11
0x2C	DMAENACLAR	DMA Channel Enable Clear	Section 8.6.12
0x30	DMAALTSET	DMA Channel Primary Alternate Set	Section 8.6.13
0x34	DMAALTCLR	DMA Channel Primary Alternate Clear	Section 8.6.14
0x38	DMAPRIOSET	DMA Channel Priority Set	Section 8.6.15
0x3C	DMAPRIOCLR	DMA Channel Priority Clear	Section 8.6.16
0x4C	DMAERRCLR	DMA Bus Error Clear	Section 8.6.17
0x510	DMACHMAP0	DMA Channel Map Select 0	Section 8.6.18
0x514	DMACHMAP1	DMA Channel Map Select 1	Section 8.6.19
0x518	DMACHMAP2	DMA Channel Map Select 2	Section 8.6.20
0x51C	DMACHMAP3	DMA Channel Map Select 3	Section 8.6.21
0xFD0	DMAPeriphID4	DMA Peripheral Identification 4	Section 8.6.22
0xFE0	DMAPeriphID0	DMA Peripheral Identification 0	Section 8.6.23
0xFE4	DMAPeriphID1	DMA Peripheral Identification 1	Section 8.6.24
0xFE8	DMAPeriphID2	DMA Peripheral Identification 2	Section 8.6.25
0xFEC	DMAPeriphID3	DMA Peripheral Identification 3	Section 8.6.26
0xFF0	DMAPrimeCellID0	DMA PrimeCell Identification 0	Section 8.6.27
0xFF4	DMAPrimeCellID1	DMA PrimeCell Identification 1	Section 8.6.28
0xFF8	DMAPrimeCellID2	DMA PrimeCell Identification 2	Section 8.6.29
0xFFC	DMAPrimeCellID3	DMA PrimeCell Identification 3	Section 8.6.30

Complex bit access types are encoded to fit into small table cells. Table 8-19 shows the codes that are used for access types in this section.

Table 8-19. UDMA Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		

Table 8-19. UDMA Access Type Codes (continued)

Access Type	Code	Description
-n		Value after reset or the default value

8.6.1 DMASTAT Register (Offset = 0x0) [reset = 0x001F0000]

DMA Status (DMASTAT)

The DMA Status (DMASTAT) register returns the status of the μDMA controller. You cannot read this register when the μDMA controller is in the reset state.

DMASTAT is shown in [Figure 8-10](#) and described in [Table 8-20](#).

Return to [Summary Table](#).

Figure 8-10. DMASTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED				DMACHANS			
R-0h				R-1Fh			
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
STATE				RESERVED			MASTEN
R-0h				R-0h			R-0h

Table 8-20. DMASTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	RESERVED	R	0x0	
20-16	DMACHANS	R	0x1F	Available μDMA Channels Minus 1 This field contains a value equal to the number of μDMA channels the μDMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 μDMA channels.
15-8	RESERVED	R	0x0	
7-4	STATE	R	0x0	Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following. 0x0 = Undefined 0x1 = Reading channel controller data. 0x2 = Reading source end pointer. 0x3 = Reading destination end pointer. 0x4 = Reading source data. 0x5 = Writing destination data. 0x6 = Waiting for μDMA request to clear. 0x7 = Writing channel controller data. 0x8 = Stalled 0x9 = Done
3-1	RESERVED	R	0x0	
0	MASTEN	R	0x0	Master Enable Status 0x0 = The μDMA controller is disabled. 0x1 = The μDMA controller is enabled.

8.6.2 DMACFG Register (Offset = 0x4) [reset = X]

DMA Configuration (DMACFG)

The DMACFG register controls the configuration of the μDMA controller.

DMACFG is shown in [Figure 8-11](#) and described in [Table 8-21](#).

Return to [Summary Table](#).

Figure 8-11. DMACFG Register

31	30	29	28	27	26	25	24
RESERVED							
W-X							
23	22	21	20	19	18	17	16
RESERVED							
W-X							
15	14	13	12	11	10	9	8
RESERVED							
W-X							
7	6	5	4	3	2	1	0
RESERVED							MASTEN
W-X							W-X

Table 8-21. DMACFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	W	X	
0	MASTEN	W	X	Controller Master Enable 0x0 = Disables the μDMA controller. 0x1 = Enables μDMA controller.

8.6.3 DMACTLBASE Register (Offset = 0x8) [reset = 0x0]

DMA Channel Control Base Pointer (DMACTLBASE)

The DMACTLBASE register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that must be assigned to the μDMA controller depends on the number of μDMA channels used and whether the alternate channel control data structure is used. See [Section 8.3.4](#) for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the μDMA controller is in the reset state.

DMACTLBASE is shown in [Figure 8-12](#) and described in [Table 8-22](#).

Return to [Summary Table](#).

Figure 8-12. DMACTLBASE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR												RESERVED																			
R/W-0h												R-0h																			

Table 8-22. DMACTLBASE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	ADDR	R/W	0x0	Channel Control Base Address This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned.
9-0	RESERVED	R	0x0	

8.6.4 DMAALTBASE Register (Offset = 0xC) [reset = 0x200]

DMA Alternate Channel Control Base Pointer (DMAALTBASE)

The DMAALTBASE register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. This register cannot be read when the μDMA controller is in the reset state.

DMAALTBASE is shown in [Figure 8-13](#) and described in [Table 8-23](#).

Return to [Summary Table](#).

Figure 8-13. DMAALTBASE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
R-200h																															

Table 8-23. DMAALTBASE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	0x200	Alternate Channel Address Pointer This field provides the base address of the alternate channel control structures.

8.6.5 DMAWAITSTAT Register (Offset = 0x10) [reset = 0x03C3CF00]

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

This read-only register indicates that the μDMA channel is waiting on a request. A peripheral can hold off the μDMA from performing a single request until the peripheral is ready for a burst request to enhance the μDMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the μDMA controller is in the reset state.

DMAWAITSTAT is shown in [Figure 8-14](#) and described in [Table 8-24](#).

Return to [Summary Table](#).

Figure 8-14. DMAWAITSTAT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAITREQ[n]																															
R-03C3CF00h																															

Table 8-24. DMAWAITSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WAITREQ[n]	R	0x03C3CF00	Channel [n] Wait Status These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0. 0x0 = The corresponding channel is not waiting on a request. 0x1 = The corresponding channel is waiting on a request.

8.6.6 DMASWREQ Register (Offset = 0x14) [reset = X]

DMA Channel Software Request (DMASWREQ)

Each bit of the DMASWREQ register represents the corresponding μDMA channel. Setting a bit generates a request for the specified μDMA channel.

DMASWREQ is shown in [Figure 8-15](#) and described in [Table 8-25](#).

Return to [Summary Table](#).

Figure 8-15. DMASWREQ Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREQ[n]																															
W-X																															

Table 8-25. DMASWREQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SWREQ[n]	W	X	Channel [n] Software Request These bits generate software requests. Bit 0 corresponds to channel 0. These bits are automatically cleared when the software request has been completed. 0x0 = No request generated. 0x1 = Generate a software request for the corresponding channel.

8.6.7 DMAUSEBURSTSET Register (Offset = 0x18) [reset = 0x0]

DMA Channel Useburst Set (DMAUSEBURSTSET)

Each bit of the DMAUSEBURSTSET register represents the corresponding μDMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

If the amount of data to transfer is a multiple of the arbitration (burst) size, the corresponding SET[n] bit is cleared after completing the final transfer. If there are fewer items remaining to transfer than the arbitration (burst) size, the μDMA controller automatically clears the corresponding SET[n] bit, allowing the remaining items to transfer using single requests. In order to resume transfers using burst requests, the corresponding bit must be set again. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to [Section 8.3.3](#) for more details about request types.

DMAUSEBURSTSET is shown in [Figure 8-16](#) and described in [Table 8-26](#).

Return to [Summary Table](#).

Figure 8-16. DMAUSEBURSTSET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET[n]																															
R/W-0h																															

Table 8-26. DMAUSEBURSTSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SET[n]	R/W	0x0	Channel [n] Useburst Set Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding CLR[n] bit in the DMAUSEBURSTCLR register. 0x0 = μDMA channel [n] responds to single or burst requests. 0x1 = μDMA channel [n] responds only to burst requests.

8.6.8 DMAUSEBURSTCLR Register (Offset = 0x1C) [reset = X]

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Each bit of the DMAUSEBURSTCLR register represents the corresponding μDMA channel. Setting a bit clears the corresponding SET[n] bit in the DMAUSEBURSTSET register.

DMAUSEBURSTCLR is shown in [Figure 8-17](#) and described in [Table 8-27](#).

Return to [Summary Table](#).

Figure 8-17. DMAUSEBURSTCLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR[n]																															
W-X																															

Table 8-27. DMAUSEBURSTCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLR[n]	W	X	Channel [n] Useburst Clear 0x0 = No effect. 0x1 = Setting a bit clears the corresponding SET[n] bit in the DMAUSEBURSTSET register meaning that μDMA channel [n] responds to single and burst requests.

8.6.9 DMAREQMASKSET Register (Offset = 0x20) [reset = 0x0]

DMA Channel Request Mask Set (DMAREQMASKSET)

Each bit of the DMAREQMASKSET register represents the corresponding μDMA channel. Setting a bit disables μDMA requests for the channel. Reading the register returns the request mask status. When a μDMA channel's request is masked, that means the peripheral can no longer request μDMA transfers. The channel can then be used for software-initiated transfers.

DMAREQMASKSET is shown in [Figure 8-18](#) and described in [Table 8-28](#).

Return to [Summary Table](#).

Figure 8-18. DMAREQMASKSET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET[n]																															
R/W-0h																															

Table 8-28. DMAREQMASKSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SET[n]	R/W	0x0	<p>Channel [n] Request Mask Set</p> <p>Bit 0 corresponds to channel 0.</p> <p>A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAREQMASKCLR register.</p> <p>0x0 = The peripheral associated with channel [n] is enabled to request μDMA transfers.</p> <p>0x1 = The peripheral associated with channel [n] is not able to request μDMA transfers. Channel [n] may be used for software-initiated transfers.</p>

8.6.10 DMAREQMASKCLR Register (Offset = 0x24) [reset = X]

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Each bit of the DMAREQMASKCLR register represents the corresponding μDMA channel. Setting a bit clears the corresponding SET[n] bit in the DMAREQMASKSET register.

DMAREQMASKCLR is shown in [Figure 8-19](#) and described in [Table 8-29](#).

Return to [Summary Table](#).

Figure 8-19. DMAREQMASKCLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR[n]																															
W-X																															

Table 8-29. DMAREQMASKCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLR[n]	W	X	Channel [n] Request Mask Clear 0x0 = No effect. 0x1 = Setting a bit clears the corresponding SET[n] bit in the DMAREQMASKSET register meaning that the peripheral associated with channel [n] is enabled to request μDMA transfers.

8.6.11 DMAENASET Register (Offset = 0x28) [reset = 0x0]

DMA Channel Enable Set (DMAENASET)

Each bit of the DMAENASET register represents the corresponding μDMA channel. Setting a bit enables the corresponding μDMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (DMAREQMASKSET), then the channel can be used for software-initiated transfers.

DMAENASET is shown in [Figure 8-20](#) and described in [Table 8-30](#).

Return to [Summary Table](#).

Figure 8-20. DMAENASET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET[n]																															
R/W-0h																															

Table 8-30. DMAENASET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SET[n]	R/W	0x0	Channel [n] Enable Set Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAENACLR register or when the end of a μDMA transfer occurs. 0x0 = μDMA Channel [n] is disabled. 0x1 = μDMA Channel [n] is enabled.

8.6.12 DMAENACLR Register (Offset = 0x2C) [reset = X]

DMA Channel Enable Clear (DMAENACLR)

Each bit of the DMAENACLR register represents the corresponding μDMA channel. Setting a bit clears the corresponding SET[n] bit in the DMAENASET register.

DMAENACLR is shown in [Figure 8-21](#) and described in [Table 8-31](#).

Return to [Summary Table](#).

Figure 8-21. DMAENACLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR[n]																															
W-X																															

Table 8-31. DMAENACLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLR[n]	W	X	<p>Clear Channel [n] Enable Clear</p> <p>The controller disables a channel when it completes the μDMA cycle.</p> <p>0x0 = No effect.</p> <p>0x1 = Setting a bit clears the corresponding SET[n] bit in the DMAENASET register meaning that channel [n] is disabled for μDMA transfers.</p>

8.6.13 DMAALTSET Register (Offset = 0x30) [reset = 0x0]

DMA Channel Primary Alternate Set (DMAALTSET)

Each bit of the DMAALTSET register represents the corresponding μDMA channel. Setting a bit configures the μDMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding μDMA channel.

DMAALTSET is shown in [Figure 8-22](#) and described in [Table 8-32](#).

Return to [Summary Table](#).

Figure 8-22. DMAALTSET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET[n]																															
R/W-0h																															

Table 8-32. DMAALTSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SET[n]	R/W	0x0	<p>Channel [n] Alternate Set</p> <p>Bit 0 corresponds to channel 0.</p> <p>A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAALTCLR register.</p> <p>For Ping-Pong and Scatter-Gather cycle types, the μDMA controller automatically sets these bits to select the alternate channel control data structure.</p> <p>0x0 = μDMA channel [n] is using the primary control structure.</p> <p>0x1 = μDMA channel [n] is using the alternate control structure.</p>

8.6.14 DMAALTCLR Register (Offset = 0x34) [reset = X]

DMA Channel Primary Alternate Clear (DMAALTCLR)

Each bit of the DMAALTCLR register represents the corresponding μDMA channel. Setting a bit clears the corresponding SET[n] bit in the DMAALTSET register.

DMAALTCLR is shown in [Figure 8-23](#) and described in [Table 8-33](#).

Return to [Summary Table](#).

Figure 8-23. DMAALTCLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR[n]																															
W-X																															

Table 8-33. DMAALTCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLR[n]	W	X	<p>Channel [n] Alternate Clear</p> <p>For Ping-Pong and Scatter-Gather cycle types, the μDMA controller automatically sets these bits to select the alternate channel control data structure.</p> <p>0x0 = No effect.</p> <p>0x1 = Setting a bit clears the corresponding SET[n] bit in the DMAALTSET register meaning that channel [n] is using the primary control structure.</p>

8.6.15 DMAPRIOSET Register (Offset = 0x38) [reset = 0x0]

DMA Channel Priority Set (DMAPRIOSET)

Each bit of the DMAPRIOSET register represents the corresponding μDMA channel. Setting a bit configures the μDMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMAPRIOSET is shown in [Figure 8-24](#) and described in [Table 8-34](#).

Return to [Summary Table](#).

Figure 8-24. DMAPRIOSET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET[n]																															
R/W-0h																															

Table 8-34. DMAPRIOSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SET[n]	R/W	0x0	Channel [n] Priority Set Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAPRIOCLR register. 0x0 = μDMA channel [n] is using the default priority level. 0x1 = μDMA channel [n] is using a high priority level.

8.6.16 DMAPRIOCLR Register (Offset = 0x3C) [reset = X]

DMA Channel Priority Clear (DMAPRIOCLR)

Each bit of the DMAPRIOCLR register represents the corresponding μDMA channel. Setting a bit clears the corresponding SET[n] bit in the DMAPRIOSET register.

DMAPRIOCLR is shown in [Figure 8-25](#) and described in [Table 8-35](#).

Return to [Summary Table](#).

Figure 8-25. DMAPRIOCLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR[n]																															
W-X																															

Table 8-35. DMAPRIOCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLR[n]	W	X	Channel [n] Priority Clear 0x0 = No effect. 0x1 = Setting a bit clears the corresponding SET[n] bit in the DMAPRIOSET register meaning that channel [n] is using the default priority level.

8.6.17 DMAERRCLR Register (Offset = 0x4C) [reset = 0x0]

DMA Bus Error Clear (DMAERRCLR)

The DMAERRCLR register is used to read and clear the μDMA bus error status. The error status is set if the μDMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the μDMA controller. The other channels are unaffected.

DMAERRCLR is shown in [Figure 8-26](#) and described in [Table 8-36](#).

Return to [Summary Table](#).

Figure 8-26. DMAERRCLR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							ERRCLR
R-0h							R/W1C-0h

Table 8-36. DMAERRCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	ERRCLR	R/W1C	0x0	μDMA Bus Error Status This bit is cleared by writing a 1 to it. 0x0 = No bus error is pending. 0x1 = A bus error is pending.

8.6.18 DMACHMAP0 Register (Offset = 0x510) [reset = 0x0]

DMA Channel Map Select 0 (DMACHMAP0)

Each 4-bit field of the DMACHMAP0 register configures the μDMA channel assignment as specified in .

DMACHMAP0 is shown in [Figure 8-27](#) and described in [Table 8-37](#).

Return to [Summary Table](#).

Figure 8-27. DMACHMAP0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH7SEL				CH6SEL				CH5SEL				CH4SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3SEL				CH2SEL				CH1SEL				CH0SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

Table 8-37. DMACHMAP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	CH7SEL	R/W	0x0	μDMA Channel 7 Source Select See for channel assignments.
27-24	CH6SEL	R/W	0x0	μDMA Channel 6 Source Select See for channel assignments.
23-20	CH5SEL	R/W	0x0	μDMA Channel 5 Source Select See for channel assignments.
19-16	CH4SEL	R/W	0x0	μDMA Channel 4 Source Select See for channel assignments.
15-12	CH3SEL	R/W	0x0	μDMA Channel 3 Source Select See for channel assignments.
11-8	CH2SEL	R/W	0x0	μDMA Channel 2 Source Select See for channel assignments.
7-4	CH1SEL	R/W	0x0	μDMA Channel 1 Source Select See for channel assignments.
3-0	CH0SEL	R/W	0x0	μDMA Channel 0 Source Select See for channel assignments.

8.6.19 DMACHMAP1 Register (Offset = 0x514) [reset = 0x0]

DMA Channel Map Select 1 (DMACHMAP1)

Each 4-bit field of the DMACHMAP1 register configures the μDMA channel assignment as specified in .

DMACHMAP1 is shown in [Figure 8-28](#) and described in [Table 8-38](#).

Return to [Summary Table](#).

Figure 8-28. DMACHMAP1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH15SEL				CH14SEL				CH13SEL				CH12SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH11SEL				CH10SEL				CH9SEL				CH8SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

Table 8-38. DMACHMAP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	CH15SEL	R/W	0x0	μDMA Channel 15 Source Select See for channel assignments.
27-24	CH14SEL	R/W	0x0	μDMA Channel 14 Source Select See for channel assignments.
23-20	CH13SEL	R/W	0x0	μDMA Channel 13 Source Select See for channel assignments.
19-16	CH12SEL	R/W	0x0	μDMA Channel 12 Source Select See for channel assignments.
15-12	CH11SEL	R/W	0x0	μDMA Channel 11 Source Select See for channel assignments.
11-8	CH10SEL	R/W	0x0	μDMA Channel 10 Source Select See for channel assignments.
7-4	CH9SEL	R/W	0x0	μDMA Channel 9 Source Select See for channel assignments.
3-0	CH8SEL	R/W	0x0	μDMA Channel 8 Source Select See for channel assignments.

8.6.20 DMACHMAP2 Register (Offset = 0x518) [reset = 0x0]

DMA Channel Map Select 2 (DMACHMAP2)

Each 4-bit field of the DMACHMAP2 register configures the μDMA channel assignment as specified in .

DMACHMAP2 is shown in [Figure 8-29](#) and described in [Table 8-39](#).

Return to [Summary Table](#).

Figure 8-29. DMACHMAP2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH23SEL				CH22SEL				CH21SEL				CH20SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH19SEL				CH18SEL				CH17SEL				CH16SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

Table 8-39. DMACHMAP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	CH23SEL	R/W	0x0	μDMA Channel 23 Source Select See for channel assignments.
27-24	CH22SEL	R/W	0x0	μDMA Channel 22 Source Select See for channel assignments.
23-20	CH21SEL	R/W	0x0	μDMA Channel 21 Source Select See for channel assignments.
19-16	CH20SEL	R/W	0x0	μDMA Channel 20 Source Select See for channel assignments.
15-12	CH19SEL	R/W	0x0	μDMA Channel 19 Source Select See for channel assignments.
11-8	CH18SEL	R/W	0x0	μDMA Channel 18 Source Select See for channel assignments.
7-4	CH17SEL	R/W	0x0	μDMA Channel 17 Source Select See for channel assignments.
3-0	CH16SEL	R/W	0x0	μDMA Channel 16 Source Select See for channel assignments.

8.6.21 DMACHMAP3 Register (Offset = 0x51C) [reset = 0x0]

DMA Channel Map Select 3 (DMACHMAP3)

Each 4-bit field of the DMACHMAP3 register configures the μDMA channel assignment as specified in .

DMACHMAP3 is shown in [Figure 8-30](#) and described in [Table 8-40](#).

Return to [Summary Table](#).

Figure 8-30. DMACHMAP3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH31SEL				CH30SEL				CH29SEL				CH28SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH27SEL				CH26SEL				CH25SEL				CH24SEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

Table 8-40. DMACHMAP3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	CH31SEL	R/W	0x0	μDMA Channel 31 Source Select See for channel assignments.
27-24	CH30SEL	R/W	0x0	μDMA Channel 30 Source Select See for channel assignments.
23-20	CH29SEL	R/W	0x0	μDMA Channel 29 Source Select See for channel assignments.
19-16	CH28SEL	R/W	0x0	μDMA Channel 28 Source Select See for channel assignments.
15-12	CH27SEL	R/W	0x0	μDMA Channel 27 Source Select See for channel assignments.
11-8	CH26SEL	R/W	0x0	μDMA Channel 26 Source Select See for channel assignments.
7-4	CH25SEL	R/W	0x0	μDMA Channel 25 Source Select See for channel assignments.
3-0	CH24SEL	R/W	0x0	μDMA Channel 24 Source Select See for channel assignments.

8.6.22 DMAPeriphID4 Register (Offset = 0xFD0) [reset = 0x4]

DMA Peripheral Identification 4 (DMAPeriphID4)

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPeriphID4 is shown in [Figure 8-31](#) and described in [Table 8-41](#).

Return to [Summary Table](#).

Figure 8-31. DMAPeriphID4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID4															
R-0h																R-4h															

Table 8-41. DMAPeriphID4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID4	R	0x4	μDMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

8.6.23 DMAPeriphID0 Register (Offset = 0xFE0) [reset = 0x30]

DMA Peripheral Identification 0 (DMAPeriphID0)

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPeriphID0 is shown in [Figure 8-32](#) and described in [Table 8-42](#).

Return to [Summary Table](#).

Figure 8-32. DMAPeriphID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PID0																	
R-0h														R-30h																	

Table 8-42. DMAPeriphID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID0	R	0x30	μDMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

8.6.24 DMAPeriphID1 Register (Offset = 0xFE4) [reset = 0xB2]

DMA Peripheral Identification 1 (DMAPeriphID1)

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPeriphID1 is shown in [Figure 8-33](#) and described in [Table 8-43](#).

Return to [Summary Table](#).

Figure 8-33. DMAPeriphID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID1															
R-0h																R-B2h															

Table 8-43. DMAPeriphID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID1	R	0xB2	μDMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

8.6.25 DMAPeriphID2 Register (Offset = 0xFE8) [reset = 0xB]

DMA Peripheral Identification 2 (DMAPeriphID2)

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPeriphID2 is shown in [Figure 8-34](#) and described in [Table 8-44](#).

Return to [Summary Table](#).

Figure 8-34. DMAPeriphID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID2							
R-0h																								R-Bh							

Table 8-44. DMAPeriphID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID2	R	0xB	μDMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

8.6.26 DMAPeriphID3 Register (Offset = 0xFEC) [reset = 0x0]

DMA Peripheral Identification 3 (DMAPeriphID3)

The DMAPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

DMAPeriphID3 is shown in [Figure 8-35](#) and described in [Table 8-45](#).

Return to [Summary Table](#).

Figure 8-35. DMAPeriphID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID3							
R-0h																								R-0h							

Table 8-45. DMAPeriphID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID3	R	0x0	μDMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

8.6.27 DMAPCellID0 Register (Offset = 0xFF0) [reset = 0xD]

DMA PrimeCell Identification 0 (DMAPCellID0)

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPCellID0 is shown in [Figure 8-36](#) and described in [Table 8-46](#).

Return to [Summary Table](#).

Figure 8-36. DMAPCellID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID0							
R-0h																								R-Dh							

Table 8-46. DMAPCellID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID0	R	0xD	μDMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

8.6.28 DMAPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]

DMA PrimeCell Identification 1 (DMAPCellID1)

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPCellID1 is shown in [Figure 8-37](#) and described in [Table 8-47](#).

Return to [Summary Table](#).

Figure 8-37. DMAPCellID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID1							
R-0h																								R-F0h							

Table 8-47. DMAPCellID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID1	R	0xF0	μDMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

8.6.29 DMAPCellID2 Register (Offset = 0xFF8) [reset = 0x5]

DMA PrimeCell Identification 2 (DMAPCellID2)

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPCellID2 is shown in [Figure 8-38](#) and described in [Table 8-48](#).

Return to [Summary Table](#).

Figure 8-38. DMAPCellID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID2							
R-0h																								R-5h							

Table 8-48. DMAPCellID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID2	R	0x5	μDMA PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

8.6.30 DMAPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]

DMA PrimeCell Identification 3 (DMAPCellID3)

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

DMAPCellID3 is shown in [Figure 8-39](#) and described in [Table 8-49](#).

Return to [Summary Table](#).

Figure 8-39. DMAPCellID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID3							
R-0h																								R-B1h							

Table 8-49. DMAPCellID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID3	R	0xB1	μDMA PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

Advance Encryption Standard Accelerator (AES)

This section describes the advanced encryption standard (AES) cryptographic hardware-accelerated module.

Topic	Page
9.1 AES Overview	659
9.2 AES Functional Description	659
9.3 AES Performance Information	672
9.4 AES Module Programming Guide	674
9.5 AES Registers	679
9.6 AES μ DMA Registers	697

9.1 AES Overview

This section introduces the AES and describes its main functions and connections in the device.

The advanced encryption standard (AES) security modules provide hardware-accelerated data encryption and decryption operations based on a binary key. The AES is a symmetric cipher module that supports a 128-bit, 192-bit, or 256-bit key in hardware for both encryption and decryption. The AES module is based on a symmetric algorithm, meaning that the encryption and decryption keys are identical. To encrypt data means to convert it from plain text to an unintelligible form called cipher text. Decrypting cipher text converts previously encrypted data back to its original plain text form.

The main features of the AES accelerator are as follows:

- Support for basic AES encryption and decryption operations:
 - Galois/counter mode (GCM), with basic GHASH operation
 - Counter mode with CBC-MAC (CCM)
 - XTS mode
- Availability of the following feedback operating modes:
 - Electronic code book mode (ECB)
 - Cipher block chaining mode (CBC)
 - Counter mode (CTR)
 - Cipher feedback mode (CFB), 128-bit
 - F8 mode
- Key sizes: 128, 192, and 256 bits
- Support for CBC_MAC and Fedora 9 (F9) authentication modes
- Basic GHASH operation (when selecting no encryption)
- Key scheduling in hardware
- Support for μ DMA transfers
- Fully synchronous design

9.2 AES Functional Description

The following sections describe the features of the AES Module.

9.2.1 AES Block Diagram

[Figure 9-1](#) shows the AES block diagram. A single-core dual-interface architecture is used.

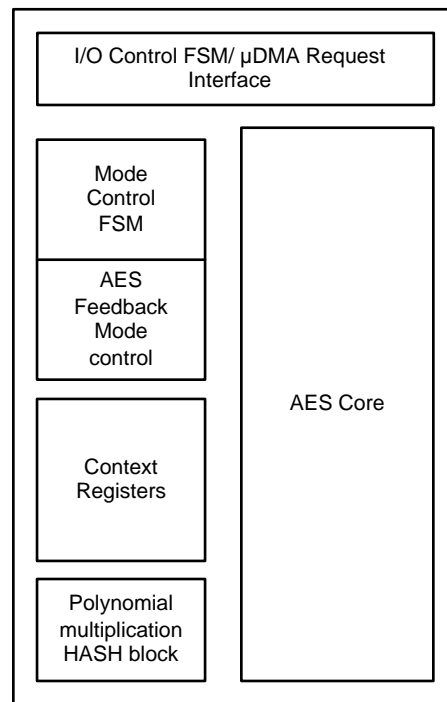


Figure 9-1. AES Block Diagram

AES is an efficient implementation of the Rijndael cipher (the AES algorithm) and a 128-bit polynomial multiplication (referred to here as GHASH, as per the AES-GCM specification). Rijndael is a block cipher in which each data block is 128 bits. The polynomial multiplication multiplies two 128-bit vectors using the smallest 128-bit irreducible polynomial, represented by the following 128-bit string: $\{0^{120}\}||10000111$. The two implementations are combined into the AES wide-bus engine.

Depending on the availability of context and data, the AES wide-bus engine is automatically triggered to process the data. The AES wide-bus engine is directly connected to the context and data registers so that it can immediately start processing when all data is available. The AES wide-bus engine also interfaces to the I/O control FSM/ μ DMA request interface.

AES comprises the following major functional blocks:

- Global control FSM and μ DMA interface
- Register interface module
- AES wide-bus engine

The AES wide-bus engine, which is the major top-level component, comprises the following functional blocks:

- Mode control FSM: Manages the data flow to and from the AES wide-bus engine and starts each encryption or decryption operation.
- Feedback modes: The logic that implements the various feedback modes supported by AES.
- GHASH core: The polynomial multiplication algorithm used for AES-GCM.
- AES key scheduler: Generates AES encryption and decryption (round) keys.
- AES encryption core: The AES encryption algorithm
- AES decryption core: The AES decryption algorithm
- Substitution-boxes (S-Boxes): Contains AES S-Box $GF(2^8)$ implementations

AES encryption requires a specific number of rounds, depending on the key length. The supported key lengths are 128, 192, and 256 bits, which require 10, 12, and 14 rounds, respectively, or 32, 38, and 44 clock cycles, respectively, because $\{\text{number of clock cycles}\} = 2 + 3 \times \{\text{number of rounds}\}$.

The larger key lengths provide greater encryption strength at the expense of additional rounds, and therefore reduced throughput. The overall throughput of the AES executing polynomial multiplication is adjusted based on the overall cryptographic performance. The AES module contains one electronic codebook (ECB) core and a dedicated 32-cycle polynomial multiplication module for performing GHASH operations. Polynomial multiplication operates in parallel with the AES core, if data is available for both modules.

Depending on the key size (128, 192, or 256 bits), this core requires 32, 38, or 44 clock cycles to process one 128-bit data block. While one data block processes, the next block can be immediately preloaded. When a block is preloaded, the previous block must finish before additional data can be loaded. Therefore, once the pipeline is full, sequential data blocks can be passed every 32, 38, or 44 clock cycles.

9.2.1.1 Interfaces

The interface signals to the AES module can be grouped into the following four categories:

- Clock signals
- Reset signals
- Register interface
- μ DMA/INT interface, used to request new context and packet data or to indicate available result data (encrypted and decrypted data or authentication result)

9.2.1.2 Register Interface

The register interface block performs all address decoding and control. However, not all registers are located in this block. The context and data input registers are in the AES wide-bus engine. The data output registers are available in this block.

9.2.1.3 AES Wide-Bus Engine

The AES wide-bus engine performs the cryptographic operations. The composition of the AES core is as follows:

- The main data path operates on the input block, performing the required substitution, shift, and mix operations.
- The key scheduler generates the round keys. A new subkey is generated and XORed with the data each round.

9.2.1.3.1 AES Key Scheduler

The AES key scheduler generates the round keys. During each round, a new subkey is generated from the input key to be XORed with the data. Round keys are generated on-the-fly and parallel to data processing to minimize register requirements.

For encryption operations, the key sequencer transfers the initial key data to the AES core. For decryption operations, the key scheduler must provide the final subkey to the AES core so it can generate the subkeys in reverse order.

9.2.1.3.2 AES Encryption Core

The AES encryption core implements the Rijndael algorithm as specified in [FIPS-197]. This core operates on the input block and performs the required substitution, shift, and mix operations. For each round, the encryption core receives the proper round key from the AES key scheduler. A fundamental component of the AES algorithm is the S-Box. The S-Box provides a unique 8-bit output for each 8-bit input. This implementation of the AES encryption core has a 64-bit data path.

9.2.1.3.3 AES Decryption Core

The architecture of the AES decryption core is generally the same as the architecture of the encryption core. One difference is that the generation of round keys for decryption requires an initial conversion of the input key (always supplied by the host in the form of an encryption key) to the corresponding decryption key. This conversion is done by performing a dummy encryption operation and storing the final round key as a decryption key. The key scheduler is then reversed to generate the round keys for the decryption operation. Consequently, for each sequence of decryption operations under the same key, a single throughput reduction equal to the time to encrypt a single block occurs. Once a decryption key is generated, subsequent decryption operations with the same key use this generated decryption key directly.

9.2.1.3.4 AES Feedback Mode Block

AES feedback mode block buffers the feedback parameters and controls the various feedback modes. For more information about the ECB, CBC, CTR, and CFB modes of operation, see the [NIST-SP800-38A](#) specification.

CTR implements the standard incrementing function, as described in the NIST-SP800-38A specification, with m set to 16 or a multiple of 32.

AES-XTS mode requires a polynomial multiplication for initialization vector (IV) generation of the AES operation. This multiplication can be simplified when the first result is available due to the definition and use of the block number within a unit. The input for the polynomial multiplication is not directly j , but α^j , where $\alpha = x^2$ in the $GF(2^{128})$ domain.

In addition, F8 encryption or decryption mode, F9, and (X)CBC-MAC authentication modes are available.

9.2.1.3.5 GHASH Block

The data sequencer manages the data flow to and from the AES core. For data input, the data sequencer monitors the input buffer until a 16-byte block is available. If the AES core is idle, the data sequencer writes this data block to the internal working registers of the AES core, thus clearing the buffer for the next block.

After completing an encryption or decryption operation, the data sequencer writes the AES output to the output buffer. If the output buffer is full at the time of completion, the AES core is held until the buffer clears. Although the data sequencer is designed to support uninterrupted packet encryption, the host must properly manage the input and output packet buffers to achieve optimal performance.

9.2.2 AES Algorithm

The AES algorithm generates block ciphers. The AES block size is 16 bytes. The AES key(s) can be coded on 128, 192, or 256 bits. Larger key sizes provide a higher level of security, but at the cost of a moderate decrease in throughput.

For the AES algorithm:

- The length of the input and output blocks is 128 bits, which is represented by $N_b = 4$, which reflects the number of 32-bit words.
- The length of the cipher key (K) is 128, 192, or 256 bits. The key length is represented by $N_k = 4, 6$, or 8 , which reflects the number of 32-bit words in the cipher key.
- The number of rounds to be performed during execution of the algorithm is dependent on the key size. The number of rounds is represented by N_r , where:
 - $N_r = 10$ when $N_k = 4$ (128-bit key)
 - $N_r = 12$ when $N_k = 6$ (192-bit key)
 - $N_r = 14$ when $N_k = 8$ (256-bit key)

[Table 9-1](#) lists the combinations of keys, blocks, and rounds.

Table 9-1. Key-Block Round Combinations

Key	Key Length (Nk)	Block Size (Nb)	Number of Rounds (Nr)
128 bits	4	4	10
192 bits	6	4	12
256 bits	8	4	14

The AES algorithm for cipher and inverse cipher uses a round function composed of four different byte-oriented transformations, as follows:

- **Byte substitution using a substitution table (S-Box)**
This transformation is a nonlinear byte substitution that operates independently on each byte of the state (the state is an intermediate processed block of 128 bits inside the AES, arranged as an array of $[4 \times N_k]$ bytes) using an S-Box. This S-Box transformation is reversible.
- **Shifting rows of the state array by different offsets**
In this transformation, the bytes in the last three rows of the state are cyclically shifted over different numbers of bytes (offsets). The first row ($r = 0$) is not shifted.
- **Mixing the data within each column of the state array**
This transformation operates on the state column-by-column, treating each column as a four-term polynomial. The columns are considered polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$.
- **Adding a round key to the state**
In this transformation, a round key is added to the state using a simple bitwise XOR operation. Each round key consists of N_b words from the key schedule.

The AES algorithm takes the cipher key (K) and performs a key expansion routine to generate a key schedule. The key expansion generates a total of $N_b \times (N_r + 1)$ words. The algorithm requires an initial set of N_b words, and each of the N_r rounds requires N_b words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted $[w_i]$, with i in the range $0 \leq i < N_b \times (N_r + 1)$.

9.2.3 AES Operating Modes

9.2.3.1 Supported Modes of Operation

9.2.3.1.1 ECB Feedback Mode

Figure 9-2 shows the basic ECB feedback mode of operation, where the input data is passed directly to the basic cryptographic core and the output is passed directly to the output buffer.

For decryption, the cryptographic core operates in reverse: the decryption data path is used for data processing, whereas encryption uses the encryption data path.

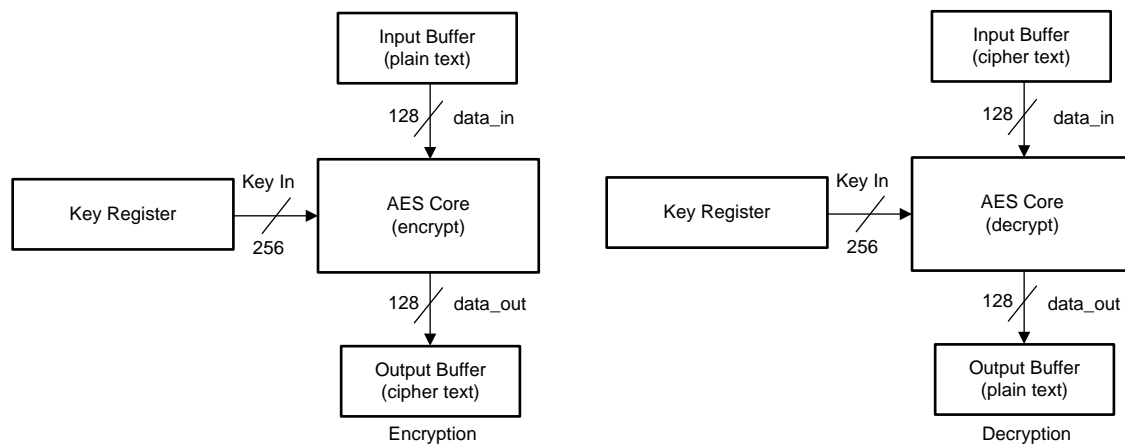


Figure 9-2. AES – ECB Feedback Mode

9.2.3.1.2 CBC Feedback Mode

Figure 9-3 shows the CBC feedback mode of operation, where the input data is XORed with the IV before it is passed to the basic cryptographic core. The output of the cryptographic core passes directly to the output buffer and becomes the next IV.

The operation is reversed for decryption, resulting in an XOR at the output of the cryptographic core. The input cipher text of the current operation is the IV for the next operation.

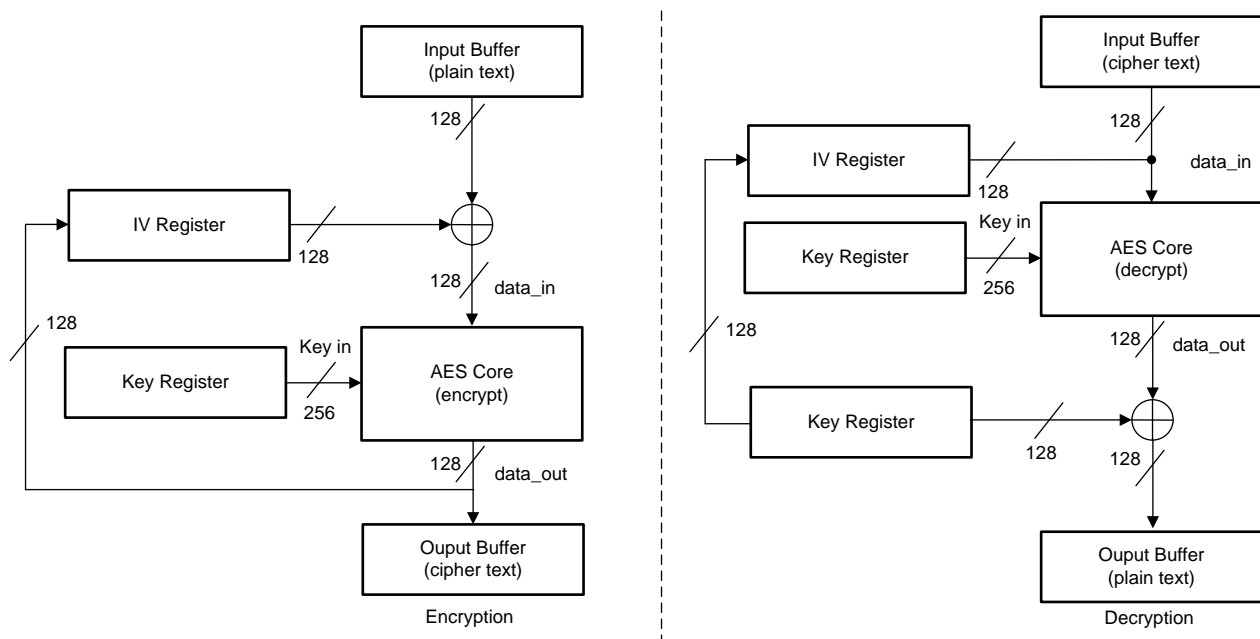


Figure 9-3. AES – CBC Feedback Mode

9.2.3.1.3 CTR and ICM Feedback Modes

Figure 9-4 shows the counter feedback (CTR/ICM) mode of operation. This operation encrypts the IV. The output of the cryptographic core (encrypted IV) is XORed with the data, therefore creating the output result.

The IV is built out of two components: a fixed part and a counter part. The counter part is incremented with each block. The counter width is selectable per context and can be 16, 32, 64, 96, or 128 bits wide. In this mode, encryption and decryption use the same operation.

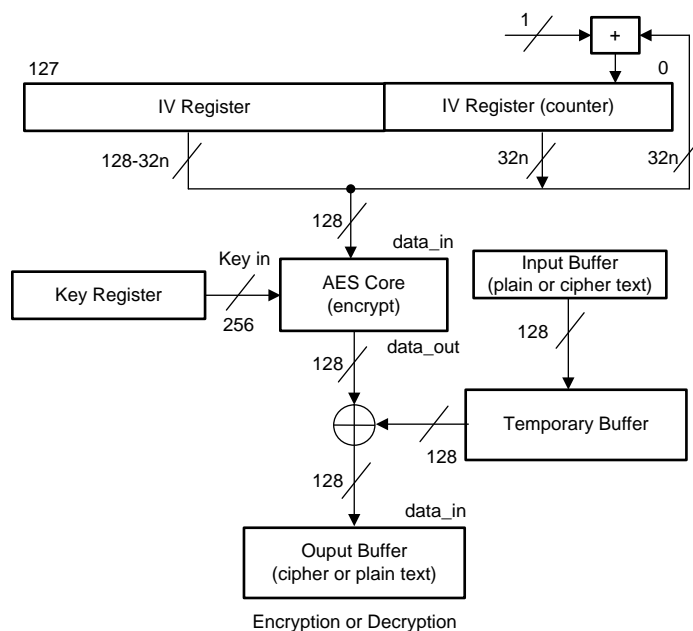


Figure 9-4. AES Encryption With CTR/ICM Mode

NOTE: The value for n can be 1, 2, 3, or 4 for CTR mode and is ½ for ICM mode.

9.2.3.1.4 CFB Mode

Figure 9-5 shows the full block (128 bits) CFB mode of operation for encryption and decryption. The input for the cryptographic core is the IV; the result is XORed with the data. The result is fed back through the IV register as the next input for the cryptographic core. The decryption operation is reversed, but the cryptographic core still performs encryption.

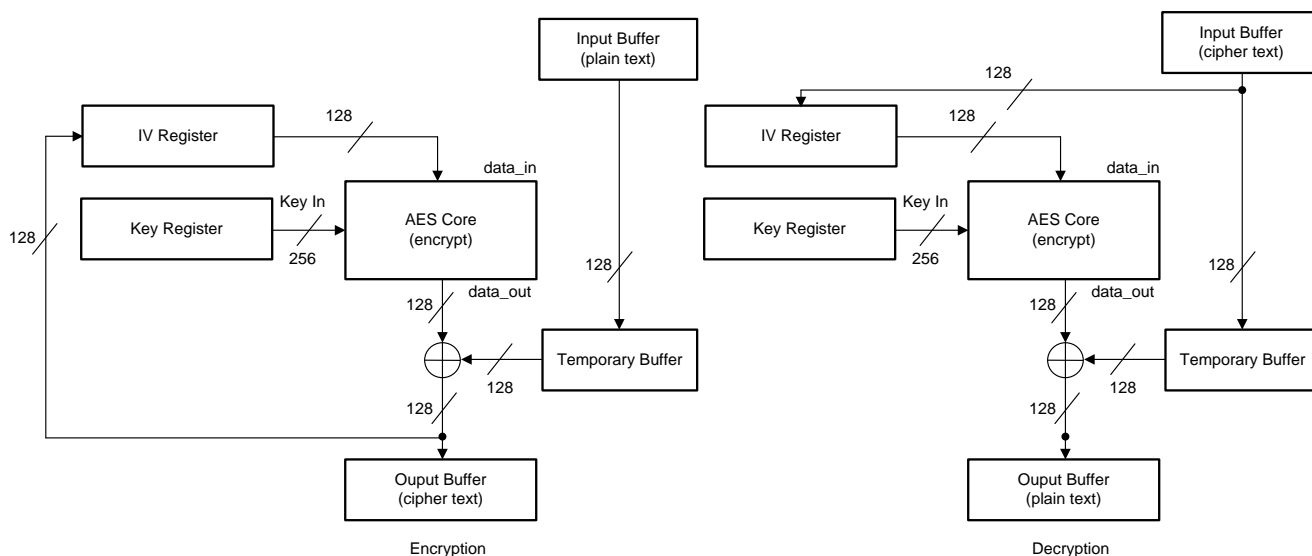


Figure 9-5. AES – CFB Feedback Mode

9.2.3.1.5 F8 Mode

Figure 9-6 shows the F8 feedback mode of operation for encryption and decryption. The input to the cryptographic core is the result of the XOR operation of the previous cryptographic core output, a constant IV, and a block counter. The output of the cryptographic core is XORed with the input to create the result. In this mode, encryption and decryption use the same operations.

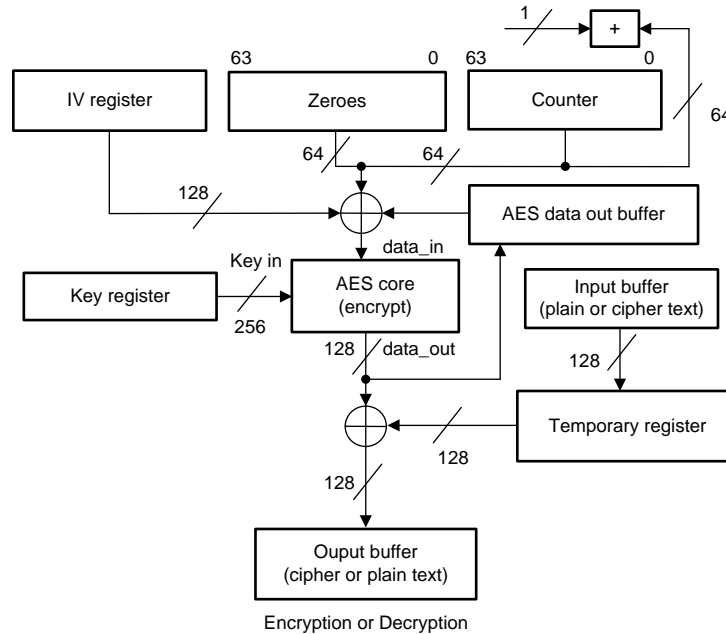


Figure 9-6. AES – F8 Mode

9.2.3.1.6 XTS Operation

Figure 9-7 shows the XTS mode of operation for encryption and decryption. The input to the cryptographic core is XORed with the IV; the output of the cryptographic core is XORed with the same IV. For decryption, the cryptographic core operates in reverse, but the XOR operations are the same.

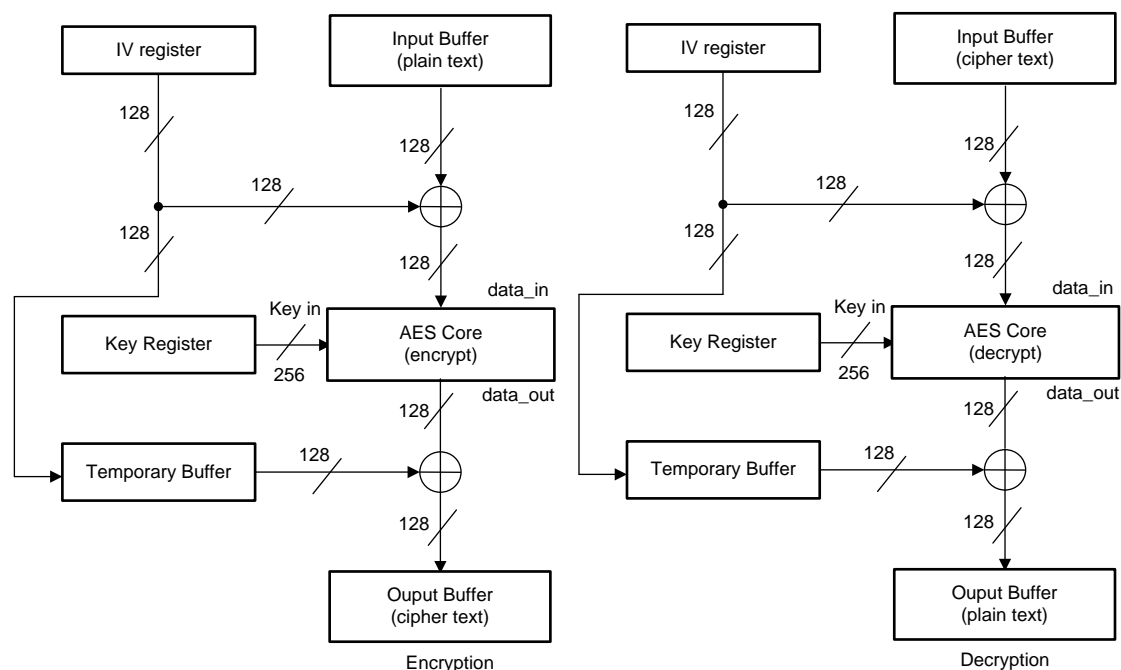


Figure 9-7. AES – XTS Operation

NOTE: The IV is created with an initial encryption, followed by an LFSR operation for each new block.

9.2.3.1.7 F9 Operation

Figure 9-8 shows the F9 authentication mode of operation, where the input to the cryptographic core is XORed with the IV, and the output is XORed with the previous result to create the next result. The cryptographic core output is fed back as IV for the next block. The result is the output of the last XOR operation of the cryptographic core output.

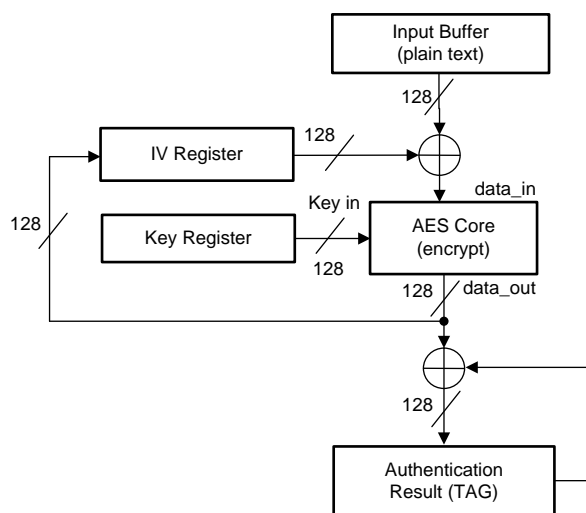


Figure 9-8. AES – F9 Operation

9.2.3.1.8 CBC-MAC Operation

Figure 9-9 shows the CBC-MAC authentication mode of operation, where the input to the cryptographic core is XORed with the IV. The cryptographic core output is then fed back as IV for the next block. The last data input block is XORed with an additional input value stored in the temporary buffer; this can be any precalculated value and is dependent on the alignment of the last input block. The result is the cryptographic core output of the last encryption operation.

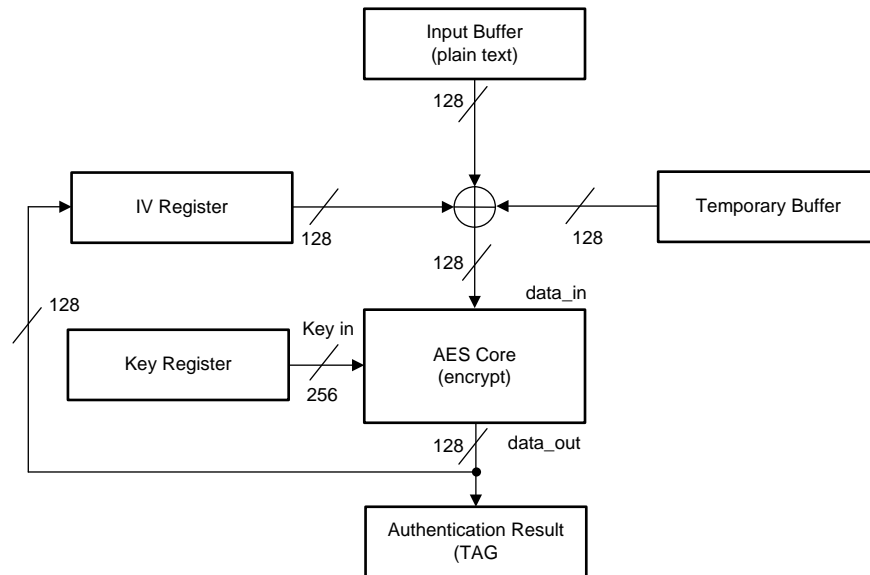


Figure 9-9. AES – CBC-MAC Authentication Mode

9.2.3.1.9 GCM Operation

Figure 9-10 shows one round of a GCM operation for encryption and decryption. A 32-bit counter is used as IV (as it is for CTR mode). The data is encrypted in the same way as in CTR mode, by XORing the cryptographic core output with the input. After the encryption or decryption, the ciphertext is XORed with the intermediate authentication result. The XORed result is used as input for the polynomial multiplication to create the next (intermediate) authentication result. For more information about the GCM protocol, see Section 9.2.3.2.1.

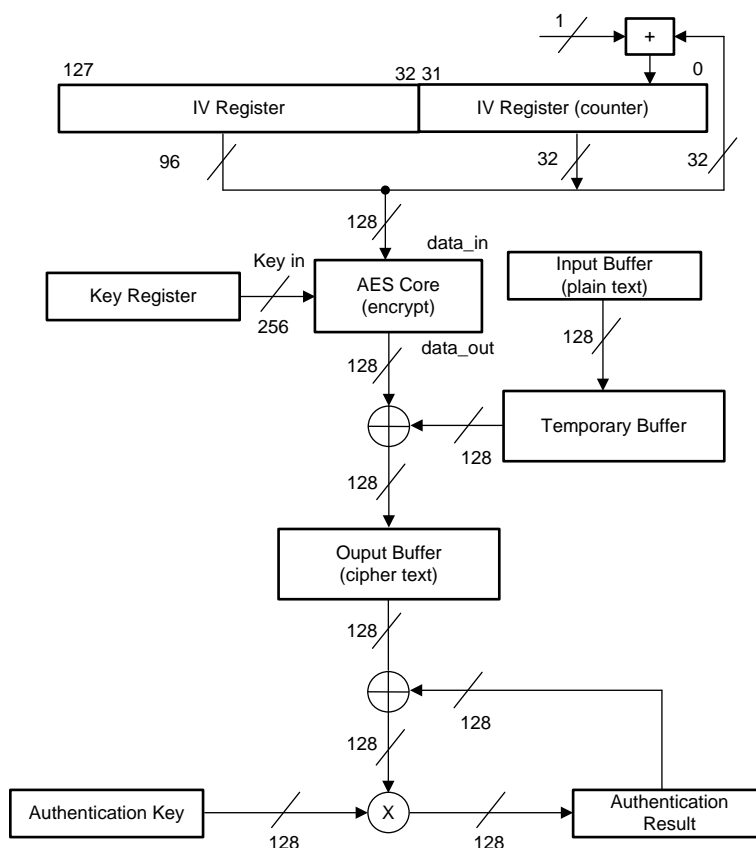
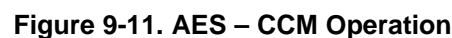


Figure 9-10. AES – GCM Operation

Figure 9-11 shows one round of a CCM (counter with CBC-MAC) operation for encryption and decryption. A 32-bit counter is used as IV (as it is for CTR mode). The data is encrypted in the same way as in CTR mode, by XORing the cryptographic core output with the input. Immediately after the encryption operation, the plaintext is XORed with the intermediate authentication result. The XOR result is used as input for a second encryption operation to calculate the next (intermediate) authentication result.



This section describes the protocols (or autonomous precalculations) supported by the AES wide-bus engine.

A GCM protocol operation is a combined operation consisting of encryption or decryption and authentication. A part of the input data stream can be authenticated only, while normally most of the input data is encrypted/decrypted and authenticated. The authentication-only data must always be in front of the data requiring encryption. Within GCM, the authentication-only data is called the additional authentication data (AAD). The AAD is fetched independently of other data.

The intermediate (temporary) result data is used as input to the remaining authentication operation. Because the authentication operation does not require the cryptographic core but only the polynomial multiplication, encryption or decryption and authentication can be performed in parallel. After encryption of the last data block, additional polynomial multiplication and encryption are required to authenticate a 128-bit length vector and finally encrypt the authentication result.

9.2.3.2.2 CCM Protocol Operation

The CCM protocol operation is a combined operation consisting of encryption or decryption and authentication. The authentication and encryption or decryption operations use the cryptographic core; these are executed sequentially on the AES core. A part of the data stream can require authentication only. The authentication-only data must always be in front of the data requiring encryption.

Authentication starts with the encryption of a predefined block B0. This block consists of flags, nonce, and message length. The next blocks contain the authentication data length concatenated with the authentication-only data. After processing the authentication-only data, the encryption or decryption operations are performed, each followed by the related authentication of the plaintext data block (which equals the input in the case of encryption, and the output in the case of decryption). The final authentication result must be encrypted using the output of the encryption of the IV block A0. This block contains the IV (consisting of flags and nonce) concatenated with the counter, which is zero for A0.

9.2.4 AES Software Reset

To perform a software reset of the AES module, write a 1 to the SOFTRESET bit in the AES System Configuration (AES_SYSCONFIG) register. The RESETDONE bit in the AES Secure System Status (AES_SYSSTATUS) register indicates that the software reset is complete when its value is 1. When the software reset completes, the SOFTRESET bit in the AES_SYSCONFIG register is automatically reset. Software must ensure that the software reset completes before doing any operations.

The behavior of the software reset is the same as the hardware reset, except that the software reset bit resets this module without affecting the reset core domain of the entire device.

9.2.5 Power Management

To save power, the application can disable the clock to the AES module when not in use. The AES is clock gated by setting the AESCFG bit in the Cryptographic Modules Clock Gating Request (CCMCGREQ) register, CRC and Cryptographic Modules (CCM) offset 0x204. The AES in addition to the DES, SHA/MD5 and CRC can also be clock gated as a group by setting the D0 bit in the CRC and Cryptographic Modules (DCGCCCM) register, System Control Module offset 0x874.

9.2.6 Hardware Requests

The AES module can assert a μ DMA request for context in, context out, input data, or output data read. The AES uDMA Interrupt Mask (AES_DMAIM) register can be set to generate interrupts during the following events:

- Context In μ DMA request (AES0 Cin)
- Context Out μ DMA request (AES0 Cout)
- Data In μ DMA request (AES0 Din)
- Data Out μ DMA request (AES0 Dout)

The AES module can be programmed to assert an interrupt when the uDMA has completed its last transfer. See [Section 9.6](#) for more information.

If context and data transfers are to be handled through software, then the AES Interrupt Enable (AES_IRQENABLE), offset 0x090, can be used to enable interrupt triggering when context out, context in, data in or data out is ready. The AES Interrupt Status (AES_IRQSTATUS), offset 0x08C, indicates when an interrupt is triggered.

Table 9-2. Interrupts and Events

Event	Description
AES_IRQSTATUS [3]: CONTEXT_OUT	Context output interrupt
AES_IRQSTATUS [2]: DATA_OUT	Data output interrupt
AES_IRQSTATUS [1]: DATA_IN	Data input interrupt
AES_IRQSTATUS [0]: CONTEXT_IN	Context input interrupt

9.3 AES Performance Information

Table 9-3 lists the performance for all supported key sizes and modes of operations. It assumes that the engine is kept fully utilized (that is, the host is supplying input blocks and retrieving output blocks in such a way that the engine never has to wait for input), and that the previous output has been retrieved before the next output is ready.

Table 9-3. AES Module Performance (Input/Output Block Size = 128)

Key Size	Mode of Operation	Cycles per Block ⁽¹⁾	Throughput (Bits per Cycle)
128	ECB encrypt or decrypt	32	4.00
	CBC-decrypt		
	CTR / ICM encrypt or decrypt		
	CFB128-decrypt		
	CBC encrypt	33	3.88
	OFB encrypt or decrypt		
	f8 encrypt or decrypt		
	CFB128 encrypt		
	XTS encrypt or decrypt		
	GCM-outbound / inbound		
	CCM encrypt or decrypt	66	1.94
	CBC-MAC ⁽²⁾	33	3.88
	f9 ⁽²⁾		
192	ECB encrypt or decrypt	38	3.37
	CBC-decrypt		
	CTR/ICM encrypt or decrypt		
	CFB128-decrypt		
	CBC encrypt	39	3.37
	OFB encrypt or decrypt		
	f8 encrypt or decrypt		
	CFB128 encrypt		
	XTS encrypt or decrypt		
	GCM-outbound / inbound		
	CCM encrypt or decrypt	78	1.64
	CBC-MAC ⁽²⁾	39	3.28
	f9 ⁽²⁾		

⁽¹⁾ Numbers for regular GCM mode (H is precalculated and Y0-encrypted need to be calculated internally using the new IV). If H needs to be calculated by the core (complete GCM mode), this number needs to be doubled. If Y0-encrypted is not calculated (forced to zero, such that the hash result is not encrypted) this number is zero.

⁽²⁾ Input Only

Table 9-3. AES Module Performance (Input/Output Block Size = 128) (continued)

Key Size	Mode of Operation	Cycles per Block ⁽¹⁾	Throughput (Bits per Cycle)
256	ECB encrypt or decrypt	44	2.91
	CBC-decrypt		
	CTR/ICM encrypt or decrypt		
	CFB128-decrypt		
	CBC encrypt	45	2.84
	OFB encrypt or decrypt		
	f8 encrypt or decrypt		
	CFB128 encrypt		
	XTS encrypt or decrypt		
	GCM-outbound inbound		
	CCM encrypt or decrypt	90	1.42
	CBC-MAC ⁽³⁾	45	2.84
	f9 ⁽³⁾		

⁽³⁾ Numbers for regular CCM mode. Dependent on the AAD length it is possible that one additional encryption needs to be done to finalize the AAD authentication; if the additional operation is required, this number needs to be doubled.

The Cycles per Block numbers do not apply to the first block after selection of a new key or mode of operation, or the last block before switching to a new algorithm. [Table 9-4](#) indicates the additional clock cycles required for changing context data.

Table 9-4. AES Module Packet Mode Switch Overhead

Title	Cycles Needed For First Block / To Finish Last Block	Total
Mode is encrypt	1 / 1	2
Mode is decrypt, key size is 128	32 / 1	33
Mode is decrypt, key size is 192	38 / 1	39
Mode is decrypt, key size is 256	44 / 1	45
Mode is XTS, key size is 128	34 / 1	35
Mode is XTS, key size is 192	40 / 1	41
Mode is XTS, key size is 256	46 / 1	47
Outbound GCM, AES key size is 128	33 ⁽¹⁾ / 52	85
Outbound GCM, AES key size is 192	39 ⁽¹⁾ / 52	91
Outbound GCM, AES key size is 256	45 ⁽¹⁾ / 52	97
Inbound GCM, AES key size is 128	33 ⁽¹⁾ / 27	60
Inbound GCM, AES key size is 192	39 ⁽¹⁾ / 27	66
Inbound GCM, AES key size is 256	45 ⁽¹⁾ / 27	72
Outbound CCM, AES key size is 128	33 ⁽¹⁾ / 33	66
Outbound CCM, AES key size is 192	39 ⁽²⁾ / 39	78
Outbound CCM, AES key size is 256	45 ⁽²⁾ / 45	90
Inbound CCM, AES key size is 128	33 ⁽²⁾ / 33	66
Inbound CCM, AES key size is 192	39 ⁽²⁾ / 39	78
Inbound CCM, AES key size is 256	45 ⁽²⁾ / 45	90

⁽¹⁾ Numbers for regular GCM mode (H is precalculated and Y0-encrypted need to be calculated internally using the new IV). If H needs to be calculated by the core (complete GCM mode), this number needs to be doubled. If Y0-encrypted is not calculated (forced to zero, such that the hash result is not encrypted) this number is zero.

⁽²⁾ Numbers for regular CCM mode. Dependent on the AAD length it is possible that one additional encryption needs to be done to finalize the AAD authentication; if the additional operation is required, this number needs to be doubled.

9.4 AES Module Programming Guide

9.4.1 AES Low - Level Programming Models

This section describes the low-level hardware programming sequences for configuring and using the AES module.

9.4.1.1 Global Initialization

The following list describes the requirements for initializing the AES and surrounding modules when the AES is used for the first time after a device reset.

1. When reset has completed, enable the AES by setting the R0 bit in the CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM), System Control offset 0x674. When the R0 bit is set in the CRC and Cryptographic Modules Peripheral Ready (PRCCM), System Control offset 0xA74 register, the AES is powered and ready to be configured.
2. Configure the AES μ DMA channels for Context In, Context Out, Data In, and/or Data Out by programming the appropriate encoding value in the DMA Channel Map Select n (DMACHMAPn) register in the μ DMA module, offset 0x510. For more information on how to program channel assignments as well as enabling burst and the configured channels, refer to [Chapter 8](#).
3. Execute a software reset by setting the SOFTRESET bit in the AES_SYSCONFIG register. When reset is complete, the RESETDONE bit reads as 1 in the AES_SYSSTATUS register.
4. If the AES channels are configured in the μ DMA, enable the required AES DMA requests by programming bits [9:5] of the AES_SYSCONFIG register, in addition to the completion interrupts in the AES DMA Interrupt Mask (AES_DMAIM) register, CCM offset 0x020.
5. Specify the size of the keys by programming the KEY_SIZE bit field in the AES_CTRL register.
6. Load AES Key 1 (AES_KEY1_n) register.
7. Load AES Key 2 (AES_KEY2_n) register if it is used by the configuration mode. See [Table 9-7](#) for information regarding which configuration modes require a load to this register.
8. Configure the AES for the appropriate encryption or decryption mode (see [Section 9.4.1.2.1](#) to [Section 9.4.1.2.10](#)).
9. Select encryption or decryption by programming the DIRECTION bit in the AES Control (AES_CTRL) register, offset 0x050.

9.4.1.2 Initialization Subsequence

The following sections list the initialization subsequences for the available encryption or decryption modes:

9.4.1.2.1 Subsequence: Initialize CCM AES Core Mode

The CCM mode initialization is as follows:

1. Define the width of the length field and the length of the authentication field by programming the CCM_L and CCM_M bit fields in the AES_CTRL register at offset 0x050.
2. Enable counter mode by setting the CTR bit in the AES_CTRL register.
3. Load the authentication data length in the AUTH field of the AES Authentication Data Length (AES_AUTH_LENGTH) register at offset 0x05C.
4. Select the IV counter by programming the CTR_WIDTH field in the AES_CTRL register.
5. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.2 Subsequence: Initialize GCM AES Core Mode

The following steps to enable GCM mode is as follows:

1. Enable counter mode by setting the CTR bit in the AES_CTRL register.
2. Load the authentication data length in the AUTH field of the AES Authentication Data Length (AES_AUTH_LENGTH) register at offset 0x05C.
3. Select the IV counter by programming the CTR_WIDTH field in the AES_CTRL register.

4. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.3 Subsequence: Initialize CBC-MAC AES Core Mode

To initialize CBC-MAC Mode:

1. Enable CBC-MAC Mode by setting the CBCMAC bit in the AES_CTRL register.
2. Select encryption mode by setting the DIRECTION bit in the AES_CTRL register at offset 0x050
3. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.4 Subsequence: Initialize F9 AES Core Mode

The steps for configuring the AES for F9 mode is as follows:

1. Enable F9 Mode by setting the F9 bit in the AES_CTRL register.
2. Set the key size to 128 bits by programming the KEY_SIZE field to 0x1 in the AES_CTRL register.
3. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.5 Subsequence: Initialize F8 AES Core Mode

The steps for configuring the AES for F8 mode is as follows:

1. Enable F8 Mode by setting the F8 bit in the AES_CTRL register.
2. Select the counter width by programming the CTR_WIDTH field in the AES_CTRL register.
3. Set the key size to 128 bits by setting the KEY_SIZE field to 0x1 in the AES_CTRL register.
4. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.6 Subsequence: Initialize XTS AES Core Mode

The steps for XTS mode configuration are as follows:

1. Enable XTS Mode by configuring the XTS field in the AES_CTRL register.
2. If the XTS field in the AES_CTRL register indicates that the AAD length is required, load the AAD length in the AES_AUTH_LENGTH register.
3. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.7 Subsequence: Initialize CFB AES Core Mode

To initialize the AES code for CFB mode:

1. Enable CFB Mode by setting the CFB bit in the AES_CTRL register.
2. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.8 Subsequence: Initialize ICM AES Core Mode

To initialize the AES code for ICM mode:

1. Enable ICM Mode by setting the ICM bit in the AES_CTRL register.
2. Configure for a 16-bit counter by programming the CTR_WIDTH to 0x0 in the AES_CTRL register.
3. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.9 Subsequence: Initialize CTR AES Core Mode

To initialize CTR mode:

1. Enable CTR Mode by setting the CTR bit in the AES_CTRL register.
2. Select counter width by programming the CTR_WIDTH in the AES_CTRL register.
3. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.2.10 Subsequence: Initialize CBC AES Core Mode

To configure CBC mode:

1. Enable CBC Mode by setting the MODE bit in the AES_CTRL register.
2. Load the AES Initialization Vector Input n (AES_IV_IN_n) registers at offset 0x040 to 0x04C.

9.4.1.3 Operational Modes Configuration

9.4.1.3.1 AES Polling Mode

9.4.1.3.1.1 Main Sequence: AES Polling Mode

Figure 9-12 shows AES polling mode. The registers used in AES polling mode are:

- AES Data RW Plaintext/Ciphertext 0 (AES_DATA_IN_0) registers, offset 0x060 to 0x06C
- AES Control (AES_CTRL) register, offset 0x050
- AES Hash Tag Out 0 (AES_TAG_OUT_0), offset 0x070

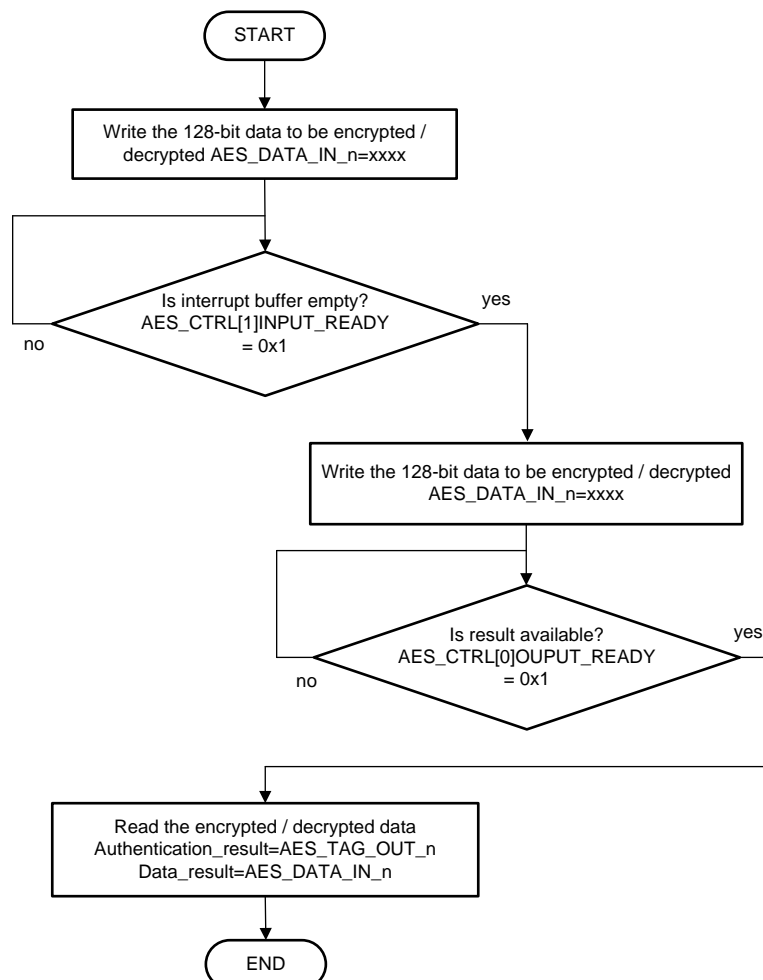


Figure 9-12. AES Polling Mode

9.4.1.3.2 AES Interrupt Mode

The application can use software interrupts to control the flow of Context In, Context Out, Data In, and Data Out requests. To enable these interrupts:

1. When the device has been initialized, following the initialization sequences described in [Section 9.4.1.1](#) and [Section 9.4.1.2](#), the application can enable the AES module interrupts through the AES Interrupt Enable (AES_IRQENABLE) register, offset 0x090. If all four interrupts must be enabled, the application can write 0x0000.000F to the AES_IRQENABLE register.
2. Load the input buffers, AES_DATA_IN_n, with data.

NOTE: If the application uses Interrupt Mode, an interrupt is generated for each block of processed data. To support larger data flow, AES μ DMA Mode should be used and the bits in the AES_IRQENABLE register should be cleared.

9.4.1.3.3 AES DMA Mode

When AES DMA Mode is enabled, the AES_IRQENABLE register should be cleared. To enable the μ DMA to transfer data follow these steps:

1. When the AES module has been initialized, enable the AES module μ DMA channels by programming the DMA Channel Map Select n (DMACHMAPn) register in the μ DMA module. Further μ DMA configuration guidelines are available in the [Chapter 8](#).
2. Configure the dma_done interrupts by programming the AES DMA Masked Interrupt Status (AES_DMAMIS) register, at CCM offset 0x028.
3. Enable the μ DMA channels in the AES by programming the μ DMA enable bits in the AES System Configuration (AES_SYSCONFIG) register, offset 0x084.
4. The input buffer registers, AES_DATA_IN_n, at offsets 0x060 to 0x06C, are loaded.

9.4.1.4 AES Events Servicing

9.4.1.4.1 Interrupt Servicing

This section describes the event servicing of the module. [Figure 9-13](#) shows the AES interrupt service. The registers used during event servicing are as follows:

- AES_IRQSTATUS
- AES_KEY1_n
- AES_KEY2_n
- AES_IV_IN_n
- AES_DATA_IN_n
- AES_TAG_OUT_n
- AES_IRQENABLE

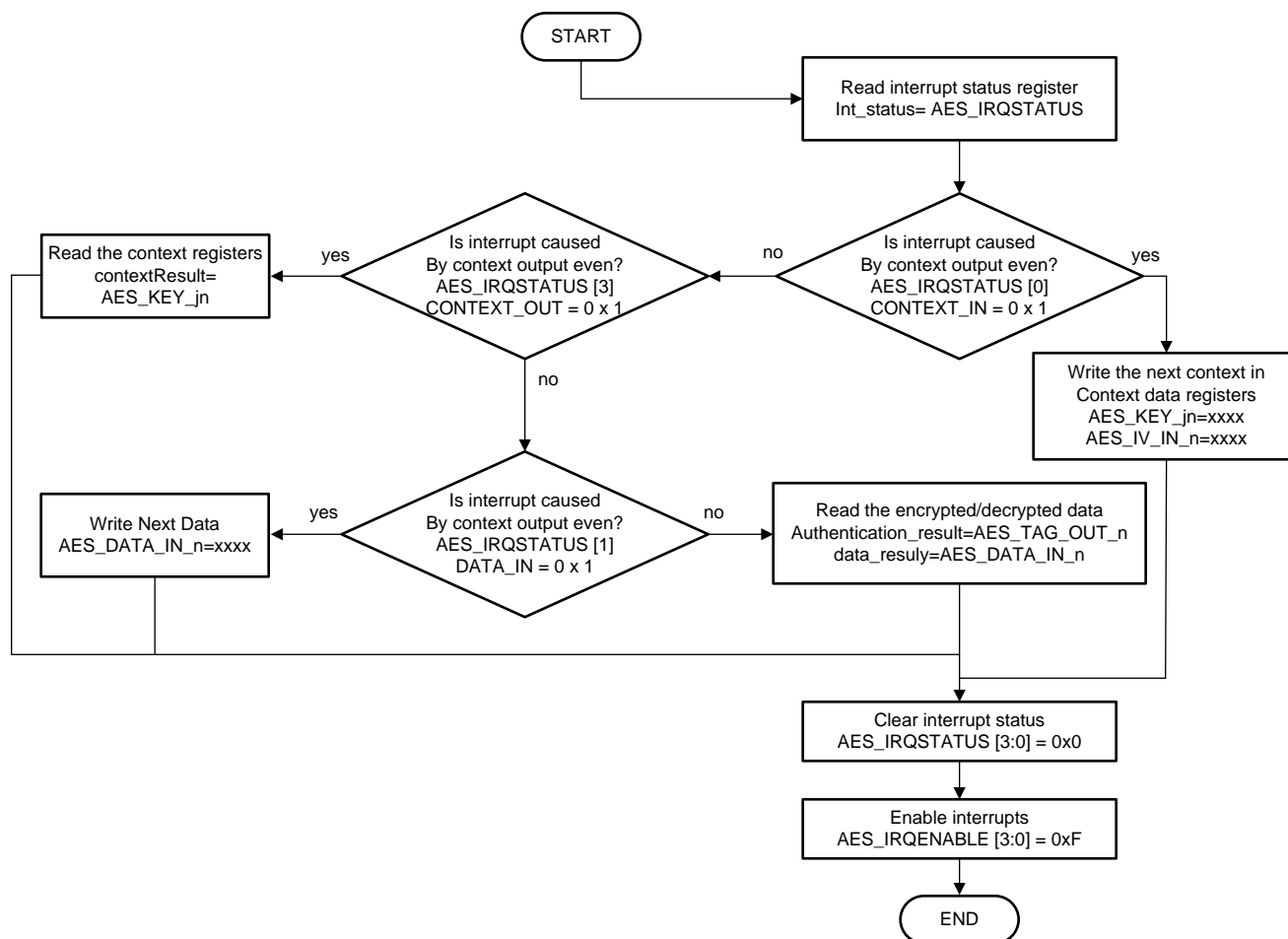


Figure 9-13. AES Interrupt Service

9.5 AES Registers

The AES module comprises registers that exist at an offset relative to the AES Module base address and a small set of AES μ DMA registers that exist at an offset relative to an Encryption Control Module base address.

The AES module register offsets are relative to the base address 0x44036000.

[Table 9-5](#) lists the memory-mapped registers for the AES. All register offset addresses not listed in [Table 9-5](#) should be considered as reserved locations and the register contents should not be modified.

Table 9-5. AES Registers

Offset	Acronym	Register Name	Section
0x000	AES_KEY2_6	AES Key 2_6	Section 9.5.1
0x004	AES_KEY2_7	AES Key 2_7	Section 9.5.1
0x008	AES_KEY2_4	AES Key 2_4	Section 9.5.1
0x00C	AES_KEY2_5	AES Key 2_5	Section 9.5.1
0x010	AES_KEY2_2	AES Key 2_2	Section 9.5.1
0x014	AES_KEY2_3	AES Key 2_3	Section 9.5.1
0x018	AES_KEY2_0	AES Key 2_0	Section 9.5.1
0x01C	AES_KEY2_1	AES Key 2_1	Section 9.5.1
0x020	AES_KEY1_6	AES Key 1_6	Section 9.5.1
0x024	AES_KEY1_7	AES Key 1_7	Section 9.5.1
0x028	AES_KEY1_4	AES Key 1_4	Section 9.5.1
0x02C	AES_KEY1_5	AES Key 1_5	Section 9.5.1
0x030	AES_KEY1_2	AES Key 1_2	Section 9.5.1
0x034	AES_KEY1_3	AES Key 1_3	Section 9.5.1
0x038	AES_KEY1_0	AES Key 1_0	Section 9.5.1
0x03C	AES_KEY1_1	AES Key 1_1	Section 9.5.1
0x40 to 0x4C	AES_IV_IN_0 to AES_IV_IN_3	AES Initialization Vector Input 0 to AES Initialization Vector Input 3	Section 9.5.2
0x50	AES_CTRL	AES Control	Section 9.5.3
0x54 to 0x58	AES_C_LENGTH_0 to AES_C_LENGTH_1	AES Crypto Data Length 0 to AES Crypto Data Length 1	Section 9.5.4
0x5C	AES_AUTH_LENGTH	AES Authentication Data Length	Section 9.5.5
0x60 to 0x6C	AES_DATA_IN_0 to AES_DATA_IN_3	AES Data R/W Plaintext/Ciphertext 0 to AES Data R/W Plaintext/Ciphertext 3	Section 9.5.6
0x70 to 0x7C	AES_TAG_OUT_0 to AES_TAG_OUT_3	AES Hash Tag Out 0 to AES Hash Tag Out 3	Section 9.5.7
0x80	AES_REVISION	AES IP Revision Identifier	Section 9.5.8
0x84	AES_SYSCONFIG	AES System Configuration	Section 9.5.9
0x88	AES_SYSSTATUS	AES System Status	Section 9.5.10
0x8C	AES_IRQSTATUS	AES Interrupt Status	Section 9.5.11
0x90	AES_IRQENABLE	AES Interrupt Enable	Section 9.5.12
0x94	AES_DIRTYBITS	AES Dirty Bits	Section 9.5.13

Complex bit access types are encoded to fit into small table cells. [Table 9-6](#) shows the codes that are used for access types in this section.

Table 9-6. AES Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		

Table 9-6. AES Access Type Codes (continued)

Access Type	Code	Description
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

9.5.1 AES_KEYn_n Register (Offset = 0x000 to 0x03C) [reset = 0x0]

This register contains the 32-bit key data for the AES module. This value can be expanded to 256 bits depending on the mode of operation. [Table 9-7](#) describes the type of key that is held in each register.

NOTE: Reads return zeros.

Table 9-7. AES Key Register Descriptions

Register Name	Address Offset	Description
AES_KEY2_6	0x000	Secure XTS second key / CBC-MAC third key
AES_KEY2_7	0x004	Secure XTS second key (MSW for 256-bit key) / CBC-MAC third key (MSW)
AES_KEY2_4	0x008	Secure XTS / CCM second key / CBC-MAC third key (LSW)
AES_KEY2_5	0x00C	Secure XTS second key (MSW for 192-bit key) / CBC-MAC third key
AES_KEY2_2	0x010	Secure XTS / CCM / CBC-MAC second key / Hash Key input
AES_KEY2_3	0x014	Secure XTS second key (MSW for 128-bit key) + CCM/CBC-MAC second key (MSW) / Hash Key input (MSW)
AES_KEY2_0	0x018	Secure XTS / CCM / CBC-MAC second key (LSW) / Hash Key input (LSW)
AES_KEY2_1	0x01C	Secure XTS / CCM / CBC-MAC second key / Hash Key input
AES_KEY1_6	0x020	Secure Key (LSW for 256-bit key)
AES_KEY1_7	0x024	Secure Key (MSW for 256-bit key)
AES_KEY1_4	0x028	Secure Key (LSW for 192-bit key)
AES_KEY1_5	0x02C	Secure Key (MSW for 192-bit key)
AES_KEY1_2	0x030	Secure Key
AES_KEY1_3	0x034	Secure Key (MSW for 128-bit key)
AES_KEY1_0	0x038	Secure Key (LSW for 128-bit key)
AES_KEY1_1	0x03C	Secure Key

AES_KEYn_n is shown in [Figure 9-14](#) and described in [Table 9-8](#).

Return to [Summary Table](#).

Figure 9-14. AES_KEYn_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																															
R/W-0x0																															

Table 9-8. AES_KEYn_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KEY	R/W	0x0	Key Data. This register contains the 32-bit key data for the AES module.

9.5.2 AES_IV_IN_0 to AES_IV_IN_3 Registers (Offset = 0x40 to 0x4C) [reset = 0x0]

AES Initialization Vector Input 0 (AES_IV_IN_0), offset 0x040

AES Initialization Vector Input 1 (AES_IV_IN_1), offset 0x044

AES Initialization Vector Input 2 (AES_IV_IN_2), offset 0x048

AES Initialization Vector Input 3 (AES_IV_IN_3), offset 0x04C

This register contains the initialization vector input.

AES_IV_IN_n is shown in [Figure 9-15](#) and described in [Table 9-9](#).

Return to [Summary Table](#).

Figure 9-15. AES_IV_IN_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0x0																															

Table 9-9. AES_IV_IN_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0x0	Initialization Vector Input. This field contains the initialization input vector. The least significant word (LSW) is represented in register AES_IV_IN_0 and the most significant word is stored in AES_IV_IN_3

9.5.3 AES_CTRL Register (Offset = 0x50) [reset = 0x80000000]

AES Control (AES_CTRL)

This register determines the mode of operation of the AES Engine.

AES_CTRL is shown in [Figure 9-16](#) and described in [Table 9-10](#).

Return to [Summary Table](#).

Figure 9-16. AES_CTRL Register

31	30	29	28	27	26	25	24
CTXTRDY	SVCTXTRDY	SAVE_CONTEXT	RESERVED			CCM_M	
R-0x1	R-0x0	R/W-0x0	R-0x0			R/W-0x0	
23	22	21	20	19	18	17	16
CCM_M		CCM_L			CCM	GCM	
R/W-0x0		R/W-0x0			R/W-0x0	R/W-0x0	
15	14	13	12	11	10	9	8
CBCMAC	F9	F8	XTS		CFB	ICM	CTR_WIDTH
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
CTR_WIDTH	CTR	MODE	KEY_SIZE		DIRECTION	INPUT_READY	OUTPUT_READY
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0	R-0x0	R-0x0

Table 9-10. AES_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CTXTRDY	R	0x1	Context Data Registers Ready 0x0 = The context data registers are not ready to be overwritten. 0x1 = The context data registers can be overwritten and the host is permitted to write the next context.
30	SVCTXTRDY	R	0x0	AES TAG/IV Blocks Ready. This bit is only asserted if the SAVE_CONTEXT bit is set to 1. This bit is mutual exclusive with the CTXTRDY bit. 0x0 = AES authentication TAG and/or IV block(s) is/are not available. 0x1 = Indicates the AES authentication TAG and /or IV block(s) is/are available for the host to retrieve.
29	SAVE_CONTEXT	R/W	0x0	TAG or Result IV Save. If this bit is set, the CONTEXT_OUT interrupt bit is set in the AES_IRQSTATUS register if the operation is finished and related signals are enabled. 0x0 = No effect. 0x1 = Indicates an authentication TAG of result IV needs to be stored as a result context.
28-25	RESERVED	R	0x0	
24-22	CCM_M	R/W	0x0	Counter with CBC-MAC (CCM). Defines M which indicates the length of the authentication field for CCM operations the authentication field length equals two times the sum of CCM-M plus one. The AES Engine always returns a 128-bit authentication field, of which the M least significant bytes are valid. All values are supported.

Table 9-10. AES_CTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-19	CCM_L	R/W	0x0	L Value. Defines L, which indicates the width of the length field for CCM operations the length field in bytes equals the value of CMM-L plus one. Supported values for L are: 0x0 = reserved 0x1 = width = 2 0x2 = reserved 0x3 = width = 4 0x7 = width = 8
18	CCM	R/W	0x0	AES-CCM Mode Enable 0x0 = AES-CCM mode is not enabled. 0x1 = AES-CCM mode enabled. This is a combined mode, using AES for both authentication and encryption. No additional mode selection is required.
17-16	GCM	R/W	0x0	AES-GCM Mode Enable. This is a combined mode, using the Galois field-multiplier GF(2 ¹²⁸) for authentication and AES-CTR mode for encryption the bits specify the GCM mode. 0x0 = No operation 0x1 = GHASH with H loaded and Y0-encrypted forced to zero 0x2 = GHASH with H loaded and Y0-encrypted calculated internally 0x3 = Autonomous GHASH (both H and Y0-encrypted calculated internally)
15	CBCMAC	R/W	0x0	AES-CBC MAC Enable. The DIRECTION bit must be set to 1 for this mode. 0x0 = AES-CBC MAC mode is not enabled. 0x1 = AES-CBC MAC mode enabled.
14	F9	R/W	0x0	AES f9 Mode Enable. The AES key size must be set to 128-bit for this mode. 0x0 = f9 mode is not enabled 0x1 = f9 mode is enabled.
13	F8	R/W	0x0	AES f8 Mode Enable. The KEY_SIZE must be set to 128-bit for this mode. 0x0 = AES f8 mode is not enabled. 0x1 = AES f8 mode is enabled.
12-11	XTS	R/W	0x0	AES-XTS Operation Enabled. The bits specify the XTS mode. 0x0 = No operation 0x1 = Previous/intermediate tweak value and j loaded (value is loaded via IV, j is loaded via the AAD length register) 0x2 = Key2, n and j are loaded (n is loaded via IV, j is loaded via the AAD length register) 0x3 = Key2 and n are loaded; j=0 (n is loaded via IV)
10	CFB	R/W	0x0	Full block AES cipher feedback mode (CFB128) Enable 0x0 = AES-CFB mode is not enabled. 0x1 = AES-CFB mode is enabled.
9	ICM	R/W	0x0	AES Integer Counter Mode (ICM) Enable. This is a counter mode with a 16-bit wide counter. 0x0 = AES-ICM mode is not enabled. 0x1 = AES-ICM mode is enabled.

Table 9-10. AES_CTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8-7	CTR_WIDTH	R/W	0x0	AES-CTR Mode Counter Width 0x0 = Counter is 32 bits 0x1 = Counter is 64 bits 0x2 = Counter is 96 bits 0x3 = Counter is 128 bits
6	CTR	R/W	0x0	Counter Mode. This bit must also be set for GCM and CCM mode, when encryption/decryption is required. 0x0 = Counter mode is not enabled. 0x1 = Counter mode is enabled.
5	MODE	R/W	0x0	ECB/CBC Mode 0x0 = ECB mode 0x1 = CBC mode
4-3	KEY_SIZE	R/W	0x0	Key Size 0x0 = reserved 0x1 = Key is 128 bits 0x2 = Key is 192 bits 0x3 = Key is 256 bits
2	DIRECTION	R/W	0x0	Encryption/Decryption Selection. If set to DIRECTION = 1, an encrypt operation is performed. If set to 0, a decrypt operation is performed. 0x0 = Decryption is selected. 0x1 = Encryption is selected.
1	INPUT_READY	R	0x0	Input Ready Status 0x0 = Input buffer is not empty. 0x1 = Indicates that the 16-byte input buffer is empty, and the host is permitted to write the next block of data.
0	OUTPUT_READY	R	0x0	Output Ready Status 0x0 = No AES output block is available. 0x1 = An AES output block is available for the host to retrieve.

9.5.4 AES_C_LENGTH_0 to AES_C_LENGTH_1 Registers (Offset = 0x54 to 0x58) [reset = 0x0]

AES Crypto Data Length 0 (AES_C_LENGTH_0), offset 0x054

AES Crypto Data Length 1 (AES_C_LENGTH_1), offset 0x058

AES Crypto Data Length n (AES_C_LENGTH_n) registers (LSW and MSW) store the cryptographic data length in bytes for all modes. The AES_C_LENGTH_0 register stores the most significant word and the AES_C_LENGTH_1 register stores the least significant word. Once processing with this context is started, this length decrements to zero. Data lengths up to $(2^{61} - 1)$ bytes are allowed. For GCM, any value up to $236 - 32$ bytes can be used. This is because a 32-bit counter mode is used; the maximum number of 128-bit blocks is $2^{32} - 2$, resulting in a maximum number of bytes of $2^{36} - 32$. A write to this register triggers the engine to start using this context. This is valid for all modes except GCM and CCM. Note that for the combined modes, this length does not include the authentication only data; the authentication length is specified in the AES_AUTH_LENGTH register below. All modes must have a length greater than 0. For the combined modes, it is permissible to have one of the lengths equal to zero. For the basic encryption modes (ECB/CBC/CTR/ICM/CFB128) it is permissible to program zero to the length field; in that case the length is assumed infinite. All data must be byte (8-bit) aligned; bit aligned data streams are not supported by the AES Engine.

NOTE: For a Host read operation, these registers return all zeroes.

AES_C_LENGTH_n is shown in [Figure 9-17](#) and described in [Table 9-11](#).

Return to [Summary Table](#).

Figure 9-17. AES_C_LENGTH_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENGTH																															
R/W-0x0																															

Table 9-11. AES_C_LENGTH_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LENGTH	R/W	0x0	Data Length. This field stores the cryptographic data length in bytes for all modes.

9.5.5 AES_AUTH_LENGTH Register (Offset = 0x5C) [reset = 0x0]

AES Authentication Data Length (AES_AUTH_LENGTH)

The AES Authentication Data Length (AES_AUTH_LENGTH) register stores the authentication data length in bytes for combined modes only (GCM or CCM). Supported AUTH lengths for CCM are from 0 to ($2^{16} - 28$) bytes. For GCM any value up to ($2^{32} - 1$) bytes can be used. Once processing with this context is started, this length decrements to zero. A write to this register triggers the engine to start using this context for GCM and CCM. For XTS, this register is optionally used to load j . Loading of j is only required if $j \neq 0$. j is a 28-bit value and must be written to bits [31-4] of this register. j represents the sequential number of the 128-bit block inside the data unit. For the first block in a unit, this value is zero. It is not required to provide a j for each new data block within a unit. Note that it is possible to start with a j unequal to zero.

NOTE: For a Host read operation, these registers return all zeroes.

AES_AUTH_LENGTH is shown in [Figure 9-18](#) and described in [Table 9-12](#).

Return to [Summary Table](#).

Figure 9-18. AES_AUTH_LENGTH Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTH																															
R/W-0x0																															

Table 9-12. AES_AUTH_LENGTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AUTH	R/W	0x0	Authentication Data Length. This field stores the authentication data length in bytes for combined modes (GCM or CCM).

9.5.6 AES_DATA_IN_0 to AES_DATA_IN_3 Registers (Offset = 0x60 to 0x6C) [reset = 0x0]

AES Data R/W Plaintext/Ciphertext 0 (AES_DATA_IN_0), offset 0x060

AES Data R/W Plaintext/Ciphertext 1 (AES_DATA_IN_1), offset 0x064

AES Data R/W Plaintext/Ciphertext 2 (AES_DATA_IN_2), offset 0x068

AES Data R/W Plaintext/Ciphertext 3 (AES_DATA_IN_3), offset 0x06C

The AES Data R/W Plaintext/Ciphertext n (AES_DATA_IN_n) registers are used to read and write plaintext/ciphertext. The AES_DATA_IN_0 register contains the most significant word; the AES_DATA_IN_3 register contains least significant word.

NOTE: The AES_DATA_IN_0 register acts as a FIFO and shifts data into the other AES_DATA_IN_n registers.

AES_DATA_IN_n is shown in [Figure 9-19](#) and described in [Table 9-13](#).

Return to [Summary Table](#).

Figure 9-19. AES_DATA_IN_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0x0																															

Table 9-13. AES_DATA_IN_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0x0	Secure Data R/W Plaintext/Ciphertext. This field holds the plaintext/ciphertext data.

9.5.7 AES_TAG_OUT_0 to AES_TAG_OUT_3 Registers (Offset = 0x70 to 0x7C) [reset = 0x0]

AES Hash Tag Out 0 (AES_TAG_OUT_0), offset 0x070

AES Hash Tag Out 1 (AES_TAG_OUT_1), offset 0x074

AES Hash Tag Out 2 (AES_TAG_OUT_2), offset 0x078

AES Hash Tag Out 3 (AES_TAG_OUT_3), offset 0x07C

This register displays the Hash result. The AES_TAG_OUT_0 register is the most significant word of the Hash and the AES_TAG_OUT_3 register is the least significant word.

AES_TAG_OUT_n is shown in [Figure 9-20](#) and described in [Table 9-14](#).

Return to [Summary Table](#).

Figure 9-20. AES_TAG_OUT_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HASH																															
R/W-0x0																															

Table 9-14. AES_TAG_OUT_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH	R/W	0x0	Hash Result. This field holds the hash result.

9.5.8 AES_REVISION Register (Offset = 0x80) [reset = 0x41]

AES IP Revision Identifier (AES_REVISION)

This register contains the IP revision number of the AES.

AES_REVISION is shown in [Figure 9-21](#) and described in [Table 9-15](#).

Return to [Summary Table](#).

Figure 9-21. AES_REVISION Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															
R-0x41																															

Table 9-15. AES_REVISION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REVISION	R	0x41	Revision number

9.5.9 AES_SYSCONFIG Register (Offset = 0x84) [reset = 0x1]

AES System Configuration (AES_SYSCONFIG)

This register controls the IDLE and reset logic.

NOTE: After one operation has completed, the AES_SYSCONFIG register must be cleared and reconfigured for the next operation to ensure proper DMA and data operation functionality.

AES_SYSCONFIG is shown in [Figure 9-22](#) and described in [Table 9-16](#).

Return to [Summary Table](#).

Figure 9-22. AES_SYSCONFIG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED			K3	KEYENC	RESERVED	MAP_CONTEXT_OUT_ON_DATA_OUT	DMA_REQ_CONTEXT_OUT_EN
R-0x0			R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
DMA_REQ_CONTEXT_IN_EN	DMA_REQ_DATA_OUT_EN	DMA_REQ_DATA_IN_EN	RESERVED			SOFTRESET	RESERVED
R/W-0x0	R/W-0x0	R/W-0x0	R-0x0			R/W-0x0	R-0x1

Table 9-16. AES_SYSCONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12	K3	R/W	0x0	K3 Select 0x0 = A regular cryptographic operation is performed. 0x1 = The K3 key is used as the key for the selected cryptographic operation. The key size should be 128-bits. This bit may only be set to one if the KEYENC bit of this register is cleared to zero.
11	KEYENC	R/W	0x0	Key Encoding 0x0 = A regular cryptographic operation is performed 0x1 = The KEK key (see KEKMODE bit) is XOR-ed with a predefined constant value before it is used as a key for the selected cryptographic operation.
10	RESERVED	R	0x0	
9	MAP_CONTEXT_OUT_ON_DATA_OUT	R/W	0x0	Map Context Out on Data Out Enable 0x0 = Original context out bit values are used. 0x1 = The DMA_REQ_CONTEXT_OUT_EN bit in this register and the CONTEXT_OUT bit in the AES_IRQENABLE register are mapped on the corresponding DATA_OUT request bits. In this case, the original CONTEXT_OUT bit values are ignored.
8	DMA_REQ_CONTEXT_OUT_EN	R/W	0x0	DMA Request Context Out Enable. If set to 1 [€] , the DMA context output request is enabled (for context data out, for example, TAG for authentication modes). The dma_done indication bits in AES_DMARIS register, at CCM module offset 0x024, identify to the application when the DMA transfer is complete. 0x0 = DMA disabled for context output request. 0x1 = DMA enabled for context output request.

Table 9-16. AES_SYSCONFIG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	DMA_REQ_CONTEXT_IN_EN	R/W	0x0	DMA Request Context In Enable. The dma_done indication bits in AES_DMARIS register, at CCM offset 0x024, identify to the application when the DMA transfer is complete. 0x0 = DMA disabled for context input request. 0x1 = DMA enabled for context input request.
6	DMA_REQ_DATA_OUT_EN	R/W	0x0	DMA Request Data Out Enable. The dma_done indication bits in AES_DMARIS register, at CCM offset 0x024, identify to the application when the DMA transfer is complete. 0x0 = DMA disabled for data output request. 0x1 = DMA enabled for data output request.
5	DMA_REQ_DATA_IN_EN	R/W	0x0	DMA Request Data In Enable. The dma_done indication bits in AES_DMARIS register, at CCM module offset 0x024, identify to the application when the DMA transfer is complete. 0x0 = DMA disabled for data input request. 0x1 = DMA enabled for data input request.
4-2	RESERVED	R	0x0	
1	SOFTRESET	R/W	0x0	Soft reset 0x0 = No operation 0x1 = Start soft reset sequence
0	RESERVED	R	0x1	

9.5.10 AES_SYSSTATUS Register (Offset = 0x88) [reset = 0x1]

AES System Status (AES_SYSSTATUS)

This register indicates if reset has completed.

AES_SYSSTATUS is shown in [Figure 9-23](#) and described in [Table 9-17](#).

Return to [Summary Table](#).

Figure 9-23. AES_SYSSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							RESETDONE
R-0x0							R-0x1

Table 9-17. AES_SYSSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	RESETDONE	R	0x1	Reset Done 0x0 = Reset is not complete. 0x1 = Reset is has completed.

9.5.11 AES_IRQSTATUS Register (Offset = 0x8C) [reset = 0x0]

AES Interrupt Status (AES_IRQSTATUS)

This register indicates the interrupt status. If one of the interrupt bits is set, the interrupt output is asserted.

AES_IRQSTATUS is shown in [Figure 9-24](#) and described in [Table 9-18](#).

Return to [Summary Table](#).

Figure 9-24. AES_IRQSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				CONTEXT_OUT	DATA_OUT	DATA_IN	CONTEXT_IN
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 9-18. AES_IRQSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	CONTEXT_OUT	R	0x0	Context Output Interrupt Status 0x0 = Authentication tag (and IV) interrupt(s) is/are not active. 0x1 = Authentication tag (and IV) interrupt(s) is/are active and the interrupt output has been triggered.
2	DATA_OUT	R	0x0	Data Out Interrupt Status 0x0 = The data out interrupt is not active. 0x1 = The data out interrupt is active and the interrupt output has been triggered.
1	DATA_IN	R	0x0	Data In Interrupt Status 0x0 = The data in interrupt is not active. 0x1 = The data in interrupt is active and the interrupt output has been triggered.
0	CONTEXT_IN	R	0x0	Context In Interrupt Status 0x0 = The context in interrupt is not active. 0x1 = The context in interrupt is active and the interrupt output has been triggered.

9.5.12 AES_IRQENABLE Register (Offset = 0x90) [reset = 0x0]

AES Interrupt Enable (AES_IRQENABLE)

This register contains an enable bit for each unique interrupt generated by the module. It matches the layout of AES_IRQSTATUS register. An interrupt is enabled when the bit in this register is set to 1. An interrupt that is enabled is propagated to the NVIC controller. All AES software interrupts need to be enabled explicitly by writing this register.

NOTE: If the application uses Interrupt Mode, an interrupt is generated for each block of processed data. To support larger data flow, AES μ DMA Mode should be used and the bits in the AES_IRQENABLE register should be cleared.

AES_IRQENABLE is shown in [Figure 9-25](#) and described in [Table 9-19](#).

Return to [Summary Table](#).

Figure 9-25. AES_IRQENABLE Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				CONTEXT_OUT	DATA_OUT	DATA_IN	CONTEXT_IN
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 9-19. AES_IRQENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	CONTEXT_OUT	R/W	0x0	Context Out Interrupt Enable 0x0 = Authentication tag (and IV) interrupt(s) is/are disabled. 0x1 = Authentication tag (and IV) interrupt(s) is/are enabled.
2	DATA_OUT	R/W	0x0	Data Out Interrupt Enable 0x0 = The data out interrupt is disabled. 0x1 = The data out interrupt is enabled.
1	DATA_IN	R/W	0x0	Data In Interrupt Enable 0x0 = The data in interrupt is disabled. 0x1 = The data in interrupt is enabled.
0	CONTEXT_IN	R/W	0x0	Context In Interrupt Enable 0x0 = The context in interrupt is disabled. 0x1 = The context in interrupt is enabled.

9.5.13 AES_DIRTYBITS Register (Offset = 0x94) [reset = 0x0]

AES Dirty Bits (AES_DIRTYBITS)

This register can be used to identify if AES registers have been read or written to.

AES_DIRTYBITS is shown in [Figure 9-26](#) and described in [Table 9-20](#).

Return to [Summary Table](#).

Figure 9-26. AES_DIRTYBITS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						S_DIRTY	S_ACCESS
R-0x0						R/W1C-0x0	R/W1C-0x0

Table 9-20. AES_DIRTYBITS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	S_DIRTY	R/W1C	0x0	AES Dirty Bit. This bit must be written to a 1 to clear. 0x0 = No AES registers have been written. 0x1 = Indicates when any of the AES_x registers have been written (except for the AES_DIRTYBITS register).
0	S_ACCESS	R/W1C	0x0	AES Access Bit. This bit must be written to a 1 to clear. 0x0 = No AES registers have been read. 0x1 = Indicates when any of the AES_x registers have been read (except for the AES_DIRTYBITS register).

9.6 AES μ DMA Registers

[Table 9-21](#) lists the memory-mapped registers for the AES μ DMA. All register offset addresses not listed in [Table 9-21](#) should be considered as reserved locations and the register contents should not be modified.

The AES μ DMA interrupt register offsets are relative to the base address 0x44030000.

Table 9-21. AES μ DMA Registers

Offset	Acronym	Register Name	Section
0x20	AES_DMAIM	AES DMA Interrupt Mask	Section 9.6.1
0x24	AES_DMARIS	AES DMA Raw Interrupt Status	Section 9.6.2
0x28	AES_DMAMIS	AES DMA Masked Interrupt Status	Section 9.6.3
0x2C	AES_DMAIC	AES DMA Interrupt Clear	Section 9.6.4

Complex bit access types are encoded to fit into small table cells. [Table 9-22](#) shows the codes that are used for access types in this section.

Table 9-22. AES μ DMA Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

9.6.1 AES_DMAIM Register (Offset = 0x20) [reset = 0x0]

AES DMA Interrupt Mask (AES_DMAIM)

The AES DMA Interrupt Mask (AES_DMAIM) register controls interrupt behavior and is used to program which interrupts are suppressed.

AES_DMAIM is shown in [Figure 9-27](#) and described in [Table 9-23](#).

Return to [Summary Table](#).

Figure 9-27. AES_DMAIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				DOUT	DIN	COUT	CIN
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 9-23. AES_DMAIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	DOUT	R/W	0x0	Data Out DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA writes the last word of the process result. 0x0 = The DOUT interrupt is suppressed and not sent to the interrupt controller. 0x1 = The DOUT interrupt is sent to the interrupt controller.
2	DIN	R/W	0x0	Data In DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA writes the last word of input data to the internal FIFO of the engine. 0x0 = The DIN interrupt is suppressed and not sent to the interrupt controller. 0x1 = The DIN interrupt is sent to the interrupt controller.
1	COUT	R/W	0x0	Context Out DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA completes the output context read from the internal register. 0x0 = The COUT interrupt is suppressed and not sent to the interrupt controller. 0x1 = The COUT interrupt is sent to the interrupt controller.
0	CIN	R/W	0x0	Context In DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA completes a context write to the internal register. 0x0 = The CIN interrupt is suppressed and not sent to the interrupt controller. 0x1 = The CIN interrupt is sent to the interrupt controller.

9.6.2 AES_DMARIS Register (Offset = 0x24) [reset = 0x0]

AES DMA Raw Interrupt Status (AES_DMARIS)

The AES DMA Raw Interrupt Status (AES_DMARIS) register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set to 1.

AES_DMARIS is shown in [Figure 9-28](#) and described in [Table 9-24](#).

Return to [Summary Table](#).

Figure 9-28. AES_DMARIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				DOUT	DIN	COUT	CIN
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 9-24. AES_DMARIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	DOUT	R/W	0x0	Data Out DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has written the last word of the process result and an interrupt has been triggered and is pending.
2	DIN	R/W	0x0	Data In DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has written the last word of input data to the internal FIFO of the engine and an interrupt has been triggered and is pending.
1	COUT	R/W	0x0	Context Out DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has completed the output context read from the internal register and an interrupt has been triggered and is pending.
0	CIN	R/W	0x0	Context In DMA Done Raw Interrupt Status 0x0 = No interrupt. 0x1 = The μ DMA has completed a context write to the internal register and an interrupt has been triggered and is pending.

9.6.3 AES_DMAMIS Register (Offset = 0x28) [reset = 0x0]

AES DMA Masked Interrupt Status (AES_DMAMIS)

The AES DMA Masked Interrupt Status (AES_DMAMIS) register displays the raw interrupts that are unmasked in the AES DMA Raw Interrupt Status (AES_DMARIS) register.

AES_DMAMIS is shown in [Figure 9-29](#) and described in [Table 9-25](#).

Return to [Summary Table](#).

Figure 9-29. AES_DMAMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				DOUT	DIN	COUT	CIN
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 9-25. AES_DMAMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	DOUT	R	0x0	Data Out DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A DOUT interrupt has occurred.
2	DIN	R	0x0	Data In DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A DIN interrupt has occurred.
1	COUT	R	0x0	Context Out DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A COUT interrupt has occurred.
0	CIN	R	0x0	Context In DMA Done Raw Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A CIN interrupt has occurred.

9.6.4 AES_DMAIC Register (Offset = 0x2C) [reset = 0x0]

AES DMA Interrupt Clear (AES_DMAIC)

The AES DMA Interrupt Clear (AES_DMAIC) register is used to clear the AES_DMARIS and AES_DMAMIS registers by writing a 1 to each register bit.

NOTE: This registers always reads as zero.

AES_DMAIC is shown in [Figure 9-30](#) and described in [Table 9-26](#).

Return to [Summary Table](#).

Figure 9-30. AES_DMAIC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				DOUT	DIN	COUT	CIN
R-0x0				W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0

Table 9-26. AES_DMAIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	DOUT	W1C	0x0	Data Out DMA Done Interrupt Clear. Writing a 1 to this bit clears the DOUT bit in the AES_DMARIS and AES_DMAMIS register.
2	DIN	W1C	0x0	Data In DMA Done Interrupt Clear. Writing a 1 to this bit clears the DIN bit in the AES_DMARIS and AES_DMAMIS register.
1	COUT	W1C	0x0	Context Out DMA Done Masked Interrupt Status. Writing a 1 to this bit clears the COUT bit in the AES_DMARIS and AES_DMAMIS register.
0	CIN	W1C	0x0	Context In DMA Done Raw Interrupt Status. Writing a 1 to this bit clears the CIN bit in the AES_DMARIS and AES_DMAMIS register.

Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter modules are included, which share 24 input channels.

The ADC module features 12-bit conversion resolution and supports 24 input channels, plus an internal temperature sensor. Each ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. In addition, the conversion value can optionally be diverted to a digital comparator module. Each ADC module provides eight digital comparators. Each digital comparator evaluates the ADC conversion value against its two user-defined values to determine the operational range of the signal. The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. A phase shifter can delay the start of sampling by a specified phase angle. When using both ADC modules, it is possible to configure the converters to start the conversions simultaneously or within a relative phase from each other, see [Section 10.3.2.6](#).

Topic	Page
10.1 Introduction	703
10.2 Block Diagram.....	703
10.3 Functional Description	704
10.4 Initialization and Configuration	717
10.5 ADC Registers.....	719

10.1 Introduction

The MSP432E4 microcontroller provides two ADC modules with each having the following features:

- 24 shared analog input channels
- 12-bit precision ADC
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of two million samples/second
- Optional, programmable phase delay
- Sample and hold window programmability
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
 - Controller (software)
 - Timers
 - Analog Comparators
 - PWM
 - GPIO
- Hardware averaging of up to 64 samples
- Eight digital comparators
- Converter uses two external reference signals (VREFA+ and VREFA-) or VDDA and GNDA as the voltage reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Dedicated channel for each sample sequencer
 - ADC module uses burst requests for DMA
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate ADC clock

10.2 Block Diagram

The MSP432E4 microcontroller contains two identical ADC modules. These two modules, ADC0 and ADC1, share the same 24 analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. [Figure 10-1](#) shows how the two modules are connected to analog inputs and the system bus.

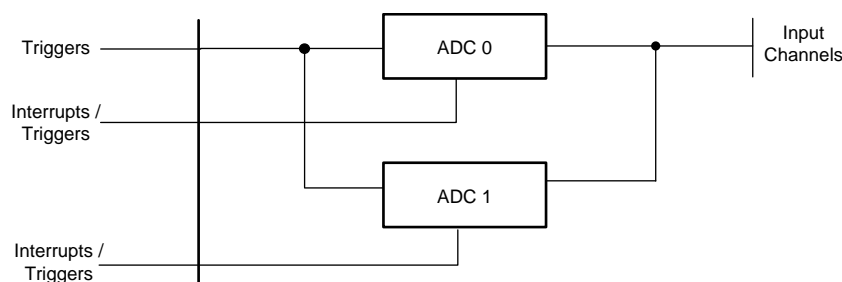


Figure 10-1. Implementation of Two ADC Blocks

[Figure 10-2](#) provides details on the internal configuration of the ADC controls and data registers.

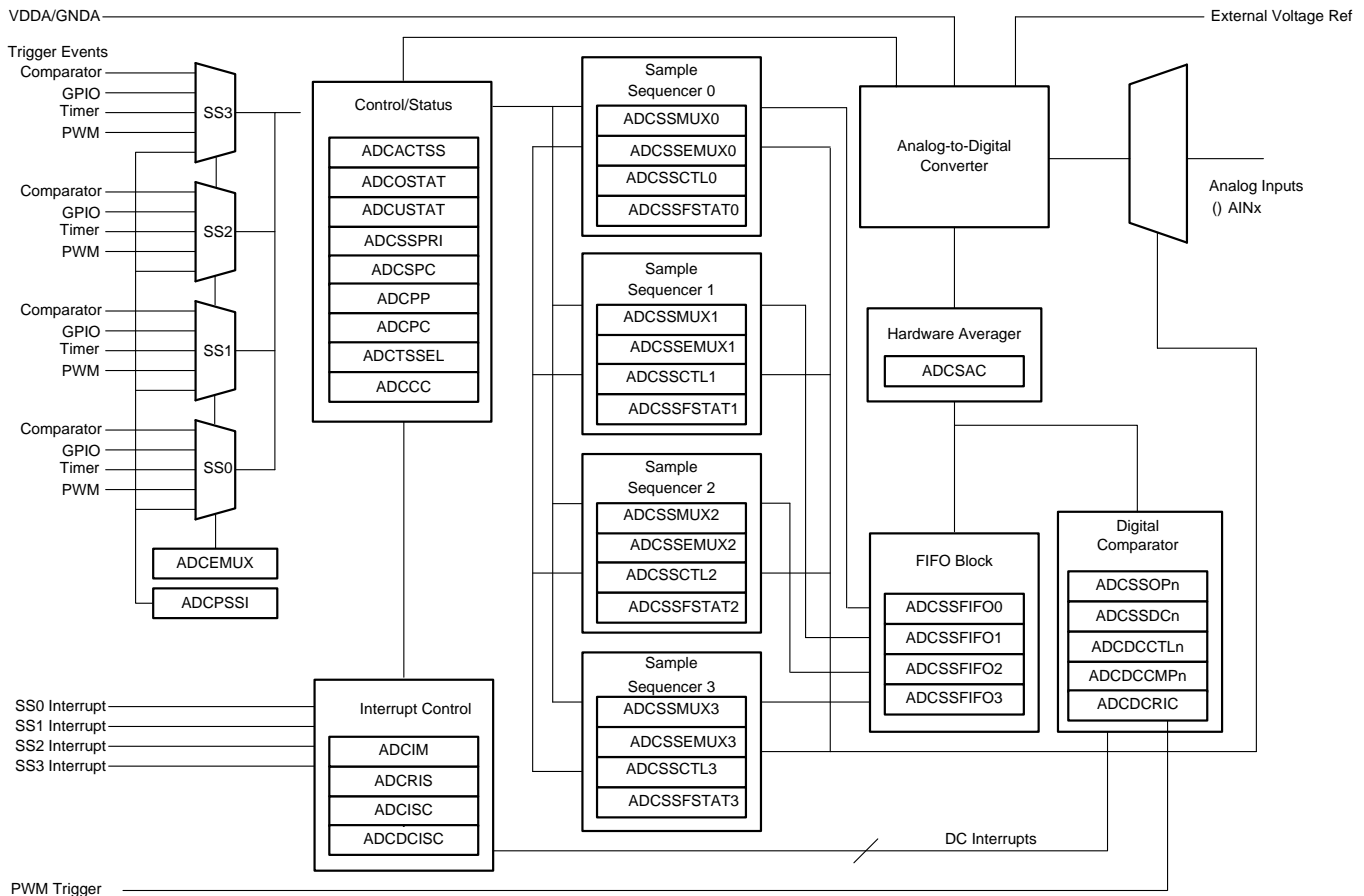


Figure 10-2. ADC Module Block Diagram

10.3 Functional Description

The ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each sample sequence is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential or single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence. In addition, the μ DMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

10.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 10-1 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. Each sample that is captured is stored in the FIFO. In this implementation, each FIFO entry is a 32-bit word, with the lower 12 bits containing the conversion result.

Table 10-1. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4

Table 10-1. Samples and FIFO Depth of Sequencers (continued)

Sequencer	Number of Samples	Depth of FIFO
SS0	8	8

For a given sample sequence, each sample is defined by bit fields in the ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn), ADC Sample Sequence Extended Input Multiplexer Select (ADCSEMUXn) and ADC Sample Sequence Control (ADCSSCTLn) registers, where "n" corresponds to the sequence number. The ADCSSMUXn and ADCSEMUXn fields select the input pin, while the ADCSSCTLn fields contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective ASENn bit in the ADC Active Sample Sequencer (ADCACTSS) register and should be configured before being enabled. Sampling is then initiated by setting the SSn bit in the ADC Processor Sample Sequence Initiate (ADCPSSI) register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the GSYNC and SYNCWAIT bits in the ADCPSSI register during the configuration of each ADC module. For more information on using these bits, see [Section 10.5.11](#).

When configuring a sample sequence, multiple uses of the same input pin within the same sequence are allowed. In the ADCSSCTLn register, the IEn bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the END bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the END bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the ADC Sample Sequence Result FIFO (ADCSSFIFO) registers. The FIFOs are simple circular buffers that read a single address to pop result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the ADC Sample Sequence FIFO Status (ADCSSFSTATn) registers along with FULL and EMPTY status flags. If a write is attempted when the FIFO is full, the write does not occur and an overflow condition is indicated. Overflow and underflow conditions are monitored using the ADCOSTAT and ADCUSTAT registers.

10.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- DMA operation
- Sequence prioritization
- Trigger configuration
- Comparator configuration
- External voltage reference
- Sample phase control
- Module clocking

10.3.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the MASK bits in the ADC Interrupt Mask (ADCIM) register. Interrupt status can be viewed at two locations: the ADC Raw Interrupt Status (ADCRIS) register, which shows the raw status of the various interrupt signals; and the ADC Interrupt Status and Clear (ADCISC) register, which shows active interrupts that are enabled by the ADCIM register. Sequencer interrupts are cleared by writing a 1 to the corresponding IN bit in ADCISC. Digital comparator interrupts are cleared by writing a 1 to the ADC Digital Comparator Interrupt Status and Clear (ADCDCISC) register.

10.3.2.2 DMA Operation

DMA may be used to increase efficiency by allowing each sample sequencer to operate independently and transfer data without processor intervention or reconfiguration.

The ADC asserts single and burst μ DMA request signals (dma_sreq and dma_req) to the μ DMA controller based on the FIFO level. The dma_req signal is generated when the FIFO in question is half-full (that is, at 4 samples for SS0, 2 samples for SS1 and SS2, and at 1 sample for SS3). If, for example, the ADCSSCTL0 register has six samples to transfer, a burst of four values occurs followed by two single transfers (dma_sreq). The dma_done signals (one per sample sequencer) are sent to the ADC to allow for a triggering of DMAINRn interrupt bits in the ADCRIS register. The μ DMA is enabled for a specific sample sequencer by setting the appropriate ADENn bit in the ADCACTSS register at offset 0x000.

To use the μ DMA with the ADC module, the application must enable the ADC channel through DMA Channel Map Select n (DMACHMAPn) register in the μ DMA.

See the [Chapter 8](#) for more details about programming the μ DMA controller.

10.3.2.3 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the ADC Sample Sequencer Priority (ADCSSPRI) register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active sample sequencer units with the same priority do not provide consistent results, so software must ensure that all active sample sequencer units have a unique priority value.

10.3.2.4 Sampling Events

Sample triggering for each sample sequencer is defined in the ADC Event Multiplexer Select (ADCEMUX) register. Trigger sources include processor (default), analog comparators, an external signal on a GPIO specified by the GPIO ADC Control (GPIOADCCTL) register, a GP Timer, a PWM generator, and continuous sampling. The processor triggers sampling by setting the SSx bits in the ADC Processor Sample Sequence Initiate (ADCPSSI) register.

Care must be taken when using the continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers. Generally, a sample sequencer using continuous sampling should be set to the lowest priority. Continuous sampling can be used with a digital comparator to cause an interrupt when a particular voltage is seen on an input.

10.3.2.5 Sample and Hold Window Control

The ADC module provides the capability of programming the sample and hold window of each step in a sequence through the ADC Sample Sequence n Sample and Hold Time (ADCSSTSHn) register. Each TSHn field can be written with a different sample and hold width, which is represented in ADC clocks.

[Table 10-2](#) lists the allowed encodings:

Table 10-2. Sample and Hold Width in ADC Clocks

TSHn Encoding	N _{SH}
0x0	4
0x1	Reserved
0x2	8
0x3	Reserved
0x4	16
0x5	Reserved
0x6	32
0x7	Reserved
0x8	64
0x9	Reserved
0xA	128

Table 10-2. Sample and Hold Width in ADC Clocks (continued)

TSHn Encoding	N _{SH}
0xB	Reserved
0xC	256
0xD-0xF	Reserved

The ADC conversion frequency is a function of the Sample and Hold number, given by the following equation:

$$f_{\text{CONV}} = 1 / ((N_{\text{SH}} + 12) \times T_{\text{ADC}})$$

where:

- N_{SH} is the sample and hold width in ADC clocks
- T_{ADC} is the ADC conversion clock period, which is the inverse of the ADC clock frequency f_{ADC}

Now, the maximum allowable external source resistance (R_S) also changes with the value of N_{SH}, as the total settling time of the input circuitry must be fast enough to settle to within the ADC resolution in a single sampling interval. The input circuitry includes the external source resistance as well as the input resistance and capacitance of the ADC (R_{ADC} and C_{ADC}).

The values for R_S and f_{CONV} for varying N_{SH} values, with f_{ADC} = 16 MHz and f_{ADC} = 32 MHz are given in tables 18-4-a and 18-4-b. The system designer must take into consideration both of these factors for optimal ADC operation.

Table 10-3. R_S and f_{CONV} Values with Varying N_{SH} Values and f_{ADC} = 16 MHz

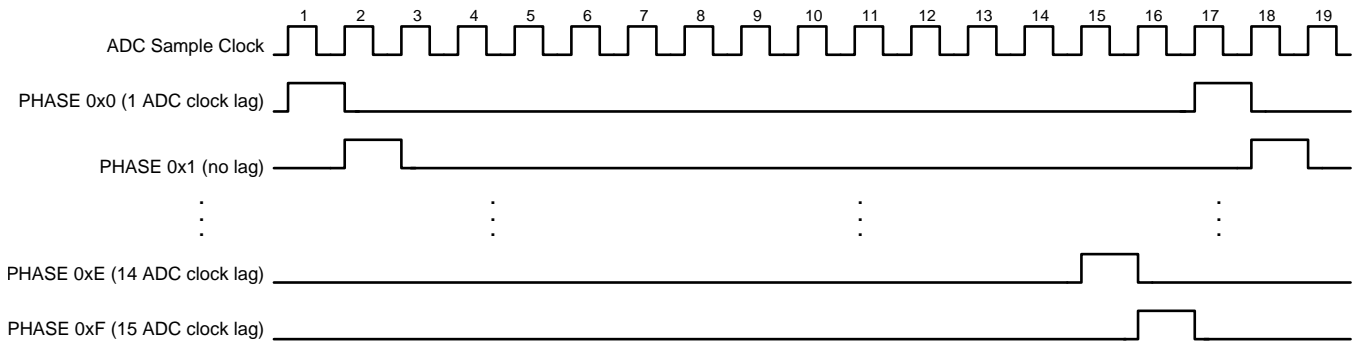
N _{SH} (Cycles)	4	8	16	32	64	128	256
f _{CONV} (ksps)	1000	800	571	364	211	114	60
R _S max (Ω)	500	3500	9500	21500	45500	93500	189500

Table 10-4. R_S and f_{CONV} Values with Varying N_{SH} Values and f_{ADC} = 32 MHz

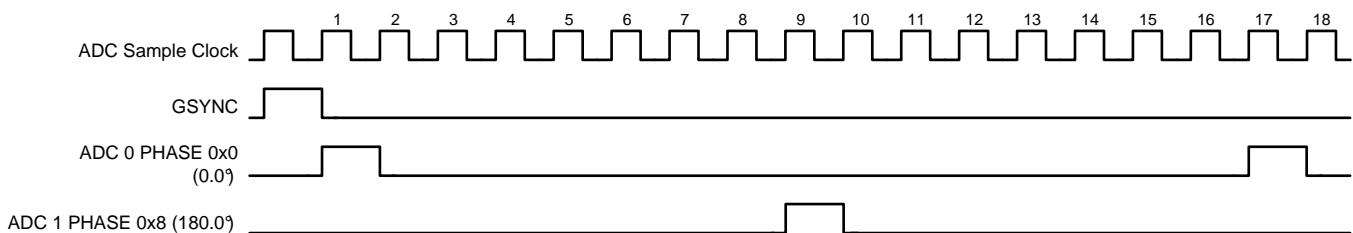
N _{SH} (Cycles)	4	8	16	32	64	128	256
f _{CONV} (ksps)	2000	1600	1143	727	421	229	119
R _S max (Ω)	250	500	3500	9500	21500	45500	93500

10.3.2.6 Sample Phase Control

The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. If the converters are running at the same sample rate, they may be configured to start the conversions coincidentally or one ADC may be programmed to lag up to 15 clock cycles relative to the other ADC. The sample time can be delayed from the standard sampling time by programming the PHASE field in the ADC Sample Phase Control (ADCSPC) register. [Figure 10-3](#) shows an example of various phase relationships.


Figure 10-3. ADC Sample Phases

This feature can be used to double the sampling rate of an input. Both ADC Module 0 and ADC Module 1 can be programmed to sample the same input. ADC module 0 can sample at the standard position (the PHASE field in the ADCSPC register is 0x0). ADC Module 1 can be configured to sample with a phase lag (PHASE is nonzero). For a sample rate of two million samples per second at 16 MHz, the TSHn field of all of the sequencer samples of both ADCs must be programmed to 0x0 and the PHASE field of one of the ADC modules must be set to 0x8. The two modules can be synchronized using the GSYNC and SYNCWAIT bits in the ADC Processor Sample Sequence Initiate (ADCPSSI) register. Software can then combine the results from the two modules to create a sample rate of two million samples/second at 16MHz as shown in Figure 10-4.


Figure 10-4. Doubling the ADC Sample Rate

Using the ADCSPC register, ADC0 and ADC1 may provide a number of interesting applications:

- Coincident continuous sampling of different signals. The sample sequence steps run coincidently in both converters. In this situation, the TSHn of matching sample steps of both ADC module sequencers must be the same and the PHASE field must be 0x0 in both ADC module ADCSPC registers. The TSHn field is found in the ADC Sample Sequence n Sample and Hold Time (ADCSSTSHn) register.
 - ADC Module 0, ADCSPC = 0x0, sampling AIN0
 - ADC Module 1, ADCSPC = 0x0, sampling AIN1

NOTE: If two ADCs are configured to sample the same signal, a skew (phase lag) must be added to one of the ADC modules to prevent coincident sampling. Phase lag can be added by programming the PHASE field in the ADCSPC register.

- Skewed sampling of the same signal. The skew is determined by both the TSHn field in the ADCSSTSHn registers and the PHASE field in the ADCSPC register. For the fastest skewed sample rate, all TSHn fields must be programmed to 0x0. If TSHn = 0x0 for all sequencers and the PHASE field of one ADC is 0x8, the configuration doubles the conversion bandwidth of a single input when software combines the results as shown in Figure 10-5.
 - ADC Module 0, ADCSPC = 0x0, sampling AIN0
 - ADC Module 1, ADCSPC = 0x8, sampling AIN0

Note that it is not required that the TSHn fields be the same in a skewed sample. If an application has varying analog input resistance, then TSHn and PHASE may vary according to operational requirements.

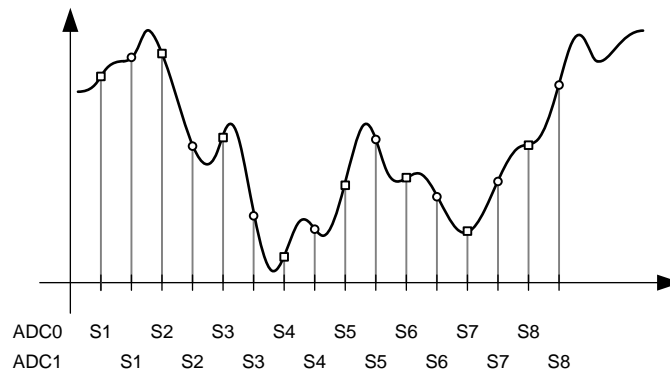


Figure 10-5. Skewed Sampling

10.3.2.7 Module Clocking

The ADC digital block is clocked by the system clock and the ADC analog block is clocked from a separate conversion clock (ADC Clock). The ADC clock frequency can be up to 32 MHz to generate a conversion rate of 2 Msps. A 16 MHz ADC clock provides a 1 Msps sampling rate. There are three sources of the ADC clock:

- Divided PLL VCO. The PLL VCO frequency can be configured to generate up to a 32-MHz clock for a conversion rate of 2 Msps. The CS field in the ADCCC register must be programmed to 0x0 to select the PLL VCO and the CLKDIV field is used to set the appropriate clock divisor for the desired frequency.
- 16 MHz PIOSC. Using the PIOSC provides a conversion rate near 1 Msps. To use the PIOSC to clock the ADC, first power up the PLL and then enable the PIOSC in the CS bit field in the ADCCC register, then disable the PLL.
- MOSC. The MOSC clock source must be 16 MHz for a 1 Msps conversion rate and 32 MHz for a 2 Msps conversion rate.

The system clock must be at the same frequency or higher than the ADC clock. All ADC modules share the same clock source to facilitate the synchronization of data samples between conversion units, the selection and programming of which is provided by the ADCCC register of ADC0. The ADC modules do not run at different conversion rates.

10.3.2.8 Busy Status

The BUSY bit of the ADCACTSS register is used to indicate when the ADC is busy with a current conversion. When there are no triggers pending which may start a new conversion in the immediate cycle or next few cycles, the BUSY bit reads as 0. Software must read the status of the BUSY bit as clear before disabling the ADC clock by writing to the ADC Run Mode Clock Gating Control (RCGCADC) register.

10.3.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the ADC Sample Averaging Control (ADCSAC) register (see [Section 10.5.12](#)). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

Figure 10-6 shows an example in which the ADCSAC register is set to 0x2 for 4x hardware oversampling and the IE1 bit is set for the sample sequence, resulting in an interrupt after the second averaged value is stored in the FIFO.

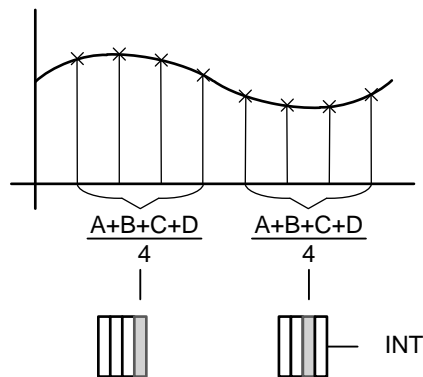


Figure 10-6. Sample Averaging Example

10.3.4 Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The successive approximation uses a switched capacitor array to perform the dual functions of sampling and holding the signal as well as providing the 12-bit DAC operation.

Figure 10-7 shows the ADC input equivalency diagram; for parameter values, see the device-specific data sheet.

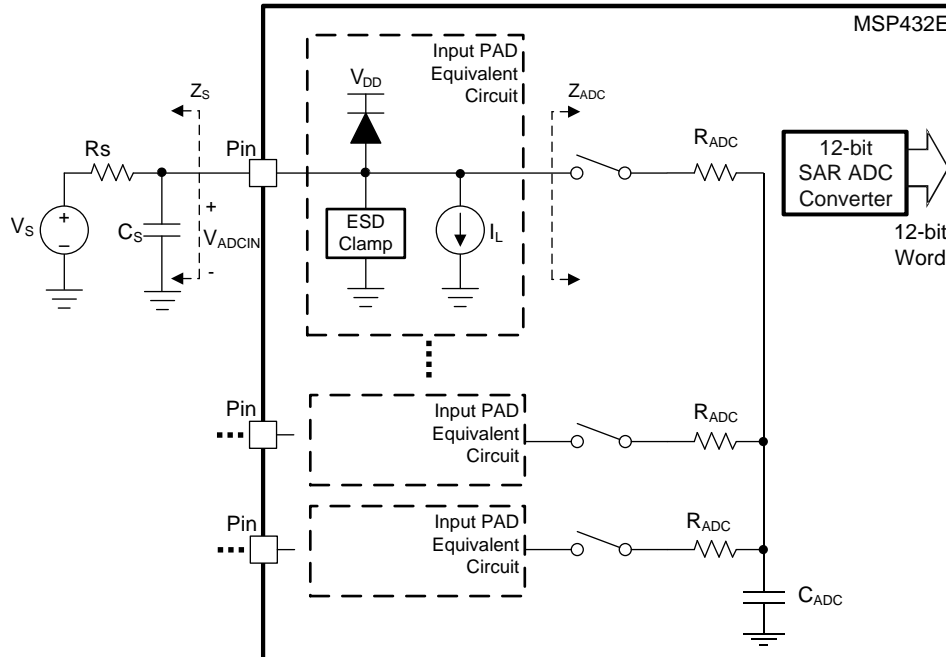


Figure 10-7. ADC Input Equivalency

The ADC operates from both the 3.3-V analog and 1.2-V digital power supplies. The ADC clock can be configured to reduce power consumption when ADC conversions are not required (see Section 4.1.6). The analog inputs are connected to the ADC through specially balanced input paths to minimize the distortion and cross-talk on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in the device-specific data sheet.

10.3.4.1 Voltage Reference

The ADC uses internal signals VREFP and VREFN as references to produce a conversion value from the selected analog input. VREFP can be connected to either VREFA+ or VDDA and VREFN can be connected to either VREFA- or GNDA as configured by the VREF bit in the ADC Control (ADCCTL) register, as shown in [Figure 10-8](#).

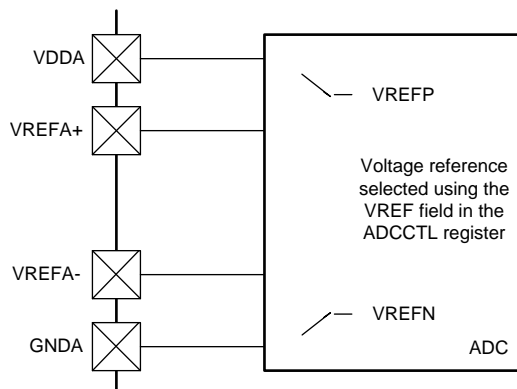


Figure 10-8. ADC Voltage Reference

The range of this conversion value is from 0x000 to 0xFFFF. In single-ended-input mode, the 0x000 value corresponds to the voltage level on VREFN; the 0xFFFF value corresponds to the voltage level on VREFP. This configuration results in a resolution that can be calculated using the following equation:

$$\text{mV per ADC code} = (\text{VREFP} - \text{VREFN}) / 4096$$

While the analog input pads can handle voltages beyond this range, the analog input voltages must remain within the specification limits in the device-specific data sheet to produce accurate results. The $V_{\text{REFA+}}$ and $V_{\text{REFA-}}$ specifications define the useful range for the external voltage references on VREFA+ and VREFA-. Care must be taken to supply a reference voltage of acceptable quality. [Figure 10-9](#) shows the ADC conversion function of the analog inputs.

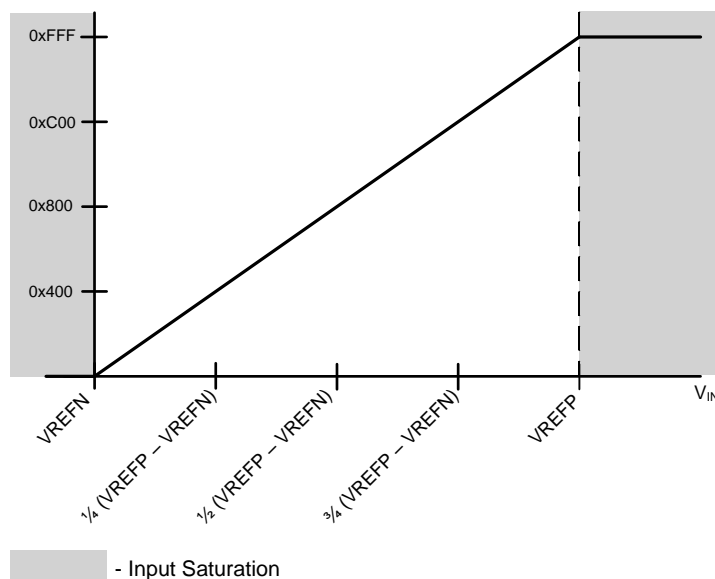


Figure 10-9. ADC Conversion Result

10.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the Dn bit in the ADCSSCTL0n register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the ADCSSMUXn register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see [Table 10-5](#)). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 10-5. Differential Sampling Pairs

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7
4	8 and 9
5	10 and 11
6	12 and 13
7	14 and 15
8	16 and 17
9	18 and 19
10	20 and 21
11	22 and 23

The voltage sampled in differential mode is the difference between the odd and even channels:

- Input Positive Voltage: $V_{IN+} = V_{IN_EVEN}$ (even channel)
- Input Negative Voltage: $V_{IN-} = V_{IN_ODD}$ (odd channel)

The input differential voltage is defined as: $V_{IN_D} = V_{IN+} - V_{IN-}$, therefore:

- If $V_{IN_D} = 0$, then the conversion result = 0x800
- If $V_{IN_D} > 0$, then the conversion result > 0x800 (range is 0x800 to 0xFFFF)
- If $V_{IN_D} < 0$, then the conversion result < 0x800 (range is 0 to 0x800)

When using differential sampling, the following definitions are relevant:

- Input Common Mode Voltage: $V_{IN_CM} = (V_{IN+} + V_{IN-}) / 2$
- Reference Positive Voltage: V_{REFP}
- Reference Negative Voltage: V_{REFN}
- Reference Differential Voltage: $V_{REF_D} = V_{REFP} - V_{REFN}$
- Reference Common Mode Voltage: $V_{REF_CM} = (V_{REFP} + V_{REFN}) / 2$

The following conditions provide optimal results in differential mode:

- Both V_{IN_EVEN} and V_{IN_ODD} must be in the range of (V_{REFP} to V_{REFN}) for a valid conversion result
- The maximum possible differential input swing, or the maximum differential range, is: $-V_{REF_D}$ to $+V_{REF_D}$, so the maximum peak-to-peak input differential signal is:

$$+V_{REF_D} - (-V_{REF_D}) = 2 \times V_{REF_D} = 2 \times (V_{REFP} - V_{REFN}) \quad (5)$$
- To take advantage of the maximum possible differential input swing, V_{IN_CM} should be very close to V_{REF_CM} (see the device-specific data sheet).

If V_{IN_CM} is not equal to V_{REF_CM} , the differential input signal may clip at either maximum or minimum voltage, because either single ended input can never be larger than V_{REFP} or smaller than V_{REFN} , and it is not possible to achieve full swing. Thus any difference in common mode between the input voltage and the reference voltage limits the differential dynamic range of the ADC.

Because the maximum peak-to-peak differential signal voltage is $2 \times (V_{REFP} - V_{REFN})$, the ADC codes are interpreted as:

$$\text{mV per ADC code} = (2 \times (V_{REFP} - V_{REFN})) / 4096 \quad (6)$$

Figure 10-10 shows how the differential voltage, ΔV , is represented in ADC codes.

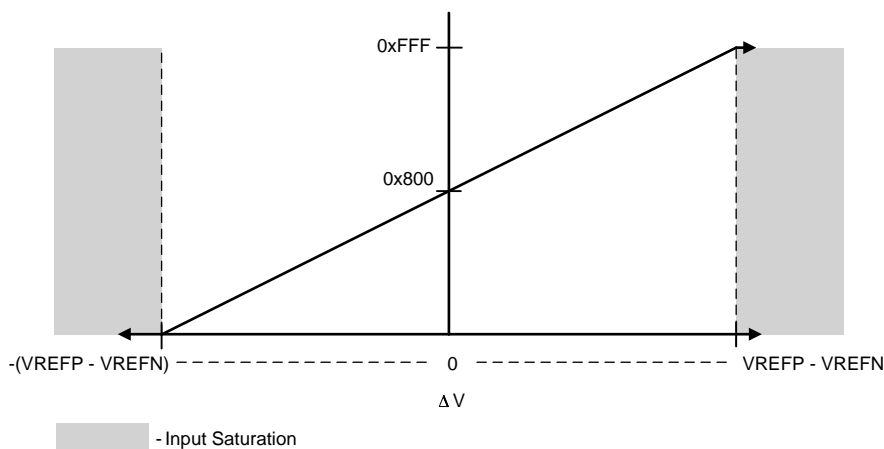


Figure 10-10. Differential Voltage Representation

10.3.6 Internal Temperature Sensor

The temperature sensor serves two primary purposes: 1) to notify the system that internal temperature is too high or low for reliable operation and 2) to provide temperature measurements for calibration of the Hibernate module RTC trim value.

The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC. In addition, the temperature sensor has a second power-down input in the 3.3 V domain which provides control by the Hibernation module.

The internal temperature sensor converts a temperature measurement into a voltage. This voltage value, V_{TSENS} , is given by the following equation (where TEMP is the temperature in °C):

$$V_{TSENS} = 2.7 - ((TEMP + 55) / 75) \quad (7)$$

This relation is shown in Figure 10-11.

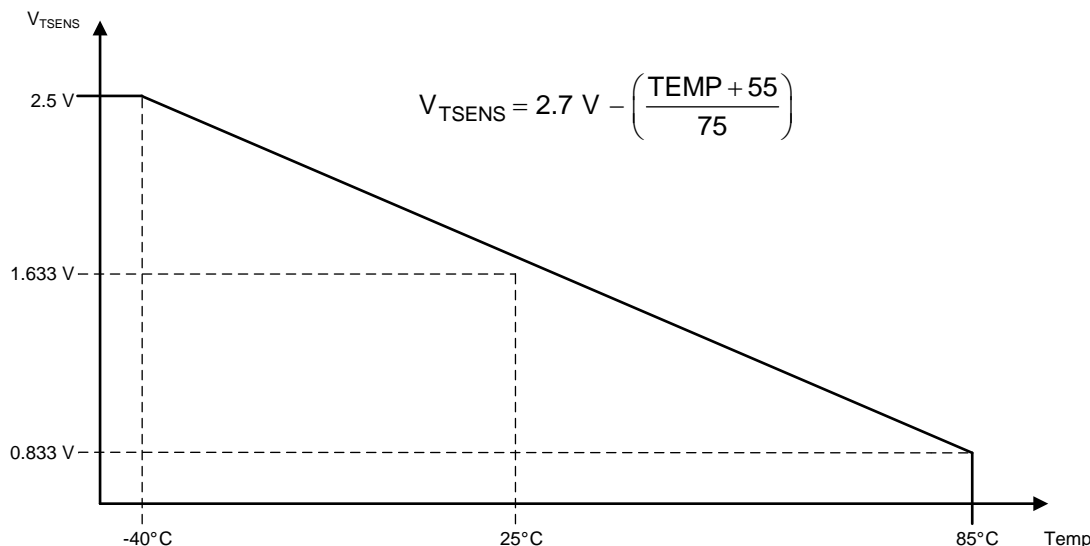


Figure 10-11. Internal Temperature Sensor Characteristic

The temperature sensor reading can be sampled in a sample sequence by setting the TS_n bit in the ADCSSCTL_n register. The sample and hold width should be configured for at least 16 ADC clocks using the ADCSSTSH_n register. The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (TEMP in °C) based on the ADC reading (ADC_{CODE}, given as an unsigned decimal number from 0 to 4095) and the maximum ADC voltage range (VREFP – VREFN) :

$$\text{TEMP} = 147.5 - ((75 \times (\text{VREFP} - \text{VREFN}) \times \text{ADC}_{\text{CODE}}) / 4096) \quad (8)$$

10.3.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, each module provides eight digital comparators.

Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the ADC Digital Comparator Range (ADCDCCMP_n) registers. The ADC can be configured to generate an interrupt depending on whether the ADC is operating within the low, mid or high-band region configured in the ADCDCCMP_n bit fields. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be additionally applied to the interrupt configuration.

10.3.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the SnDCOP bits in the ADC Sample Sequence n Operation (ADCSSOP_n) register. These selected ADC conversions are used by their respective digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

10.3.7.1.1 Interrupts

The digital comparator interrupt function is enabled by setting the CIE bit in the ADC Digital Comparator Control (ADCDCCTL_n) register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the DCONSS_x bit is set in the ADCIM register, an interrupt is sent to the interrupt controller.

NOTE: For a 1 to 2 Msps rate, as the system clock frequency approaches the ADC clock frequency, it is recommended that the application use the μ DMA to store conversion data from the FIFO to memory before processing rather than an interrupt-driven single data read. Using the μ DMA to store multiple samples before interrupting the processor amortizes interrupt overhead across multiple transfers and prevents loss of sample data.

NOTE: Only a single DCONSS_n bit should be set at any given time. Setting more than one of these bits results in the INRDC bit from the ADCRIS register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines. It is recommended that when interrupts are used, they are enabled on alternating samples or at the end of the sample sequence.

10.3.7.1.2 Triggers

The digital comparator trigger function is enabled by setting the CTE bit in the ADCDCCTL_n register. This bit enables the trigger function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, the corresponding digital comparator trigger to the PWM module is asserted.

10.3.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the CIM or CTM field in the ADCDCCTLn register.

10.3.7.2.1 Always Mode

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

10.3.7.2.2 Once Mode

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

10.3.7.2.3 Hysteresis-Always Mode

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2) a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

10.3.7.2.4 Hysteresis-Once Mode

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

10.3.7.3 Function Ranges

The two comparison values, COMP0 and COMP1, in the ADC Digital Comparator Range (ADCDCCMPn) register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than COMP0), mid-band (greater than COMP0 but less than or equal to COMP1), and high-band (greater than or equal to COMP1) regions. COMP0 and COMP1 may be programmed to the same value, effectively creating two regions, but COMP1 must always be greater than or equal to the value of COMP0. A COMP1 value that is less than COMP0 generates unpredictable results.

10.3.7.3.1 Low-Band Operation

To operate in the low-band region, the CIC field or the CTC field in the ADCDCCTLn register must be programmed to 0x0. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in [Figure 10-12](#). Note that a 0 in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is deasserted and a 1 indicates that the signal is asserted.

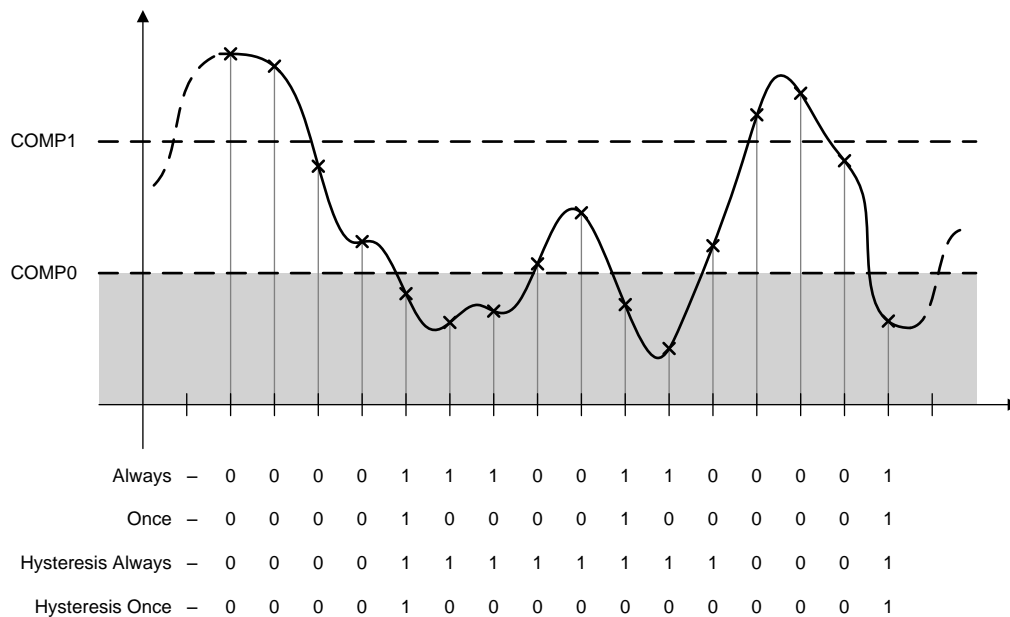


Figure 10-12. Low-Band Operation (CIC = 0x0 or CTC = 0x0)

10.3.7.3.2 Mid-Band Operation

To operate in the mid-band region, the CIC field or the CTC field in the ADCDCCTLn register must be programmed to 0x1. This setting causes interrupts or triggers to be generated in the mid-band region according to the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 10-13. Note that a 0 in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

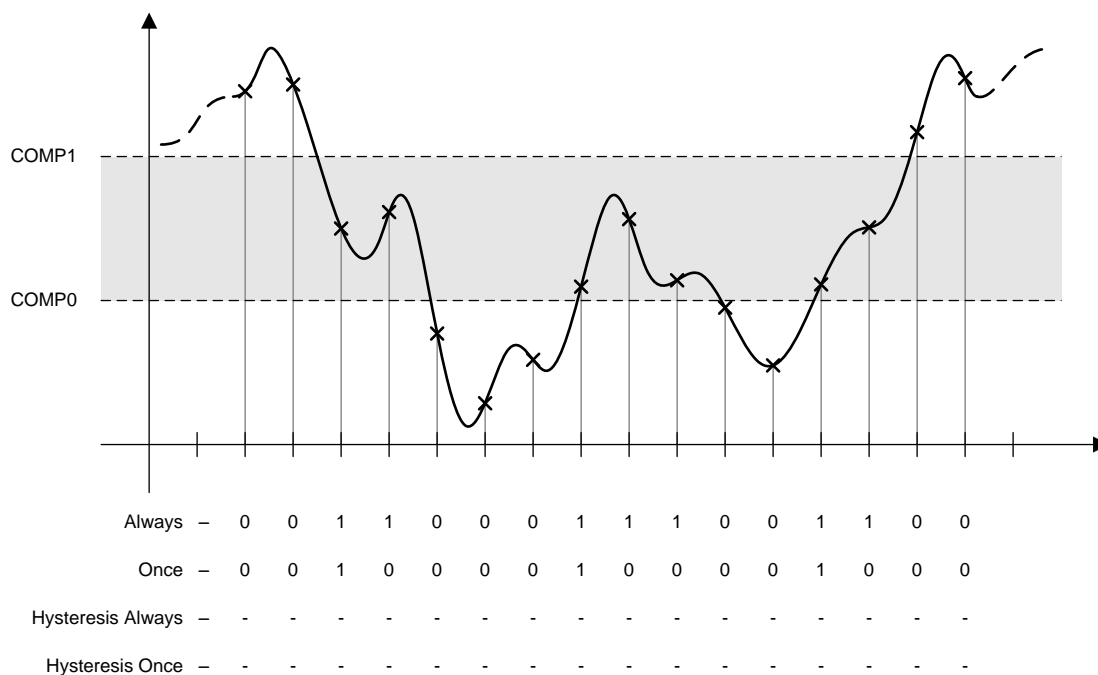


Figure 10-13. Mid-Band Operation (CIC = 0x1 or CTC = 0x1)

10.3.7.3.3 High-Band Operation

To operate in the high-band region, the CIC field or the CTC field in the ADCDCCTLn register must be programmed to 0x3. This setting causes interrupts or triggers to be generated in the high-band region according to the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 10-14. Note that a 0 in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is deasserted and a 1 indicates that the signal is asserted.

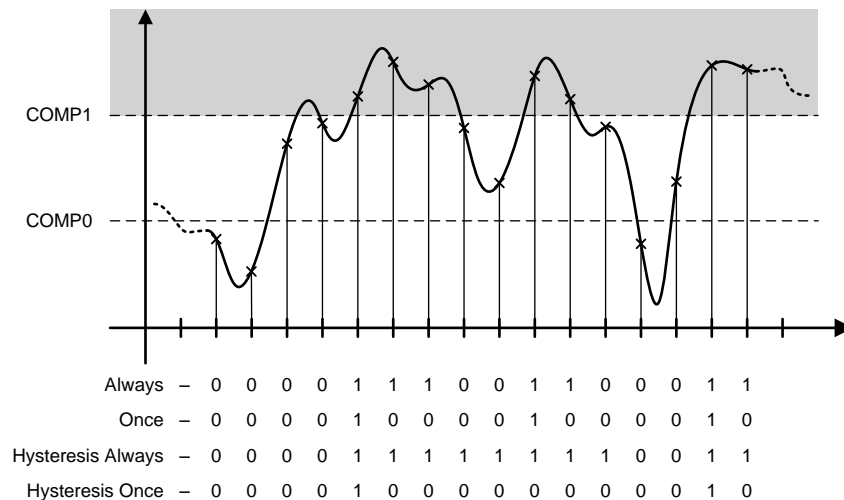


Figure 10-14. High-Band Operation (CIC = 0x3 or CTC = 0x3)

10.4 Initialization and Configuration

10.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock using the RCGCADC register (see [Section 4.2.97](#)).
2. Enable the clock to the appropriate GPIO modules using the RCGCGPIO register (see [Section 4.2.87](#)). To find out which GPIO ports to enable, see the device-specific data sheet.
3. Set the GPIO AFSEL bits for the ADC input pins (see [Section 17.5.10](#)). To determine which GPIOs to configure, see the device-specific data sheet.
4. Configure the AINx signals to be analog inputs by clearing the corresponding DEN bit in the GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)).
5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the GPIOAMSEL register (see [Section 17.5.21](#)) in the associated GPIO block.
6. If required by the application, reconfigure the sample sequencer priorities in the ADCSSPRI register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

10.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by clearing the corresponding ASENn bit in the ADCACTSS register. Programming of the sample sequencers is allowed without having them enabled.

Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.

2. Configure the trigger event for the sample sequencer in the ADCEMUX register.
3. When using a PWM generator as the trigger source, use the ADC Trigger Source Select (ADCTSSEL) register to specify in which PWM module the generator is located. The default register reset selects PWM module 0 for all generators.
4. For each sample in the sample sequence, configure the corresponding input source in the ADCSSMUXn and ADCSSEMUXn registers.
5. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the ADCSSCTLn register. When programming the last nibble, ensure that the END bit is set. Failure to set the END bit causes unpredictable behavior.
6. If interrupts are to be used, set the corresponding MASK bit in the ADCIM register.
7. Enable the sample sequencer logic by setting the corresponding ASENn bit in the ADCACTSS register.

10.5 ADC Registers

[Table 10-6](#) lists the memory-mapped registers for the ADC. All register offset addresses not listed in [Table 10-6](#) should be considered as reserved locations and the register contents should not be modified.

The offset listed is relative to the base address of the ADC module:

- ADC0: 0x40038000
- ADC1: 0x40039000

The ADC module clock must be enabled before the registers can be programmed (see [Section 4.2.97](#)). There must be a delay of 3 system clock cycles after the ADC module clock is enabled before any ADC module registers are accessed.

Table 10-6. ADC Registers

Offset	Acronym	Register Name	Section
0x0	ADCACTSS	ADC Active Sample Sequencer	Section 10.5.1
0x4	ADCRIS	ADC Raw Interrupt Status	Section 10.5.2
0x8	ADCIM	ADC Interrupt Mask	Section 10.5.3
0xC	ADCISC	ADC Interrupt Status and Clear	Section 10.5.4
0x10	ADCOSTAT	ADC Overflow Status	Section 10.5.5
0x14	ADCEMUX	ADC Event Multiplexer Select	Section 10.5.6
0x18	ADCUSTAT	ADC Underflow Status	Section 10.5.7
0x1C	ADCTSSEL	ADC Trigger Source Select	Section 10.5.8
0x20	ADCSSPRI	ADC Sample Sequencer Priority	Section 10.5.9
0x24	ADCSPC	ADC Sample Phase Control	Section 10.5.10
0x28	ADCPSSI	ADC Processor Sample Sequence Initiate	Section 10.5.11
0x30	ADCSAC	ADC Sample Averaging Control	Section 10.5.12
0x34	ADCDCISC	ADC Digital Comparator Interrupt Status and Clear	Section 10.5.13
0x38	ADCCTL	ADC Control	Section 10.5.14
0x40	ADCSSMUX0	ADC Sample Sequence Input Multiplexer Select 0	Section 10.5.15
0x44	ADCSSCTL0	ADC Sample Sequence Control 0	Section 10.5.16
0x48	ADCSSFIFO0	ADC Sample Sequence Result FIFO 0	Section 10.5.17
0x4C	ADCSSFSTAT0	ADC Sample Sequence FIFO 0 Status	Section 10.5.18
0x50	ADCSSOP0	ADC Sample Sequence 0 Operation	Section 10.5.19
0x54	ADCSSDC0	ADC Sample Sequence 0 Digital Comparator Select	Section 10.5.20
0x58	ADCSSEMUX0	ADC Sample Sequence Extended Input Multiplexer Select 0	Section 10.5.21
0x5C	ADCSSTSH0	ADC Sample Sequence 0 Sample and Hold Time	Section 10.5.22
0x60	ADCSSMUX1	ADC Sample Sequence Input Multiplexer Select 1	Section 10.5.23
0x64	ADCSSCTL1	ADC Sample Sequence Control 1	Section 10.5.24
0x68	ADCSSFIFO1	ADC Sample Sequence Result FIFO 1	Section 10.5.17
0x6C	ADCSSFSTAT1	ADC Sample Sequence FIFO 1 Status	Section 10.5.18
0x70	ADCSSOP1	ADC Sample Sequence 1 Operation	Section 10.5.25
0x74	ADCSSDC1	ADC Sample Sequence 1 Digital Comparator Select	Section 10.5.26
0x78	ADCSSEMUX1	ADC Sample Sequence Extended Input Multiplexer Select 1	Section 10.5.27
0x7C	ADCSSTSH1	ADC Sample Sequence 1 Sample and Hold Time	Section 10.5.28
0x80	ADCSSMUX2	ADC Sample Sequence Input Multiplexer Select 2	Section 10.5.23
0x84	ADCSSCTL2	ADC Sample Sequence Control 2	Section 10.5.24
0x88	ADCSSFIFO2	ADC Sample Sequence Result FIFO 2	Section 10.5.17
0x8C	ADCSSFSTAT2	ADC Sample Sequence FIFO 2 Status	Section 10.5.18
0x90	ADCSSOP2	ADC Sample Sequence 2 Operation	Section 10.5.25
0x94	ADCSSDC2	ADC Sample Sequence 2 Digital Comparator Select	Section 10.5.26

Table 10-6. ADC Registers (continued)

Offset	Acronym	Register Name	Section
0x098	ADCSSEMUX2	ADC Sample Sequence Extended Input Multiplexer Select 2	Section 10.5.27
0x09C	ADCSSTSH2	ADC Sample Sequence 2 Sample and Hold Time	Section 10.5.28
0xA0	ADCSSMUX3	ADC Sample Sequence Input Multiplexer Select 3	Section 10.5.29
0xA4	ADCSSCTL3	ADC Sample Sequence Control 3	Section 10.5.30
0x0A8	ADCSSFIFO3	ADC Sample Sequence Result FIFO 3	Section 10.5.17
0x0AC	ADCSSFSTAT3	ADC Sample Sequence FIFO 3 Status	Section 10.5.18
0xB0	ADCSSOP3	ADC Sample Sequence 3 Operation	Section 10.5.31
0xB4	ADCSSDC3	ADC Sample Sequence 3 Digital Comparator Select	Section 10.5.32
0xB8	ADCSSEMUX3	ADC Sample Sequence Extended Input Multiplexer Select 3	Section 10.5.33
0xBC	ADCSSTSH3	ADC Sample Sequence 3 Sample and Hold Time	Section 10.5.34
0xD00	ADCDCRIC	ADC Digital Comparator Reset Initial Conditions	Section 10.5.35
0xE00 to 0xE1C	ADCDCCTL0 to ADCDCCTL7	ADC Digital Comparator Control 0 to ADC Digital Comparator Control 7	Section 10.5.36
0xE40 to 0xE5C	ADCDCCMP0 to ADCDCCMP7	ADC Digital Comparator Range 0 to ADC Digital Comparator Range 7	Section 10.5.37
0xFC0	ADCPP	ADC Peripheral Properties	Section 10.5.38
0xFC4	ADCP	ADC Peripheral Configuration	Section 10.5.39
0xFC8	ADCC	ADC Clock Configuration	Section 10.5.40

Complex bit access types are encoded to fit into small table cells. [Table 10-7](#) shows the codes that are used for access types in this section.

Table 10-7. ADC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

10.5.1 ADCACTSS Register (Offset = 0x0) [reset = 0x0]

ADC Active Sample Sequencer (ADCACTSS)

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADCACTSS is shown in [Figure 10-15](#) and described in [Table 10-8](#).

Return to [Summary Table](#).

Figure 10-15. ADCACTSS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							BUSY
R-0x0							R-0x0
15	14	13	12	11	10	9	8
RESERVED				ADEN3	ADEN2	ADEN1	ADEN0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED				ASEN3	ASEN2	ASEN1	ASEN0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 10-8. ADCACTSS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	BUSY	R	0x0	ADC Busy. To use the BUSY bit, the ADC Event Multiplexer Select (ADCEMUX) register must be programmed such that no trigger is selected (bit field encoding is 0xE). The NEVER encoding in the ADCEMUX register allows the ADC to safely be put in Deep-Sleep mode. 0x0 = ADC is idle 0x1 = ADC is busy
15-12	RESERVED	R	0x0	
11	ADEN3	R/W	0x0	ADC SS3 DMA Enable. 0x0 = DMA for Sample Sequencer 3 is disabled. 0x1 = DMA for Sample Sequencer 3 is enabled.
10	ADEN2	R/W	0x0	ADC SS2 DMA Enable. 0x0 = DMA for Sample Sequencer 2 is disabled. 0x1 = DMA for Sample Sequencer 2 is enabled.
9	ADEN1	R/W	0x0	ADC SS1 DMA Enable. 0x0 = DMA for Sample Sequencer 1 is disabled. 0x1 = DMA for Sample Sequencer 1 is enabled.
8	ADEN0	R/W	0x0	ADC SS1 DMA Enable. 0x0 = DMA for Sample Sequencer 1 is disabled. 0x1 = DMA for Sample Sequencer 1 is enabled.
7-4	RESERVED	R	0x0	
3	ASEN3	R/W	0x0	ADC SS3 Enable. 0x0 = Sample Sequencer 3 is disabled. 0x1 = Sample Sequencer 3 is enabled.
2	ASEN2	R/W	0x0	ADC SS2 Enable. 0x0 = Sample Sequencer 2 is disabled. 0x1 = Sample Sequencer 2 is enabled.

Table 10-8. ADCACTSS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	ASEN1	R/W	0x0	ADC SS1 Enable. 0x0 = Sample Sequencer 1 is disabled. 0x1 = Sample Sequencer 1 is enabled.
0	ASEN0	R/W	0x0	ADC SS0 Enable. 0x0 = Sample Sequencer 0 is disabled. 0x1 = Sample Sequencer 0 is enabled.

10.5.2 ADCRIS Register (Offset = 0x4) [reset = 0x0]

ADC Raw Interrupt Status (ADCRIS)

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

ADCRIS is shown in [Figure 10-16](#) and described in [Table 10-9](#).

Return to [Summary Table](#).

Figure 10-16. ADCRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							INRDC
R-0x0							R-0x0
15	14	13	12	11	10	9	8
RESERVED				DMAINR3	DMAINR2	DMAINR1	DMAINR0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED				INR3	INR2	INR1	INR0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 10-9. ADCRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	INRDC	R	0x0	Digital Comparator Raw Interrupt Status. 0x0 = All bits in the ADCDCISC register are clear. 0x1 = At least one bit in the ADCDCISC register is set, meaning that a digital comparator interrupt has occurred.
15-12	RESERVED	R	0x0	
11	DMAINR3	R	0x0	SS3 DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMAINR3 bit in the ADCISC register. 0x0 = The DMA interrupt has not occurred. 0x1 = The sample sequence 3 DMA interrupt is asserted.
10	DMAINR2	R	0x0	SS2 DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMAINR2 bit in the ADCISC register. 0x0 = The DMA interrupt has not occurred. 0x1 = The sample sequence 2 DMA interrupt is asserted.
9	DMAINR1	R	0x0	SS1 DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMAINR1 bit in the ADCISC register. 0x0 = The DMA interrupt has not occurred. 0x1 = The sample sequence 1 DMA interrupt is asserted.
8	DMAINR0	R	0x0	SS0 DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMAINR0 bit in the ADCISC register. 0x0 = The DMA interrupt has not occurred. 0x1 = The sample sequence 0 DMA interrupt is asserted.
7-4	RESERVED	R	0x0	

Table 10-9. ADCRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	INR3	R	0x0	SS3 Raw Interrupt Status. This bit is cleared by writing a 1 to the IN3 bit in the ADCISC register. 0x0 = An interrupt has not occurred. 0x1 = A sample has completed conversion and the respective ADCSSCTL3 IEn bit is set, enabling a raw interrupt.
2	INR2	R	0x0	SS2 Raw Interrupt Status. This bit is cleared by writing a 1 to the IN2 bit in the ADCISC register. 0x0 = An interrupt has not occurred. 0x1 = A sample has completed conversion and the respective ADCSSCTL2 IEn bit is set, enabling a raw interrupt.
1	INR1	R	0x0	SS1 Raw Interrupt Status. This bit is cleared by writing a 1 to the IN1 bit in the ADCISC register. 0x0 = An interrupt has not occurred. 0x1 = A sample has completed conversion and the respective ADCSSCTL1 IEn bit is set, enabling a raw interrupt.
0	INR0	R	0x0	SS0 Raw Interrupt Status. This bit is cleared by writing a 1 to the IN0 bit in the ADCISC register. 0x0 = An interrupt has not occurred. 0x1 = A sample has completed conversion and the respective ADCSSCTL0 IEn bit is set, enabling a raw interrupt.

10.5.3 ADCIM Register (Offset = 0x8) [reset = 0x0]

ADC Interrupt Mask (ADCIM)

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently.

NOTE: For a 1 to 2 Msps rate, as the system clock frequency approaches the ADC clock frequency, it is recommended that the application use the μ DMA to store conversion data from the FIFO to memory before processing rather than an interrupt-driven single data read. Using the μ DMA to store multiple samples before interrupting the processor amortizes interrupt overhead across multiple transfers and prevents loss of sample data.

NOTE: Only a single DCONSSn bit should be set at any given time. Setting more than one of these bits results in the INRDC bit from the ADCRIS register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines. It is recommended that when interrupts are used, they are enabled on alternating samples or at the end of the sample sequence.

ADCIM is shown in [Figure 10-17](#) and described in [Table 10-10](#).

Return to [Summary Table](#).

Figure 10-17. ADCIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				DCONSS3	DCONSS2	DCONSS1	DCONSS0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED				DMAMASK3	DMAMASK2	DMAMASK1	DMAMASK0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED				MASK3	MASK2	MASK1	MASK0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 10-10. ADCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	DCONSS3	R/W	0x0	Digital Comparator Interrupt on SS3. 0x0 = The status of the digital comparators does not affect the SS3 interrupt status. 0x1 = The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS3 interrupt line.
18	DCONSS2	R/W	0x0	Digital Comparator Interrupt on SS2. 0x0 = The status of the digital comparators does not affect the SS2 interrupt status. 0x1 = The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS2 interrupt line.

Table 10-10. ADCIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	DCONSS1	R/W	0x0	Digital Comparator Interrupt on SS1. 0x0 = The status of the digital comparators does not affect the SS1 interrupt status. 0x1 = The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS1 interrupt line.
16	DCONSS0	R/W	0x0	Digital Comparator Interrupt on SS0. 0x0 = The status of the digital comparators does not affect the SS0 interrupt status. 0x1 = The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS0 interrupt line.
15-12	RESERVED	R	0x0	
11	DMAMASK3	R/W	0x0	SS3 DMA Interrupt Mask. 0x0 = The status of Sample Sequencer 3 DMA does not affect the SS3 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 3 DMA (ADCRIS register DMAINR3 bit) is sent to the interrupt controller.
10	DMAMASK2	R/W	0x0	SS2 DMA Interrupt Mask. 0x0 = The status of Sample Sequencer 2 DMA does not affect the SS2 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 2 DMA (ADCRIS register DMAINR2 bit) is sent to the interrupt controller.
9	DMAMASK1	R/W	0x0	SS1 DMA Interrupt Mask. 0x0 = The status of Sample Sequencer 1 DMA does not affect the SS1 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 1 DMA (ADCRIS register DMAINR1 bit) is sent to the interrupt controller.
8	DMAMASK0	R/W	0x0	SS0 DMA Interrupt Mask. 0x0 = The status of Sample Sequencer 0 DMA does not affect the SS0 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 0 DMA (ADCRIS register DMAINR0 bit) is sent to the interrupt controller.
7-4	RESERVED	R	0x0	
3	MASK3	R/W	0x0	SS3 Interrupt Mask. 0x0 = The status of Sample Sequencer 3 does not affect the SS3 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 3 (ADCRIS register INR3 bit) is sent to the interrupt controller.
2	MASK2	R/W	0x0	SS2 Interrupt Mask. 0x0 = The status of Sample Sequencer 2 does not affect the SS2 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 2 (ADCRIS register INR2 bit) is sent to the interrupt controller.
1	MASK1	R/W	0x0	SS1 Interrupt Mask. 0x0 = The status of Sample Sequencer 1 does not affect the SS1 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 1 (ADCRIS register INR1 bit) is sent to the interrupt controller.
0	MASK0	R/W	0x0	SS0 Interrupt Mask. 0x0 = The status of Sample Sequencer 0 does not affect the SS0 interrupt status. 0x1 = The raw interrupt signal from Sample Sequencer 0 (ADCRIS register INR0 bit) is sent to the interrupt controller.

10.5.4 ADCISC Register (Offset = 0xC) [reset = 0x0]

ADC Interrupt Status and Clear (ADCISC)

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective INR and MASK bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the ADCDCISC register. If software is polling the ADCRIS instead of generating interrupts, the sample sequence INRn bits are still cleared via the ADCISC register, even if the INn bit is not set.

ADCISC is shown in [Figure 10-18](#) and described in [Table 10-11](#).

Return to [Summary Table](#).

Figure 10-18. ADCISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				DCINSS3	DCINSS2	DCINSS1	DCINSS0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED				DMAIN3	DMAIN2	DMAIN1	DMAIN0
R-0x0				R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0
7	6	5	4	3	2	1	0
RESERVED				IN3	IN2	IN1	IN0
R-0x0				R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 10-11. ADCISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	DCINSS3	R	0x0	Digital Comparator Interrupt Status on SS3. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INRDC bit in the ADCRIS register and the DCONSS3 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
18	DCINSS2	R	0x0	Digital Comparator Interrupt Status on SS2. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INRDC bit in the ADCRIS register and the DCONSS2 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
17	DCINSS1	R	0x0	Digital Comparator Interrupt Status on SS1. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INRDC bit in the ADCRIS register and the DCONSS1 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.

Table 10-11. ADCISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
16	DCINSS0	R	0x0	Digital Comparator Interrupt Status on SS0. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INRDC bit in the ADCRIS register and the DCONSS0 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
15-12	RESERVED	R	0x0	
11	DMAIN3	R/W1C	0x0	SS3 DMA Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR3 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the DMAINR3 bit in the ADCRIS register and the DMAMASK3 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
10	DMAIN2	R/W1C	0x0	SS2 DMA Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR2 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the DMAINR2 bit in the ADCRIS register and the DMAMASK2 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
9	DMAIN1	R/W1C	0x0	SS1 DMA Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR1 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the DMAINR1 bit in the ADCRIS register and the DMAMASK1 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
8	DMAIN0	R/W1C	0x0	SS0 DMA Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR0 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the DMAINR0 bit in the ADCRIS register and the DMAMASK0 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
7-4	RESERVED	R	0x0	
3	IN3	R/W1C	0x0	SS3 Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INR3 bit in the ADCRIS register and the MASK3 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
2	IN2	R/W1C	0x0	SS2 Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INR2 bit in the ADCRIS register and the MASK2 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.
1	IN1	R/W1C	0x0	SS1 Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit in the ADCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = Both the INR1 bit in the ADCRIS register and the MASK1 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.

Table 10-11. ADCISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	IN0	R/W1C	0x0	<p>SS0 Interrupt Status and Clear.</p> <p>This bit is cleared by writing a 1.</p> <p>Clearing this bit also clears the INR0 bit in the ADCRIS register.</p> <p>0x0 = No interrupt has occurred or the interrupt is masked.</p> <p>0x1 = Both the INR0 bit in the ADCRIS register and the MASK0 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.</p>

10.5.5 ADCOSTAT Register (Offset = 0x10) [reset = 0x0]

ADC Overflow Status (ADCOSTAT)

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADCOSTAT is shown in [Figure 10-19](#) and described in [Table 10-12](#).

Return to [Summary Table](#).

Figure 10-19. ADCOSTAT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OV3	OV2	OV1	OV0
R-0x0												R/W1 C-0x0	R/W1 C-0x0	R/W1 C-0x0	R/W1 C-0x0

Table 10-12. ADCOSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	OV3	R/W1C	0x0	SS3 FIFO Overflow. This bit is cleared by writing a 1. 0x0 = The FIFO has not overflowed. 0x1 = The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
2	OV2	R/W1C	0x0	SS2 FIFO Overflow. This bit is cleared by writing a 1. 0x0 = The FIFO has not overflowed. 0x1 = The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
1	OV1	R/W1C	0x0	SS1 FIFO Overflow. This bit is cleared by writing a 1. 0x0 = The FIFO has not overflowed. 0x1 = The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
0	OV0	R/W1C	0x0	SS0 FIFO Overflow. This bit is cleared by writing a 1. 0x0 = The FIFO has not overflowed. 0x1 = The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.

10.5.6 ADCEMUX Register (Offset = 0x14) [reset = 0x0]

ADC Event Multiplexer Select (ADCEMUX)

The ADCEMUX selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADCEMUX is shown in [Figure 10-20](#) and described in [Table 10-13](#).

Return to [Summary Table](#).

Figure 10-20. ADCEMUX Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EM3				EM2				EM1				EM0			
R-0x0																R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-13. ADCEMUX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-12	EM3	R/W	0x0	<p>SS3 Trigger Select.</p> <p>This field selects the trigger source for Sample Sequencer 3.</p> <p>0x0 = Processor (default). The trigger is initiated by setting the SSn bit in the ADCPSSI register.</p> <p>0x1 = Analog Comparator 0. This trigger is configured by the Analog Comparator Control 0 (ACCTL0) register ().</p> <p>0x2 = Analog Comparator 1. This trigger is configured by the Analog Comparator Control 1 (ACCTL1) register ().</p> <p>0x3 = Analog Comparator 2. This trigger is configured by the Analog Comparator Control 2 (ACCTL2) register ().</p> <p>0x4 = External (GPIO Pins). This trigger is connected to the GPIO interrupt for the corresponding GPIO (see).</p> <p>0x5 = Timer. In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register ().</p> <p>0x6 = PWM generator 0. The PWM generator 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register ().</p> <p>0x7 = PWM generator 1. The PWM generator 1 trigger can be configured with the PWM1INTEN register ().</p> <p>0x8 = PWM generator 2. The PWM generator 2 trigger can be configured with the PWM2INTEN register ().</p> <p>0x9 = PWM generator 3. The PWM generator 3 trigger can be configured with the PWM3INTEN register ().</p> <p>0xA = Reserved</p> <p>0xC = Reserved</p> <p>0xD = Reserved</p> <p>0xE = Never Trigger (No triggers are allowed to the ADC digital interface)</p> <p>0xF = Always (continuously sample)</p>

Table 10-13. ADCMUX Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-8	EM2	R/W	0x0	<p>SS2 Trigger Select. This field selects the trigger source for Sample Sequencer 2.</p> <p>0x0 = Processor (default). The trigger is initiated by setting the SSn bit in the ADCPSSI register.</p> <p>0x1 = Analog Comparator 0. This trigger is configured by the Analog Comparator Control 0 (ACCTL0) register ().</p> <p>0x2 = Analog Comparator 1. This trigger is configured by the Analog Comparator Control 1 (ACCTL1) register ().</p> <p>0x3 = Analog Comparator 2. This trigger is configured by the Analog Comparator Control 2 (ACCTL2) register ().</p> <p>0x4 = External (GPIO Pins). This trigger is connected to the GPIO interrupt for the corresponding GPIO (see).</p> <p>0x5 = Timer. In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register ().</p> <p>0x6 = PWM generator 0. The PWM generator 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register ().</p> <p>0x7 = PWM generator 1. The PWM generator 1 trigger can be configured with the PWM1INTEN register ().</p> <p>0x8 = PWM generator 2. The PWM generator 2 trigger can be configured with the PWM2INTEN register ().</p> <p>0x9 = PWM generator 3. The PWM generator 3 trigger can be configured with the PWM3INTEN register ().</p> <p>0xA = Reserved</p> <p>0xC = Reserved</p> <p>0xD = Reserved</p> <p>0xE = Never Trigger (No triggers are allowed to the ADC digital interface)</p> <p>0xF = Always (continuously sample)</p>
7-4	EM1	R/W	0x0	<p>SS1 Trigger Select. This field selects the trigger source for Sample Sequencer 1.</p> <p>0x0 = Processor (default). The trigger is initiated by setting the SSn bit in the ADCPSSI register.</p> <p>0x1 = Analog Comparator 0. This trigger is configured by the Analog Comparator Control 0 (ACCTL0) register ().</p> <p>0x2 = Analog Comparator 1. This trigger is configured by the Analog Comparator Control 1 (ACCTL1) register ().</p> <p>0x3 = Analog Comparator 2. This trigger is configured by the Analog Comparator Control 2 (ACCTL2) register ().</p> <p>0x4 = External (GPIO Pins). This trigger is connected to the GPIO interrupt for the corresponding GPIO (see).</p> <p>0x5 = Timer. In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register ().</p> <p>0x6 = PWM generator 0. The PWM generator 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register ().</p> <p>0x7 = PWM generator 1. The PWM generator 1 trigger can be configured with the PWM1INTEN register ().</p> <p>0x8 = PWM generator 2. The PWM generator 2 trigger can be configured with the PWM2INTEN register ().</p> <p>0x9 = PWM generator 3. The PWM generator 3 trigger can be configured with the PWM3INTEN register ().</p> <p>0xA = Reserved</p> <p>0xC = Reserved</p> <p>0xD = Reserved</p> <p>0xE = Never Trigger (No triggers are allowed to the ADC digital interface)</p> <p>0xF = Always (continuously sample)</p>

Table 10-13. ADCEMUX Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	EM0	R/W	0x0	<p>SS0 Trigger Select.</p> <p>This field selects the trigger source for Sample Sequencer 0</p> <p>0x0 = Reserved</p> <p>0x1 = Analog Comparator 0. This trigger is configured by the Analog Comparator Control 0 (ACCTL0) register ().</p> <p>0x2 = Analog Comparator 1. This trigger is configured by the Analog Comparator Control 1 (ACCTL1) register ().</p> <p>0x3 = Analog Comparator 2. This trigger is configured by the Analog Comparator Control 2 (ACCTL2) register ().</p> <p>0x4 = External (GPIO Pins). This trigger is connected to the GPIO interrupt for the corresponding GPIO (see).</p> <p>0x5 = Timer. In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register ().</p> <p>0x6 = PWM generator 0. The PWM generator 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register ().</p> <p>0x7 = PWM generator 1. The PWM generator 1 trigger can be configured with the PWM1INTEN register ().</p> <p>0x8 = PWM generator 2. The PWM generator 2 trigger can be configured with the PWM2INTEN register ().</p> <p>0x9 = PWM generator 3. The PWM generator 3 trigger can be configured with the PWM3INTEN register ().</p> <p>0xA = Reserved</p> <p>0xC = Reserved</p> <p>0xD = Reserved</p> <p>0xE = Never Trigger (No triggers are allowed to the ADC digital interface)</p> <p>0xF = Always (continuously sample)</p>

10.5.7 ADCUSTAT Register (Offset = 0x18) [reset = 0x0]

ADC Underflow Status (ADCUSTAT)

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADCUSTAT is shown in [Figure 10-21](#) and described in [Table 10-14](#).

Return to [Summary Table](#).

Figure 10-21. ADCUSTAT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												UV3	UV2	UV1	UV0
R-0x0												R/W1 C-0x0	R/W1 C-0x0	R/W1 C-0x0	R/W1 C-0x0

Table 10-14. ADCUSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	UV3	R/W1C	0x0	SS3 FIFO Underflow. The valid configurations for this field are shown below. This bit is cleared by writing a 1. 0x0 = The FIFO has not underflowed. 0x1 = The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.
2	UV2	R/W1C	0x0	SS2 FIFO Underflow. The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
1	UV1	R/W1C	0x0	SS1 FIFO Underflow. The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
0	UV0	R/W1C	0x0	SS0 FIFO Underflow. The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.

10.5.8 ADCTSSEL Register (Offset = 0x1C) [reset = 0x0]

ADC Trigger Source Select (ADCTSSEL)

If a PWM Generator n is selected as a trigger source through the EMn bit field in the ADC Event Multiplexer Select (ADCEMUX) register, the ADCTSSEL register is programmed to identify in which PWM module instance the generator creating the trigger is located. The register resets to 0x00000000, which selects PWM module 0 for all generators. Field PS3 selects the PWM module that maps to Generator 3; PS2 selects the PWM module that maps to Generator 2, and so on.

ADCTSSEL is shown in [Figure 10-22](#) and described in [Table 10-15](#).

Return to [Summary Table](#).

Figure 10-22. ADCTSSEL Register

31	30	29	28	27	26	25	24
RESERVED		PS3		RESERVED			
R-0x0		R/W-0x0		R-0x0			
23	22	21	20	19	18	17	16
RESERVED		PS2		RESERVED			
R-0x0		R/W-0x0		R-0x0			
15	14	13	12	11	10	9	8
RESERVED		PS1		RESERVED			
R-0x0		R/W-0x0		R-0x0			
7	6	5	4	3	2	1	0
RESERVED		PS0		RESERVED			
R-0x0		R/W-0x0		R-0x0			

Table 10-15. ADCTSSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0x0	
29-28	PS3	R/W	0x0	Generator 3 PWM Module Trigger Select. This field selects in which PWM module the generator 3 trigger is located. 0x0 = Use Generator 3 (and its trigger) in PWM module 0 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
27-22	RESERVED	R	0x0	
21-20	PS2	R/W	0x0	Generator 2 PWM Module Trigger Select. This field selects in which PWM module the Generator 2 trigger is located. 0x0 = Use Generator 2 (and its trigger) in PWM module 0 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
19-14	RESERVED	R	0x0	
13-12	PS1	R/W	0x0	Generator 1 PWM Module Trigger Select. This field selects in which PWM module the Generator 1 trigger is located. 0x0 = Use Generator 1 (and its trigger) in PWM module 0 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
11-6	RESERVED	R	0x0	

Table 10-15. ADCTSSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	PS0	R/W	0x0	Generator 0 PWM Module Trigger Select. This field selects in which PWM module the Generator 0 trigger is located. 0x0 = Use Generator 0 (and its trigger) in PWM module 0 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
3-0	RESERVED	R	0x0	

10.5.9 ADCSSPRI Register (Offset = 0x20) [reset = 0x3210]

ADC Sample Sequencer Priority (ADCSSPRI)

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADCSSPRI is shown in [Figure 10-23](#) and described in [Table 10-16](#).

Return to [Summary Table](#).

Figure 10-23. ADCSSPRI Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		SS3		RESERVED		SS2	
R-0x0		R/W-0x3		R-0x0		R/W-0x2	
7	6	5	4	3	2	1	0
RESERVED		SS1		RESERVED		SS0	
R-0x0		R/W-0x1		R-0x0		R/W-0x0	

Table 10-16. ADCSSPRI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13-12	SS3	R/W	0x3	SS3 Priority. This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11-10	RESERVED	R	0x0	
9-8	SS2	R/W	0x2	SS2 Priority. This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
7-6	RESERVED	R	0x0	
5-4	SS1	R/W	0x1	SS1 Priority. This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
3-2	RESERVED	R	0x0	
1-0	SS0	R/W	0x0	SS0 Priority. This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.

10.5.10 ADCSPC Register (Offset = 0x24) [reset = 0x0]

ADC Sample Phase Control (ADCSPC)

The ADC Sample Phase Control (ADCSPC) register is used to insert a delay in ADC module sampling. This feature can be used with the SYNCWAIT and GSYNC bit in the ADCPSSI register to provide concurrent sampling of two different signals by two different ADC modules or skewed sampling of two ADC modules to increase the effective sampling rate. For concurrent sampling, the PHASE field of each ADC module must be the same and the sample and hold times (TSHn) for the matching sample steps of each ADC must be the same. For example, both ADC0 and ADC1 would program PHASE = 0x0 in the ADCSPC register and might both have the following configuration for their ADCSSTSH0 register:

- TSH7 = 0x4
- TSH6 = 0x2
- TSH5 = 0x2
- TSH4 = 0x8
- TSH3 = 0x6
- TSH2 = 0x2
- TSH1 = 0x4
- TSH0 = 0x2

For skewed sampling with a consistent phase lag, the TSHn field in the ADCSSTSHn register must be the same for all sample steps of an ADC and for both ADC Modules. The desired lag can be calculated by adding the sample and hold time (TSHn) to the twelve clock conversion time to determine the total number of clocks in a sample period. For example to create a 180.0 degree phase lag, the PHASE of the lagging ADC is calculated as:

$$\text{PHASE} = (\text{TSHn} + 12)/2, \text{ where TSHn is in ADC_Clocks}$$

For situations where a predictable phase lag is not required, sample and hold times (TSHn) of ADC modules can vary.

NOTE: Care should be taken when the PHASE field is non-zero, as the resulting delay in sampling the AINx input may result in undesirable system consequences. The time from ADC trigger to sample is increased and could make the response time longer than anticipated. The added latency could have ramifications in the system design. Designers should carefully consider the impact of this delay.

ADCSPC is shown in [Figure 10-24](#) and described in [Table 10-17](#).

Return to [Summary Table](#).

Figure 10-24. ADCSPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												PHASE			
R-0x0												R/W-0x0			

Table 10-17. ADCSPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	PHASE	R/W	0x0	<p>Phase Lag . This field selects the sample phase lag from the standard sample time.</p> <p>0x0 = The ADC samples are concurrent. 0x1 = The ADC sample lags by 1 ADC clock 0x2 = The ADC sample lags by 2 ADC clocks 0x3 = The ADC sample lags by 3 ADC clocks 0x4 = The ADC sample lags by 4 clocks 0x5 = The ADC sample lags by 5 clocks 0x6 = The ADC sample lags by 6 clocks 0x7 = The ADC sample lags by 7 clocks 0x8 = The ADC sample lags by 8 clocks 0x9 = The ADC sample lags by 9 clocks 0xA = The ADC sample lags by 10 clocks 0xC = The ADC sample lags by 12 clocks 0xD = The ADC sample lags by 13 clocks 0xE = The ADC sample lags by 14 clocks 0xF = The ADC sample lags by 15 clocks</p>

10.5.11 ADCPSSI Register (Offset = 0x28) [reset = X]

ADC Processor Sample Sequence Initiate (ADCPSSI)

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in ADCSSPRI dictate execution order.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The ADCPSSI register for that module should then be written. The appropriate SS bits should be set along with the SYNCWAIT bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its ADCPSSI register should be written with the appropriate SS bits set along with the GSYNC bit. All of the ADC modules then begin concurrent sampling according to their configuration.

ADCPSSI is shown in [Figure 10-25](#) and described in [Table 10-18](#).

Return to [Summary Table](#).

Figure 10-25. ADCPSSI Register

31	30	29	28	27	26	25	24
GSYNC	RESERVED			SYNCWAIT	RESERVED		
R/W-X	R-0x0			R/W-0x0	R-0x0		
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				SS3	SS2	SS1	SS0
R-0x0				W-X	W-X	W-X	W-X

Table 10-18. ADCPSSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GSYNC	R/W	X	Global Synchronize. 0x0 = This bit is cleared once sampling has been initiated. 0x1 = This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SSn bit and the SYNCWAIT bit starts sampling once this bit is written.
30-28	RESERVED	R	0x0	
27	SYNCWAIT	R/W	0x0	Synchronize Wait. 0x0 = Sampling begins when a sample sequence has been initiated. 0x1 = This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.
26-4	RESERVED	R	0x0	
3	SS3	W	X	SS3 Initiate. Only a write by software is valid a read of this register returns no meaningful data. 0x0 = No effect 0x1 = Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.
2	SS2	W	X	SS2 Initiate. Only a write by software is valid a read of this register returns no meaningful data. 0x0 = No effect 0x1 = Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.

Table 10-18. ADCPSSI Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	SS1	W	X	SS1 Initiate. Only a write by software is valid a read of this register returns no meaningful data. 0x0 = No effect 0x1 = Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.
0	SS0	W	X	SS0 Initiate. Only a write by software is valid a read of this register returns no meaningful data. 0x0 = No effect 0x1 = Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.

10.5.12 ADCSAC Register (Offset = 0x30) [reset = 0x0]

ADC Sample Averaging Control (ADCSAC)

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{AVG} consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG=7 provides unpredictable results.

ADCSAC is shown in [Figure 10-26](#) and described in [Table 10-19](#).

Return to [Summary Table](#).

Figure 10-26. ADCSAC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED																															AVG	
R-0x0																															R/W-0x0	

Table 10-19. ADCSAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2-0	AVG	R/W	0x0	<p>Hardware Averaging Control. Specifies the amount of hardware averaging that will be applied to ADC samples. The AVG field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results.</p> <p>0x0 = No hardware oversampling 0x1 = 2x hardware oversampling 0x2 = 4x hardware oversampling 0x3 = 8x hardware oversampling 0x4 = 16x hardware oversampling 0x5 = 32x hardware oversampling 0x6 = 64x hardware oversampling 0x7 = Reserved</p>

10.5.13 ADCDCISC Register (Offset = 0x34) [reset = 0x0]

ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

This register provides status and acknowledgement of digital comparator interrupt s. One bit is provided for each comparator.

ADCDCISC is shown in [Figure 10-27](#) and described in [Table 10-20](#).

Return to [Summary Table](#).

Figure 10-27. ADCDCISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 10-20. ADCDCISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	DCINT7	R/W1C	0x0	Digital Comparator 7 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 7 has generated an interrupt.
6	DCINT6	R/W1C	0x0	Digital Comparator 6 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 6 has generated an interrupt.
5	DCINT5	R/W1C	0x0	Digital Comparator 5 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 5 has generated an interrupt.
4	DCINT4	R/W1C	0x0	Digital Comparator 4 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 4 has generated an interrupt.
3	DCINT3	R/W1C	0x0	Digital Comparator 3 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 3 has generated an interrupt.
2	DCINT2	R/W1C	0x0	Digital Comparator 2 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 2 has generated an interrupt.
1	DCINT1	R/W1C	0x0	Digital Comparator 1 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 1 has generated an interrupt.

Table 10-20. ADCDCISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DCINT0	R/W1C	0x0	Digital Comparator 0 Interrupt Status and Clear. This bit is cleared by writing a 1. 0x0 = No interrupt. 0x1 = Digital Comparator 0 has generated an interrupt.

10.5.14 ADCCTL Register (Offset = 0x38) [reset = 0x0]

ADC Control (ADCCTL)

This register configures the voltage reference. The voltage references for the conversion can be VREFA+ and VREFA- or VDDA and GNDA. Values set in this register apply to all ADC modules, it is not possible to set one module to use internal references and another to use external references.

ADCCTL is shown in [Figure 10-28](#) and described in [Table 10-21](#).

Return to [Summary Table](#).

Figure 10-28. ADCCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							VREF
R-0x0							R/W-0x0

Table 10-21. ADCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	VREF	R/W	0x0	Voltage Reference Select. 0x0 = VDDA and GNDA are the voltage references for all ADC modules. 0x1 = The external VREFA+ and VREFA- inputs are the voltage references for all ADC modules.

10.5.15 ADCSSMUX0 Register (Offset = 0x40) [reset = 0x0]

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSEMUX0)

This register, along with the ADCSSEMUX0 register, defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. If the corresponding EMUXn bit in the ADCSSEMUX0 register is set, the MUXn field in this register selects from AIN[23:16]. When the corresponding EMUXn bit is clear, the MUXn field selects from AIN[15:0]. This register is 32 bits wide and contains information for eight possible samples.

NOTE: Channels AIN[31:24] do not exist on this microcontroller. Configuring MUXn to be 0x8 -0xF when the corresponding EMUXn bit is set results in undefined behavior.

ADCSSEMUX0 is shown in [Figure 10-29](#) and described in [Table 10-22](#).

Return to [Summary Table](#).

Figure 10-29. ADCSSMUX0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7				MUX6				MUX5				MUX4				MUX3				MUX2				MUX1				MUX0			
R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-22. ADCSSMUX0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MUX7	R/W	0x0	8th Sample Input Select. The MUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 0x1 when EMUX7 is clear indicates the input is AIN1. A value of 0x1 when EMUX7 is set indicates the input is AIN17. If differential sampling is enabled (the D7 bit in the ADCSSCTL0 register is set), this field must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
27-24	MUX6	R/W	0x0	7th Sample Input Select. The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23-20	MUX5	R/W	0x0	6th Sample Input Select. The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19-16	MUX4	R/W	0x0	5th Sample Input Select. The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15-12	MUX3	R/W	0x0	4th Sample Input Select. The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11-8	MUX2	R/W	0x0	3rd Sample Input Select. The MUX2 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Table 10-22. ADCSSMUX0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-4	MUX1	R/W	0x0	2nd Sample Input Select. The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3-0	MUX0	R/W	0x0	1st Sample Input Select. The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

10.5.16 ADCSSCTL0 Register (Offset = 0x44) [reset = 0x0]

ADC Sample Sequence Control 0 (ADCSSCTL0)

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, eighth sample, or any sample in between. This register is 32 bits wide and contains information for eight possible samples.

ADCSSCTL0 is shown in [Figure 10-30](#) and described in [Table 10-23](#).

Return to [Summary Table](#).

Figure 10-30. ADCSSCTL0 Register

31	30	29	28	27	26	25	24
TS7	IE7	END7	D7	TS6	IE6	END6	D6
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
TS5	IE5	END5	D5	TS4	IE4	END4	D4
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
TS3	IE3	END3	D3	TS2	IE2	END2	D2
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
TS1	IE1	END1	D1	TS0	IE0	END0	D0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 10-23. ADCSSCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	TS7	R/W	0x0	8th Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the eighth sample of the sample sequence. 0x1 = The temperature sensor is read during the eighth sample of the sample sequence.
30	IE7	R/W	0x0	8th Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
29	END7	R/W	0x0	8th Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The eighth sample is the last sample of the sequence.
28	D7	R/W	0x0	8th Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS7 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
27	TS6	R/W	0x0	7th Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the seventh sample of the sample sequence. 0x1 = The temperature sensor is read during the seventh sample of the sample sequence.

Table 10-23. ADCSSCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	IE6	R/W	0x0	7th Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the seventh sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
25	END6	R/W	0x0	7th Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The seventh sample is the last sample of the sequence.
24	D6	R/W	0x0	7th Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS6 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
23	TS5	R/W	0x0	6th Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the sixth sample of the sample sequence. 0x1 = The temperature sensor is read during the sixth sample of the sample sequence.
22	IE5	R/W	0x0	6th Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the sixth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
21	END5	R/W	0x0	6th Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The sixth sample is the last sample of the sequence.
20	D5	R/W	0x0	6th Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS5 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
19	TS4	R/W	0x0	5th Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the fifth sample of the sample sequence. 0x1 = The temperature sensor is read during the fifth sample of the sample sequence.
18	IE4	R/W	0x0	5th Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the fifth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.

Table 10-23. ADCSSCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	END4	R/W	0x0	5th Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The fifth sample is the last sample of the sequence.
16	D4	R/W	0x0	5th Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS4 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
15	TS3	R/W	0x0	4th Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the fourth sample of the sample sequence. 0x1 = The temperature sensor is read during the fourth sample of the sample sequence.
14	IE3	R/W	0x0	4th Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
13	END3	R/W	0x0	4th Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The fourth sample is the last sample of the sequence.
12	D3	R/W	0x0	4th Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS3 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
11	TS2	R/W	0x0	3rd Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the third sample of the sample sequence. 0x1 = The temperature sensor is read during the third sample of the sample sequence.
10	IE2	R/W	0x0	3rd Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
9	END2	R/W	0x0	3rd Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The third sample is the last sample of the sequence.

Table 10-23. ADCSSCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	D2	R/W	0x0	3rd Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS2 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
7	TS1	R/W	0x0	2nd Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the second sample of the sample sequence. 0x1 = The temperature sensor is read during the second sample of the sample sequence.
6	IE1	R/W	0x0	2nd Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
5	END1	R/W	0x0	2nd Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The second sample is the last sample of the sequence.
4	D1	R/W	0x0	2nd Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS1 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
3	TS0	R/W	0x0	1st Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the first sample of the sample sequence. 0x1 = The temperature sensor is read during the first sample of the sample sequence.
2	IE0	R/W	0x0	1st Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the first sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
1	END0	R/W	0x0	1st Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The first sample is the last sample of the sequence.
0	D0	R/W	0x0	1st Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".

10.5.17 ADCSSFIFO0 to ADCSSFIFO3 Registers [reset = X]

ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

This register contains the conversion results for samples collected with the sample sequencer (the ADCSSFIFO0 register is used for Sample Sequencer 0, ADCSSFIFO1 for Sequencer 1, ADCSSFIFO2 for Sequencer 2, and ADCSSFIFO3 for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the ADCOSTAT and ADCUSTAT registers.

ADCSSFIFO_n is shown in [Figure 10-31](#) and described in [Table 10-24](#).

Return to [Summary Table](#).

Figure 10-31. ADCSSFIFO_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
RESERVED																				DATA																
R-0x0																				R-X																

Table 10-24. ADCSSFIFO_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11-0	DATA	R	X	Conversion Result Data

10.5.18 ADCSSFSTAT0 to ADCSSFSTAT3 Registers [reset = 0x00000100]

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO with the head and tail pointers both pointing to index 0. The ADCSSFSTAT0 register provides status on FIFO0, which has 8 entries; ADCSSFSTAT1 on FIFO1, which has 4 entries; ADCSSFSTAT2 on FIFO2, which has 4 entries; and ADCSSFSTAT3 on FIFO3 which has a single entry.

ADCSSFSTATn is shown in [Figure 10-32](#) and described in [Table 10-25](#).

Return to [Summary Table](#).

Figure 10-32. ADCSSFSTATn Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED			FULL	RESERVED			EMPTY
R-0x0			R-0x0	R-0x0			R-0x1
7	6	5	4	3	2	1	0
HPTR				TPTR			
R-0x0				R-0x0			

Table 10-25. ADCSSFSTATn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12	FULL	R	0x0	FIFO Full 0x0 = The FIFO is not currently full. 0x1 = The FIFO is currently full.
11-9	RESERVED	R	0x0	
8	EMPTY	R	0x1	FIFO Empty 0x0 = The FIFO is not currently empty. 0x1 = The FIFO is currently empty.
7-4	HPTR	R	0x0	FIFO Head Pointer. This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written. Valid values are 0x 0-0x7 for FIFO0 0x 0-0x3 for FIFO1 and FIFO2 and 0x0 for FIFO3.
3-0	TPTR	R	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read. Valid values are 0x 0-0x7 for FIFO0 0x 0-0x3 for FIFO1 and FIFO2 and 0x0 for FIFO3.

10.5.19 ADCSSOP0 Register (Offset = 0x50) [reset = 0x0]

ADC Sample Sequence 0 Operation (ADCSSOP0)

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

ADCSSOP0 is shown in [Figure 10-33](#) and described in [Table 10-26](#).

Return to [Summary Table](#).

Figure 10-33. ADCSSOP0 Register

31	30	29	28	27	26	25	24
RESERVED			S7DCOP	RESERVED			S6DCOP
R-0x0			R/W-0x0	R-0x0			R/W-0x0
23	22	21	20	19	18	17	16
RESERVED			S5DCOP	RESERVED			S4DCOP
R-0x0			R/W-0x0	R-0x0			R/W-0x0
15	14	13	12	11	10	9	8
RESERVED			S3DCOP	RESERVED			S2DCOP
R-0x0			R/W-0x0	R-0x0			R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			S1DCOP	RESERVED			S0DCOP
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 10-26. ADCSSOP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0x0	
28	S7DCOP	R/W	0x0	Sample 7 Digital Comparator Operation 0x0 = The eighth sample is saved in Sample Sequence FIFO0. 0x1 = The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the ADCSSDC0 register, and the value is not written to the FIFO.
27-25	RESERVED	R	0x0	
24	S6DCOP	R/W	0x0	Sample 6 Digital Comparator Operation. Same definition as S7DCOP but used during the seventh sample.
23-21	RESERVED	R	0x0	
20	S5DCOP	R/W	0x0	Sample 5 Digital Comparator Operation. Same definition as S7DCOP but used during the sixth sample.
19-17	RESERVED	R	0x0	
16	S4DCOP	R/W	0x0	Sample 4 Digital Comparator Operation. Same definition as S7DCOP but used during the fifth sample.
15-13	RESERVED	R	0x0	
12	S3DCOP	R/W	0x0	Sample 3 Digital Comparator Operation. Same definition as S7DCOP but used during the fourth sample.
11-9	RESERVED	R	0x0	
8	S2DCOP	R/W	0x0	Sample 2 Digital Comparator Operation. Same definition as S7DCOP but used during the third sample.
7-5	RESERVED	R	0x0	
4	S1DCOP	R/W	0x0	Sample 1 Digital Comparator Operation. Same definition as S7DCOP but used during the second sample.
3-1	RESERVED	R	0x0	
0	S0DCOP	R/W	0x0	Sample 0 Digital Comparator Operation. Same definition as S7DCOP but used during the first sample.

10.5.20 ADCSSDC0 Register (Offset = 0x54) [reset = 0x0]

ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding SnDCOP bit in the ADCSSOP0 register is set.

ADCSSDC0 is shown in [Figure 10-34](#) and described in [Table 10-27](#).

Return to [Summary Table](#).

Figure 10-34. ADCSSDC0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL			
R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-27. ADCSSDC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	S7DCSEL	R/W	0x0	Sample 7 Digital Comparator Select. When the S7DCOP bit in the ADCSSOP0 register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0. Values not listed are reserved. 0x0 = Digital Comparator Unit 0 (ADCDCCMP0 and ADCDCCTL0) 0x1 = Digital Comparator Unit 1 (ADCDCCMP1 and ADCDCCTL1) 0x2 = Digital Comparator Unit 2 (ADCDCCMP2 and ADCDCCTL2) 0x3 = Digital Comparator Unit 3 (ADCDCCMP3 and ADCDCCTL3) 0x4 = Digital Comparator Unit 4 (ADCDCCMP4 and ADCDCCTL4) 0x5 = Digital Comparator Unit 5 (ADCDCCMP5 and ADCDCCTL5) 0x6 = Digital Comparator Unit 6 (ADCDCCMP6 and ADCDCCTL6) 0x7 = Digital Comparator Unit 7 (ADCDCCMP7 and ADCDCCTL7)
27-24	S6DCSEL	R/W	0x0	Sample 6 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the seventh sample.
23-20	S5DCSEL	R/W	0x0	Sample 5 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the sixth sample.
19-16	S4DCSEL	R/W	0x0	Sample 4 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the fifth sample.
15-12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the fourth sample.
11-8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the third sample.
7-4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the second sample.
3-0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select. This field has the same encodings as S7DCSEL but is used during the first sample.

10.5.21 ADCSSEMUX0 Register (Offset = 0x58) [reset = 0x0]

ADC Sample Sequence Extended Input Multiplexer Select 0 (ADCSSEMUX0)

This register, along with the ADCSSMUX0 register, defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. If a bit in this register is set, the corresponding MUXn field in the ADCSSMUX0 register selects from AIN[23:16]. When a bit in this register is clear, the corresponding MUXn field selects from AIN[15:0]. This register is 32 bits wide and contains information for eight possible samples.

This register is not used when the differential channel designation is used (the Dn bit is set in the ADCSSCTL0 register) because the ADCSSMUX0 register can select all the available pairs.

ADCSSEMUX0 is shown in [Figure 10-35](#) and described in [Table 10-28](#).

Return to [Summary Table](#).

Figure 10-35. ADCSSEMUX0 Register

31	30	29	28	27	26	25	24
RESERVED			EMUX7	RESERVED			EMUX6
R-0x0			R/W-0x0	R-0x0			R/W-0x0
23	22	21	20	19	18	17	16
RESERVED			EMUX5	RESERVED			EMUX4
R-0x0			R/W-0x0	R-0x0			R/W-0x0
15	14	13	12	11	10	9	8
RESERVED			EMUX3	RESERVED			EMUX2
R-0x0			R/W-0x0	R-0x0			R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			EMUX1	RESERVED			EMUX0
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 10-28. ADCSSEMUX0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0x0	
28	EMUX7	R/W	0x0	8th Sample Input Select (Upper Bit). The EMUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. 0x0 = The eighth sample input is selected from AIN[15:0] using the ADCSSMUX0 register. For example, if the MUX7 field is 0x0, AIN0 is selected. 0x1 = The eighth sample input is selected from AIN[23:16] using the ADCSSMUX0 register. For example, if the MUX7 field is 0x0, AIN16 is selected.
27-25	RESERVED	R	0x0	
24	EMUX6	R/W	0x0	7th Sample Input Select (Upper Bit). The EMUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.
23-21	RESERVED	R	0x0	
20	EMUX5	R/W	0x0	6th Sample Input Select (Upper Bit). The EMUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.
19-17	RESERVED	R	0x0	
16	EMUX4	R/W	0x0	5th Sample Input Select (Upper Bit). The EMUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.
15-13	RESERVED	R	0x0	

Table 10-28. ADCSSEMUX0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	EMUX3	R/W	0x0	4th Sample Input Select (Upper Bit). The EMUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.
11-9	RESERVED	R	0x0	
8	EMUX2	R/W	0x0	3rd Sample Input Select (Upper Bit). The EMUX2 field is used during the third sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.
7-5	RESERVED	R	0x0	
4	EMUX1	R/W	0x0	2th Sample Input Select (Upper Bit). The EMUX1 field is used during the second sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.
3-1	RESERVED	R	0x0	
0	EMUX0	R/W	0x0	1st Sample Input Select (Upper Bit). The EMUX0 field is used during the first sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7.

10.5.22 ADCSSTSH0 Register (Offset = 0x5C) [reset = 0x0]

ADC Sample Sequence 0 Sample and Hold Time (ADCSSTSH0)

This register controls the sample period size for each sample of sequencer 0. Each sample and hold period select specifies the time allocated to the sample and hold circuit as shown by the encodings in [Table 10-2](#).

NOTE: If sampling the internal temperature sensor, the sample and hold width should be at least 16 ADC clocks (TSHn = 0x4).

Table 10-29. Sample and Hold Width in ADC Clocks

TSHn Encoding	N _{SH}
0x0	4
0x1	Reserved
0x2	8
0x3	Reserved
0x4	16
0x5	Reserved
0x6	32
0x7	Reserved
0x8	64
0x9	Reserved
0xA	128
0xB	Reserved
0xC	256
0xD to 0xF	Reserved

ADCSSTSH0 is shown in [Figure 10-36](#) and described in [Table 10-30](#).

Return to [Summary Table](#).

Figure 10-36. ADCSSTSH0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSH7				TSH6				TSH5				TSH4				TSH3				TSH2				TSH1				TSH0			
R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-30. ADCSSTSH0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	TSH7	R/W	0x0	8th Sample and Hold Period Select. The TSH7 field is used during the eighth sample of a sequence executed with the sample sequencer.
27-24	TSH6	R/W	0x0	7th Sample and Hold Period Select. The TSH6 field is used during the seventh sample of a sequence executed with the sample sequencer.
23-20	TSH5	R/W	0x0	6th Sample and Hold Period Select. The TSH5 field is used during the sixth sample of a sequence executed with the sample sequencer.
19-16	TSH4	R/W	0x0	5th Sample and Hold Period Select. The TSH4 field is used during the fifth sample of a sequence executed with the sample sequencer.
15-12	TSH3	R/W	0x0	4th Sample and Hold Period Select. The TSH3 field is used during the fourth sample of a sequence executed with the sample sequencer.

Table 10-30. ADCSSTSH0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-8	TSH2	R/W	0x0	3rd Sample and Hold Period Select. The TSH2 field is used during the third sample of a sequence executed with the sample sequencer.
7-4	TSH1	R/W	0x0	2nd Sample and Hold Period Select. The TSH1 field is used during the second sample of a sequence executed with the sample sequencer.
3-0	TSH0	R/W	0x0	1st Sample and Hold Period Select. The TSH0 field is used during the first sample of a sequence executed with the sample sequencer.

10.5.23 ADCSSMUX1 and ADCSSMUX2 Registers [reset = 0x0]

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register, along with the ADCSSEMUX1 or ADCSSEMUX2 register, defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. If the corresponding EMUXn bit in the ADCSSEMUX1 or ADCSSEMUX2 register is set, the MUXn field in this register selects from AIN[23:16]. When the corresponding EMUXn bit is clear, the MUXn field selects from AIN[15:0]. These registers are 16 bits wide and contain information for four possible samples. See the ADCSSMUX0 register on [Section 10.5.15](#) for detailed bit descriptions. The ADCSSMUX1 register affects Sample Sequencer 1 and the ADCSSMUX2 register affects Sample Sequencer 2.

NOTE: Channels AIN[31:24] do not exist on this microcontroller. Configuring MUXn to be 0x8 to 0xF when the corresponding EMUXn bit is set results in undefined behavior.

ADCSSMUXn is shown in [Figure 10-37](#) and described in [Table 10-31](#).

Return to [Summary Table](#).

Figure 10-37. ADCSSMUXn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MUX3				MUX2				MUX1				MUX0			
R-0x0																R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-31. ADCSSMUXn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-12	MUX3	R/W	0x0	4th Sample Input Select
11-8	MUX2	R/W	0x0	3rd Sample Input Select
7-4	MUX1	R/W	0x0	2nd Sample Input Select
3-0	MUX0	R/W	0x0	1st Sample Input Select

10.5.24 ADCSSCTL1 and ADCSSCTL2 Registers [reset = 0x0]

ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the ADCSSCTL0 register on [Section 10.5.16](#) for detailed bit descriptions. The ADCSSCTL1 register configures Sample Sequencer 1 and the ADCSSCTL2 register configures Sample Sequencer 2.

ADCSSCTLn is shown in [Figure 10-38](#) and described in [Table 10-32](#).

Return to [Summary Table](#).

Figure 10-38. ADCSSCTLn Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
TS3	IE3	END3	D3	TS2	IE2	END2	D2
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
TS1	IE1	END1	D1	TS0	IE0	END0	D0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 10-32. ADCSSCTLn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	TS3	R/W	0x0	4th Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the fourth sample of the sample sequence. 0x1 = The temperature sensor is read during the fourth sample of the sample sequence.
14	IE3	R/W	0x0	4th Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
13	END3	R/W	0x0	4th Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The fourth sample is the last sample of the sequence.
12	D3	R/W	0x0	4th Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS3 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".

Table 10-32. ADCSSCTLn Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	TS2	R/W	0x0	3rd Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the third sample of the sample sequence. 0x1 = The temperature sensor is read during the third sample of the sample sequence.
10	IE2	R/W	0x0	3rd Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
9	END2	R/W	0x0	3rd Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The third sample is the last sample of the sequence.
8	D2	R/W	0x0	3rd Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS2 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
7	TS1	R/W	0x0	2nd Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the second sample of the sample sequence. 0x1 = The temperature sensor is read during the second sample of the sample sequence.
6	IE1	R/W	0x0	2nd Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
5	END1	R/W	0x0	2nd Sample is End of Sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. 0x0 = Another sample in the sequence is the final sample. 0x1 = The second sample is the last sample of the sequence.
4	D1	R/W	0x0	2nd Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS1 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".
3	TS0	R/W	0x0	1st Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the first sample of the sample sequence. 0x1 = The temperature sensor is read during the first sample of the sample sequence.

Table 10-32. ADCSSCTLn Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	IE0	R/W	0x0	<p>1st Sample Interrupt Enable.</p> <p>It is legal to have multiple samples within a sequence generate interrupts.</p> <p>0x0 = The raw interrupt is not asserted to the interrupt controller.</p> <p>0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of the first sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.</p>
1	END0	R/W	0x0	<p>1st Sample is End of Sequence.</p> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p> <p>0x0 = Another sample in the sequence is the final sample.</p> <p>0x1 = The first sample is the last sample of the sequence.</p>
0	D0	R/W	0x0	<p>1st Sample Differential Input Select Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.</p> <p>0x0 = The analog inputs are not differentially sampled.</p> <p>0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".</p>

10.5.25 ADCSSOP1 and ADCSSOP2 Registers [reset = 0x0]

ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The ADCSSOP1 register controls Sample Sequencer 1 and the ADCSSOP2 register controls Sample Sequencer 2.

ADCSSOP1 is shown in [Figure 10-39](#) and described in [Table 10-33](#).

Return to [Summary Table](#).

Figure 10-39. ADCSSOP1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED			S3DCOP	RESERVED			S2DCOP
R-0x0			R/W-0x0	R-0x0			R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			S1DCOP	RESERVED			S0DCOP
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 10-33. ADCSSOP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12	S3DCOP	R/W	0x0	Sample 3 Digital Comparator Operation. 0x0 = The fourth sample is saved in Sample Sequence FIFO. 0x1 = The fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the ADCSSDC0n register, and the value is not written to the FIFO.
11-9	RESERVED	R	0x0	
8	S2DCOP	R/W	0x0	Sample 2 Digital Comparator Operation. Same definition as S3DCOP but used during the third sample.
7-5	RESERVED	R	0x0	
4	S1DCOP	R/W	0x0	Sample 1 Digital Comparator Operation. Same definition as S3DCOP but used during the second sample.
3-1	RESERVED	R	0x0	
0	S0DCOP	R/W	0x0	Sample 0 Digital Comparator Operation. Same definition as S3DCOP but used during the first sample.

10.5.26 ADCSSDC1 and ADCSSDC2 Registers [reset = 0x0]

ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding SnDCOP bit in the ADCSSOPn register is set. The ADCSSDC1 register controls the selection for Sample Sequencer 1 and the ADCSSDC2 register controls the selection for Sample Sequencer 2.

ADCSSDCn is shown in [Figure 10-40](#) and described in [Table 10-34](#).

Return to [Summary Table](#).

Figure 10-40. ADCSSDCn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-34. ADCSSDCn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select. When the S3DCOP bit in the ADCSSOPn register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer n. Values not listed are reserved. 0x0 = Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0) 0x1 = Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1) 0x2 = Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2) 0x3 = Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3) 0x4 = Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4) 0x5 = Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5) 0x6 = Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6) 0x7 = Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7)
11-8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select. This field has the same encodings as S3DCSEL but is used during the third sample.
7-4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select. This field has the same encodings as S3DCSEL but is used during the second sample.
3-0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select. This field has the same encodings as S3DCSEL but is used during the first sample.

10.5.27 ADCSSEMUX1 and ADCSSEMUX2 Registers [reset = 0x0]

ADC Sample Sequence Extended Input Multiplexer Select 1 (ADCSSEMUX1), offset 0x078

ADC Sample Sequence Extended Input Multiplexer Select 2 (ADCSSEMUX2), offset 0x098

This register, along with the ADCSSMUX1 or ADCSSMUX2 register, defines the analog input configuration for each sample in a sequence executed with either Sample Sequencer 1 or 2. If a bit in this register is set, the corresponding MUXn field in the ADCSSMUX1 or ADCSSMUX2 register selects from AIN[23:16]. When a bit in this register is clear, the corresponding MUXn field selects from AIN[15:0]. This register is 16 bits wide and contains information for four possible samples. The ADCSSEMUX1 register controls Sample Sequencer 1 and the ADCSSEMUX2 register controls Sample Sequencer 2.

This register is not used when the differential channel designation is used (the Dn bit is set in the ADCSSCTL1 or ADCSSCTL2 register) because the ADCSSMUX1 or ADCSSMUX2 register can select all the available pairs.

ADCSSEMUXn is shown in [Figure 10-41](#) and described in [Table 10-35](#).

Return to [Summary Table](#).

Figure 10-41. ADCSSEMUXn Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED			EMUX3	RESERVED			EMUX2
R-0x0			R/W-0x0	R-0x0			R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			EMUX1	RESERVED			EMUX0
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 10-35. ADCSSEMUXn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12	EMUX3	R/W	0x0	4th Sample Input Select (Upper Bit). The EMUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. 0x0 = The fourth sample input is selected from AIN[15:0] using the ADCSSMUX1 or ADCSSMUX2 register. For example, if the MUX3 field is 0x0, AIN0 is selected. 0x1 = The fourth sample input is selected from AIN[23:16] using the ADCSSMUX1 or ADCSSMUX2 register. For example, if the MUX3 field is 0x0, AIN16 is selected.
11-9	RESERVED	R	0x0	
8	EMUX2	R/W	0x0	3rd Sample Input Select (Upper Bit). The EMUX2 field is used during the third sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX3.
7-5	RESERVED	R	0x0	
4	EMUX1	R/W	0x0	2th Sample Input Select (Upper Bit). The EMUX1 field is used during the second sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX3.
3-1	RESERVED	R	0x0	

Table 10-35. ADCSSEMUXn Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	EMUX0	R/W	0x0	1st Sample Input Select (Upper Bit). The EMUX0 field is used during the first sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX3.

10.5.28 ADCSSTSH1 and ADCSSTSH2 Registers [reset = 0x0]

ADC Sample Sequence 1 Sample and Hold Time (ADCSSTSH1), offset 0x07C

ADC Sample Sequence 2 Sample and Hold Time (ADCSSTSH2), offset 0x09C

These registers control the sample period size for each sample step of sequencer 1 and sequencer 2. Each sample and hold period select specifies the time allocated to the sample and hold circuit as shown by the encodings in [Table 10-2](#).

NOTE: If sampling the internal temperature sensor, the sample and hold width should be at least 16 ADC clocks (TSHn = 0x4).

Table 10-36. Sample and Hold Width in ADC Clocks

TSHn Encoding	N _{SH}
0x0	4
0x1	Reserved
0x2	8
0x3	Reserved
0x4	16
0x5	Reserved
0x6	32
0x7	Reserved
0x8	64
0x9	Reserved
0xA	128
0xB	Reserved
0xC	256
0xD-0xF	Reserved

ADCSSTSHn is shown in [Figure 10-42](#) and described in [Table 10-37](#).

Return to [Summary Table](#).

Figure 10-42. ADCSSTSHn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TSH3				TSH2				TSH1				TSH0			
R-0x0																R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			

Table 10-37. ADCSSTSHn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-12	TSH3	R/W	0x0	4th Sample and Hold Period Select. The TSH3 field is used during the fourth sample of a sequence executed with the sample sequencer.
11-8	TSH2	R/W	0x0	3rd Sample and Hold Period Select. The TSH2 field is used during the third sample of a sequence executed with the sample sequencer.
7-4	TSH1	R/W	0x0	2nd Sample and Hold Period Select. The TSH1 field is used during the second sample of a sequence executed with the sample sequencer.
3-0	TSH0	R/W	0x0	1st Sample and Hold Period Select. The TSH0 field is used during the first sample of a sequence executed with the sample sequencer.

10.5.29 ADCSSMUX3 Register (Offset = 0xA0) [reset = 0x0]

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

This register, along with the ADCSSEMUX3 register, defines the analog input configuration for the sample in a sequence executed with Sample Sequencer 3. If the EMUX0 bit in the ADCSSEMUX3 register is set, the MUX0 field in this register selects from AIN[23:16]. When the EMUX0 bit is clear, the MUX0 field selects from AIN[15:0]. This register is four bits wide and contains information for one possible sample. See the ADCSSMUX0 register in [Section 10.5.15](#) for detailed bit descriptions.

ADCSSMUX3 is shown in [Figure 10-43](#) and described in [Table 10-38](#).

Return to [Summary Table](#).

Figure 10-43. ADCSSMUX3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												MUX0			
R-0x0																												R/W-0x0			

Table 10-38. ADCSSMUX3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	MUX0	R/W	0x0	1st Sample Input Select.

10.5.30 ADCSSCTL3 Register (Offset = 0xA4) [reset = 0x0]

ADC Sample Sequence Control 3 (ADCSSCTL3)

This register contains the configuration information for a sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the ADCSSCTL0 register in [Section 10.5.16](#) for detailed bit descriptions.

NOTE: When configuring a sample sequence in this register, the END0 bit must be set.

ADCSSCTL3 is shown in [Figure 10-44](#) and described in [Table 10-39](#).

Return to [Summary Table](#).

Figure 10-44. ADCSSCTL3 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				TS0	IE0	END0	D0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 10-39. ADCSSCTL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	TS0	R/W	0x0	1st Sample Temp Sensor Select. 0x0 = The input pin specified by the ADCSSMUXn register is read during the first sample of the sample sequence. 0x1 = The temperature sensor is read during the first sample of the sample sequence.
2	IE0	R/W	0x0	Sample Interrupt Enable. It is legal to have multiple samples within a sequence generate interrupts. 0x0 = The raw interrupt is not asserted to the interrupt controller. 0x1 = The raw interrupt signal (INR0 bit) is asserted at the end of this sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.
1	END0	R/W	0x0	End of Sequence. This bit must be set before initiating a single sample sequence. 0x0 = Sampling and conversion continues. 0x1 = This is the end of sequence.
0	D0	R/W	0x0	Sample Differential Input Select. Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set. 0x0 = The analog inputs are not differentially sampled. 0x1 = The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".

10.5.31 ADCSSOP3 Register (Offset = 0xB0) [reset = 0x0]

ADC Sample Sequence 3 Operation (ADCSSOP3)

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

ADCSSOP3 is shown in [Figure 10-45](#) and described in [Table 10-40](#).

Return to [Summary Table](#).

Figure 10-45. ADCSSOP3 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							S0DCOP
R-0x0							R/W-0x0

Table 10-40. ADCSSOP3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	S0DCOP	R/W	0x0	Sample 0 Digital Comparator Operation. 0x0 = The sample is saved in Sample Sequence FIFO3. 0x1 = The sample is sent to the digital comparator unit specified by the S0DCSEL bit in the ADCSSDC03 register, and the value is not written to the FIFO.

10.5.32 ADCSSDC3 Register (Offset = 0xB4) [reset = 0x0]

ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding SnDCOP bit in the ADCSSOP3 register is set.

ADCSSDC3 is shown in [Figure 10-46](#) and described in [Table 10-41](#).

Return to [Summary Table](#).

Figure 10-46. ADCSSDC3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												S0DCSEL			
R-0x0												R/W-0x0			

Table 10-41. ADCSSDC3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	S0DCSEL	R/W	0x0	<p>Sample 0 Digital Comparator Select.</p> <p>When the S0DCOP bit in the ADCSSOP3 register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3. Values not listed are reserved.</p> <p>0x0 = Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0)</p> <p>0x1 = Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1)</p> <p>0x2 = Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2)</p> <p>0x3 = Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3)</p> <p>0x4 = Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4)</p> <p>0x5 = Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5)</p> <p>0x6 = Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6)</p> <p>0x7 = Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7)</p>

10.5.33 ADCSSEMUX3 Register (Offset = 0xB8) [reset = 0x0]

ADC Sample Sequence Extended Input Multiplexer Select 3 (ADCSSEMUX3)

This register, along with the ADCSSMUX3 register, defines the analog input configuration for the sample in a sequence executed with Sample Sequencer 3. If EMUX0 is set, the MUX0 field in the ADCSSMUX3 register selects from AIN[23:16]. When EMUX0 is clear, the MUX0 field selects from AIN[15:0]. This register is 1 bit wide and contains information for one possible sample.

This register is not used when the differential channel designation is used (the Dn bit is set in the ADCSSCTL3 register) because the ADCSSMUX3 register can select all the available pairs.

ADCSSEMUX3 is shown in [Figure 10-47](#) and described in [Table 10-42](#).

Return to [Summary Table](#).

Figure 10-47. ADCSSEMUX3 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							EMUX0
R-0x0							R/W-0x0

Table 10-42. ADCSSEMUX3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	EMUX0	R/W	0x0	1st Sample Input Select (Upper Bit). The EMUX0 field is used during the only sample of a sequence executed with the sample sequencer. 0x0 = The sample input is selected from AIN[15:0] using the ADCSSMUX3 register. For example, if the MUX0 field is 0x0, AIN0 is selected. 0x1 = The sample input is selected from AIN[23:16] using the ADCSSMUX3 register. For example, if the MUX0 field is 0x0, AIN16 is selected.

10.5.34 ADCSSTSH3 Register (Offset = 0xBC) [reset = 0x0]

ADC Sample Sequence 3 Sample and Hold Time (ADCSSTSH3)

This register controls the sample period size for the sample in sequencer 3. The sample and hold period select specifies the time allocated to the sample and hold circuit as shown by the encodings in [Table 10-2](#)

NOTE: If sampling the internal temperature sensor, the sample and hold width should be at least 16 ADC clocks (TSHn = 0x4).

Table 10-43. Sample and Hold Width in ADC Clocks

TSHn Encoding	N _{SH}
0x0	4
0x1	Reserved
0x2	8
0x3	Reserved
0x4	16
0x5	Reserved
0x6	32
0x7	Reserved
0x8	64
0x9	Reserved
0xA	128
0xB	Reserved
0xC	256
0xD-0xF	Reserved

ADCSSTSH3 is shown in [Figure 10-48](#) and described in [Table 10-44](#).

Return to [Summary Table](#).

Figure 10-48. ADCSSTSH3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												TSH0			
R-0x0																												R/W-0x0			

Table 10-44. ADCSSTSH3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	TSH0	R/W	0x0	1st Sample and Hold Period Select. The TSH0 field is used during the first sample of a sequence executed with the sample sequencer.

10.5.35 ADCDCRIC Register (Offset = 0xD00) [reset = 0x0]

ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

ADCDCRIC is shown in [Figure 10-49](#) and described in [Table 10-45](#).

Return to [Summary Table](#).

Figure 10-49. ADCDCRIC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0

Table 10-45. ADCDCRIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0x0	
23	DCTRIG7	W	0x0	Digital Comparator Trigger 7. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. After setting this bit, software should wait until the bit clears before continuing. 0x0 = No effect 0x1 = Resets the Digital Comparator 7 trigger unit to its initial conditions.
22	DCTRIG6	W	0x0	Digital Comparator Trigger 6. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 6 trigger unit to its initial conditions.
21	DCTRIG5	W	0x0	Digital Comparator Trigger 5. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 5 trigger unit to its initial conditions.

Table 10-45. ADCDCRIC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
20	DCTRIG4	W	0x0	Digital Comparator Trigger 4. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 4 trigger unit to its initial conditions.
19	DCTRIG3	W	0x0	Digital Comparator Trigger 3. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 3 trigger unit to its initial conditions.
18	DCTRIG2	W	0x0	Digital Comparator Trigger 2. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 2 trigger unit to its initial conditions.
17	DCTRIG1	W	0x0	Digital Comparator Trigger 1. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 1 trigger unit to its initial conditions.
16	DCTRIG0	W	0x0	Digital Comparator Trigger 0. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 0 trigger unit to its initial conditions.
15-8	RESERVED	R	0x0	
7	DCINT7	W	0x0	Digital Comparator Interrupt 7. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 7 interrupt unit to its initial conditions.

Table 10-45. ADCDCRIC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	DCINT6	W	0x0	Digital Comparator Interrupt 6. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 6 interrupt unit to its initial conditions.
5	DCINT5	W	0x0	Digital Comparator Interrupt 5. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 5 interrupt unit to its initial conditions.
4	DCINT4	W	0x0	Digital Comparator Interrupt 4. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 4 interrupt unit to its initial conditions.
3	DCINT3	W	0x0	Digital Comparator Interrupt 3. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 3 interrupt unit to its initial conditions.
2	DCINT2	W	0x0	Digital Comparator Interrupt 2. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 2 interrupt unit to its initial conditions.
1	DCINT1	W	0x0	Digital Comparator Interrupt 1. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. 0x0 = No effect 0x1 = Resets the Digital Comparator 1 interrupt unit to its initial conditions.

Table 10-45. ADCDCRIC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DCINT0	W	0x0	<p>Digital Comparator Interrupt 0. When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> <p>0x0 = No effect</p> <p>0x1 = Resets the Digital Comparator 0 interrupt unit to its initial conditions.</p>

10.5.36 ADCDCCTL0 to ADCDCCTL7 Registers [reset = 0x0]

ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00

ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04

ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08

ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C

ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10

ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14

ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18

ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt or PWM trigger. See [Section 21.3.6](#) for more information on using the ADC digital comparators to trigger a PWM generator.

ADCDCCTLn is shown in [Figure 10-50](#) and described in [Table 10-46](#).

Return to [Summary Table](#).

Figure 10-50. ADCDCCTLn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				CTE		CTC		CTM		RESERVED		CIE	CIC		CIM
R-0x0				R/W-0x0		R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	R/W-0x0		R/W-0x0

Table 10-46. ADCDCCTLn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12	CTE	R/W	0x0	Comparison Trigger Enable. 0x0 = Disables the trigger function state machine. ADC conversion data is ignored by the trigger function. 0x1 = Enables the trigger function state machine. The ADC conversion data is used to determine if a trigger should be generated according to the programming of the CTC and CTM fields.
11-10	CTC	R/W	0x0	Comparison Trigger Condition. This field specifies the operational region in which a trigger is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the ADCDCCMPx registers. 0x0 = Low BandADC Data < COMP0 ≤ COMP1 0x1 = Mid BandCOMP0 < ADC Data ≤ COMP1 0x2 = Reserved 0x3 = High BandCOMP0 ≤ COMP1 ≤ ADC Data

Table 10-46. ADCDCCTLn Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	CTM	R/W	0x0	<p>Comparison Trigger Mode. This field specifies the mode by which the trigger comparison is made.</p> <p>0x0 = AlwaysThis mode generates a trigger every time the ADC conversion data falls within the selected operational region.</p> <p>0x1 = OnceThis mode generates a trigger the first time that the ADC conversion data enters the selected operational region.</p> <p>0x2 = Hysteresis AlwaysThis mode generates a trigger when the ADC conversion data falls within the selected operational region and continues to generate the trigger until the hysteresis condition is cleared by entering the opposite operational region. The hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> <p>0x3 = Hysteresis OnceThis mode generates a trigger the first time that the ADC conversion data falls within the selected operational region. No additional triggers are generated until the hysteresis condition is cleared by entering the opposite operational region. The hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p>
7-5	RESERVED	R	0x0	
4	CIE	R/W	0x0	<p>Comparison Interrupt Enable.</p> <p>0x0 = Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.</p> <p>0x1 = Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the CIC and CIM fields.</p>
3-2	CIC	R/W	0x0	<p>Comparison Interrupt Condition. This field specifies the operational region in which an interrupt is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the ADCDCCMPx registers.</p> <p>0x0 = Low Band. $ADC\ Data < COMP0 \leq COMP1$</p> <p>0x1 = Mid Band. $COMP0 \leq ADC\ Data < COMP1$</p> <p>0x2 = Reserved</p> <p>0x3 = High Band. $COMP0 < COMP1 \leq ADC\ Data$</p>
1-0	CIM	R/W	0x0	<p>Comparison Interrupt Mode. This field specifies the mode by which the interrupt comparison is made.</p> <p>0x0 = Always. This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.</p> <p>0x1 = Once. This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.</p> <p>0x2 = Hysteresis Always. This mode generates an interrupt when the ADC conversion data falls within the selected operational region and continues to generate the interrupt until the hysteresis condition is cleared by entering the opposite operational region. The hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> <p>0x3 = Hysteresis Once. This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region. The hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p>

10.5.37 ADCDCCMP0 to ADCDCCMP7 Registers [reset = 0x0]

ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40

ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44

ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48

ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C

ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50

ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54

ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58

ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region.

NOTE: The value in the COMP1 field must be greater than or equal to the value in the COMP0 field or unexpected results can occur.

ADCDCCMPn is shown in [Figure 10-51](#) and described in [Table 10-47](#).

Return to [Summary Table](#).

Figure 10-51. ADCDCCMPn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								COMP1							
R-0x0								R/W-0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								COMP0							
R-0x0								R/W-0x0							

Table 10-47. ADCDCCMPn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0x0	
27-16	COMP1	R/W	0x0	Compare 1. The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region. The value of COMP1 must be greater than or equal to the value of COMP0.
15-12	RESERVED	R	0x0	
11-0	COMP0	R/W	0x0	Compare 0. The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region.

10.5.38 ADCPP Register (Offset = 0xFC0) [reset = 0x01B02187]

ADC Peripheral Properties (ADCPP)

The ADCPP register provides information regarding the properties of the ADC module.

ADCPP is shown in [Figure 10-52](#) and described in [Table 10-48](#).

Return to [Summary Table](#).

Figure 10-52. ADCPP Register

31	30	29	28	27	26	25	24
RESERVED							APSHT
R-0x0							R-0x1
23	22	21	20	19	18	17	16
TS	RSL				TYPE		
R-0x1	R-0xC				R-0x0		
15	14	13	12	11	10	9	8
DC						CH	
R-0x8						R-0x18	
7	6	5	4	3	2	1	0
CH				MCR			
R-0x18				R-0x7			

Table 10-48. ADCPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0x0	
24	APSHT	R	0x1	Application-Programmable Sample-and-Hold Time. This bit indicates the ADC has the capability of allowing the application to adjust the sample and hold window period.
23	TS	R	0x1	Temperature Sensor. This field provides the similar information as the legacy DC1 register TEMPSNS bit. 0x0 = The ADC module does not have a temperature sensor. 0x1 = The ADC module has a temperature sensor.
22-18	RSL	R	0xC	Resolution. This field specifies the maximum number of binary bits used to represent the converted sample. The field is encoded as a binary value, in the range of 0 to 32 bits.
17-16	TYPE	R	0x0	ADC Architecture. 0x0 = SAR 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
15-10	DC	R	0x8	Digital Comparator Count. This field specifies the number of ADC digital comparators available to the converter. The field is encoded as a binary value, in the range of 0 to 63. This field provides similar information to the legacy DC9 register ADCnDCn bits.
9-4	CH	R	0x18	ADC Channel Count. This field specifies the number of ADC input channels available to the converter. This field is encoded as a binary value, in the range of 0 to 63. This field provides similar information to the legacy DC3 and DC8 register ADCnAlNn bits.

Table 10-48. ADCPP Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	MCR	R	0x7	<p>Maximum Conversion Rate. This field specifies the maximum value that may be programmed into the ADCPC register's CR field.</p> <p>0x0 = Reserved 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Full conversion rate (FCONV) as defined by TADC and NSH. 0x8 = Reserved 0x9 = Reserved 0xA = Reserved 0xC = Reserved 0xD = Reserved 0xE = Reserved 0xF = Reserved</p>

10.5.39 ADCPC Register (Offset = 0xFC4) [reset = 0x7]

ADC Peripheral Configuration (ADCPC)

The ADCPC register provides information regarding the configuration of the peripheral.

ADCPC is shown in [Figure 10-53](#) and described in [Table 10-49](#).

Return to [Summary Table](#).

Figure 10-53. ADCPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												MCR			
R-0x0																												R/W-0x7			

Table 10-49. ADCPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	MCR	R/W	0x7	<p>Conversion Rate.</p> <p>This field specifies the relative sample rate of the ADC and is used in run, sleep, and deep-sleep modes.</p> <p>It allows the application to reduce the rate at which conversions are generated relative to the maximum conversion rate.</p> <p>0x0 = Reserved</p> <p>0x1 = Eighth conversion rate. After a conversion completes, the logic pauses for 112 TADC periods before starting the next conversion.</p> <p>0x2 = Reserved</p> <p>0x3 = Quarter conversion rate. After a conversion completes, the logic pauses for 48 TADC periods before starting the next conversion.</p> <p>0x4 = Reserved</p> <p>0x5 = Half conversion rate. After a conversion completes, the logic pauses for 16 TADC periods before starting the next conversion.</p> <p>0x6 = Reserved</p> <p>0x7 = Full conversion rate (FCONV) as defined by TADC and NSH.</p> <p>0x8 = Reserved</p> <p>0x9 = Reserved</p> <p>0xA = Reserved</p> <p>0xC = Reserved</p> <p>0xD = Reserved</p> <p>0xE = Reserved</p> <p>0xF = Reserved</p>

10.5.40 ADCCC Register (Offset = 0xFC8) [reset = 0x1]

ADC Clock Configuration (ADCCC)

The ADCCC register controls the clock source for the ADC module.

ADCCC is shown in [Figure 10-54](#) and described in [Table 10-50](#).

Return to [Summary Table](#).

Figure 10-54. ADCCC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												CLKDIV				CS															
R-0x0												R/W-0x0				R/W-0x1															

Table 10-50. ADCCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9-4	CLKDIV	R/W	0x0	PLL VCO Clock Divisor 0x0 = /1 $Nh = /(N + 1)$ 0x1 = /2 0x2 = /3
3-0	CS	R/W	0x1	ADC Clock Source 0x0 = PLL VCO divided by CLKDIV 0x1 = Alternate clock source as defined by ALTCLKCFG register in System Control Module. 0x2 = MOSC 0x3 = Reserved 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved 0x8 = Reserved 0x9 = Reserved 0xA = Reserved 0xC = Reserved 0xD = Reserved 0xE = Reserved 0xF = Reserved

Controller Area Network (CAN) Module

This chapter describes the CAN module.

Topic	Page
11.1 Introduction	787
11.2 Block Diagram.....	787
11.3 Functional Description	788
11.4 CAN Registers.....	803

11.1 Introduction

Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can use a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally created for automotive purposes, CAN is also used in many embedded control applications (such as industrial and medical). Bit rates up to 1 Mbps are possible at network lengths less than 40 meters. Decreased bit rates allow longer network distances (for example, 125 kbps at 500 meters).

Two CAN units support the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CANnTX and CANnRX signals

11.2 Block Diagram

Figure 11-1 shows the CAN Controller block diagram.

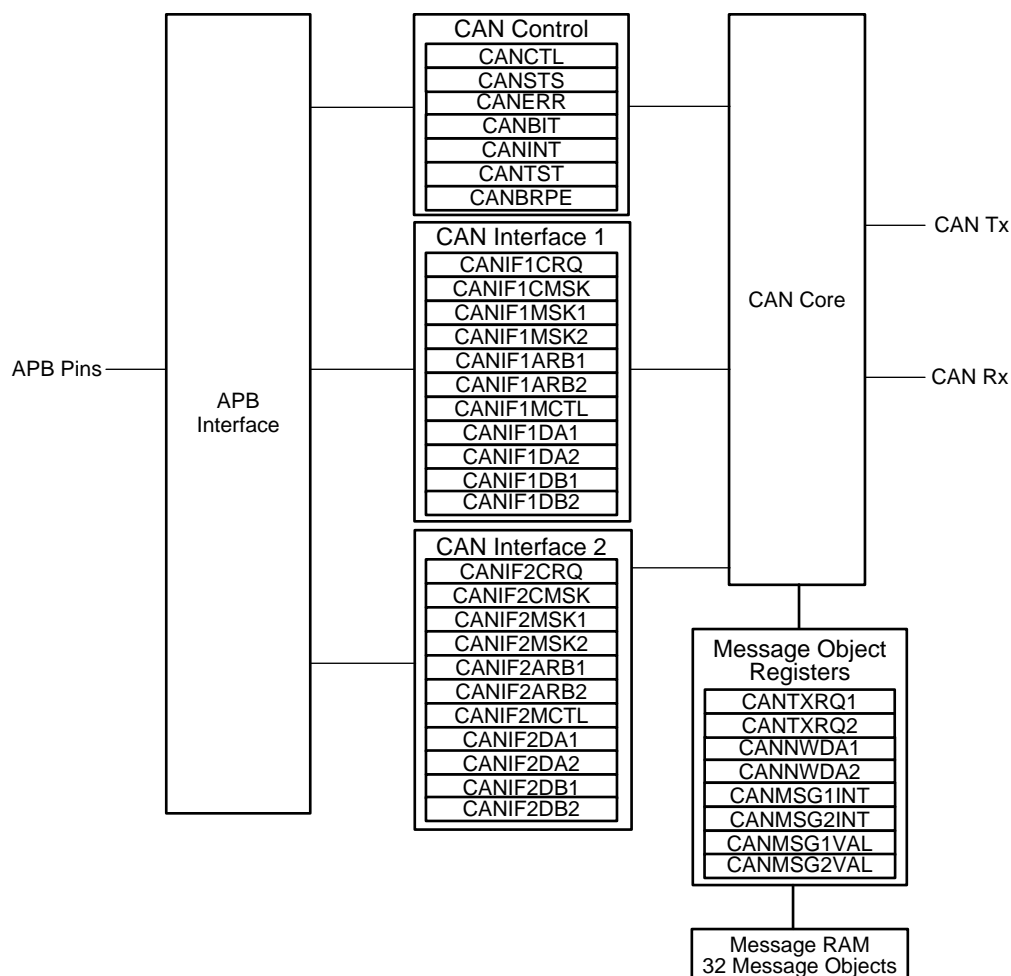


Figure 11-1. CAN Controller Block Diagram

11.3 Functional Description

The CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

A data frame contains data for transmission, whereas a remote frame contains no data and is used to request the transmission of a specific message object. The CAN data frame or remote frame is constructed as shown in [Figure 11-2](#).

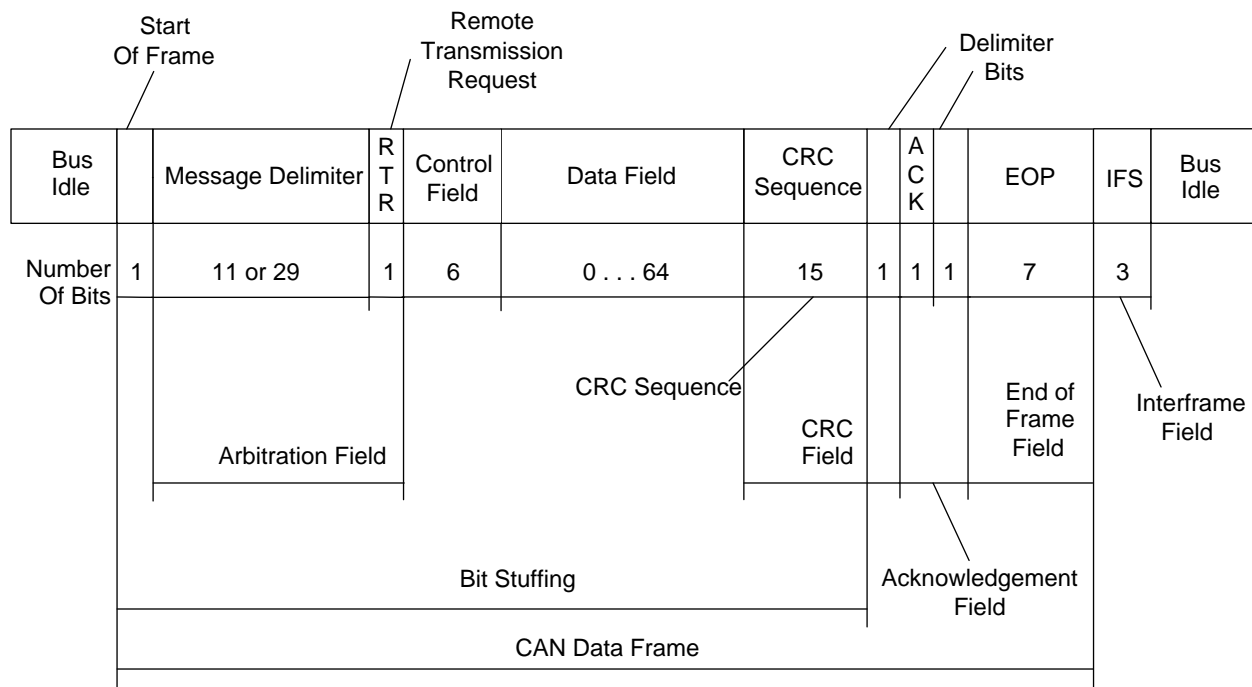


Figure 11-2. CAN Data Frame or Remote Frame

The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These memory blocks are accessed using either of the CAN message object register interfaces.

The message memory is not directly accessible in the MSP432E4 memory map, so the CAN controller provides an interface to communicate with the message memory using two CAN interface register sets for communicating with the message objects. These two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that must be processed. In general, one interface is used for transmit data and one for receive data.

11.3.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the RCGCCAN register (see [Section 4.2.96](#)). In addition, the clock to the appropriate GPIO module must be enabled using the RCGCGPIO register (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet. Set the GPIO AFSEL bits for the appropriate pins (see [Section 17.5.10](#)). Configure the PMcN fields in the GPIOPCTL register to assign the CAN signals to the appropriate pins. See [Section 17.5.22](#) and the device-specific data sheet.

Software initialization is started by setting the INIT bit in the CAN Control (CANCTL) register (with software or by a hardware reset) or by going bus-off, which occurs when the error counter of the transmitter exceeds a count of 255. While INIT is set, all message transfers to and from the CAN bus are stopped and the CANnTX signal is held High. Entering the initialization state does not change the configuration of the CAN controller, the message objects, or the error counters. However, some configuration registers are only accessible while in the initialization state.

To initialize the CAN controller, set the CAN Bit Timing (CANBIT) register and configure each message object. If a message object is not needed, label it as not valid by clearing the MSGVAL bit in the CAN IFn Arbitration 2 (CANIFnARB2) register. Otherwise, the whole message object must be initialized, as the fields of the message object may not have valid information, causing unexpected results. Both the INIT and CCE bits in the CANCTL register must be set in order to access the CANBIT register and the CAN Baud Rate Prescaler Extension (CANBRPE) register to configure the bit timing. To leave the initialization state, the INIT bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (indicating a bus idle condition) before it takes part in bus activities and starts message transfers. Message object initialization does not require the CAN to be in the initialization state and can be done on the fly. However, message objects should all be configured to particular identifiers or set to not valid before message transfer starts. To change the configuration of a message object during normal operation, clear the MSGVAL bit in the CANIFnARB2 register to indicate that the message object is not valid during the change. When the configuration is completed, set the MSGVAL bit again to indicate that the message object is once again valid.

11.3.2 Operation

Two sets of CAN Interface registers (CANIF1x and CANIF2x) are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

After the CAN module is initialized and the INIT bit in the CANCTL register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the filtering process of the message handler, and if it passes through the filter, is stored in the message object specified by the MNUM bit in the CAN IFn Command Request (CANIFnCRQ) register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the MSK bits in the CAN IFn Mask 1 and CAN IFn Mask 2 (CANIFnMSKn) registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time through the CAN Interface registers. The message handler ensures data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. Message objects can be used for one-time data transfers or can be permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate TXRQST bit in the CAN Transmission Request n (CANTXRQn) register and the NEWDAT bit in the CAN New Data n (CANNWDAn) register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object must be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (MNUM) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the RMTEN bit in the CAN IFn Message Control (CANIFnMCTL) register. A matching received remote frame causes the TXRQST bit to be set, and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. A remote frame can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, CANIFnMSKn, configure which groups of frames are identified as remote frame requests. The UMASK bit in the CANIFnMCTL register enables the MSK bits in the CANIFnMSKn register to filter which frames are identified as a remote frame request. The MXTD bit in the CANIFnMSK2 register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

11.3.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if a data transfer is not occurring between the CAN interface registers and message RAM, the valid message object with the highest priority that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's NEWDAT bit in the CANNWDAn register is cleared. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the TXRQST bit in the CANTXRQn register is cleared. If the CAN controller is configured to interrupt on a successful transmission of a message object, (the TXIE bit in the CAN IFn Message Control (CANIFnMCTL) register is set), the INTPND bit in the CANIFnMCTL register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is retransmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

11.3.4 Configuring a Transmit Message Object

The following steps explain how to configure a transmit message object.

1. In the CAN IFn Command Mask (CANIFnCMSK) register:
 - a. Set the WRNRD bit to specify a write to the CANIFnCMSK register; specify whether to transfer the IDMASK, DIR, and MXTD of the message object into the CAN IFn registers using the MASK bit.
 - b. Specify whether to transfer the ID, DIR, XTD, and MSGVAL of the message object into the interface registers using the ARB bit.
 - c. Specify whether to transfer the control bits from the CANIFnMCTL (CAN message control) register into the interface registers using the CONTROL bit.
 - d. Specify whether to clear the INTPND bit (which indicates if the given message is the source of an interrupt) in the CANIFnMCTL register using the CLRINTPND bit.
 - e. Specify whether to clear the NEWDAT bit in the CANNWDAn register (which indicates if new data is available for each of 32 message objects) using the NEWDAT bit.
 - f. Specify which bytes to transfer using the DATAA and DATAB bits. The DATAA bit in CANIFnCMSK enables transfer of data bytes 0 to 3 to or from the message object. Similarly, the DATAB bit controls data bytes 4 to 7.
2. In the CANIFnMSK1 register, use the MSK[15:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. The bits MSK[15:0] in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. For these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the CANIFnMCTL register.
3. In the CANIFnMSK2 register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message

identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the CANIFnMCTL register.

4. For a 29-bit identifier:
 - a. Configure ID[15:0] in the CANIFnARB1 register for bits [15:0] of the message identifier.
 - b. Configure ID[12:0] in the CANIFnARB2 register for bits [28:16] of the message identifier.
 - c. Set the XTD bit to indicate an extended identifier.
 - d. Set the DIR bit to indicate transmit.
 - e. Set the MSGVAL bit to indicate that the message object is valid.
5. For an 11-bit identifier:
 - a. Disregard the CANIFnARB1 register.
 - b. Configure ID[12:2] in the CANIFnARB2 register for bits [10:0] of the message identifier.
 - c. Clear the XTD bit to indicate a standard identifier.
 - d. Set the DIR bit to indicate transmit.
 - e. Set the MSGVAL bit to indicate that the message object is valid.
6. In the CANIFnMCTL register:
 - a. Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the CANIFnMSK1 and CANIFnMSK2 registers) for acceptance filtering.
 - b. Optionally set the TXIE bit to enable the INTPND bit to be set after a successful transmission.
 - c. Optionally set the RMTEN bit to enable the TXRQST bit to be set on the reception of a matching remote frame allowing automatic transmission.
 - d. Set the EOB bit for a single message object.
 - e. Configure the DLC[3:0] field to specify the size of the data frame. Take care during this configuration not to set the NEWDAT, MSGLST, INTPND, or TXRQST bits.
7. Load the data to be transmitted into the CAN IFn Data (CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2) registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the CANIFnDA1 register.
8. Program the number of the message object to be transmitted in the MNUM field in the CAN IFn Command Request (CANIFnCRQ) register.
9. When everything is properly configured, set the TXRQST bit in the CANIFnMCTL register. When this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Setting the RMTEN bit in the CANIFnMCTL register can also start message transmission if a matching remote frame has been received.

11.3.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time using the CAN Interface Registers and neither the MSGVAL bit in the CANIFnARB2 register nor the TXRQST bits in the CANIFnMCTL register need to be cleared before the update.

Even if only some of the data bytes are to be updated, all four bytes of the corresponding CANIFnDAn / CANIFnDBn register must be valid before the content of that register is transferred to the message object. Either the CPU must write all four bytes into the CANIFnDAn / CANIFnDBn register or the message object is transferred to the CANIFnDAn / CANIFnDBn register before the CPU writes the new data bytes.

To only update the data in a message object, the WRNRD, DATAA, and DATAB bits in the CANIFnMSKn register are set, followed by writing the updated data into CANIFnDA1, CANIFnDA2, CANIFnDB1, and CANIFnDB2 registers, and then the number of the message object is written to the MNUM field in the CAN IFn Command Request (CANIFnCRQ) register. To begin transmission of the new data as soon as possible, set the TXRQST bit in the CANIFnMSKn register.

To prevent the clearing of the TXRQST bit in the CANIFnMCTL register at the end of a transmission that may already be in progress while the data is updated, the NEWDAT and TXRQST bits must be set at the same time in the CANIFnMCTL register. When these bits are set at the same time, NEWDAT is cleared as soon as the new transmission has started.

11.3.6 Accepting Received Message Objects

When the arbitration and control field (the ID and XTD bits in the CANIFnARB2 and the RMTEN and DLC[3:0] bits of the CANIFnMCTL register) of an incoming message is completely shifted into the CAN controller, the message handling capability of the controller starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the controller uses the acceptance filtering programmed through the mask bits in the CANIFnMSKn register and enabled using the UMASK bit in the CANIFnMCTL register. Each valid message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

11.3.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLC bits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The NEWDAT bit of the CANIFnMCTL register is set to indicate that new data has been received. The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages. If the CAN controller receives a message and the NEWDAT bit is already set, the MSGLSST bit in the CANIFnMCTL register is set to indicate that the previous data was lost. If the system requires an interrupt on successful reception of a frame, the RXIE bit of the CANIFnMCTL register should be set. In this case, the INTPND bit of the same register is set, causing the CANINT register to point to the message object that just received a message. The TXRQST bit of this message object should be cleared to prevent the transmission of a remote frame.

11.3.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object must be considered (see [Table 11-1](#)):

Table 11-1. Message Object Configurations

Configuration in CANIFnMCTL	Description
<ul style="list-style-type: none"> DIR = 1 (direction = transmit); programmed in the CANIFnARB2 register RMTEN = 1 (set the TXRQST bit of the CANIFnMCTL register at reception of the frame to enable transmission) UMASK = 1 or 0 	At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.
<ul style="list-style-type: none"> DIR = 1 (direction = transmit); programmed in the CANIFnARB2 register RMTEN = 0 (do not change the TXRQST bit of the CANIFnMCTL register at reception of the frame) UMASK = 0 (ignore mask in the CANIFnMSKn register) 	At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and nothing indicates that the remote frame ever happened.
<ul style="list-style-type: none"> DIR = 1 (direction = transmit); programmed in the CANIFnARB2 register RMTEN = 0 (do not change the TXRQST bit of the CANIFnMCTL register at reception of the frame) UMASK = 1 (use mask (MSK, MXTD, and MDIR in the CANIFnMSKn register) for acceptance filtering) 	At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. The arbitration and control field (ID + XTD + RMTEN + DLC) from the shift register is stored into the message object in the message RAM, and the NEWDAT bit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame. This mode is useful for a remote data request from another CAN device for which the controller does not have readily available data. The software must fill the data and answer the frame manually.

11.3.9 Receive and Transmit Priority

The receive and transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This prioritization is separate from that of the message identifier which is enforced by the CAN bus. As a result, if message object 1 and message object 2 both have valid messages to be transmitted, message object 1 is always transmitted first regardless of the message identifier in the message object itself.

11.3.10 Configuring a Receive Message Object

The following steps show how to configure a receive message object.

1. Program the CAN IFn Command Mask (CANIFnCMSK) register as described in the [Section 11.3.4](#) section, except that the WRNRD bit is set to specify a write to the message RAM.
2. Program the CANIFnMSK1 and CANIFnMSK2 registers as described in the [Section 11.3.4](#) section to configure which bits are used for acceptance filtering. Note that for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the CANIFnMCTL register.
3. In the CANIFnMSK2 register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. For these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the CANIFnMCTL register.
4. Program the CANIFnARB1 and CANIFnARB2 registers as described in [Section 11.3.4](#) to program XTD and ID bits for the message identifier to be received; set the MSGVAL bit to indicate a valid message; and clear the DIR bit to specify receive.
5. In the CANIFnMCTL register:
 - a. Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the CANIFnMSK1 and CANIFnMSK2 registers) for acceptance filtering.
 - b. Optionally set the RXIE bit to enable the INTPND bit to be set after a successful reception.
 - c. Clear the RMTEN bit to leave the TXRQST bit unchanged.
 - d. Set the EOB bit for a single message object.
 - e. Configure the DLC[3:0] field to specify the size of the data frame.

Take care during this configuration not to set the NEWDAT, MSGLST, INTPND, or TXRQST bits.

6. Program the number of the message object to be received in the MNUM field in the CAN IFn Command Request (CANIFnCRQ) register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received data length code and eight data bytes in the CANIFnDA1, CANIFnDA2, CANIFnDB1, and CANIFnDB2 registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the CANIFnDA1 register. If the data length code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, CANIFnMSKn, configure which groups of frames are received by a message object. The UMASK bit in the CANIFnMCTL register enables the MSK bits in the CANIFnMSKn register to filter which frames are received. The MXTD bit in the CANIFnMSK2 register should be set if only 29-bit extended identifiers are expected by this message object.

11.3.11 Handling of Received Message Objects

The CPU may read a received message any time through the CAN interface registers, because the data consistency is ensured by the message handler state machine.

Typically, the CPU first writes 0x007F to the CANIFnCMSK register and then writes the number of the message object to the CANIFnCRQ register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (CANIFnMSKn, CANIFnARBn, and CANIFnMCTL). Additionally, the NEWDAT and INTPND bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt generated by this message object.

If the message object uses masks for acceptance filtering, the CANIFnARBn registers show the full, unmasked ID for the received message.

The NEWDAT bit in the CANIFnMCTL register shows whether a new message has been received since the last time this message object was read. The MSGLST bit in the CANIFnMCTL register shows whether more than one message has been received since the last time this message object was read. MSGLST is not automatically cleared, and should be cleared by software after reading its status.

Using a remote frame, the CPU may request new data from another CAN node on the CAN bus. Setting the TXRQST bit of a receive object causes the transmission of a remote frame with the identifier of the receive object. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TXRQST bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data slightly earlier than expected.

11.3.11.1 Configuration of a FIFO Buffer

With the exception of the EOB bit in the CANIFnMCTL register, the configuration of receive message objects belonging to a FIFO buffer is the same as the configuration of a single receive message object (see [Section 11.3.10](#)). To concatenate two or more message objects into a FIFO buffer, the identifiers and masks (if used) of these message objects must be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest message object number is the first message object in a FIFO buffer. The EOB bit of all message objects of a FIFO buffer except the last one must be cleared. The EOB bit of the last message object of a FIFO buffer is set, indicating it is the last entry in the buffer.

11.3.11.2 Reception of Messages With FIFO Buffers

Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NEWDAT of the CANIFnMCTL register bit of this message object is set. By setting NEWDAT while EOB is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. Until all of the preceding message objects have been released by clearing the NEWDAT bit, all further messages for this FIFO buffer are written into the last message object of the FIFO buffer and therefore overwrite previous messages.

11.3.11.3 Reading From a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the CANIFnCRQ register, the TXRQST and CLRINTPND bits in the CANIFnCMSK register should be set such that the NEWDAT and INTPEND bits in the CANIFnMCTL register are cleared after the read. The values of these bits in the CANIFnMCTL register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. When reading from the FIFO buffer, the user should be aware that a new received message is placed in the message object with the lowest message number for which the NEWDAT bit of the CANIFnMCTL register is clear. As a result, the order of the received messages in the FIFO is not ensured. [Figure 11-3](#) shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

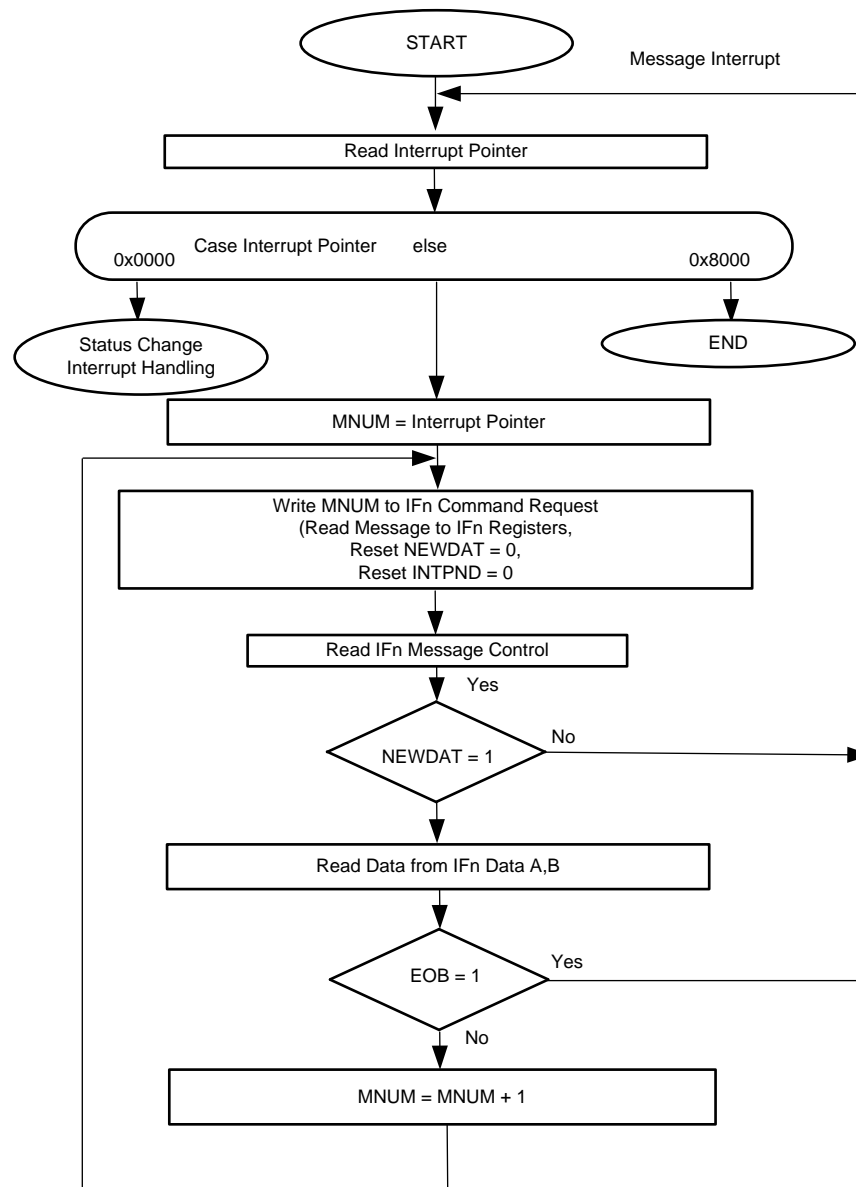


Figure 11-3. Message Objects in a FIFO Buffer

11.3.12 Handling of Interrupts

If several interrupts are pending, the CAN Interrupt (CANINT) register points to the pending interrupt with the highest priority, disregarding their chronological order. The status interrupt has the highest priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's INTPND bit in the CANIFnMCTL register or by reading the CAN Status (CANSTS) register. The status Interrupt is cleared by reading the CANSTS register.

The interrupt identifier INTID in the CANINT register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the INTID field is different from 0, then an interrupt is pending. If the IE bit is set in the CANCTL register, the interrupt line to the interrupt controller is active. The interrupt line remains active until the INTID field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until IE is cleared, which disables interrupts from the CAN controller.

The INTID field of the CANINT register points to the pending message interrupt with the highest interrupt priority. The SIE bit in the CANCTL register controls whether a change of the RXOK, TXOK, and LEC bits in the CANSTS register can cause an interrupt. The EIE bit in the CANCTL register controls whether a change of the BOFF and EWARN bits in the CANSTS register can cause an interrupt. The IE bit in the CANCTL register controls whether any interrupt from the CAN controller actually generates an interrupt to the interrupt controller. The CANINT register is updated even when the IE bit in the CANCTL register is clear, but the interrupt is not indicated to the CPU.

A value of 0x8000 (INTID = 0x8000, which means the interrupt identifier is a status interrupt) in the CANINT register indicates that an interrupt is pending because the CAN module has updated (but not necessarily changed) the CANSTS register, indicating that either an error or status interrupt has been generated. A write access to the CANSTS register can clear the RXOK, TXOK, and LEC bits in that same register; however, the only way to clear the source of a status interrupt is to read the CANSTS register.

The source of an interrupt can be determined in two ways during interrupt handling. The first is to read the INTID bit in the CANINT register to determine the highest priority interrupt that is pending, and the second is to read the CAN Message Interrupt Pending (CANMSGnINT) register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the INTPND bit of the message object at the same time by setting the CLRINTPND bit in the CANIFnCMSK register. Once the INTPND bit has been cleared, the CANINT register contains the message number for the next message object with a pending interrupt.

11.3.13 Test Mode

A Test Mode is provided which allows various diagnostics to be performed. Test Mode is entered by setting the TEST bit in the CANCTL register. Once in Test Mode, the TX[1:0], LBACK, SILENT and BASIC bits in the CAN Test (CANTST) register can be used to put the CAN controller into the various diagnostic modes. The RX bit in the CANTST register allows monitoring of the CANnRX signal. All CANTST register functions are disabled when the TEST bit is cleared.

11.3.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the SILENT bit in the CANTST register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

11.3.13.2 Loopback Mode

Loopback mode is useful for self-test functions. In Loopback Mode, the CAN Controller internally routes the CANnTX signal on to the CANnRX signal and treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into the message buffer. The CAN Controller is put in Loopback Mode by setting the LBACK bit in the CANTST register. To be independent from external stimulation, the CAN Controller ignores acknowledge errors (a recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. The actual value of the CANnRX signal is disregarded by the CAN Controller. The transmitted messages can be monitored on the CANnTX signal.

11.3.13.3 Loopback Combined with Silent Mode

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the CANnTX and CANnRX signals. In this mode, the CANnRX signal is disconnected from the CAN Controller and the CANnTX signal is held recessive. This mode is enabled by setting both the LBACK and SILENT bits in the CANTST register.

11.3.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the BUSY bit of the CANIF1CRQ register. The CANIF1 registers are locked while the BUSY bit is set. The BUSY bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the BUSY bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the BUSY bit in the CANIF1CRQ register while the CANIF1 registers are locked. If the CPU has cleared the BUSY bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register are stored in the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the BUSY bit of the CANIF2CRQ register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the CANIFnCMSK registers are not evaluated. The message number of the CANIFnCRQ registers is also not evaluated. In the CANIF2MCTL register, the NEWDAT and MSGLST bits retain their function, the DLC[3:0] field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the BASIC bit in the CANTST register.

11.3.13.5 Transmit Control

Software can directly override control of the CANnTX signal in four different ways, as follows:

- CANnTX is controlled by the CAN Controller.
- The sample point is driven on the CANnTX signal to monitor the bit timing.
- CANnTX drives a low value.
- CANnTX drives a high value.

The last two functions, combined with the readable CAN receive pin CANnRX, can be used to check the physical layer of the CAN bus.

The Transmit Control function is enabled by programming the TX[1:0] field in the CANTST register. The three test functions for the CANnTX signal interfere with all CAN protocol functions. TX[1:0] must be cleared when CAN message transfer or Loopback Mode, Silent Mode, or Basic Mode are selected.

11.3.14 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

11.3.15 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 kbps up to 1000 kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see [Figure 11-4](#)): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see [Table 11-2](#)). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's input clock (f_{sys}) and the Baud Rate Prescaler (BRP):

$$t_q = \text{BRP} / f_{sys}$$

The f_{sys} input clock is the system clock frequency as configured by the RSCLKCFG register (see [Section 4.2.11](#)).

The Synchronization Segment Sync is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync and the Sync is called the phase error of that edge.

The Propagation Time Segment Prop is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase1 and Phase2 surround the Sample Point.

The (Re)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range must be considered.

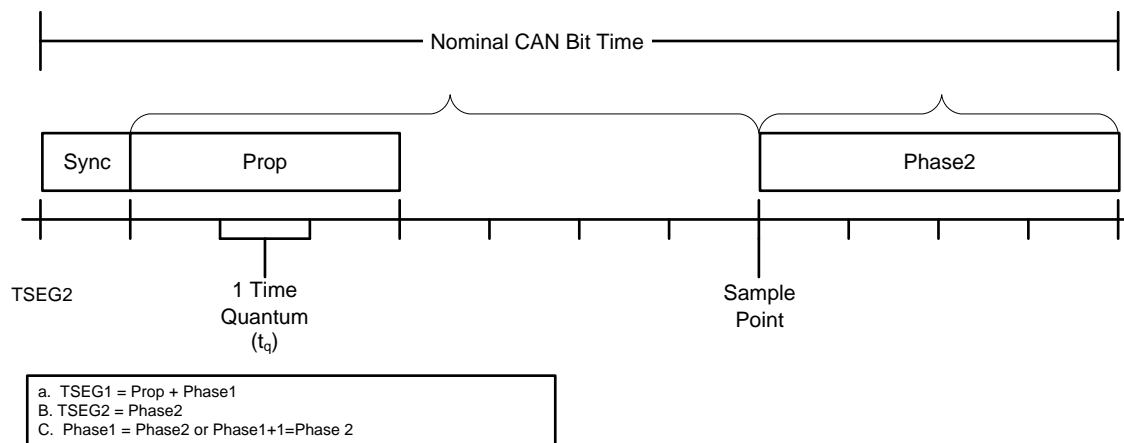


Figure 11-4. CAN Bit Time

Table 11-2. CAN Protocol Ranges

Parameter	Range	Remark
BRP	[1 to 64]	Defines the length of the time quantum t_q . The CANBRPE register can be used to extend the range to 1024.
Sync	1 t_q	Fixed length, synchronization of bus input to system clock
Prop	[1 to 8] t_q	Compensates for the physical delay times
Phase1	[1 to 8] t_q	May be lengthened temporarily by synchronization
Phase2	[1 to 8] t_q	May be shortened temporarily by synchronization
SJW	[1 to 4] t_q	May not be longer than either Phase Buffer Segment

The bit timing configuration is programmed in two register bytes in the CANBIT register. In the CANBIT register, the four components TSEG2, TSEG1, SJW, and BRP must be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1 to n], values in the range of [0 to n-1] are programmed. That way, for example, SJW (functional range of [1 to 4]) is represented by only two bits in the SJW bit field. [Table 11-3](#) shows the relationship between the CANBIT register values and the parameters.

Table 11-3. CANBIT Register Values

CANBIT Register Field	Setting
TSEG2	Phase2 – 1
TSEG1	Prop + Phase1 – 1
SJW	SJW – 1
BRP	BRP

Therefore, the length of the bit time is (programmed values):

$$[TSEG1 + TSEG2 + 3] t_q$$

or (functional values):

$$[Sync + Prop + Phase1 + Phase2] t_q$$

The data in the CANBIT register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. In addition, the controller generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. The bit value is received or transmitted at the sample point. The information processing time (IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

The IPT is application-specific but may not be longer than $2 t_q$; the IPT of the CAN is $0 t_q$. Its length is the lower limit of the programmed length of Phase2. In case of synchronization, Phase2 may be shortened to a value less than IPT, which does not affect bus timing.

11.3.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the required bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is Prop. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop is converted into time quanta (rounded up to the nearest integer multiple of t_q).

Sync is $1 t_q$ long (fixed), which leaves $(\text{bit time} - \text{Prop} - 1) t_q$ for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, that is, $\text{Phase2} = \text{Phase1}$, else $\text{Phase2} = \text{Phase1} + 1$.

The minimum nominal length of Phase2 has to be regarded as well. Phase2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of $[0 \text{ to } 2] t_q$.

The length of the synchronization jump width is set to the least of 4, Phase1, or Phase2.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given in [Equation 9](#):

$$(1 - df) \times f_{nom} \leq f_{osc} \leq (1 + df) \times f_{nom}$$

where

- df = Maximum tolerance of oscillator frequency
- f_{osc} = Actual oscillator frequency
- f_{nom} = Nominal oscillator frequency

(9)

Maximum frequency tolerance must take into account [Equation 10](#) and [Equation 11](#):

$$df \leq \frac{(\text{Phase_seg1, Phase_seg2})_{\min}}{2 \times (13 \times t_{\text{bit}} - \text{Phase_Seg2})}$$

(10)

$$df_{\max} = 2 \times df \times f_{nom}$$

where

- Phase1 and Phase2 are from [Table 11-2](#)
- t_{bit} = Bit Time
- df_{\max} = Maximum difference between two oscillators

(11)

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate must be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol-compliant configuration of the CAN bit timing.

11.3.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, and the bit rate is 1 Mbps.

$$\text{bit time} = 1 \mu\text{s} = n \times t_q = 5 \times t_q$$

where

- $t_q = 200 \text{ ns}$
- $t_q = (\text{baud rate prescaler}) / \text{CAN clock}$
- Baud rate prescaler = $t_q \times \text{CAN clock}$
- Baud rate prescaler = $200 \times 10^9 / 25 \times 10^6 = 8$
- $t_{\text{Sync}} = 1 \times t_q = 200 \text{ ns}$

(12)

\\fixed at 1 time quanta

- delay of bus driver = 50 ns
- delay of receiver circuit = 30 ns
- delay of bus line (40 m) = 220 ns

$$t_{\text{Prop}} 400 \text{ ns} = 2 \times t_q$$

(13)

\\400 is next integer multiple of t_q

$$\text{bit time} = t_{\text{Sync}} + t_{\text{Tseg1}} + t_{\text{Tseg2}} = 5 \times t_q$$

(14)

$$\text{bit time} = t_{\text{Sync}} + t_{\text{Prop}} + t_{\text{Phase 1}} + t_{\text{Phase 2}}$$

(15)

$$t_{\text{Phase 1}} + t_{\text{Phase 2}} = \text{bit time} - t_{\text{Sync}} - t_{\text{Prop}}$$

(16)

$$t_{\text{Phase 1}} + t_{\text{Phase 2}} = (5 \times t_q) - (1 \times t_q) - (2 \times t_q)$$

(17)

$$t_{\text{Phase 1}} + t_{\text{Phase 2}} = 2 \times t_q$$

(18)

$$t_{\text{Phase 1}} = 1 \times t_q$$

(19)

$$t_{\text{Phase 2}} = 1 \times t_q$$

(20)

\\tPhase2 = tPhase1

$$tTSeg1 = tProp + tPhase1$$

where

- $tTSeg1 = (2 \times t_q) + (1 \times t_q)$
- $tTSeg1 = 3 \times t_q$ (21)

$$tTSeg2 = tPhase2$$

where

- $tTSeg2 = (\text{Information Processing Time} + 1) \times t_q$
- $tTSeg2 = 1 \times t_q$ (22)

\\Assumes IPT = 0

$$tSJW = 1 \times t_q \quad (23)$$

\\Least of 4, Phase1 and Phase2

In the previous examples, the bit field values for the CANBIT register are as follows (see [Table 11-4](#)):

Table 11-4. CANBIT Register Values for High Baud Rate Example

Bit	Value
TSEG2	$= TSeg2 - 1$ $= 1 - 1$ $= 0$
TSEG1	$= TSeg1 - 1$ $= 3 - 1$ $= 2$
SJW	$= SJW - 1$ $= 1 - 1$ $= 0$
BRP	$= \text{Baud rate prescaler} - 1$ $= 5 - 1$ $= 4$

The final value programmed into the CANBIT register = 0x0204.

11.3.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, and the bit rate is 100 kbps.

$$\text{bit time} = 10 \mu\text{s} = n \times t_q = 10 \times t_q \quad (24)$$

$$t_q = 1 \mu\text{s} \quad (25)$$

$$t_q = (\text{Baud rate Prescaler}) / \text{CAN Clock} \quad (26)$$

$$\text{Baud rate Prescaler} = t_q \times \text{CAN Clock} \quad (27)$$

$$\text{Baud rate Prescaler} = 1E-6 \times 50E6 = 50 \quad (28)$$

$$tSync = 1 \times t_q = 1 \mu\text{s} \quad (29)$$

\\fixed at 1 time quanta

$$\text{delay of bus driver } 200 \text{ ns} \quad (30)$$

$$\text{delay of receiver circuit } 80 \text{ ns} \quad (31)$$

$$\text{delay of bus line (40 m) } 220 \text{ ns} \quad (32)$$

$$tProp \ 1 \mu\text{s} = 1 \times t_q \quad (33)$$

\\1 μs is next integer multiple of tq

$$\text{bit time} = tSync + tTSeg1 + tTSeg2 = 10 \times t_q \quad (34)$$

$$\text{bit time} = tSync + tProp + tPhase1 + tPhase2 \quad (35)$$

$$tPhase1 + tPhase2 = \text{bit time} - tSync - tProp \quad (36)$$

$$tPhase1 + tPhase2 = (10 \times t_q) - (1 \times t_q) - (1 \times t_q) \quad (37)$$

$$tPhase1 + tPhase2 = 8 \times t_q \quad (38)$$

$$tPhase1 = 4 \times t_q \quad (39)$$

$$tPhase2 = 4 \times t_q \quad (40)$$

```

\\tPhase1 = tPhase2
tTSeg1 = tProp + tPhase1 (41)
tTSeg1 = (1 × tq) + (4 × tq) (42)
tTSeg1 = 5 × tq (43)
tTSeg2 = tPhase2 (44)
tTSeg2 = (Information Processing Time + 4) × tq (45)
tTSeg2 = 4 × tq (46)
\\Assumes IPT = 0
tSJW = 4 × tq (47)
\\Least of 4, Phase1, and Phase2

```

Table 11-5. CANBIT Register Values for Low Baud Rate Example

Bit	Value
TSEG2	= TSeg2 – 1 = 4 – 1 = 3
TSEG1	= TSeg1 – 1 = 5 – 1 = 4
SJW	= SJW – 1 = 4 – 1 = 3
BRP	= Baud rate prescaler – 1 = 50 – 1 = 49

The final value programmed into the CANBIT register = 0x34F1.

11.4 CAN Registers

Table 11-6 lists the memory-mapped registers for the CAN. All register offset addresses not listed in Table 11-6 should be considered as RESERVED locations and the register contents should not be modified.

All address offsets are relative to the base address of the CAN module:

- CAN0: 0x40040000
- CAN1: 0x40041000

The CAN controller clock must be enabled before the registers can be programmed. There must be a delay of 3 system clock cycles after the CAN module clock is enabled before any CAN module registers are accessed.

Table 11-6. CAN Registers

Offset	Acronym	Register Name	Section
0x0	CANCTL	CAN Control	Section 11.4.1
0x4	CANSTS	CAN Status	Section 11.4.2
0x8	CANERR	CAN Error Counter	Section 11.4.3
0xC	CANBIT	CAN Bit Timing	Section 11.4.4
0x10	CANINT	CAN Interrupt	Section 11.4.5
0x14	CANTST	CAN Test	Section 11.4.6
0x18	CANBRPE	CAN Baud Rate Prescaler Extension	Section 11.4.7
0x20	CANIF1CRQ	CAN IF1 Command Request	Section 11.4.8
0x24	CANIF1CMSK	CAN IF1 Command Mask	Section 11.4.9
0x28	CANIF1MSK1	CAN IF1 Mask 1	Section 11.4.10
0x2C	CANIF1MSK2	CAN IF1 Mask 2	Section 11.4.11
0x30	CANIF1ARB1	CAN IF1 Arbitration 1	Section 11.4.12
0x34	CANIF1ARB2	CAN IF1 Arbitration 2	Section 11.4.13
0x38	CANIF1MCTL	CAN IF1 Message Control	Section 11.4.14
0x3C	CANIF1DA1	CAN IF1 Data A1	Section 11.4.15
0x40	CANIF1DA2	CAN IF1 Data A2	Section 11.4.15
0x44	CANIF1DB1	CAN IF1 Data B1	Section 11.4.15
0x48	CANIF1DB2	CAN IF1 Data B2	Section 11.4.15
0x80	CANIF2CRQ	CAN IF2 Command Request	Section 11.4.8
0x84	CANIF2CMSK	CAN IF2 Command Mask	Section 11.4.9
0x88	CANIF2MSK1	CAN IF2 Mask 1	Section 11.4.10
0x8C	CANIF2MSK2	CAN IF2 Mask 2	Section 11.4.11
0x90	CANIF2ARB1	CAN IF2 Arbitration 1	Section 11.4.12
0x94	CANIF2ARB2	CAN IF2 Arbitration 2	Section 11.4.13
0x98	CANIF2MCTL	CAN IF2 Message Control	Section 11.4.14
0x9C	CANIF2DA1	CAN IF2 Data A1	Section 11.4.15
0xA0	CANIF2DA2	CAN IF2 Data A2	Section 11.4.15
0xA4	CANIF2DB1	CAN IF2 Data B1	Section 11.4.15
0xA8	CANIF2DB2	CAN IF2 Data B2	Section 11.4.15
0x100	CANTXRQ1	CAN Transmission Request 1	Section 11.4.16
0x104	CANTXRQ2	CAN Transmission Request 2	Section 11.4.16
0x120	CANNWDA1	CAN New Data 1	Section 11.4.17
0x124	CANNWDA2	CAN New Data 2	Section 11.4.17
0x140	CANMSG1INT	CAN Message 1 Interrupt Pending	Section 11.4.18
0x144	CANMSG2INT	CAN Message 2 Interrupt Pending	Section 11.4.18
0x160	CANMSG1VAL	CAN Message 1 Valid	Section 11.4.19

Table 11-6. CAN Registers (continued)

Offset	Acronym	Register Name	Section
0x164	CANMSG2VAL	CAN Message 2 Valid	Section 11.4.19

Complex bit access types are encoded to fit into small table cells. [Table 11-7](#) shows the codes that are used for access types in this section.

Table 11-7. CAN Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

11.4.1 CANCTL Register (Offset = 0x0) [reset = 0x1]

CAN Control (CANCTL)

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of bus idle (129 * 11 consecutive high bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after INIT is cleared, each time a sequence of 11 high bits has been monitored, a BITERROR0 code is written to the CANSTS register (the LEC field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CANCTL is shown in [Figure 11-5](#) and described in [Table 11-8](#).

Return to [Summary Table](#).

Figure 11-5. CANCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
TEST	CCE	DAR	RESERVED	EIE	SIE	IE	INIT
R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1

Table 11-8. CANCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	TEST	R/W	0x0	Test Mode Enable. 0x0 = The CAN controller is operating normally. 0x1 = The CAN controller is in test mode.
6	CCE	R/W	0x0	Configuration Change Enable. 0x0 = Write accesses to the CANBIT register are not allowed. 0x1 = Write accesses to the CANBIT register are allowed if the INIT bit is 1.
5	DAR	R/W	0x0	Disable Automatic Retransmission. 0x0 = Auto-retransmission of disturbed messages is enabled. 0x1 = Auto-retransmission is disabled.
4	RESERVED	R	0x0	
3	EIE	R/W	0x0	Error Interrupt Enable 0x0 = No error status interrupt is generated. 0x1 = A change in the BOFF or EWARN bits in the CANSTS register generates an interrupt.

Table 11-8. CANCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	SIE	R/W	0x0	Status Interrupt Enable. 0x0 = No status interrupt is generated. 0x1 = An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the CANSTS register generates an interrupt.
1	IE	R/W	0x0	CAN Interrupt Enable 0x0 = Interrupts disabled. 0x1 = Interrupts enabled.
0	INIT	R/W	0x1	Initialization. 0x0 = Normal operation. 0x1 = Initialization started.

11.4.2 CANSTS Register (Offset = 0x4) [reset = 0x0]

CAN Status (CANSTS)

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 0x7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits, and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if the corresponding enable bits in the CAN Control (CANCTL) register are set. A change of the EPASS bit or a write to the RXOK, TXOK, or LEC bits does not generate an interrupt.

Reading the CAN Status (CANSTS) register clears the CAN Interrupt (CANINT) register, if it is pending.

CANSTS is shown in [Figure 11-6](#) and described in [Table 11-9](#).

Return to [Summary Table](#).

Figure 11-6. CANSTS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
R-0x0	R-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0		

Table 11-9. CANSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	BOFF	R	0x0	Bus-Off Status. 0x0 = The CAN controller is not in bus-off state. 0x1 = The CAN controller is in bus-off state.
6	EWARN	R	0x0	Warning Status. 0x0 = Both error counters are below the error warning limit of 96. 0x1 = At least one of the error counters has reached the error warning limit of 96.
5	EPASS	R	0x0	Error Passive. 0x0 = The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127. 0x1 = The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.
4	RXOK	R/W	0x0	Received a Message Successfully. This bit must be cleared by writing a 0 to it. 0x0 = Since this bit was last cleared, no message has been successfully received. 0x1 = Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.

Table 11-9. CANSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	TXOK	R/W	0x0	<p>Transmitted a Message Successfully. This bit must be cleared by writing a 0 to it.</p> <p>0x0 = Since this bit was last cleared, no message has been successfully transmitted.</p> <p>0x1 = Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.</p>
2-0	LEC	R/W	0x0	<p>Last Error Code. This is the type of the last error to occur on the CAN bus.</p> <p>0x0 = No Error</p> <p>0x1 = Stuff ErrorMore than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.</p> <p>0x2 = Format ErrorA fixed format part of the received frame has the wrong format.</p> <p>0x3 = ACK ErrorThe message transmitted was not acknowledged by another node.</p> <p>0x4 = Bit 1 ErrorWhen a message is transmitted, the CAN controller monitors the data lines to detect any conflicts. When the arbitration field is transmitted, data conflicts are a part of the arbitration protocol. When other frame fields are transmitted, data conflicts are considered errors.A Bit 1 Error indicates that the device wanted to send a High level (logical 1) but the monitored bus value was Low (logical 0).</p> <p>0x5 = Bit 0 ErrorA Bit 0 Error indicates that the device wanted to send a Low level (logical 0), but the monitored bus value was High (logical 1).During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. By checking for this status, software can monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.</p> <p>0x6 = CRC ErrorThe CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.</p> <p>0x7 = No EventWhen the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field.</p>

11.4.3 CANERR Register (Offset = 0x8) [reset = 0x0]

CAN Error Counter (CANERR)

This register contains the error counter values, which can be used to analyze the cause of an error.

CANERR is shown in [Figure 11-7](#) and described in [Table 11-10](#).

Return to [Summary Table](#).

Figure 11-7. CANERR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP		REC						TEC							
R-0x0		R-0x0						R-0x0							

Table 11-10. CANERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	RP	R	0x0	Received Error Passive. 0x0 = The Receive Error counter is below the Error Passive level (127 or less). 0x1 = The Receive Error counter has reached the Error Passive level (128 or greater).
14-8	REC	R	0x0	Receive Error Counter. This field contains the state of the receiver error counter (0 to 127).
7-0	TEC	R	0x0	Transmit Error Counter. This field contains the state of the transmit error counter (0 to 255).

11.4.4 CANBIT Register (Offset = 0xC) [reset = 0x2301]

CAN Bit Timing (CANBIT)

This register is used to program the bit width and bit quantum. Values are programmed to the system clock frequency. This register is write-enabled by setting the CCE and INIT bits in the CANCTL register. See [Section 11.3.15](#) for more information.

CANBIT is shown in [Figure 11-8](#) and described in [Table 11-11](#).

Return to [Summary Table](#).

Figure 11-8. CANBIT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESE RVED	TSEG2			TSEG1				SJW		BRP					
R-0x0	R/W-0x2			R/W-0x3				R/W-0x0		R/W-0x1					

Table 11-11. CANBIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0x0	
14-12	TSEG2	R/W	0x2	Time Segment after Sample Point. 0x00 to 0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x2 means that 3 (2+1) bit time quanta are defined for Phase2 (see Figure 11-4). The bit time quanta is defined by the BRP field.
11-8	TSEG1	R/W	0x3	Time Segment Before Sample Point. 0x00 to 0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x3 means that 4 (3+1) bit time quanta are defined for Phase1 (see Figure 11-4). The bit time quanta is defined by the BRP field.
7-6	SJW	R/W	0x0	(Re)Synchronization Jump Width. 0x00 to 0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of TSEG2 or TSEG1 by the value in SJW. So the reset value of 0 adjusts the length by 1 bit time quanta.
5-0	BRP	R/W	0x1	Baud Rate Prescaler. The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. 0x00 to 0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1). The CANBRPE register can be used to further divide the bit time.

11.4.5 CANINT Register (Offset = 0x10) [reset = 0x0]

CAN Interrupt (CANINT)

This register indicates the source of the interrupt.

If several interrupts are pending, the CAN Interrupt (CANINT) register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the INTID field is not 0x0000 (the default) and the IE bit in the CANCTL register is set, the interrupt is active. The interrupt line remains active until the INTID field is cleared by reading the CANSTS register, or until the IE bit in the CANCTL register is cleared.

NOTE: Reading the CAN Status (CANSTS) register clears the CAN Interrupt (CANINT) register, if it is pending.

CANINT is shown in [Figure 11-9](#) and described in [Table 11-12](#).

Return to [Summary Table](#).

Figure 11-9. CANINT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																INTID															
R-0x0																R-0x0															

Table 11-12. CANINT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	INTID	R	0x0	<p>Interrupt Identifier.</p> <p>The number in this field indicates the source of the interrupt.</p> <p>0x0000 = No interrupt pending</p> <p>0x0001 to 0x0020 = Number of the message object that caused the interrupt</p> <p>0x0021 to 0x7FFF = Reserved</p> <p>0x8000 = Status Interrupt</p> <p>0x8001 to 0xFFFF = Reserved</p>

11.4.6 CANTST Register (Offset = 0x14) [reset = X]

CAN Test (CANTST)

This register is used for self-test and external pin access. It is write-enabled by setting the TEST bit in the CANCTL register. Different test functions may be combined, however, CAN transfers are affected if the TX bits in this register are not zero.

CANTST is shown in [Figure 11-10](#) and described in [Table 11-13](#).

Return to [Summary Table](#).

Figure 11-10. CANTST Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RX	TX		LBACK	SILENT	BASIC	RESERVED	
R-0x0	R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	

Table 11-13. CANTST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	RX	R	0x0	Receive Observation. 0x0 = The CANnRx pin is low. 0x1 = The CANnRx pin is high.
6-5	TX	R/W	0x0	Transmit Control. Overrides control of the CANnTx pin. 0x0 = CAN Module ControlCANnTx is controlled by the CAN module; default operation 0x1 = Sample PointThe sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing. 0x2 = Driven LowCANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus. 0x3 = Driven HighCANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.
4	LBACK	R/W	0x0	Loopback Mode. 0x0 = Loopback mode is disabled. 0x1 = Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.
3	SILENT	R/W	0x0	Silent Mode. 0x0 = Silent mode is disabled. 0x1 = Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode.
2	BASIC	R/W	0x0	Basic Mode. 0x0 = Basic mode is disabled. 0x1 = Basic mode is enabled. In basic mode, software should use the CANIF1 registers as the transmit buffer and use the CANIF2 registers as the receive buffer.
1-0	RESERVED	R	0x0	

11.4.7 CANBRPE Register (Offset = 0x18) [reset = X]

CAN Baud Rate Prescaler Extension (CANBRPE)

This register is used to further divide the bit time set with the BRP bit in the CANBIT register. It is write-enabled by setting the CCE bit in the CANCTL register.

CANBRPE is shown in [Figure 11-11](#) and described in [Table 11-14](#).

Return to [Summary Table](#).

Figure 11-11. CANBRPE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												BRPE			
R-0x0																												R/W-0x0			

Table 11-14. CANBRPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	BRPE	R/W	0x0	Baud Rate Prescaler Extension. 0x00 to 0x0F: Extend the BRP bit in the CANBIT register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (LSBs).

11.4.8 CANIFnCRQ Register (Offset = 0x20) [reset = 0x1]

CAN IF1 Command Request (CANIF1CRQ), offset 0x020

CAN IF2 Command Request (CANIF2CRQ), offset 0x080

A message transfer is started as soon as there is a write of the message object number to the MNUM field when the TXRQST bit in the CANIF1MCTL register is set. With this write operation, the BUSY bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then clears the BUSY bit.

CANIFnCRQ is shown in [Figure 11-12](#) and described in [Table 11-15](#).

Return to [Summary Table](#).

Figure 11-12. CANIFnCRQ Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
BUSY	RESERVED						
R-0x0	R-0x0						
7	6	5	4	3	2	1	0
RESERVED		MNUM					
R-0x0		R/W-0x1					

Table 11-15. CANIFnCRQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	BUSY	R	0x0	Busy Flag. 0x0 = This bit is cleared when read/write action has finished. 0x1 = This bit is set when a write occurs to the message number in this register.
14-6	RESERVED	R	0x0	
5-0	MNUM	R/W	0x1	Message Number. Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32. 0x00 = Reserved. 0 is not a valid message number; it is interpreted as 0x20, or object 32. 0x01 to 0x20 = Message Number. Indicates specified message object 1 to 32. 0x21 to 0x3F = Reserved. Not a valid message number; values are shifted and it is interpreted as 0x01 to 0x1F.

11.4.9 CANIFnCMSK Register [reset = 0x0]

CAN IF1 Command Mask (CANIF1CMSK), offset 0x024

CAN IF2 Command Mask (CANIF2CMSK), offset 0x084

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND or NEWDAT bits are set, the interrupt pending or new data flags in the message object buffer are cleared.

CANIFnCMSK is shown in [Figure 11-13](#) and described in [Table 11-16](#).

Return to [Summary Table](#).

Figure 11-13. CANIFnCMSK Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 11-16. CANIFnCMSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	WRNRD	R/W	0x0	Write, Not Read. Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND or NEWDAT bits are set. 0x0 = Transfer the data in the CAN message object specified by the MNUM field in the CANIFnCRQ register into the CANIFn registers. 0x1 = Transfer the data in the CANIFn registers to the CAN message object specified by the MNUM field in the CAN Command Request (CANIFnCRQ).
6	MASK	R/W	0x0	Access Mask Bits. 0x0 = Mask bits unchanged. 0x1 = Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.
5	ARB	R/W	0x0	Access Arbitration Bits. 0x0 = Arbitration bits unchanged. 0x1 = Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.
4	CONTROL	R/W	0x0	Access Control Bits. 0x0 = Control bits unchanged. 0x1 = Transfer control bits from the CANIFnMCTL register into the Interface registers.

Table 11-16. CANIFnCMSK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	CLRINTPND	R/W	0x0	<p>Clear Interrupt Pending Bit. The function of this bit depends on the configuration of the WRNRD bit.</p> <p>0x0 = If WRNRD is clear, the interrupt pending status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, the INTPND bit in the message object remains unchanged.</p> <p>0x1 = If WRNRD is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, the INTPND bit is cleared in the message object.</p>
2	NEWDAT / TXRQST	R/W	0x0	<p>NEWDAT / TXRQST Bit. The function of this bit depends on the configuration of the WRNRD bit.</p> <p>0x0 = If WRNRD is clear, the value of the new data status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, a transmission is not requested.</p> <p>0x1 = If WRNRD is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, a transmission is requested. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored.</p>
1	DATAA	R/W	0x0	<p>Access Data Byte 0 to 3. The function of this bit depends on the configuration of the WRNRD bit.</p> <p>0x0 = Data bytes 0 to 3 are unchanged.</p> <p>0x1 = If WRNRD is clear, transfer data bytes 0 to 3 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 0 to 3 in message object to CANIFnDA1 and CANIFnDA2.</p>
0	DATAB	R/W	0x0	<p>Access Data Byte 4 to 7. The function of this bit depends on the configuration of the WRNRD bit as follows:</p> <p>0x0 = Data bytes 4 to 7 are unchanged.</p> <p>0x1 = If WRNRD is clear, transfer data bytes 4 to 7 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 4 to 7 in message object to CANIFnDA1 and CANIFnDA2.</p>

11.4.10 CANIFnMSK1 Register [reset = 0xFFFF]

CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (CANIFnDAn), arbitration information (CANIFnARBn), and control information (CANIFnMCTL) to the message object in the message RAM. The mask is used with the ID bit in the CANIFnARBn register for acceptance filtering. Additional mask information is contained in the CANIFnMSK2 register.

CANIFnMSK1 is shown in [Figure 11-14](#) and described in [Table 11-17](#).

Return to [Summary Table](#).

Figure 11-14. CANIFnMSK1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MSK															
R-0x0																R/W-0xFFFF															

Table 11-17. CANIFnMSK1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	MSK	R/W	0xFFFF	Identifier Mask. When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The MSK field in the CANIFnMSK2 register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored. 0x0 = The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering. 0x1 = The corresponding identifier field (ID) is used for acceptance filtering.

11.4.11 CANIFnMSK2 Register [reset = 0xE0FF]

CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C

CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C

This register holds extended mask information that accompanies the CANIFnMSK1 register.

CANIFnMSK2 is shown in [Figure 11-15](#) and described in [Table 11-18](#).

Return to [Summary Table](#).

Figure 11-15. CANIFnMSK2 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
MXTD	MDIR	RESERVED	MSK				
R/W-0x1	R/W-0x1	R-0x1	R/W-0xFF				
7	6	5	4	3	2	1	0
MSK							
R/W-0xFF							

Table 11-18. CANIFnMSK2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	MXTD	R/W	0x1	Mask Extended Identifier. 0x0 = The extended identifier bit (XTD in the CANIFnARB2 register) has no effect on the acceptance filtering. 0x1 = The extended identifier bit XTD is used for acceptance filtering.
14	MDIR	R/W	0x1	Mask Message Direction. 0x0 = The message direction bit (DIR in the CANIFnARB2 register) has no effect for acceptance filtering. 0x1 = The message direction bit DIR is used for acceptance filtering.
13	RESERVED	R	0x1	
12-0	MSK	R/W	0xFF	Identifier Mask. When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The MSK field in the CANIFnMSK1 register are used for bits [15:0] of the ID. When using an 11-bit identifier, MSK[12:2] are used for bits [10:0] of the ID. 0x0 = The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering. 0x1 = The corresponding identifier field (ID) is used for acceptance filtering.

11.4.12 CANIFnARB1 Register [reset = 0x0]

CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

These registers hold the identifiers for acceptance filtering.

CANIFnARB1 is shown in [Figure 11-16](#) and described in [Table 11-19](#).

Return to [Summary Table](#).

Figure 11-16. CANIFnARB1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																ID															
R-0x0																R/W-0x0															

Table 11-19. CANIFnARB1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	ID	R/W	0x0	<p>Message Identifier.</p> <p>This bit field is used with the ID field in the CANIFnARB2 register to create the message identifier.</p> <p>When using a 29-bit identifier, bits 15:0 of the CANIFnARB1 register are [15:0] of the ID, while bits 12:0 of the CANIFnARB2 register are [28:16] of the ID.</p> <p>When using an 11-bit identifier, these bits are not used.</p>

11.4.13 CANIFnARB2 Register [reset = 0x0]

CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094

These registers hold information for acceptance filtering.

CANIFnARB2 is shown in [Figure 11-17](#) and described in [Table 11-20](#).

Return to [Summary Table](#).

Figure 11-17. CANIFnARB2 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
MSGVAL	XTD	DIR	ID				
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0				
7	6	5	4	3	2	1	0
ID							
R/W-0x0							

Table 11-20. CANIFnARB2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	MSGVAL	R/W	0x0	<p>Message Valid. All unused message objects should have this bit cleared during initialization and before clearing the INIT bit in the CANCTL register. The MSGVAL bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the ID fields in the CANIFnARBn registers, the XTD and DIR bits in the CANIFnARB2 register, or the DLC field in the CANIFnMCTL register.</p> <p>0x0 = The message object is ignored by the message handler. 0x1 = The message object is configured and ready to be considered by the message handler within the CAN controller.</p>
14	XTD	R/W	0x0	<p>Extended Identifier.</p> <p>0x0 = An 11-bit Standard Identifier is used for this message object. 0x1 = A 29-bit Extended Identifier is used for this message object.</p>
13	DIR	R/W	0x0	<p>Message Direction.</p> <p>0x0 = Receive. When the TXRQST bit in the CANIFnMCTL register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object. 0x1 = Transmit. When the TXRQST bit in the CANIFnMCTL register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN =1).</p>
12-0	ID	R/W	0x0	<p>Message Identifier.</p> <p>This bit field is used with the ID field in the CANIFnARB2 register to create the message identifier.</p> <p>When using a 29-bit identifier, ID[15:0] of the CANIFnARB1 register are [15:0] of the ID, while these bits, ID[12:0], are [28:16] of the ID. When using an 11-bit identifier, ID[12:2] are used for bits [10:0] of the ID. The ID field in the CANIFnARB1 register is ignored.</p>

11.4.14 CANIFnMCTL Register [reset = 0x0]

CAN IF1 Message Control (CANIF1MCTL), offset 0x038

CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CANIFnMCTL is shown in [Figure 11-18](#) and described in [Table 11-21](#).

Return to [Summary Table](#).

Figure 11-18. CANIFnMCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
EOB	RESERVED			DLC			
R/W-0x0	R-0x0			R/W-0x0			

Table 11-21. CANIFnMCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	NEWDAT	R/W	0x0	New Data. 0x0 = No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU. 0x1 = The message handler or the CPU has written new data into the data portion of this message object.
14	MSGLST	R/W	0x0	Message Lost. This bit is only valid for message objects when the DIR bit in the CANIFnARB2 register is clear (receive). 0x0 = No message was lost since the last time this bit was cleared by the CPU. 0x1 = The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message.
13	INTPND	R/W	0x0	Interrupt Pending. 0x0 = This message object is not the source of an interrupt. 0x1 = This message object is the source of an interrupt. The interrupt identifier in the CANINT register points to this message object if there is not another interrupt source with a higher priority.
12	UMASK	R/W	0x0	Use Acceptance Mask. 0x0 = Mask is ignored. 0x1 = Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering.
11	TXIE	R/W	0x0	Transmit Interrupt Enable 0x0 = The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame. 0x1 = The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame.

Table 11-21. CANIFnMCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	RXIE	R/W	0x0	Receive Interrupt Enable. 0x0 = The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame. 0x1 = The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame.
9	RMTEN	R/W	0x0	Remote Enable. 0x0 = At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged. 0x1 = At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set.
8	TXRQST	R/W	0x0	Transmit Request. If the WRNRD and TXRQST bits in the CANIFnCMSK register are set, this bit is ignored. 0x0 = This message object is not waiting for transmission. 0x1 = The transmission of this message object is requested and is not yet done.
7	EOB	R/W	0x0	End of Buffer. This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set. 0x0 = Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer. 0x1 = Single message object or last message object of a FIFO Buffer.
6-4	RESERVED	R	0x0	
3-0	DLC	R/W	0x0	Data Length Code. 0x0 to 0x8 = Specifies the number of bytes in the data frame. 0x9 to 0xF = Defaults to a data frame with 8 bytes. The DLC field in the CANIFnMCTL register of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it writes DLC to the value given by the received message.

11.4.15 CANIFnDnn Register [reset = 0x0]

CAN IF1 Data A1 (CANIF1DA1), offset 0x03C

CAN IF1 Data A2 (CANIF1DA2), offset 0x040

CAN IF1 Data B1 (CANIF1DB1), offset 0x044

CAN IF1 Data B2 (CANIF1DB2), offset 0x048

CAN IF2 Data A1 (CANIF2DA1), offset 0x09C

CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0

CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4

CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8

These registers contain the data to be sent or that has been received. In a CAN data frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CANIFnDnn is shown in [Figure 11-19](#) and described in [Table 11-22](#).

Return to [Summary Table](#).

Figure 11-19. CANIFnDnn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DATA															
R-0x0																R/W-0x0															

Table 11-22. CANIFnDnn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	DATA	R/W	0x0	Data. The CANIFnDA1 registers contain data bytes 1 and 0; CANIFnDA2 data bytes 3 and 2; CANIFnDB1 data bytes 5 and 4; and CANIFnDB2 data bytes 7 and 6.

11.4.16 CANTXRQn Register [reset = 0x0]

CAN Transmission Request 1 (CANTXRQ1), offset 0x100

CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The CANTXRQ1 and CANTXRQ2 registers hold the TXRQST bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TXRQST bit of a specific message object can be changed by three sources: (1) the CPU via the CANIFnMCTL register, (2) the message handler state machine after the reception of a remote frame, or (3) the message handler state machine after a successful transmission.

The CANTXRQ1 register contains the TXRQST bits of the first 16 message objects in the message RAM; the CANTXRQ2 register contains the TXRQST bits of the second 16 message objects.

CANTXRQn is shown in [Figure 11-20](#) and described in [Table 11-23](#).

Return to [Summary Table](#).

Figure 11-20. CANTXRQn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TXRQST															
R-0x0																R-0x0															

Table 11-23. CANTXRQn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	TXRQST	R	0x0	Transmission Request Bits. 0x0 = The corresponding message object is not waiting for transmission. 0x1 = The transmission of the corresponding message object is requested and is not yet done.

11.4.17 CANNWDAn Register [reset = 0x0]

CAN New Data 1 (CANNWDA1), offset 0x120

CAN New Data 2 (CANNWDA2), offset 0x124

The CANNWDA1 and CANNWDA2 registers hold the NEWDAT bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The NEWDAT bit of a specific message object can be changed by three sources: (1) the CPU via the CANIFnMCTL register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The CANNWDA1 register contains the NEWDAT bits of the first 16 message objects in the message RAM; the CANNWDA2 register contains the NEWDAT bits of the second 16 message objects.

CANNWDAn is shown in [Figure 11-21](#) and described in [Table 11-24](#).

Return to [Summary Table](#).

Figure 11-21. CANNWDAn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																NEWDAT															
R-0x0																R-0x0															

Table 11-24. CANNWDAn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	NEWDAT	R	0x0	<p>New Data Bits.</p> <p>0x0 = No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU.</p> <p>0x1 = The message handler or the CPU has written new data into the data portion of the corresponding message object.</p>

11.4.18 CANMSGnINT Register [reset = 0x0]

CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

The CANMSG1INT and CANMSG2INT registers hold the INTPND bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The INTPND bit of a specific message object can be changed through two sources: (1) the CPU via the CANIFnMCTL register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the CANINT register.

The CANMSG1INT register contains the INTPND bits of the first 16 message objects in the message RAM; the CANMSG2INT register contains the INTPND bits of the second 16 message objects.

CANMSGnINT is shown in [Figure 11-22](#) and described in [Table 11-25](#).

Return to [Summary Table](#).

Figure 11-22. CANMSGnINT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																INTPND															
R-0x0																R-0x0															

Table 11-25. CANMSGnINT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	INTPND	R	0x0	Interrupt Pending Bits. 0x0 = The corresponding message object is not the source of an interrupt. 0x1 = The corresponding message object is the source of an interrupt.

11.4.19 CANMSGnVAL Register [reset = 0x0]

CAN Message 1 Valid (CANMSG1VAL), offset 0x160

CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The CANMSG1VAL and CANMSG2VAL registers hold the MSGVAL bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message valid bit of a specific message object can be changed with the CANIFnARB2 register.

The CANMSG1VAL register contains the MSGVAL bits of the first 16 message objects in the message RAM; the CANMSG2VAL register contains the MSGVAL bits of the second 16 message objects in the message RAM.

CANMSGnVAL is shown in [Figure 11-23](#) and described in [Table 11-26](#).

Return to [Summary Table](#).

Figure 11-23. CANMSGnVAL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MSGVAL															
R-0x0																R-0x0															

Table 11-26. CANMSGnVAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	MSGVAL	R	0x0	<p>Message Valid Bits.</p> <p>0x0 = The corresponding message object is not configured and is ignored by the message handler.</p> <p>0x1 = The corresponding message object is configured and should be considered by the message handler.</p>

Analog Comparators

This chapter describes the analog comparators.

Topic	Page
12.1 Introduction	829
12.2 Block Diagram.....	830
12.3 Functional Description	830
12.4 Initialization and Configuration	833
12.5 Comparator Registers	834

12.1 Introduction

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result.

NOTE: Not all comparators have the option to drive an output pin. See the device-specific data sheet for more information.

The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application through interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic are separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

Three independent integrated analog comparators support the following functions:

- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of the following voltages:
 - An individual external reference voltage
 - A shared single external reference voltage
 - A shared internal reference voltage

12.2 Block Diagram

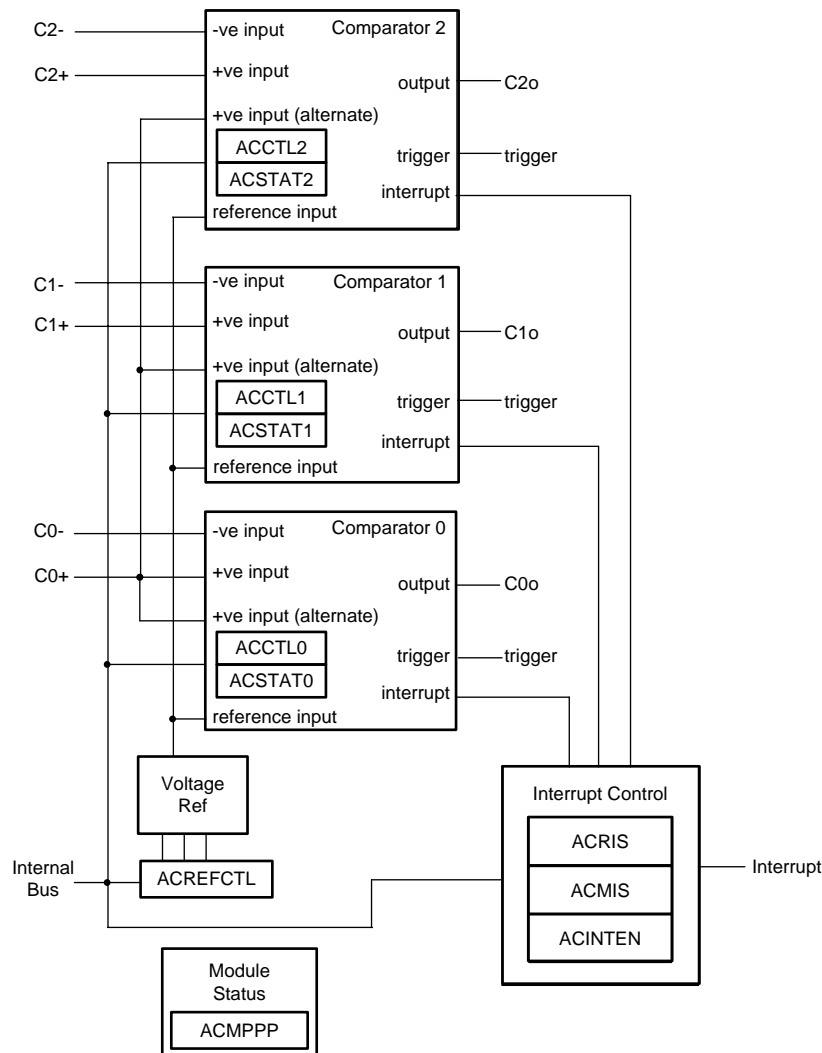


Figure 12-1. Analog Comparator Module Block Diagram

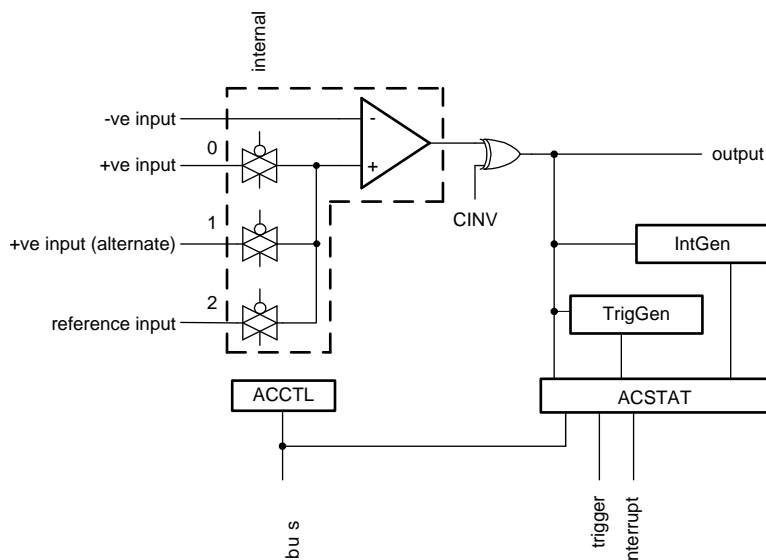
NOTE: This block diagram depicts the maximum number of analog comparators and comparator outputs for the family of microcontrollers; the number for this specific device may vary. See [Section 12.5.7](#) for what is included on this device.

12.3 Functional Description

The comparator compares the VIN⁻ and VIN⁺ inputs to produce an output, VOUT.

VIN⁻ < VIN⁺, VOUT = 1 VIN⁻ > VIN⁺, VOUT = 0

As shown in [Figure 12-2](#), the input source for VIN⁻ is an external input, Cn⁻, where n is the analog comparator number. In addition to an external input, Cn⁺, input sources for VIN⁺ can be the C0⁺ or an internal reference, V_{IREF}.



Copyright © 2017, Texas Instruments Incorporated

Figure 12-2. Structure of Comparator Unit

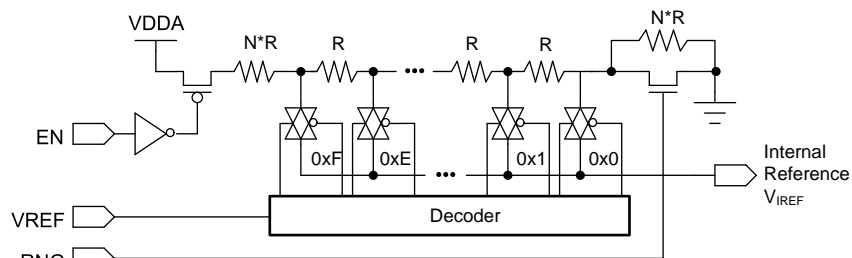
A comparator is configured through two status and control registers, Analog Comparator Control (ACCTL) and Analog Comparator Status (ACSTAT). The internal reference is configured through one control register, Analog Comparator Reference Voltage Control (ACREFCTL). Interrupt status and control are configured through three registers, Analog Comparator Masked Interrupt Status (ACMIS), Analog Comparator Raw Interrupt Status (ACRIS), and Analog Comparator Interrupt Enable (ACINTEN).

Typically, the comparator output is used internally to generate an interrupt as controlled by the ISEN bit in the ACCCTL register. The output may also be used to drive one of the external pins (Cno), or generate an analog-to-digital converter (ADC) trigger.

NOTE: The ASRCP bits in the ACCCTL register must be set before using the analog comparators.

12.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 12-3. The internal reference is controlled by a single configuration register (ACREFCTL).



Copyright © 2017, Texas Instruments Incorporated

Figure 12-3. Comparator Internal Reference Structure

NOTE: In the figure above, $N \times R$ represents a multiple of the R value that produces the results specified in Table 12-1.

The internal reference can be programmed in one of two modes (low range or high range) depending on the RNG bit in the ACREFTL register. When RNG is clear, the internal reference is in high-range mode, and when RNG is set the internal reference is in low-range mode.

In each range, the internal reference, V_{IREF} , has 16 preprogrammed thresholds or step values. The threshold to be used to compare the external input voltage against is selected using the VREF field in the ACREFTL register.

In the high-range mode, the V_{IREF} threshold voltages start at the ideal high-range starting voltage of $V_{DDA} / 4.2$ and increase in ideal constant voltage steps of $V_{DDA} / 29.4$.

In the low-range mode, the V_{IREF} threshold voltages start at 0 V and increase in ideal constant voltage steps of $V_{DDA} / 22.12$. [Table 12-1](#) summarizes the ideal V_{IREF} step voltages for each mode and their dependence on the RNG and VREF fields.

Table 12-1. Internal Reference Voltage and ACREFTL Field Values

ACREFTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN = 0	RNG = X	0 V (GND) for any value of VREF. It is recommended that RNG = 1 and VREF = 0 to minimize noise on the reference ground.
EN = 1	RNG = 0	V_{IREF} High Range: 16 voltage threshold values indexed by VREF = 0x0.. 0xF Ideal starting voltage (VREF =0): $V_{DDA} / 4.2$ Ideal step size: $V_{DDA} / 29.4$ Ideal V_{IREF} threshold values: $V_{IREF} (VREF) = V_{DDA} / 4.2 + VREF \times (V_{DDA} / 29.4)$, for VREF = 0x0.. 0xF For minimum and maximum V_{IREF} threshold values, see Table 12-2 .
	RNG = 1	V_{IREF} Low Range: 16 voltage threshold values indexed by VREF = 0x0.. 0xF Ideal starting voltage (VREF =0): 0 V Ideal step size: $V_{DDA} / 22.12$ Ideal V_{IREF} threshold values: $V_{IREF} (VREF) = VREF \times (V_{DDA} / 22.12)$, for VREF = 0x0.. 0xF For minimum and maximum V_{IREF} threshold values, see Table 12-3 .

Note that the values shown in [Table 12-1](#) are the ideal values of the V_{IREF} thresholds. These values actually vary between minimum and maximum values for each threshold step, depending on process and temperature. The minimum and maximum values for each step are given by:

- $V_{IREF}(VREF) [Min] = \text{Ideal } V_{IREF}(VREF) (\text{Ideal Step size } 2 \text{ mV}) / 2$
- $V_{IREF}(VREF) [Max] = \text{Ideal } V_{IREF}(VREF) + (\text{Ideal Step size } 2 \text{ mV}) / 2$

Examples of minimum and maximum V_{IREF} values for $V_{DDA} = 3.3 \text{ V}$ for high and low ranges, are shown in [Table 12-2](#) and [Table 12-3](#). Note that these examples are only valid for $V_{DDA} = 3.3 \text{ V}$; values scale up and down with V_{DDA} .

Table 12-2. Analog Comparator Voltage Reference Characteristics ($V_{DDA} = 3.3 \text{ V}$, EN = 1, and RNG = 0)

VREF Value	$V_{IREF} Min$	Ideal V_{IREF}	$V_{IREF} Max$	Unit
0x0	0.731	0.786	0.841	V
0x1	0.843	0.898	0.953	V
0x2	0.955	1.010	1.065	V
0x3	1.067	1.122	1.178	V
0x4	1.180	1.235	1.290	V
0x5	1.292	1.347	1.402	V
0x6	1.404	1.459	1.514	V
0x7	1.516	1.571	1.627	V
0x8	1.629	1.684	1.739	V
0x9	1.741	1.796	1.851	V
0xA	1.853	1.908	1.963	V

Table 12-2. Analog Comparator Voltage Reference Characteristics ($V_{DDA} = 3.3\text{ V}$, $EN = 1$, and $RNG = 0$) (continued)

VREF Value	V _{IREF} Min	Ideal V _{IREF}	V _{IREF} Max	Unit
0xB	1.965	2.020	2.076	V
0xC	2.078	2.133	2.188	V
0xD	2.190	2.245	2.300	V
0xE	2.302	2.357	2.412	V
0xF	2.414	2.469	2.525	V

Table 12-3. Analog Comparator Voltage Reference Characteristics ($V_{DDA} = 3.3\text{ V}$, $EN = 1$, and $RNG = 1$)

VREF Value	V _{IREF} Min	Ideal V _{IREF}	V _{IREF} Max	Unit
0x0	0.000	0.000	0.074	V
0x1	0.076	0.149	0.223	V
0x2	0.225	0.298	0.372	V
0x3	0.374	0.448	0.521	V
0x4	0.523	0.597	0.670	V
0x5	0.672	0.746	0.820	V
0x6	0.822	0.895	0.969	V
0x7	0.971	1.044	1.118	V
0x8	1.120	1.193	1.267	V
0x9	1.269	1.343	1.416	V
0xA	1.418	1.492	1.565	V
0xB	1.567	1.641	1.715	V
0xC	1.717	1.790	1.864	V
0xD	1.866	1.939	2.013	V
0xE	2.015	2.089	2.162	V
0xF	2.164	2.238	2.311	V

12.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator clock by writing a value of 0x0000.0001 to the RCGCACMP register in the System Control module (see [Section 4.2.98](#)).
2. Enable the clock to the appropriate GPIO modules using the RCGCGPIO register (see [Section 4.2.87](#)). To find out which GPIO ports to enable, see the device-specific data sheet.
3. Enable the GPIO port and pin (in the GPIO module) associated with the input signals as GPIO inputs. To determine which GPIO to configure, see the device-specific data sheet.
4. Configure the PMCN fields in the GPIOPCTL register to assign the analog comparator output signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).
5. Configure the internal voltage reference to 1.65 V by writing the ACREFCTL register with the value 0x0000.030C.
6. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the ACCTLn register with the value of 0x0000.040C.
7. Delay for 10 μ s.
8. Read the comparator output value by reading the OVAL value of the ACSTATn register.
9. Change the level of the comparator negative input signal C- to see the OVAL value change.

12.5 Comparator Registers

Table 12-4 lists the memory-mapped registers for the ACMP. All register offset addresses not listed in Table 12-4 should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the Analog Comparator base address of 0x4003C000. Note that the analog comparator clock must be enabled before the registers can be programmed (see Section 4.2.98). There must be a delay of 3 system clock cycles after the analog comparator module clock is enabled before any analog comparator module registers are accessed.

Table 12-4. Comparator Registers

Offset	Acronym	Register Name	Section
0x0	ACMIS	Analog Comparator Masked Interrupt Status	Section 12.5.1
0x4	ACRIS	Analog Comparator Raw Interrupt Status	Section 12.5.2
0x8	ACINTEN	Analog Comparator Interrupt Enable	Section 12.5.3
0x10	ACREFCTL	Analog Comparator Reference Voltage Control	Section 12.5.4
0x20	ACSTAT0	Analog Comparator Status 0	Section 12.5.5
0x24	ACCTL0	Analog Comparator Control 0	Section 12.5.6
0x40	ACSTAT1	Analog Comparator Status 1	Section 12.5.5
0x44	ACCTL1	Analog Comparator Control 1	Section 12.5.6
0x60	ACSTAT2	Analog Comparator Status 2	Section 12.5.5
0x64	ACCTL2	Analog Comparator Control 2	Section 12.5.6
0xFC0	ACMPPP	Analog Comparator Peripheral Properties	Section 12.5.7

Complex bit access types are encoded to fit into small table cells. Table 12-5 shows the codes that are used for access types in this section.

Table 12-5. ACMP Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

12.5.1 ACMIS Register (Offset = 0x0) [reset = 0x0]

Analog Comparator Masked Interrupt Status (ACMIS)

This register provides a summary of the interrupt status (masked) of the comparators.

ACMIS is shown in [Figure 12-4](#) and described in [Table 12-6](#).

Return to [Summary Table](#).

Figure 12-4. ACMIS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													IN2	IN1	IN0
R-0x0													R/W1 C-0x0	R/W1 C-0x0	R/W1 C-0x0

Table 12-6. ACMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31:3	RESERVED	R	0x0	
2	IN2	R/W1C	0x0	Comparator 2 Masked Interrupt Status. This bit is cleared by writing a 1. Clearing this bit also clears the IN2 bit in the ACRIS register.
1	IN1	R/W1C	0x0	Comparator 1 Masked Interrupt Status. This bit is cleared by writing a 1. Clearing this bit also clears the IN1 bit in the ACRIS register.
0	IN0	R/W1C	0x0	Comparator 0 Masked Interrupt Status. This bit is cleared by writing a 1. Clearing this bit also clears the IN0 bit in the ACRIS register.

12.5.2 ACRIS Register (Offset = 0x4) [reset = 0x0]

Analog Comparator Raw Interrupt Status (ACRIS)

This register provides a summary of the interrupt status (raw) of the comparators. The bits in this register must be enabled to generate interrupts using the ACINTEN register.

ACRIS is shown in [Figure 12-5](#) and described in [Table 12-7](#).

Return to [Summary Table](#).

Figure 12-5. ACRIS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													IN2	IN1	IN0
R-0x0													R-0x0	R-0x0	R-0x0

Table 12-7. ACRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31:3	RESERVED	R	0x0	
2	IN2	R	0x0	Comparator 2 Interrupt Status. This bit is cleared by writing a 1 to the IN2 bit in the ACMIS register.
1	IN1	R	0x0	Comparator 1 Interrupt Status. This bit is cleared by writing a 1 to the IN1 bit in the ACMIS register.
0	IN0	R	0x0	Comparator 0 Interrupt Status. This bit is cleared by writing a 1 to the IN0 bit in the ACMIS register.

12.5.3 ACINTEN Register (Offset = 0x8) [reset = 0x0]

Analog Comparator Interrupt Enable (ACINTEN)

This register provides the interrupt enable for the comparators.

ACINTEN is shown in [Figure 12-6](#) and described in [Table 12-8](#).

Return to [Summary Table](#).

Figure 12-6. ACINTEN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													IN2	IN1	IN0
R-0x0													R/W-0x0	R/W-0x0	R/W-0x0

Table 12-8. ACINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31:3	RESERVED	R	0x0	
2	IN2	R/W	0x0	Comparator 2 Interrupt Enable.
1	IN1	R/W	0x0	Comparator 1 Interrupt Enable.
0	IN0	R/W	0x0	Comparator 0 Interrupt Enable.

12.5.4 ACREFCTL Register (Offset = 0x10) [reset = 0x0]

Analog Comparator Reference Voltage Control (ACREFCTL)

This register specifies whether the resistor ladder is powered on as well as the range and tap.

ACREFCTL is shown in [Figure 12-7](#) and described in [Table 12-9](#).

Return to [Summary Table](#).

Figure 12-7. ACREFCTL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						EN	RNG	RESERVED				VREF			
R-0x0						R/W-0x0	R/W-0x0	R-0x0				R/W-0x0			

Table 12-9. ACREFCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31:10	RESERVED	R	0x0	
9	EN	R/W	0x0	Resistor Ladder Enable. This bit is cleared at reset so that the internal reference consumes the least amount of power if it is not used.
8	RNG	R/W	0x0	Resistor Ladder Range.
7:4	RESERVED	R	0x0	
3:0	VREF	R/W	0x0	Resistor Ladder Voltage Ref. The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See for some output reference voltage examples.

12.5.5 ACSTATn Register [reset = 0x0]

Analog Comparator Status 0 (ACSTAT0), offset 0x020

Analog Comparator Status 1 (ACSTAT1), offset 0x040

Analog Comparator Status 2 (ACSTAT2), offset 0x060

These registers specify the current output value of the comparator.

ACSTATn is shown in [Figure 12-8](#) and described in [Table 12-10](#).

Return to [Summary Table](#).

Figure 12-8. ACSTATn Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						OVAL	RESERVED
R-0x0						R-0x0	R-0x0

Table 12-10. ACSTATn Register Field Descriptions

Bit	Field	Type	Reset	Description
31:2	RESERVED	R	0x0	
1	OVAL	R	0x0	Comparator Output Value. VIN - is the voltage on the Cn- pin. VIN+ is the voltage on the Cn+ pin, the C0+ pin, or the internal voltage reference (V _{IREF}) as defined by the ASRCP bit in the ACCTL register.
0	RESERVED	R	0x0	

12.5.6 ACCTLn Register [reset = 0x0]

Analog Comparator Control 0 (ACCTL0), offset 0x024

Analog Comparator Control 1 (ACCTL1), offset 0x044

Analog Comparator Control 2 (ACCTL2), offset 0x064

These registers configure the comparator's input and output.

ACCTLn is shown in [Figure 12-9](#) and described in [Table 12-11](#).

Return to [Summary Table](#).

Figure 12-9. ACCTLn Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				TOEN	ASRCP		RESERVED
R-0x0				R/W-0x0	R/W-0x0		R-0x0
7	6	5	4	3	2	1	0
TSLVAL	TSEN		ISLVAL	ISEN		CINV	RESERVED
R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0		R/W-0x0	R-0x0

Table 12-11. ACCTLn Register Field Descriptions

Bit	Field	Type	Reset	Description
31:12	RESERVED	R	0x0	
11	TOEN	R/W	0x0	Trigger Output Enable.
10:9	ASRCP	R/W	0x0	Analog Source Positive. The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows:
8	RESERVED	R	0x0	
7	TSLVAL	R/W	0x0	Trigger Sense Level Value.
6:5	TSEN	R/W	0x0	Trigger Sense. The TSEN field specifies the sense of the comparator output that generates an ADC event.
4	ISLVAL	R/W	0x0	Interrupt Sense Level Value.
3:2	ISEN	R/W	0x0	Interrupt Sense. The ISEN field specifies the sense of the comparator output that generates an interrupt.
1	CINV	R/W	0x0	Comparator Output Invert.
0	RESERVED	R	0x0	

12.5.7 ACMPPP Register (Offset = 0xFC0) [reset = 0x70007]

Analog Comparator Peripheral Properties (ACMPPPP)

The ACMPPP register provides information regarding the properties of the analog comparator module.

ACMPPPP is shown in [Figure 12-10](#) and described in [Table 12-12](#).

Return to [Summary Table](#).

Figure 12-10. ACMPPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED					C2O	C1O	C0O
R-0x0					R-0x1	R-0x1	R-0x1
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					CMP2	CMP1	CMP0
R-0x0					R-0x1	R-0x1	R-0x1

Table 12-12. ACMPPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31:19	RESERVED	R	0x0	
18	C2O	R	0x1	Comparator Output 2 Present.
17	C1O	R	0x1	Comparator Output 1 Present.
16	C0O	R	0x1	Comparator Output 0 Present.
15:3	RESERVED	R	0x0	
2	CMP2	R	0x1	Comparator 2 Present.
1	CMP1	R	0x1	Comparator 1 Present.
0	CMP0	R	0x1	Comparator 0 Present.

Cyclical Redundancy Check (CRC)

The Cyclical Redundancy Check (CRC) computation module can be used for message transfer and safety system checks, as well as in conjunction with the AES and DES modules.

Topic	Page
13.1 Introduction	843
13.2 Functional Description	843
13.3 Initialization and Configuration	844
13.4 CRC Registers.....	846

13.1 Introduction

The following features are supported:

- Support four major CRC forms:
 - CRC16-CCITT as used by ITU-T X.25
 - CRC16-IBM as used by USB and ANSI
 - CRC32-IEEE as used by IEEE 802.3 and MPEG 2
 - CRC32C as used by G.Hn
- Allows word and byte feed
- Supports automatic initialization and manual initialization
- Supports MSb and LSb
- Supports CCITT post-processing
- Can be fed by μ DMA, flash memory, and code

13.2 Functional Description

The following sections describe the features of CRC.

13.2.1 CRC Support

The purpose of the CRC engine is to accelerate CRC and TCP checksum operation. The result of the CRC operation is a 32- and 16-bit signature which can be used to check the sanity of data. The required mode of operation is selected through the TYPE bit in the CRC Control (CRCCTRL) register, offset 0x400. A μ DMA software channel can be used to burst data into the CRC module. CRCs are computed combinatorially in one clock.

The CRC module contains all of the control registers to which the input context interfaces. Because CRC calculations are a single cycle, as soon as data is written to CRC Data Input (CRCDIN) register, the result of CRC/CSUM is updated in the CRC SEED/Context (CRCSEED) register, offset 0x410. The input data is computed by the selected CRC polynomial or CSUM.

13.2.1.1 CRC Checksum Engine

Software can offload the CRC and checksum task to the CRC checksum engine accelerator. The accelerator has registers that need to be programmed to initiate processing. These registers should be fed with data to calculate CRC/CSUM. Software should configure the μ DMA channel for data movement through the DMA Channel Map Select n (DMACHMAPn) register in the μ DMA module. Further μ DMA configuration guidelines are available in [Chapter 8](#).

The starting seed for the CRC and checksum operation is programmed in the CRC SEED/Context (CRCSEED) register at offset 0x410. Depending on the encoding of the INIT field in the CRCCTRL register, the value of the SEED field can be initialized to any one of the following:

- A unique context value written to the CRCSEED register (INIT = 0x0)
- All 0s (INIT = 0x2)
- All 1s (INIT = 0x3)

When the operation is complete, software should read the result from the CRC Post Processing Result (CRCRSLTPP) register, offset 0x418, and a software channel μ DMA interrupt should be used to identify completion.

13.2.1.2 Data Size

The CRC module supports data being fed 32-bit words and 8 bits at a time and can dynamically switch back and forth. The data size is configured by programming the SIZE bit in the CRCCTRL register, offset 0x400.

Because CRC is a division on a long stream of bits, the application must take into consideration the bit order. When processing message data that is read out by words, bit order is not an issue. For example, if the data value in the message is 0x12345678, the most significant eight byte is 0x12 (00010010 in binary). If the data is processed as bytes, 0x12, 0x34, 0x56, and 0x78 are copied into memory in that order and the word is stored as 0x78563412, where 0x12 is written as byte 0, 0x34 is written as byte 1, and so on.

13.2.1.3 Endian Configuration

The following endian configuration is provided by the ENDIAN field in the CRCCTRL register:

- Swap byte in half-word
- Swap half word

Input data width is four bytes, hence the configuration only affects the four-byte word. [Table 13-1](#) lists the configurations that the ENDIAN bit field supports, assuming the input word is {B3, B2, B1, B0}.

Table 13-1. Endian Configuration

ENDIAN Encoding	Definition	Configuration
0x0	Configuration unchanged.	{B3, B2, B1, B0}
0x1	Bytes are swapped in half-words but half-words are not swapped.	{B2, B3, B0, B1}
0x2	Half-words are swapped but bytes are not swapped in half-word.	{B1, B0, B3, B2}
0x3	Bytes are swapped in half-words and half-words are swapped.	{B0, B1, B2, B3}

Bit reversal is supported by the BR bit in the CRCCTRL register. The bit reversal operation works in tandem with endian control. For example, the above table with the BR option set would look like this:

Table 13-2. Endian Configuration With Bit Reversal

ENDIAN Encoding	Initial Endian Configuration	Configuration With Bit Reversal (BR = 1)
0x0	Configuration unchanged. {B3[31:24], B2[23:16], B1[15:8], B0[7:0]}	B3[24:31],B2[16:23],B1[8:15],B0[0:7]
0x1	Bytes are swapped in half-words but half-words are not swapped. {B2[23:16], B3[31:24], B0[7:0], B1[15:8]}	B2[16:23],B3[24:31],B0[0:7],B1[8:15]
0x2	Half-words are swapped but bytes are not swapped in half-word. {B1[15:8], B0[7:0], B3[31:24], B2[23:16]}	B1[8:15],B0[0:7],B3[24:31],B2[16:23]
0x3	Bytes are swapped in half-words and half-words are swapped. {B0[7:0], B1[15:8], B2[23:16], B3[31:24]}	B0[0:7],B1[8:15],B2[16:23],B3[24:31]

13.3 Initialization and Configuration

The following describes the initialization and configuration procedures of the CRC module.

13.3.1 CRC Initialization and Configuration

The CRC engine works in push through mode, which means it works on streaming data. This section describes the steps for initializing the CRC module:

1. Enable the CRC by setting the R0 bit in the CRC and Cryptographic Module s (RCGCCM) register, System Control offset 0x674.
2. Configure the desired CRC data size, bit order, endian configuration and CRC type by programming the CRC Control (CRCCTRL) register, offset 0x400.
3. If the CRC value has not been initialized to all 0s or all 1s using the INIT field in the CRCCTRL register, program the initial value in the CRC SEED/Context (CRCSEED) register, offset 0x410.
4. Repeatedly write the DATAIN field in the CRC Data Input (CRCDIN) register, offset 0x414. If the SIZE bit in the CRCCTRL register is set to select byte, the CRC engine operates in byte mode and only the least significant byte is used for CRC calculation.

5. When CRC is finished, read the CRCSEED register for the final result. If using post-processing, the raw CRC result is stored in the CRCSEED register and the final, post-processed result can be read from the CRC Post-Processing Result (CRCRSLTPP) register, offset 0x418. Post-processing options are selectable through the OBR and OLVN bits of the CRCCTRL register.

Alternatively a software μ DMA channel can be configured to copy data from the source into the CRCDIN register. When configuring the μ DMA, the destination should be configured to not increment. For more information on how to configure the μ DMA, see [Chapter 8](#).

13.3.1.1 Data Endian Convention for the CRC Engine

If the input stream is expressed as a byte stream, D_{in} , where $D_{in} = \{D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{11}, D_{12}, D_{13}, D_{14}, D_{15}, D_{16}, \dots\}$, then data should be fed to the CRC engine as follows:

- If operating in Byte mode, the CRCDIN register should be written in the following order:
 - {00, 00, 00, D_0 }
 - {00, 00, 00, D_1 }
 - {00, 00, 00, D_2 }
 - {00, 00, 00, D_3 }
 - {00, 00, 00, D_4 }
 - {00, 00, 00, D_5 }
 - {00, 00, 00, D_6 }
 -
 -
- If operating in word mode, the CRCDIN register should be written in the following order:
 - { D_3, D_2, D_1, D_0 }
 - { D_7, D_6, D_5, D_4 }
 - { D_{11}, D_{10}, D_9, D_8 }
 -
 -

13.4 CRC Registers

Table 13-3 lists the memory-mapped registers for the CRC. All register offset addresses not listed in Table 13-3 should be considered as reserved locations and the register contents should not be modified.

The offset listed is a hexadecimal increment to the register's address, relative to the base address of 0x44030000.

NOTE: The CRC module can only be accessed through privileged mode. If the μ DMA is used for CRC transfers, then the DMA Channel Control (DMACHCTL) register also needs to be programmed to allow for privileged accesses.

Table 13-3. CRC Registers

Offset	Acronym	Register Name	Section
400h	CRCCTRL	CRC Control	Section 13.4.1
410h	CRCSEED	CRC SEED/Context	Section 13.4.2
414h	CRCDIN	CRC Data Input	Section 13.4.3
418h	CRCRSLTPP	CRC Post Processing Result	Section 13.4.4

Complex bit access types are encoded to fit into small table cells. Table 13-4 shows the codes that are used for access types in this section.

Table 13-4. CRC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

13.4.1 CRCCTRL Register (Offset = 400h) [reset = 0h]

CRC Control (CRCCTRL)

The CRC Control (CRCCTRL) register is used to configure control of the CRC.

CRCCTRL is shown in [Figure 13-1](#) and described in [Table 13-5](#).

Return to [Summary Table](#).

Figure 13-1. CRCCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED	INIT		SIZE	RESERVED		RESINV	OBR
R-0h	R/W-0h		R/W-0h	R-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
BR	RESERVED	ENDIAN		TYPE			
R/W-0h	R-0h	R/W-0h		R/W-0h			

Table 13-5. CRCCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	
14-13	INIT	R/W	0h	CRC Initialization. Determines initialization value of CRC. This field is self-clearing. With the first write to the CRC Data Input (CRCDIN) register, this value clears to zero and remains zero for the rest of the operation unless written again. 0h = Use the CRCSEED register context as the starting value 1h = reserved 2h = Initialize to all 0s 3h = Initialize to all 1s
12	SIZE	R/W	0h	Input Data Size. 0h = 32-bit (word) 1h = 8-bit (byte)
11-10	RESERVED	R	0h	
9	RESINV	R/W	0h	Result Inverse Enable. 0h = No effect 1h = Invert the result bits before storing in the CRCRSLTPP register.
8	OBR	R/W	0h	Output Reverse Enable See Table 13-2 for more information regarding bit reversal. 0h = No change to result. 1h = Bit reverse the output result byte before storing to CRCRSLTPP register. The reversal is applied to all bytes in a word.
7	BR	R/W	0h	Bit reverse enable. See Table 13-2 for more information regarding bit reversal. 0h = No change to result. 1h = Bit reverse the input byte for all bytes in a word.
6	RESERVED	R	0h	

Table 13-5. CRCCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	ENDIAN	R/W	0h	<p>Endian Control.</p> <p>This field is used to program the endian configuration. The encodings below are with respect to an input word = (B3, B2, B1, B0).</p> <p>See Table 13-1 for more information regarding endian configuration and control.</p> <p>0h = Configuration unchanged. (B3, B2, B1, B0)</p> <p>1h = Bytes are swapped in half-words but half-words are not swapped (B2, B3, B0, B1)</p> <p>2h = Half-words are swapped but bytes are not swapped in half-word. (B1, B0, B3, B2)</p> <p>3h = Bytes are swapped in half-words and half-words are swapped. (B0, B1, B2, B3)</p>
3-0	TYPE	R/W	0h	<p>Operation Type.</p> <p>The TYPE value in the CRCCTRL register should be exclusive.</p> <p>0h = Polynomial 0x8005</p> <p>1h = Polynomial 0x1021</p> <p>2h = Polynomial 0x4C11DB7</p> <p>3h = Polynomial 0x1EDC6F41</p> <p>8h = TCP checksum</p>

13.4.2 CRCSEED Register (Offset = 410h) [reset = 0h]

CRC SEED/Context (CRCSEED)

The CRC SEED/Context (CRCSEED) register is initially written with one of the following three values depending on the encoding of the INIT field in the CRCCTRL register:

- The context value written to the CRCSEED register. This encoding is for SEED values from a previous CRC calculation or a specific protocol. (INIT = 0x0)
- 0x00000000 (INIT = 0x2)
- 0x11111111 (INIT = 0x3)

CRCSEED is shown in [Figure 13-2](#) and described in [Table 13-6](#).

Return to [Summary Table](#).

Figure 13-2. CRCSEED Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															
R/W-0h																															

Table 13-6. CRCSEED Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SEED	R/W	0h	SEED/Context Value. This register contains the starting seed of the CRC and checksum operation. This register also holds the latest result of CRC or checksum operation.

13.4.3 CRCDIN Register (Offset = 414h) [reset = 0h]

CRC Data Input (CRCDIN)

The application or μ DMA writes the CRC Data Input (CRCDIN) register with the next byte or word to compute.

CRCDIN is shown in [Figure 13-3](#) and described in [Table 13-7](#).

Return to [Summary Table](#).

Figure 13-3. CRCDIN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN																															
R/W-0h																															

Table 13-7. CRCDIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATAIN	R/W	0h	Data Input. This register contains the input data value for the CRC or checksum operation.

13.4.4 CRCRSLTPP Register (Offset = 418h) [reset = 0h]

CRC Post Processing Result (CRCRSLTPP)

This register contains the post-processed CRC result as configured by the CRCCTRL register.

CRCRSLTPP is shown in [Figure 13-4](#) and described in [Table 13-8](#).

Return to [Summary Table](#).

Figure 13-4. CRCRSLTPP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSLTPP																															
R-0h																															

Table 13-8. CRCRSLTPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RSLTPP	R	0h	Post Processing Result. This register contains the post-processed CRC result.

Data Encryption Standard Accelerator (DES)

The DES module provides hardware accelerated data encryption and decryption functions. The module runs either the single DES or the triple DES (3DES) algorithm in compliance with the [FIPS 46-3 standard](#) and supports electronic codebook (ECB), cipher block chaining (CBC), and cipher feedback (CFB) modes of operation. It does not support the output feedback (OFB) mode of operation in hardware.

Topic	Page
14.1 Introduction	853
14.2 DES Functional Description	853
14.3 DES Block Diagram	854
14.4 Software Reset	856
14.5 DES Supported Modes of Operation	856
14.6 DES Module Programming Guide – Low Level Programming Models.....	858
14.7 DES Registers	863
14.8 DES μDMA Registers	877

14.1 Introduction

The purpose of the DES algorithm is to encrypt (encipher) or decrypt (decipher) binary coded information. Encrypting data converts it to an unintelligible form called cipher text. Decrypting cipher text converts the data back to its original form called plain text. DES is a symmetrical algorithm in that the encryption and decryption keys are identical. Each triple DES encrypt or decrypt operation is a compound of DES encrypt and decrypt operations.

The DES accelerator includes the following main features:

- DES or 3DES encryption and decryption
- Feedback modes: ECB, CBC, CFB
- Host interrupt or μ DMA driven modes of operation. μ DMA support for data and context in and result out.
- Fully synchronous design
- Internal wide-bus interface

14.2 DES Functional Description

The DES module is an efficient implementation of a DES block cipher. Block ciphers, as opposed to stream ciphers, operate on blocks of plain text and cipher text. The DES block size is 8 bytes. The DES key consists of 64 binary digits, but only 56 bits are actually used directly by the algorithm. The other 8 bits are used for error detection.

The 64-bit block of input data to be enciphered is initially permuted, then passed through 16 iterations of a calculation that uses a cipher function and finally permuted to the inverse of the initial permutation. At each of the 16 iterations, a 48-bit key computed from the 64-bit input key is applied to one of the 32-bit sub-blocks of the 64-bit input block using the cipher function. The 48-bit key value changes for each iteration. The result of the cipher function is a 32-bit sub-block, which is concatenated with the second 32-bit input sub-block. The resulting 64-bit output block of each iteration feeds back as the input of the next iteration. To decipher, it is only necessary to apply the same algorithm to the enciphered message block, taking care that each iteration of the computation will use the same 48-bit key which was used during enciphering.

The triple DES is the DES used three times in a row (also known as DES-EDE). It uses three keys key1, key2, and key3, so that key length is 168 bits effective: a 64-bit block plaintext is encrypted with key1, decrypted with key2, and encrypted with key3, and a 64-bit ciphertext is decrypted with key1, encrypted with key2, and decrypted with key3.

The following is the three keying options defined in ANSI X9.52 for DES-EDE:

- The three keys key1, key2, and key3 are independent.
- key1 and key2 are independent, but key1 = key3
- key1 = key2 = key3

The first option provides highest level of security; the last option is compatible with single DES. See [Table 14-1](#) for key use.

Table 14-1. Key Repartition

Mode	Key1_L	Key1_H	Key2_L	Key2_H	Key3_L	Key3_H
64 Bit (DES)	✓	✓	X	X	X	X
192 Bit (3DES)	✓	✓	✓	✓	✓	✓

ECB, CBC, and CFB modes can be used with DES and 3DES modes.

14.3 DES Block Diagram

Figure 14-1 shows the module architecture, which consists of primary blocks. The DES module includes a register interface and a μ DMA and interrupt interface.

Depending on the availability of context and data, the DES engine is automatically triggered to process the data. The DES engine is directly connected to the context and data registers such that it can immediately start processing when all data is available.

Packets (blocks of 64 bits) must be parsed into blocks and sequentially fed into the DES, which can buffer the block currently being processed as well as an additional block that may be queued in advance.

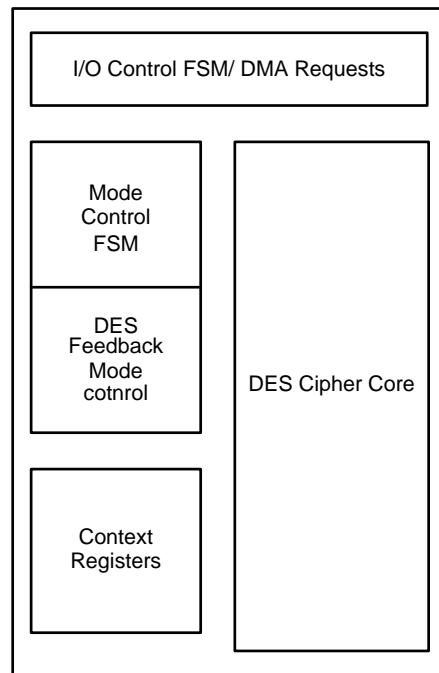


Figure 14-1. DES Block Diagram

14.3.1 μ DMA Control

The μ DMA and interrupt request logic is controlled by the DES Engine. The DES Engine can have multiple μ DMA request signals active in parallel.

There are three DMA channels available:

- DMA context in – Request a new context (DES0 Cin)
- DMA data in – Request input data (DES0 Din)
- DMA data out – Request output data read (DES0 Dout)

14.3.2 Interrupt Control

One interrupt for the DES is sent to the interrupt controller. This interrupt is an OR of the enabled interrupt bits in the DES Interrupt Status (DES_IRQSTATUS) register. These bits are enabled through the DES Interrupt Enable (DES_IRQENABLE) register. The following events can generate an interrupt bit to be set in the DES_IRQSTATUS register:

- New context input required
- Data input required
- Data output ready

14.3.3 Register Interface

The Register Interface block performs all address decoding and control; however, not all registers are available in this block. The context and data input registers are in the DES engine.

14.3.4 DES Engine

The DES buffered engine consists of the following major functional blocks:

- Cipher core: The DES algorithm
- Mode control FSM: Manages the data flow to and from the DES buffered engine and starts each encrypt or decrypt operation
- DES feedback mode block: The logic that implements the various feedback modes supported by the DES buffered engine

14.3.4.1 Mode Control FSM

The mode control FSM manages the data flow to and from the DES engine. This block also sends a start pulse to the encrypt or decrypt core and triggers the core to use the new mode keys when a new context is needed. This module also controls the 3DES operation, such that the DES core module is triggered three times (with different keys) before the result data becomes available.

14.3.4.2 DES Feedback Mode Block

The DES feedback mode block buffers the input and output blocks and contains all logic to implement the 3DES and various feedback modes. See the FIPS-81 document for details on the ECB, CBC, and CFB modes of operation.

By itself, the DES cipher core outputs data compliant for ECB encryption. However, most applications use DES with feedback. Feedback provides additional security by randomizing repeated patterns in the plain text, which could otherwise be exploited to attack the cipher text. The DES buffered engine supports ECB, CBC, and CFB modes of operations for the DES and 3DES algorithm.

3DES mode performs the DES algorithm three times on a single block and uses a different key for each invocation of the DES algorithm, greatly increasing security of the cipher text but at the cost of a 3x reduction in throughput. The DES buffered engine implements a 3DES logic wrapper around the 2-round DES core, enabling seamless 3DES encryption.

14.3.4.3 DES Cipher Core

The DES cipher core implements the DES algorithm as specified in the FIPS 46-3. The core operates on the input block and performs the required substitution, shift, and mix operations. The core also applies the correct key-scheduling.

Inherently, considerable parallelism is possible with the DES algorithm. This is exploited in two ways. For high performance, the 64 bits composing the data block are processed concurrently (4-round implementation). For low gate count, resources are shared on both the main data and key paths (1-round implementation).

A fundamental component of the DES algorithm is the substitution box (S-Box). The S-Box provides a unique 4-bit output for each 6-bit input. The S-Box design is a primary factor for both performance and gate count. The DES Cipher Core has a standard lookup table S-Box that allows room for the synthesizer to optimize on timing or gate count.

14.4 Software Reset

Table 14-2 lists the resets used in the DES module.

Table 14-2. DES Reset Description

Type	Name	Source	Description
Hardware	DES_RST	PRCM	Global reset
Software	DES_SYSCONFIG [1] SOFTRESET	Internal	Starts the soft reset sequence. When the DES_SYSSTATUS/DES_P_SYSSTATUS[0] RESETDONE bit goes to 1, the soft reset sequence is finished.

14.5 DES Supported Modes of Operation

14.5.1 ECB Feedback Mode

Figure 14-2 shows the basic ECB feedback mode of operation, where the input data is passed directly to the basic cryptographic core and the output of the cryptographic core is passed directly to the output buffer. For decryption the DES core operates in reverse, this means the decrypt key sequence is used for the data processing, where encryption uses the encrypt key sequence.

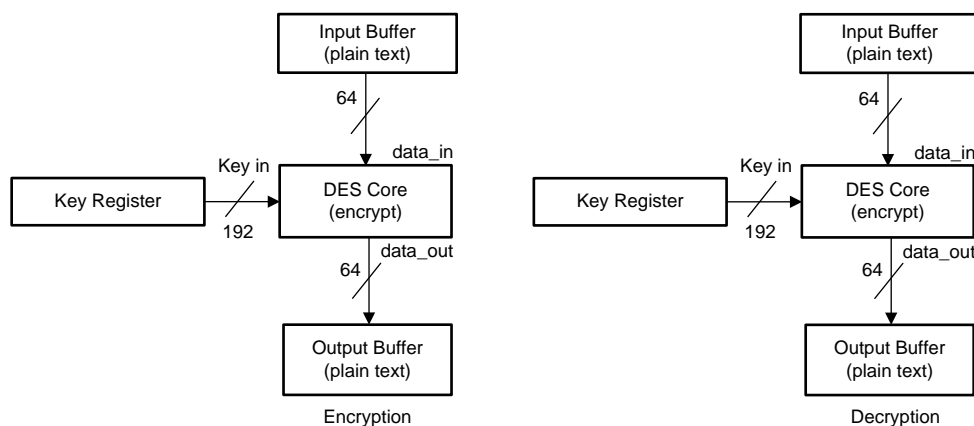


Figure 14-2. DES – ECB Feedback Mode

14.5.1.1 CBC Feedback Mode

Figure 14-3 shows the CBC feedback mode of operation, where the input data is XORed with the initialization vector (IV) before it is passed to the basic crypto core. The output of the crypto core is passed directly to the output buffer. For decryption the operation is reversed, resulting in an XOR at the output of the crypto core.

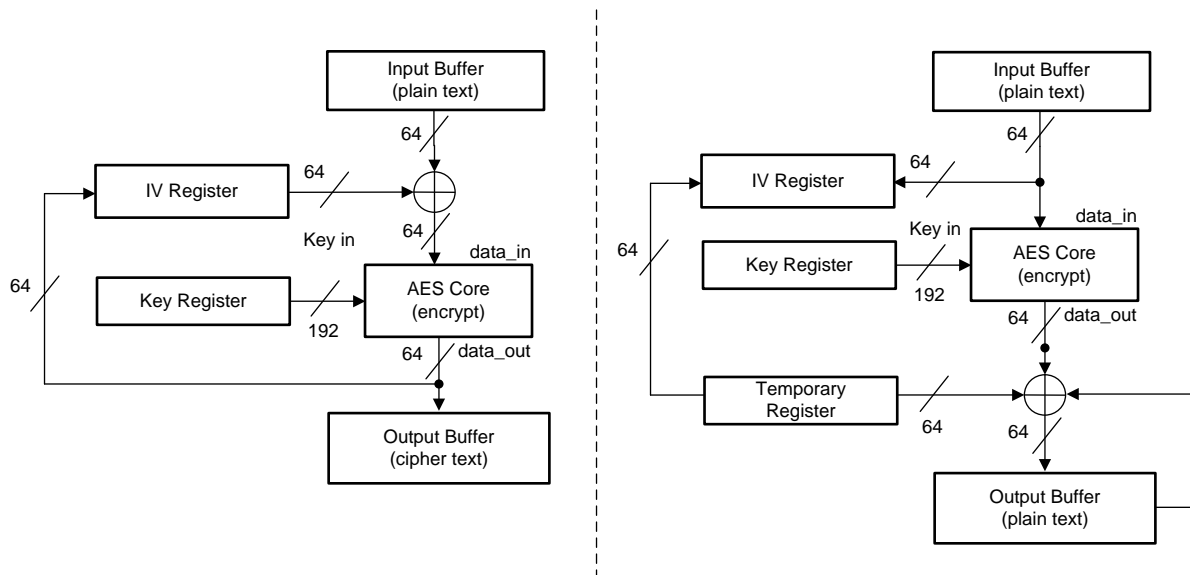


Figure 14-3. DES3DES – CBC Feedback Mode

14.5.1.2 CFB Feedback Mode

Figure 14-4 shows the CFB mode of operation for encryption and decryption. The input for the crypto core is the IV; the result is XORed with the data. The result is fed back using the IV register, as the next input for the crypto core. The decrypt operation is reversed, but the crypto core is still performing an encryption.

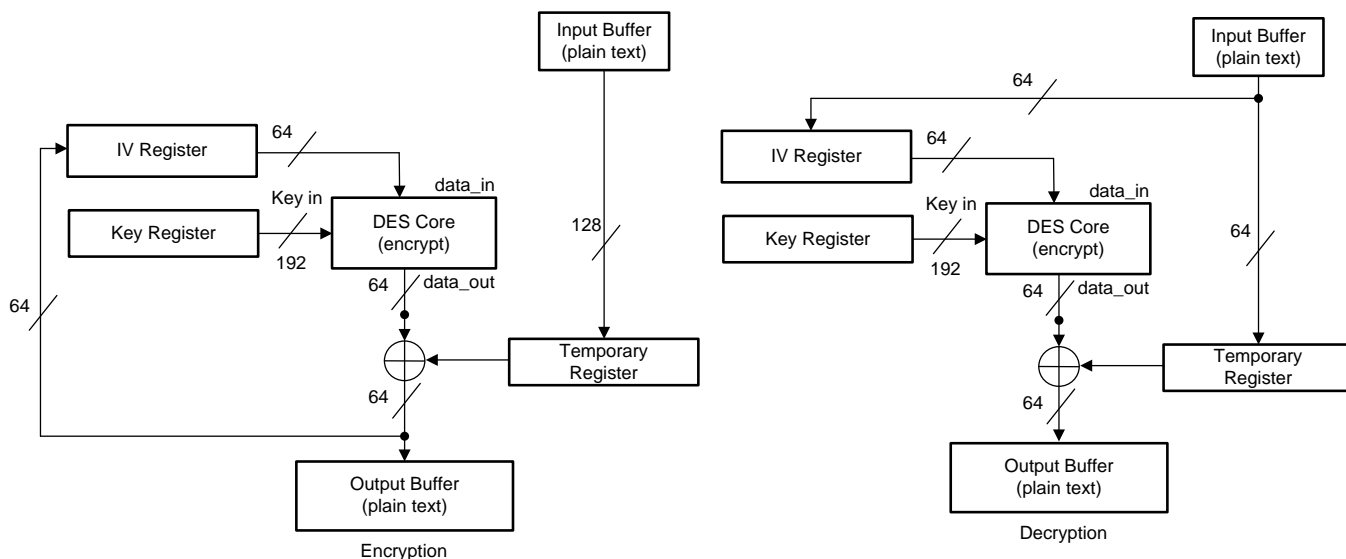


Figure 14-4. DES3DES – CFB Feedback Mode

14.6 DES Module Programming Guide – Low Level Programming Models

14.6.1 Surrounding Modules Global Initialization

14.6.1.1 Main Sequence – DES Global Initialization

The following procedure initializes the DES after a POR (see [Table 14-3](#)).

1. Execute software reset.
2. Wait for reset to complete.
3. Select force idle mode.
4. Select the algorithm type (DES or 3DES).
5. Select the operating mode (ECB, CBC, or CFB).
6. IF: ECB operating mode is not selected
7. Load the initialization vector LSW.
8. Load the initialization vector MSW.
9. Define the cryptographic data length.
10. ENDIF
11. Select encryption or decryption.

Table 14-3. DES Global Initialization

Step	Register/Bit Field / Programming Model	Value
1	DES_SYSCONFIG[1] SOFTRESET	0x1
2	DES_SYSSTATUS[0] RESETDONE	= 0x1
3	DES_SYSCONFIG[3:2] SIDLE	0x0
4	See the programming models	-
5	DES_CTRL[5:4] MODE	-
6	DES_CTRL[5:4] MODE	≠ 0x0
7	DES_IV_L[31:0] IV_L	-
8	DES_IV_H[31:0] IV_H	-
9	DES_LENGTH[31:0] LENGTH	-
10		-
11	DES_CTRL[2] DIRECTION	-

14.6.1.2 Subsequence – Configure the DES Algorithm Type

The following subsequence details the steps to configure the DES algorithm type settings (see [Table 14-4](#)).

1. Load key 1 LSW.
2. Load key 1 MSW.
3. Select DES algorithm.

Table 14-4. DES Algorithm Type Configuration

Step	Register/Bit Field / Programming Model	Value
1	DES_KEY1_L[31:0] KEY1_L	-
2	DES_KEY1_H[31:0] KEY1_H	-
3	DES_CTRL[3] TDES	0x0

14.6.1.3 Subsequence – Configure the 3DES Algorithm Type

This following sequence details the steps to configure the 3DES algorithm type settings (see [Table 14-5](#)).

1. Load key 1 LSW.
2. Load key 1 MSW.
3. Load key 2 LSW.
4. Load key 2 MSW.
5. Load key 3 LSW.
6. Load key 3 MSW.
7. Select 3DES algorithm.

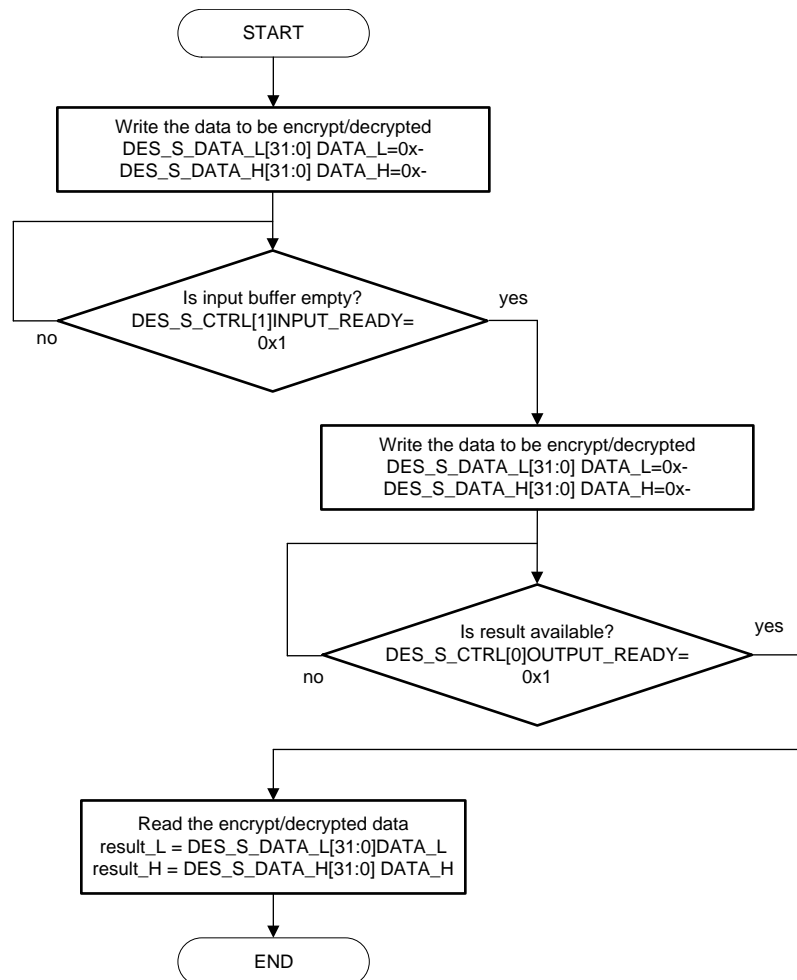
Table 14-5. 3DES Algorithm Type Configuration

Step	Register/Bit Field / Programming Model	Value
1	DES_KEY1_L[31:0] KEY1_L	-
2	DES_KEY1_H[31:0] KEY1_H	-
3	DES_KEY2_L[31:0] KEY2_L	-
4	DES_KEY2_H[31:0] KEY2_H	-
5	DES_KEY3_L[31:0] KEY3_L	-
6	DES_KEY3_H[31:0] KEY3_H	-
7	DES_CTRL[3] TDES	0x1

14.6.2 Operational Modes Configuration

14.6.2.1 Main Sequence – DES Polling Mode

[Figure 14-5](#) shows DES polling mode. The registers used in DES polling mode are: DES_DATA_L, DES_DATA_H, and DES_CTRL.


Figure 14-5. DES Polling Mode

14.6.2.2 DES Interrupt Mode

The following lists the DES interrupt mode steps (see [Table 14-6](#)).

1. Enable DES module interrupts.
2. Load the input buffer data LSW register.
3. Load the input buffer data HSW register.

Table 14-6. DES Interrupt Mode

Step	Register / Bit Field / Programming Model	Value
1	DES_IRQENABLE[2:0]	0x7
2	DES_DATA_L[31:0] DATA_L	-
3	DES_DATA_H[31:0] DATA_H	-

14.6.2.3 DES Interrupt DMA Mode

The following lists the DES DMA mode steps (see [Table 14-7](#)).

1. Enable DES module DMA requests.
2. Load the input buffer data LSW register.
3. Load the input buffer data HSW register.

Table 14-7. DES DMA Mode

Step	Register / Bit Field / Programming Model	Value
1	DES_SYSCONFIG[7:5]	0x7
2	DES_DATA_L[31:0] DATA_L	-
3	DES_DATA_H[31:0] DATA_H	-

14.6.3 DES Events Servicing

14.6.3.1 Interrupt Servicing

This section describes the event servicing of the module. [Figure 14-6](#) shows the DES interrupt service. The registers used during event servicing are: DES_IRQSTATUS, DES_DATA_L, and DES_DATA_H.

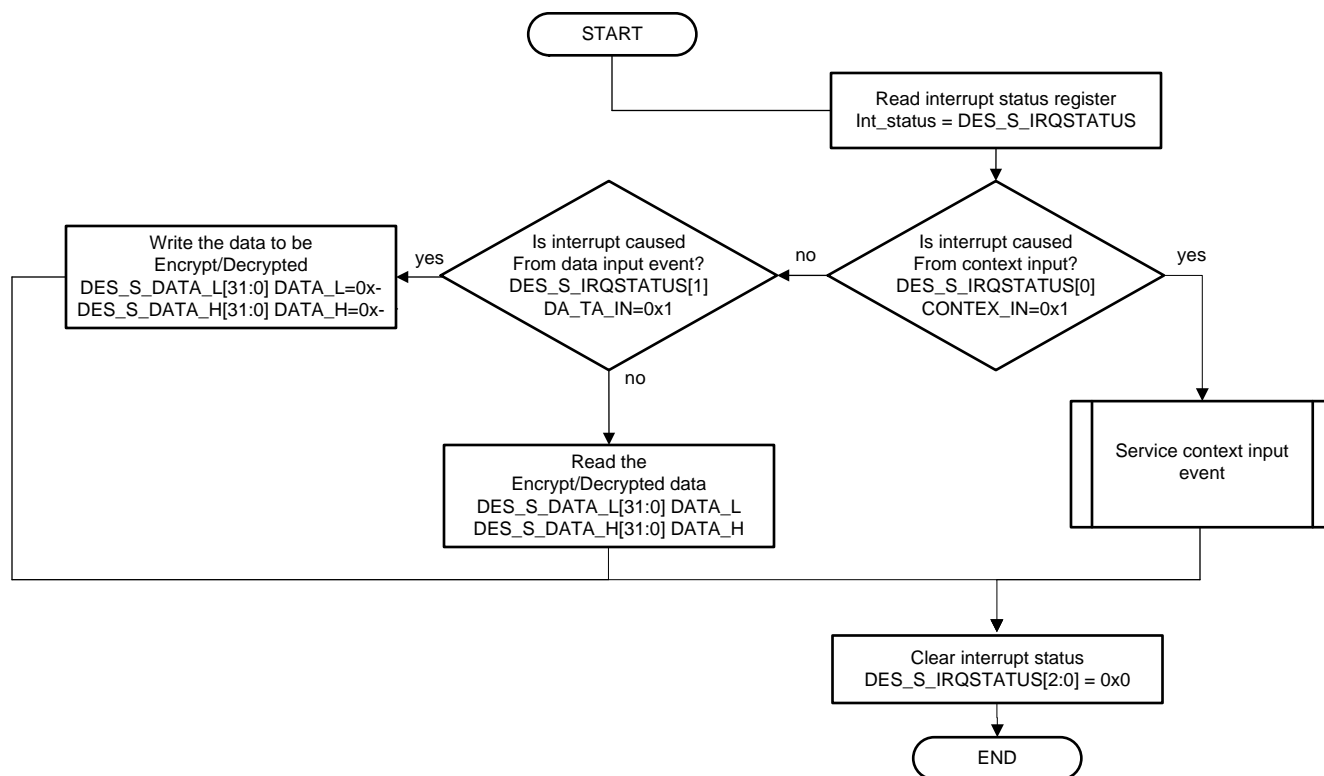


Figure 14-6. DES Interrupt Service

14.6.3.2 Context Input Event Servicing

This section describes the context input event servicing of the module, as shown in Figure 14-7. The registers used during event servicing are: DES_CTRL, DES_SYSCONFIG, DES_KEY1_L, DES_KEY1_H, DES_KEY2_L, DES_KEY2_H, DES_KEY3_L, DES_KEY3_H, DES_IV_L, DES_IV_H, and DES_LENGTH.

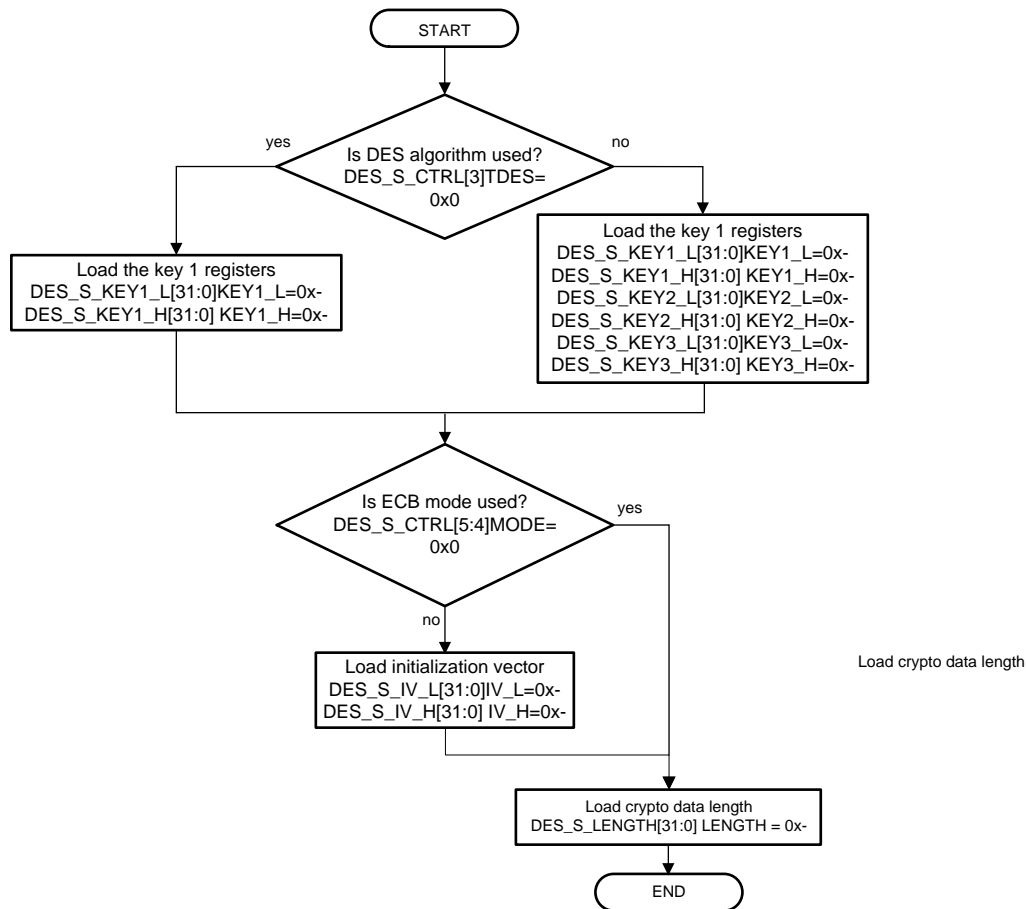


Figure 14-7. DES Context Input Event Service

14.7 DES Registers

The DES module registers are at an offset relative to the DES module base address, and a small set of DES μ DMA registers are at an offset relative to an Encryption Control module base address.

The DES module register offsets are relative to the base address 0x44038000.

The Encryption Control register offsets are relative to the base address 0x44030000.

NOTE: The DES registers are limited to 32-bit data accesses; 8- and 16-bit accesses are not allowed and can corrupt register contents.

Table 14-8 lists the memory-mapped registers for the DES. All register offset addresses not listed in Table 14-8 should be considered as reserved locations and the register contents should not be modified.

Table 14-8. DES Registers

Offset	Acronym	Register Name	Section
0x00	DES_KEY3_L	DES Key 3 LSW for 192-Bit Key	Section 14.7.1
0x04	DES_KEY3_H	DES Key 3 MSW for 192-Bit Key	Section 14.7.1
0x08	DES_KEY2_L	DES Key 2 LSW for 128-Bit Key	Section 14.7.1
0x0C	DES_KEY2_H	DES Key 2 MSW for 128-Bit Key	Section 14.7.1
0x10	DES_KEY1_L	DES Key 1 LSW for 64-Bit Key	Section 14.7.1
0x14	DES_KEY1_H	DES Key 1 MSW for 64-Bit Key	Section 14.7.1
0x18	DES_IV_L	DES Initialization Vector	Section 14.7.2
0x1C	DES_IV_H	DES Initialization Vector	Section 14.7.3
0x20	DES_CTRL	DES Control	Section 14.7.4
0x24	DES_LENGTH	DES Cryptographic Data Length	Section 14.7.5
0x28	DES_DATA_L	DES LSW Data RW	Section 14.7.6
0x2C	DES_DATA_H	DES MSW Data RW	Section 14.7.7
0x30	DES_REVISION	DES Revision Number	Section 14.7.8
0x34	DES_SYSCONFIG	DES System Configuration	Section 14.7.9
0x38	DES_SYSSTATUS	DES System Status	Section 14.7.10
0x3C	DES_IRQSTATUS	DES Interrupt Status	Section 14.7.11
0x40	DES_IRQENABLE	DES Interrupt Enable	Section 14.7.12
0x44	DES_DIRTYBITS	DES Dirty Bits	Section 14.7.13

Complex bit access types are encoded to fit into small table cells. Table 14-9 shows the codes that are used for access types in this section.

Table 14-9. DES Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

14.7.1 DES_KEYn_n Registers (Offset = 0x00 to 0x14) [reset = 0x0]

DES Key 3 LSW for 192-Bit Key (DES_KEY3_L), offset 0x00

DES Key 3 MSW for 192-Bit Key (DES_KEY3_H), offset 0x04

DES Key 2 LSW for 128-Bit Key (DES_KEY2_L), offset 0x08

DES Key 2 MSW for 128-Bit Key (DES_KEY2_H), offset 0x0C

DES Key 1 LSW for 64-Bit Key (DES_KEY1_L), offset 0x10

DES Key 1 MSW for 64-Bit Key (DES_KEY1_H), offset 0x14

DES_KEYn_n is shown in [Figure 14-8](#) and described in [Table 14-10](#).

Return to [Summary Table](#).

Figure 14-8. DES_KEYn_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																															
R/W-0x0																															

Table 14-10. DES_KEYn_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KEY	R/W	0x0	Key data

14.7.2 DES_IV_L Register (Offset = 0x18) [reset = 0x0]

DES Initialization Vector (DES_IV_L)

Least significant word of the initialization vector.

DES_IV_L is shown in [Figure 14-9](#) and described in [Table 14-11](#).

Return to [Summary Table](#).

Figure 14-9. DES_IV_L Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV_L																															
R/W-0x0																															

Table 14-11. DES_IV_L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IV_L	R/W	0x0	Initialization vector for CBC, CFB modes (LSW)

14.7.3 DES_IV_H Register (Offset = 0x1C) [reset = 0x0]

DES Initialization Vector (DES_IV_H)

Most significant word of the initialization vector.

DES_IV_H is shown in [Figure 14-10](#) and described in [Table 14-12](#).

Return to [Summary Table](#).

Figure 14-10. DES_IV_H Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV_H																															
R/W-0x0																															

Table 14-12. DES_IV_H Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IV_H	R/W	0x0	Initialization vector for CBC, CFB modes (MSW)

14.7.4 DES_CTRL Register (Offset = 0x20) [reset = 0x80000000]

DES Control (DES_CTRL)

DES_CTRL is shown in [Figure 14-11](#) and described in [Table 14-13](#).

Return to [Summary Table](#).

Figure 14-11. DES_CTRL Register

31	30	29	28	27	26	25	24
CONTEXT	RESERVED						
R-0x1	R-0x0						
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		MODE		TDES	DIRECTION	INPUT_READY	OUTPUT_READY
R-0x0		R/W-0x0		R/W-0x0	R/W-0x0	R-0x0	R-0x0

Table 14-13. DES_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CONTEXT	R	0x1	If 1, this read-only status bit indicates that the context data registers can be overwritten and the host is permitted to write the next context.
30-6	RESERVED	R	0x0	
5-4	MODE	R/W	0x0	Select CBC, ECB or CFB mode 0x0 = ECB mode 0x1 = CBC mode 0x2 = CFB mode 0x3 = reserved
3	TDES	R/W	0x0	Select DES or triple DES encryption/decryption. 0x0 = DES mode 0x1 = TDESmode
2	DIRECTION	R/W	0x0	Select encryption/decryption 0x0 = decryption is selected 0x1 = Encryption is selected
1	INPUT_READY	R	0x0	When 1, ready to encrypt or decrypt data
0	OUTPUT_READY	R	0x0	When 1, Data decrypted/encrypted ready

14.7.5 DES_LENGTH Register (Offset = 0x24) [reset = 0x0]

DES Cryptographic Data Length (DES_LENGTH)

Indicates the cryptographic data length in bytes for all modes. Once processing is started with this context, this length decrements to zero. Data lengths up to (232 - 1) bytes are allowed. A write to this register triggers the engine to start using this context.

NOTE: A read of this register returns all zeros.

DES_LENGTH is shown in [Figure 14-12](#) and described in [Table 14-14](#).

Return to [Summary Table](#).

Figure 14-12. DES_LENGTH Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENGTH																															
R/W-0x0																															

Table 14-14. DES_LENGTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LENGTH	R/W	0x0	Cryptographic data length in bytes for all modes

14.7.6 DES_DATA_L Register (Offset = 0x28) [reset = 0x0]

DES LSW Data RW (DES_DATA_L)

Data register (LSW) to read or write encrypted or decrypted data.

DES_DATA_L is shown in [Figure 14-13](#) and described in [Table 14-15](#).

Return to [Summary Table](#).

Figure 14-13. DES_DATA_L Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_L																															
R/W-0x0																															

Table 14-15. DES_DATA_L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA_L	R/W	0x0	Data for encryption/decryption, LSW

14.7.7 DES_DATA_H Register (Offset = 0x2C) [reset = 0x0]

DES MSW Data RW (DES_DATA_H)

Data register (MSW) to read or write encrypted or decrypted data.

DES_DATA_H is shown in [Figure 14-14](#) and described in [Table 14-16](#).

Return to [Summary Table](#).

Figure 14-14. DES_DATA_H Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_H																															
R/W-0x0																															

Table 14-16. DES_DATA_H Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA_H	R/W	0x0	Data for encryption/decryption, MSW

14.7.8 DES_REVISION Register (Offset = 0x30) [reset = 0x21]

DES Revision Number (DES_REVISION)

DES_REVISION is shown in [Figure 14-15](#) and described in [Table 14-17](#).

Return to [Summary Table](#).

Figure 14-15. DES_REVISION Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															
R-0x21																															

Table 14-17. DES_REVISION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REVISION	R	0x21	Revision number

14.7.9 DES_SYSCONFIG Register (Offset = 0x34) [reset = 0x1]

DES System Configuration (DES_SYSCONFIG)

NOTE: After one operation has completed, the DES_SYSCONFIG register must be cleared and re-configured for the next operation to ensure proper DMA and data operation functionality.

DES_SYSCONFIG is shown in [Figure 14-16](#) and described in [Table 14-18](#).

Return to [Summary Table](#).

Figure 14-16. DES_SYSCONFIG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
DMA_REQ_CONTEXT_IN_EN	DMA_REQ_DATA_OUT_EN	DMA_REQ_DATA_IN_EN	RESERVED	SIDLE		SOFTRESET	reserved-1
R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0		R/W-0x0	R-0x1

Table 14-18. DES_SYSCONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	DMA_REQ_CONTEXT_IN_EN	R/W	0x0	DMA Request Context In Enable 0x0 = DMA disabled 0x1 = DMA enabled
6	DMA_REQ_DATA_OUT_EN	R/W	0x0	DMA Request Data Out Enable 0x0 = DMA disabled 0x1 = DMA enabled
5	DMA_REQ_DATA_IN_EN	R/W	0x0	DMA Request Data In Enable 0x0 = DMA disabled 0x1 = DMA enabled
4	RESERVED	R	0x0	
3-2	SIDLE	R/W	0x0	Sidle mode 0x0 = reserved
1	SOFTRESET	R/W	0x0	Soft reset 0x0 = No operation 0x1 = Start soft reset sequence
0	RESERVED	R	0x1	

14.7.10 DES_SYSSTATUS Register (Offset = 0x38) [reset = 0x1]

DES System Status (DES_SYSSTATUS)

DES_SYSSTATUS is shown in [Figure 14-17](#) and described in [Table 14-19](#).

Return to [Summary Table](#).

Figure 14-17. DES_SYSSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							RESETDONE
R-0x0							R-0x1

Table 14-19. DES_SYSSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	RESETDONE	R	0x1	Reset Done 0x0 = Reset is not complete 0x1 = Reset done

14.7.11 DES_IRQSTATUS Register (Offset = 0x3C) [reset = 0x0]

DES Interrupt Status (DES_IRQSTATUS)

This register indicates the interrupt status. If one of the interrupt bits is set the interrupt output will be asserted.

DES_IRQSTATUS is shown in [Figure 14-18](#) and described in [Table 14-20](#).

Return to [Summary Table](#).

Figure 14-18. DES_IRQSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DATA_OUT	DATA_IN	CONTEX_IN
R-0x0					R-0x0	R-0x0	R-0x0

Table 14-20. DES_IRQSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DATA_OUT	R	0x0	This bit indicates data output interrupt is active and triggers the interrupt output.
1	DATA_IN	R	0x0	This bit indicates data input interrupt is active and triggers the interrupt output.
0	CONTEX_IN	R	0x0	This bit indicates context interrupt is active and triggers the interrupt output.

14.7.12 DES_IRQENABLE Register (Offset = 0x40) [reset = 0x0]

DES Interrupt Enable (DES_IRQENABLE)

This register contains an enable bit for each unique interrupt generated by the module. It matches the layout of DES_IRQSTATUS register. An interrupt is enabled when the bit in this register is set to 1.

DES_IRQENABLE is shown in [Figure 14-19](#) and described in [Table 14-21](#).

Return to [Summary Table](#).

Figure 14-19. DES_IRQENABLE Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					M_DATA_OUT	M_DATA_IN	M_CONTEX_IN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 14-21. DES_IRQENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	M_DATA_OUT	R/W	0x0	If this bit is set to 1 the data output interrupt is enabled.
1	M_DATA_IN	R/W	0x0	If this bit is set to 1 the data input interrupt is enabled.
0	M_CONTEX_IN	R/W	0x0	If this bit is set to 1 the context interrupt is enabled.

14.7.13 DES_DIRTYBITS Register (Offset = 0x44) [reset = 0x0]

DES Dirty Bits (DES_DIRTYBITS)

DES_DIRTYBITS is shown in [Figure 14-20](#) and described in [Table 14-22](#).

Return to [Summary Table](#).

Figure 14-20. DES_DIRTYBITS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						S_DIRTY	S_ACCESS
R-0x0						R/W1C-0x0	R/W1C-0x0

Table 14-22. DES_DIRTYBITS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	S_DIRTY	R/W1C	0x0	This bit is set to 1 by the module if any of the DES_* registers is written. Except DES_DIRTYBITS and DES_LOCKDOWN.
0	S_ACCESS	R/W1C	0x0	This bit is set to 1 by the module if any of the DES_* registers is read. Except DES_DIRTYBITS and DES_LOCKDOWN.

14.8 DES μ DMA Registers

This section lists and describes the DES μ DMA registers, in numerical order by address offset. Registers in this section are relative to the base address of 0x44030000.

Table 14-23 lists the memory-mapped registers for the DES_UDMA. All register offset addresses not listed in Table 14-23 should be considered as reserved locations and the register contents should not be modified.

Table 14-23. DES μ DMA Registers

Offset	Acronym	Register Name	Section
0x30	DES_DMAIM	DES DMA Interrupt Mask	Section 14.8.1
0x34	DES_DMARIS	DES DMA Raw Interrupt Status	Section 14.8.2
0x38	DES_DMAMIS	DES DMA Masked Interrupt Status	Section 14.8.3
0x3C	DES_DMAIC	DES DMA Interrupt Clear	Section 14.8.4

Complex bit access types are encoded to fit into small table cells. Table 14-24 shows the codes that are used for access types in this section.

Table 14-24. DES μ DMA Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

14.8.1 DES_DMAIM Register (Offset = 0x30) [reset = 0x0]

DES DMA Interrupt Mask (DES_DMAIM)

The DES DMA Interrupt Mask register control interrupt behavior and are used to program which interrupts are suppressed.

DES_DMAIM is shown in [Figure 14-21](#) and described in [Table 14-25](#).

Return to [Summary Table](#).

Figure 14-21. DES_DMAIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DOUT	DIN	CIN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 14-25. DES_DMAIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DOUT	R/W	0x0	<p>Data Out DMA Done Interrupt Mask</p> <p>If this bit is unmasked, an interrupt is generated when the μDMA writes the last word of the process result.</p> <p>0x0 = The DOUT interrupt is suppressed and not sent to the interrupt controller.</p> <p>0x1 = The DOUT interrupt is sent to the interrupt controller.</p>
1	DIN	R/W	0x0	<p>Data In DMA Done Interrupt Mask</p> <p>If this bit is unmasked, an interrupt is generated when the μDMA writes the last word of input data to the internal FIFO of the engine.</p> <p>0x0 = The DIN interrupt is suppressed and not sent to the interrupt controller.</p> <p>0x1 = The DIN interrupt is sent to the interrupt controller.</p>
0	CIN	R/W	0x0	<p>Context In DMA Done Interrupt Mask</p> <p>If this bit is unmasked, an interrupt is generated when the μDMA completes a context write to the internal register.</p> <p>0x0 = The CIN interrupt is suppressed and not sent to the interrupt controller.</p> <p>0x1 = The CIN interrupt is sent to the interrupt controller.</p>

14.8.2 DES_DMARIS Register (Offset = 0x34) [reset = 0x0]

DES DMA Raw Interrupt Status (DES_DMARIS)

The DES DMA Raw Interrupt Status register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set to 1.

DES_DMARIS is shown in [Figure 14-22](#) and described in [Table 14-26](#).

Return to [Summary Table](#).

Figure 14-22. DES_DMARIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DOUT	DIN	CIN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 14-26. DES_DMARIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DOUT	R/W	0x0	Data Out DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has written the last word of the process result and an interrupt has been triggered and is pending.
1	DIN	R/W	0x0	Data In DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has written the last word of input data to the internal FIFO of the engine and an interrupt has been triggered and is pending.
0	CIN	R/W	0x0	Context In DMA Done Raw Interrupt Status 0x0 = No interrupt. 0x1 = The μ DMA has completed a context write to the internal register and an interrupt has been triggered and is pending.

14.8.3 DES_DMAMIS Register (Offset = 0x38) [reset = 0x0]

DES DMA Masked Interrupt Status (DES_DMAMIS)

The DES DMA Masked Interrupt Status register displays the raw interrupts that are unmasked in the DES DMA Raw Interrupt Status (DES_DMARIS) register.

DES_DMAMIS is shown in [Figure 14-23](#) and described in [Table 14-27](#).

Return to [Summary Table](#).

Figure 14-23. DES_DMAMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DOUT	DIN	CIN
R-0x0					R-0x0	R-0x0	R-0x0

Table 14-27. DES_DMAMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DOUT	R	0x0	Data Out DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A DOUT interrupt has occurred.
1	DIN	R	0x0	Data In DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A DIN interrupt has occurred.
0	CIN	R	0x0	Context In DMA Done Raw Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A CIN interrupt has occurred.

14.8.4 DES_DMAIC Register (Offset = 0x3C) [reset = 0x0]

DES DMA Interrupt Clear (DES_DMAIC)

The DES DMA Interrupt Clear register is used to clear the DES_DMARIS and DES_DMAMIS registers by writing a 1 to the corresponding register bit.

NOTE: This registers always reads as zero.

DES_DMAIC is shown in [Figure 14-24](#) and described in [Table 14-28](#).

Return to [Summary Table](#).

Figure 14-24. DES_DMAIC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DOUT	DIN	CIN
R-0x0					W1C-0x0	W1C-0x0	W1C-0x0

Table 14-28. DES_DMAIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DOUT	W1C	0x0	Data Out DMA Done Interrupt Clear Writing a 1 to this bit clears the DOUT bit in the DES_DMARIS and DES_DMAMIS register.
1	DIN	W1C	0x0	Data In DMA Done Interrupt Clear Writing a 1 to this bit clears the DIN bit in the DES_DMARIS and DES_DMAMIS register.
0	CIN	W1C	0x0	Context In DMA Done Raw Interrupt Status Writing a 1 to this bit clears the CIN bit in the DES_DMARIS and DES_DMAMIS register.

Ethernet Controller

This chapter describes the Ethernet Controller module.

Topic	Page
15.1 Introduction	883
15.2 Block Diagram.....	884
15.3 Functional Description	884
15.4 Ethernet PHY	929
15.5 Initialization and Configuration	935
15.6 EMAC Registers	939
15.7 MII Management (EPHY) Registers	1045

15.1 Introduction

The Ethernet Controller has the following features:

- Conforms to the IEEE 802.3 specification
 - 10BASE-T/100BASE-TX IEEE 802.3 compliant
 - Supports 10/100-Mbps data transmission rates
 - Supports full-duplex and half-duplex (CSMA/CD) operation
 - Supports flow control and back pressure
 - Full-featured and enhanced auto-negotiation
 - Supports IEEE 802.1Q VLAN tag detection
- Conforms to IEEE 1588-2002 Timestamp Precision Time Protocol (PTP) and IEEE 1588-2008 Advanced Timestamp specification
 - Transmit and receive frame time stamping
 - PTP
 - Flexible pulse per second output
 - Supports coarse and fine correction methods
- Multiple addressing modes
 - Four MAC address filters
 - Programmable 64-bit Hash filter for multicast address filtering
 - Promiscuous mode support
- Processor offloading
 - Programmable insertion (TX) or deletion (RX) of preamble and start-of-frame data
 - Programmable generation (TX) or deletion (RX) of CRC and pad data
 - IP header and hardware checksum checking (IPv4, IPv6, TCP/UDP/ICMP)
- Highly configurable
 - LED activity selection
 - Supports network statistics with RMON/MIB counters
 - Supports Magic Packet and wakeup frames
- Efficient transfers using integrated Direct Memory Access (DMA)
 - Dual-buffer (ring) or linked-list (chained) descriptors
 - Round-robin or fixed priority arbitration between TX/RX
 - Descriptors support up to 8-kB transfer blocks size
 - Programmable interrupts for flexible system implementation
- Physical media manipulation
 - MDI/MDI-X cross-over support
 - Register-programmable transmit amplitude
 - Automatic polarity correction and 10BASE-T signal reception
- MII and RMI interface support

15.2 Block Diagram

Figure 15-1 shows the block diagram of the Ethernet MAC with an integrated PHY interface.

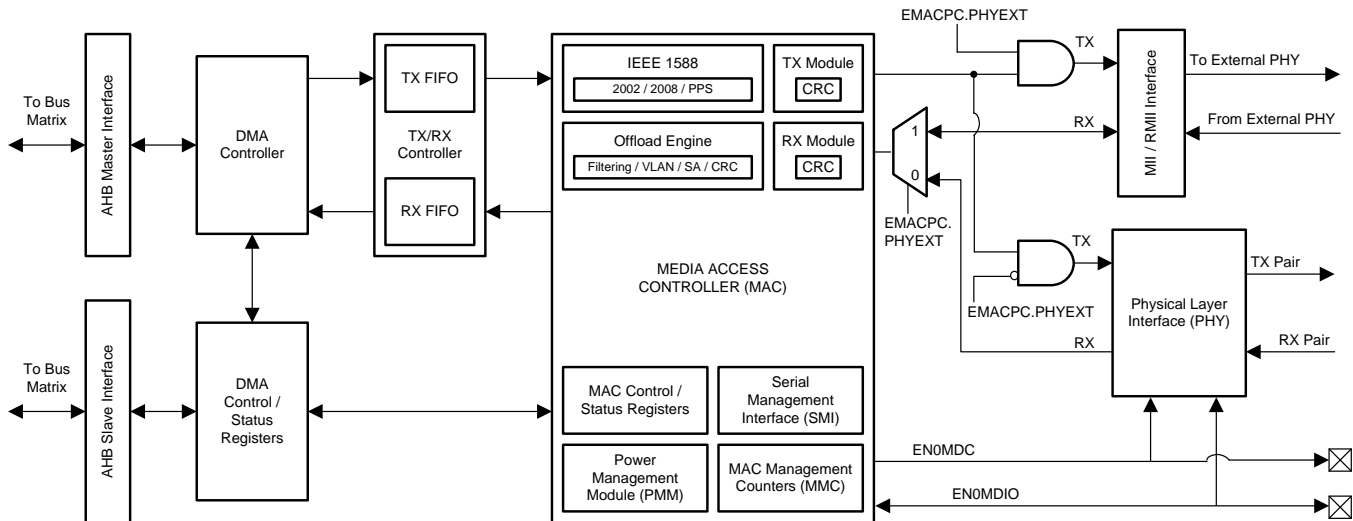


Figure 15-1. Ethernet MAC With Integrated PHY Interface

15.3 Functional Description

The Ethernet Controller comprises the following submodules:

- Clock Control
- Media-Independent Interface (MII) / Reduced Media-Independent Interface (RMII) Module
- DMA Controller
- Transmit/Receive Controller (TX/RX Controller)
- Media Access Controller (MAC)
- AHB Bus Interface
- PHY Interface

The following sections describe the features and functions of each submodule.

15.3.1 Ethernet Clock Control

Available clock sources are dependent on the interface chosen. The following sections describe the clock control for the various interfaces.

15.3.1.1 PHY Interface

The Ethernet Controller module and Integrated PHY receive two clock inputs, as follows:

- A gated system clock acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SYSCLK frequency for run, sleep, and deep sleep modes is programmed into the System Control module. See [Chapter 4](#) for more information on programming SYSCLK and enabling the Ethernet MAC.
- The PHY receives the main oscillator (MOSC) which must be 25 MHz \pm 50 ppm for proper operation. The MOSC source can be a single-ended source or a crystal. [Figure 15-2](#) shows the clock inputs to the Ethernet Controller module.

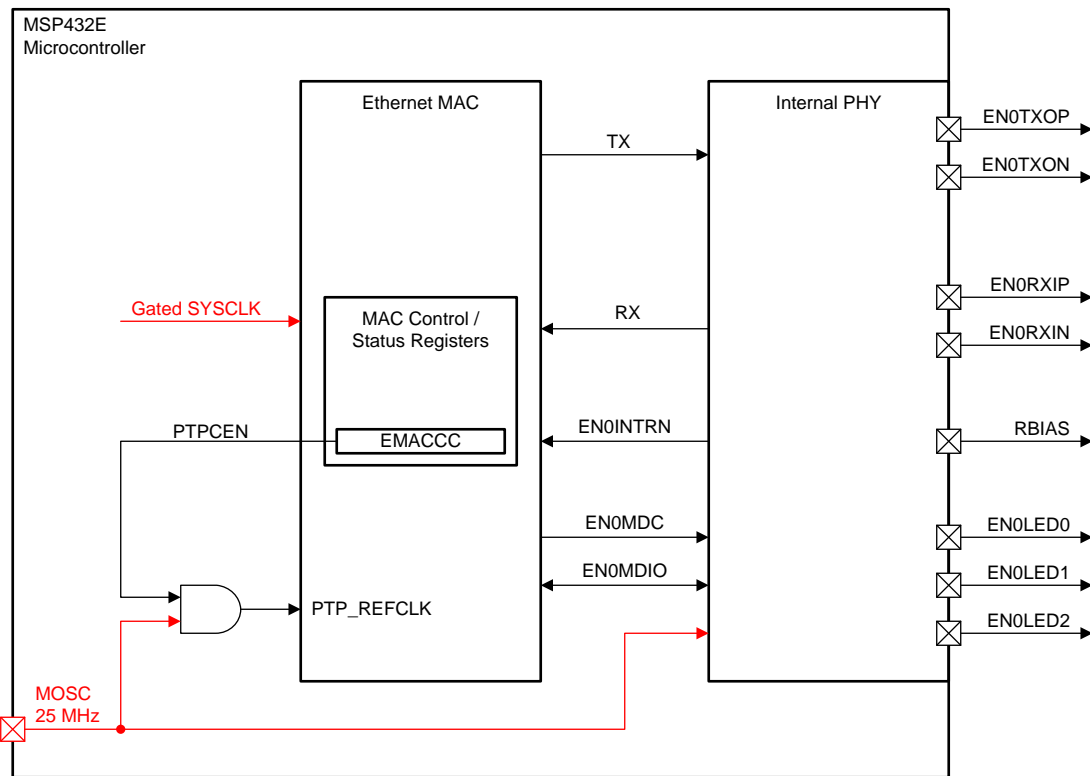


Figure 15-2. Ethernet MAC and PHY Clock Structure

15.3.1.2 Media-Independent Interface (MII)

Four clock inputs are driven into the Ethernet MAC when the MII configuration is enabled. The clocks are described as follows:

- Gated system clock (SYSCLK): The SYSCLK signal acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SYSCLK frequency for Run, Sleep, and Deep Sleep mode is programmed in the System Control module. See [Chapter 4](#) for more information on programming SYSCLK and enabling the Ethernet MAC.
- MOSC: A gated version of the MOSC clock is provided as the PTP reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the PTP module has been enabled by setting the PTPCEN bit in the EMACCC register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. See [Section 15.3.6](#) for more information.
- EN0RXCK: This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz, depending on whether the device is operating at 10 Mbps or 100 Mbps.
- EN0TXCK: This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz, depending on whether the device is operating at 10 Mbps or 100 Mbps.

Figure 15-3 shows the clock inputs for an MII.

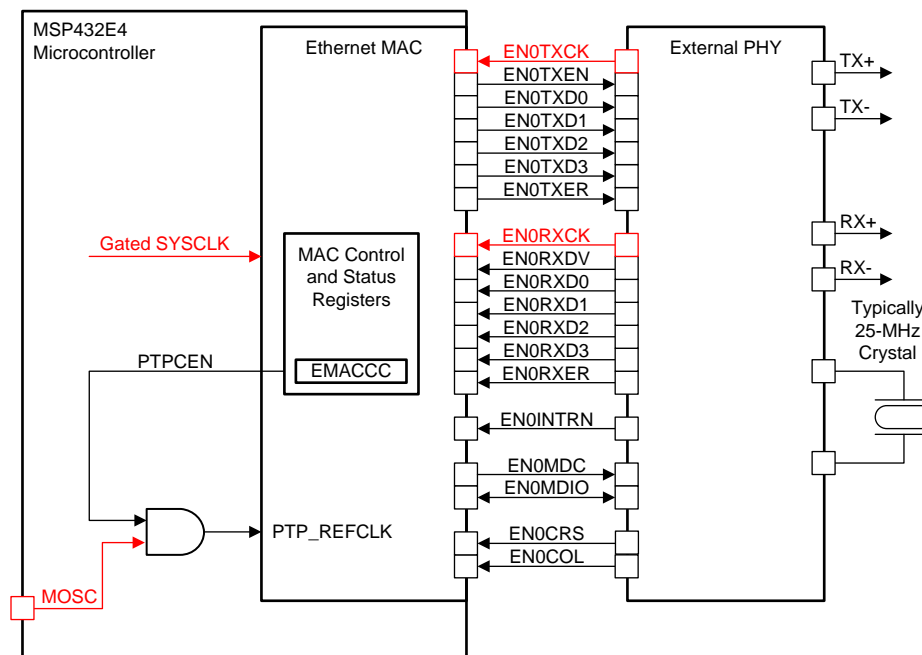


Figure 15-3. MII Clock Structure

15.3.1.3 Reduced Media-Independent Interface (RMII)

Three clock sources interface to the Ethernet MAC in an RMII configuration, as follows:

- **Gated system clock (SYSCLK):** The SYSCLK signal acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SYSCLK frequency for Run, Sleep, and Deep Sleep mode is programmed in the System Control module. See [Chapter 4](#) for more information on programming SYSCLK and enabling the Ethernet MAC.
- **MOSC:** A gated version of the MOSC clock is provided as the Precision Time Protocol (PTP) reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the PTP module has been enabled by setting the PTPCEN bit in the EMACCC register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. See [Section 15.3.6](#) for more information.
- **EN0RREF_CLK:** When using RMII, a 50-MHz external reference clock must drive the EN0RREF_CLK input signal and the external PHY. Depending on the configuration of the FES bit in the Ethernet MAC Configuration (EMACCFG) register, the reference clock input (EN0RREF_CLK) is divided by 20 for 10 Mbps, or 2 for 100 Mbps operation, and used as the clock for receive and transmit data.

[Figure 15-4](#) shows the clock inputs to the RMII clock.

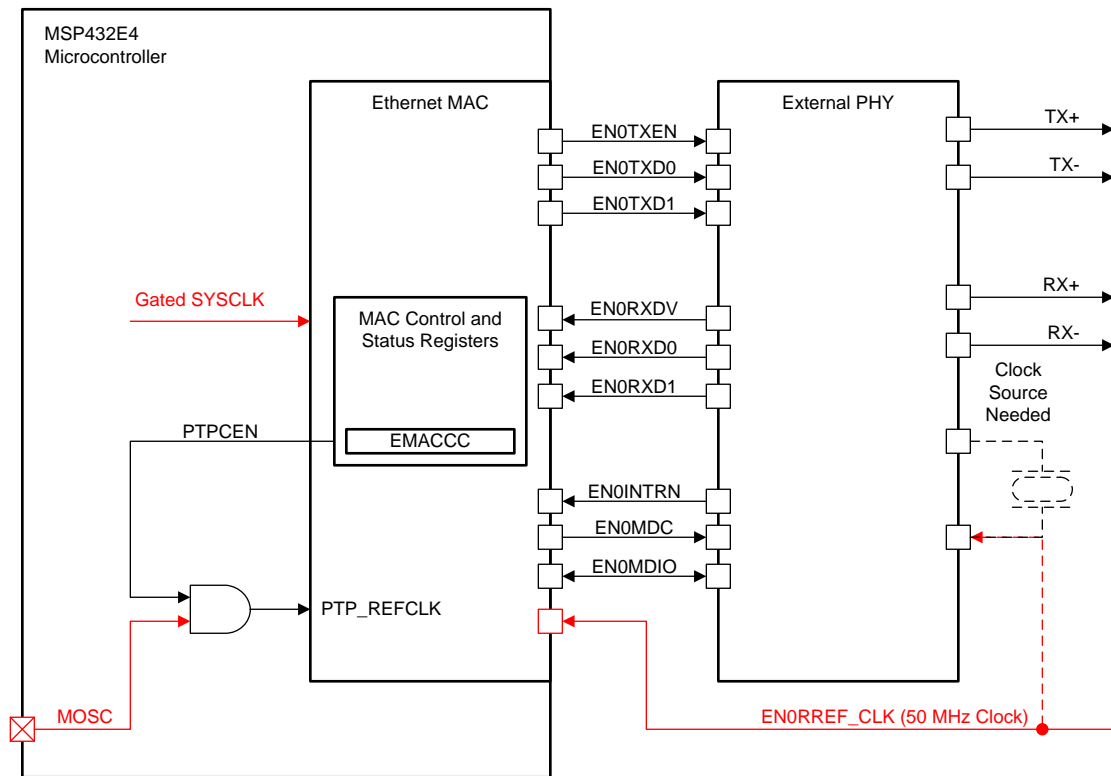


Figure 15-4. RMII Clock Structure

15.3.2 MII and RMII Signals

The MAC Module has the capability of providing an MII or RMII depending on the interface selected in the Ethernet MAC Peripheral Configuration (EMACPC) register, at offset 0xFC4. Except for EN0RREF_CLK, the signals used for RMII mode are a subset of the MII signals. Therefore, [Table 15-1](#) lists the MII signals that are used in RMII mode and the RMII function to which they correspond.

Table 15-1. MII and RMII Signals

MII Signal	RMII Signal	RMII Standard Name and Function
N/A	EN0RREF_CLK	REF_CLK: Synchronous clock reference for receive, transmit and control
EN0TXCK	Not used	N/A
EN0TXD3	Not used	N/A
EN0TXD2	Not used	N/A
EN0TXD1	EN0TXD1	TXD1: Transmit Data 1
EN0TXD0	EN0TXD0	TXD0: Transmit Data 0
EN0TXEN	EN0TXEN	TEX_EN: Transmit Enable
EN0TXER	Not used	N/A
EN0RXCK	Not used	N/A
EN0RXD3	Not used	N/A
EN0RXD2	Not used	N/A
EN0RXD1	EN0RXD1	RXD1: Receive Data 1
EN0RXD0	EN0RXD0	RXD0: Receive Data 0
EN0RXDV	EN0RXDV	CRS_DV: Carrier Sense/Receive Data Valid
EN0RXER	Not used	RX_ER: Receive Error
EN0COL	Not used	N/A

Table 15-1. MII and RMII Signals (continued)

MI Signal	RMII Signal	RMII Standard Name and Function
EN0CRS	Not used	N/A
EN0MDC	EN0MDC	MDC: Management Data Clock
EN0MDIO	EN0MDIO	MDIO: Management Data Input/Output
EN0PPS	EN0PPS	Pulse-Per-Second (PPS) Output (optional, this is not a standard RMII signal)
EN0INTRN	EN0INTRN	Interrupt to Ethernet PHY (optional, this is not a standard RMII signal)

15.3.3 DMA Controller

The integrated DMA of the Ethernet Controller is used to optimize data transfer between the MAC and system SRAM. The DMA has independent transmit and receive engines.

The DMA transmit engine transfers data from system memory to the Ethernet TX/RX Controller, while the receive engine transfers data from the RX FIFO to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. Fixed burst lengths of 1, 4, 8, or 16 words are supported with reinitiation of bursts when retry or burst termination responses occur. For a burst retry, if the remaining address count is greater than 1 and the RIB bit in the Ethernet MAC DMA Bus Mode (EMACDMABUSMOD) register is clear, then the transfer resends data in one continuous burst. When one transfer is left, it is done as a single burst and the transaction is terminated immediately afterward. If the RIB bit in the EMACDMABUSMOD register is set, the DMA sends the remaining data in fixed burst sizes of 1, 4, 8, or 16 words.

The application may also choose between solely fixed bursts or mixed bursts by the DMA. If the MB bit is set and the FB bit is clear in the EMACDMABUSMOD register, then the DMA uses fixed bursts for burst sizes less than 16 and a full, non-divided burst for lengths greater than 16. Fixed burst lengths allow for more DMA bus arbitration with other masters. Maximum burst transfer lengths can be programmed for both the receive and transmit channels of the DMA through the PBL, RPBL, and 8xPBL bit fields in the EMACDMABUSMOD register.

The DMA Controller requests a read transfer only when it can accept the received burst data completely. Data read from the bus is always pushed into the DMA without any delay or busy cycles. The DMA requests write transfers only when it has sufficient data to transfer the burst completely. When operating in fixed burst length mode, the DMA interface continues to burst with dummy data until the specified length is completed. The Ethernet controller can be programmed to interrupt the CPU in situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The integrated Ethernet DMA communicates through two data structures:

- Control and Status registers
- Descriptor lists and data buffers

The DMA writes data frames received by the MAC to the receive buffer in system memory and transfers data frames for transmission from system memory to the MAC. Descriptors that reside in the system memory act as pointers to these buffers.

There are two descriptor lists: one for reception and one for transmission. The base address of each list is written into the Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR) register at offset 0xC0C and the Ethernet Mac Transmit Descriptor List Address (EMACTXDLADDR) register at offset 0xC10, respectively.

The descriptor structure can contain up to 8 words (32 bytes). These are described in more detail in [Section 15.3.3.5](#). A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by enabling second address chaining in both the Receive and Transmit descriptors (RDES0[14] and TDES0[20]). The descriptor lists reside in the SRAM address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used at different physical addresses rather than contiguous buffers in memory.

The data buffer also resides in the physical memory space and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data and buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when the end-of-frame is detected. Data chaining can be enabled or disabled through the descriptors.

NOTE: The EMAC DMA Controller only has access to internal system SRAM.

15.3.3.1 Burst Access

The DMA attempts to execute fixed length Burst transfers if the FB bit is set in the EMACDMABUSMOD register. The maximum burst length is indicated and limited by the PBL field in the EMACDMABUSMOD register. The Receive and Transmit descriptors are always accessed in the maximum possible (limited by PBL) burst-size for the bytes to be read.

The TX DMA initiates a transfer only when there is sufficient space in the FIFO to accommodate the configured burst or remaining bytes of the end of a frame. When the DMA is configured for fixed-length burst, it transfers data using the best combination of fixed burst sizes of 4, 8, or 16 and single transactions. Otherwise when the FB bit is clear in the EMACDMABUSMOD register, the DMA transfers data as a continuous undefined burst and single transactions.

The RX DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in RX FIFO or when the end-of-frame (when it is less than the configured burst length) is detected in the RX FIFO. The DMA indicates the start address and the number of transfers required to the system. When the FB bit is set in the EMACDMABUSMOD register, then it transfers data using the best combination of fixed burst sizes of 4, 8, or 16 and single transactions. If the end-of frame is reached before the fixed-burst ends, then dummy transfers are performed in order to complete the fixed-burst. Otherwise, if the FB bit is clear, the DMA transfers data using INCR (undefined length) and SINGLE transactions. When the DMA is configured for address-aligned transfers, both DMA engines ensure that the first burst transfer on the system bus is less than or equal to the size of the configured PBL in the EMACDMABUSMOD register. Thus, all subsequent transfers start at an address that is aligned to the configured PBL. The DMA can only align the address for burst transfers up to size 16 because only bursts of 16 are supported.

15.3.3.2 Data Buffer Alignment

The transmit and receive data buffers do not have any restrictions on the start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates write transfers, with address aligned to the bus width and dummy data (old data) in the byte lanes that are not valid. This typically happens during the transfer of the beginning or end of an Ethernet frame. The software driver should discard the dummy bytes based on the start address of the buffer and size of the frame.

For example, if the transmit buffer address is 0x0000.0FF2, and 15 bytes need to be transferred, then the DMA reads five full words from address 0x0000.0FF0, but when transferring data to the TX FIFO, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures that it transfers a full 32-bit data to the TX FIFO, unless it is the end of frame.

If the receive buffer address is 0x0000.0FF2 and 15 bytes of a received frame need to be transferred, then the DMA writes five full words from address 0x0000.0FF0. However, the first two bytes of first transfer and the last three bytes of the fifth transfer have dummy data. The DMA considers the offset address only if it is the first Receive buffer of the frame. The DMA ignores the offset address and performs full word writes for the middle and the last Receive buffer of the frame.

15.3.3.3 Buffer Size Calculations

The DMA does not update the size fields in the Transmit and Receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations. The TX DMA transfers the exact number of bytes (indicated by buffer size fields of TDES1) to the MAC. If a descriptor is marked as first (FS bit of TDES0 is set), then the DMA marks the first transfer from the buffer as the start-of-frame (SOF). If a descriptor is marked as last (LS bit of TDES0), then the DMA marks the last transfer from that data buffer as the end-of frame (EOF).

The RX DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the RX/TX Controller. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffers are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.

NOTE: Even when the start address of a receive buffer is not aligned to the data width of 32-bit system bus, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1024-byte (1KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the Receive descriptor to have a 0x1002 offset. The Receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1022 bytes, even though the buffer size is programmed as 1024 bytes, because of the start address offset.

15.3.3.4 DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Transmit and Receive channel accesses. The DMA can be configured to arbitrate in a round-robin or fixed-priority configuration. When the DA bit of the EMACDMABUSMOD register is clear, the DMA arbiter allocates the data bus in the ratio set by the PR bit field of the EMACDMABUSMOD register when both the TX and RX DMA request access at the same time. When the DA bit is set, the RX DMA always has priority over the TX DMA for data access by default. However if the TXPR bit of the EMACDMABUSMOD register is also set, then the TX DMA always gets priority over the RX DMA.

15.3.3.5 Enhanced and Alternate Descriptors

Enhanced Descriptors can contain up to eight words (32 bytes) and buffers of up to 8KB (useful for Jumbo frames). Enhanced Descriptors support IEEE 1588-2008 Advanced Timestamp and IPC Full Checksum (Type 2) Offload. These enhanced features are enabled with the TSEN bit of the EMACTIMSTCTRL register and the IPC bit of the EMACCFG register, respectively.

When using Enhanced Descriptors, set the descriptor size to eight words with the Alternate Descriptor Size (ATDS) bit in the Ethernet MAC DMA Bus Mode (EMACDMABUSMOD) register. If these enhanced features are not enabled, the extended descriptors (DES4 to DES7) are not required. Therefore, the software can use Alternate Descriptors with a default size of 16 bytes (4 words). For alternate descriptors, the software should clear the Alternate Descriptor Size (ATDS) bit in the Ethernet MAC DMA Bus Mode (EMACDMABUSMOD) register.

See [Section 15.3.3.5.1](#) and [Section 15.3.3.5.2](#) for more details on the descriptor structure.

15.3.3.5.1 Enhanced Transmit Descriptor

The MAC requires at least one descriptor for a transmit frame. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit frame basis. [Figure 15-5](#) shows the enhanced transmit descriptor. Software must program the control bits TDES0[31:18] during descriptor initialization. When the DMA updates the descriptor, it writes back all the control bits to their initialized value, clears the OWN bit and updates the status bits.

With advanced timestamp support, the snapshot of the timestamp to be taken can be enabled for a given frame by setting bit 25 (TTSE) of TDES0. When the descriptor is closed (that is, when the OWN bit is cleared), the timestamp is written into TDES6 and TDES7.

NOTE: When the Advanced Timestamp feature is enabled, software should set the ATDS bit of the Ethernet MAC DMA Bus Mode (EMACDMABUSMOD) register, offset 0xC00, so that the DMA operates with extended descriptor size. When this control bit is reset to the default (0), the TDES4-TDES7 descriptor space is not valid and only Alternate Descriptors are available, with a default size of 16 bytes (four words).

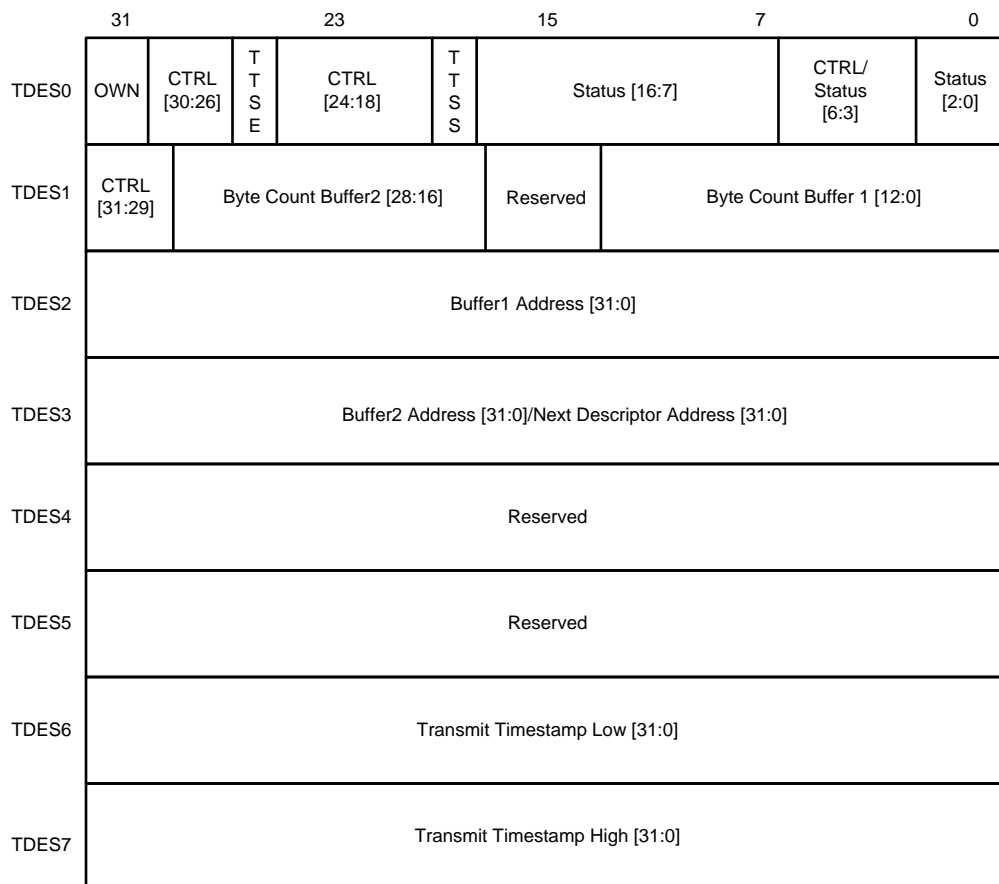


Figure 15-5. Enhanced Transmit Descriptor Structure

The following tables list the Enhanced Transmit Descriptors.

- Transmit Descriptor 0 (TDES0) contains the transmitted frame status and the descriptor ownership information (see [Table 15-2](#)).
- TDES1 contains the buffer sizes and other bits which control the descriptor chain or ring and the frame being transferred (see [Table 15-3](#)).
- TDES2 contains the address pointer to the first buffer of the descriptor (see [Table 15-4](#)).
- TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor

(see [Table 15-5](#)).

- TDES6 and TDES7 contain the timestamp (see [Table 15-6](#) and [Table 15-7](#)).

Table 15-2. Enhanced Transmit Descriptor 0 (TDES0)

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30	IC: Interrupt on Completion When set this bit sets the Transmit Interrupt (TI) bit in the EMACDMARIS register when the frame contained in this descriptor has been transmitted.
29	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1 or TBS2 field in TDES1 should have a non-zero value.
28	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
27	DC: Disable CRC When set, the MAC does not append a Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.
26	DP: Disable Padding When set, the MAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set.
25	TTSE: Transmit Timestamp Enable When set, this bit enables IEEE1588 hardware timestamping for the transmit frame referenced by the descriptor. This bit is only valid when the First Segment Control bit (TDES0[28]) is set.
24	CRCR: CRC Replacement Control When set, the MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The CPU should ensure that the CRC bytes are present in the frame being transferred from the Transmit Buffer. CRC replacement is done only when Bit 27 (DC) is set to 1.
23:22	CIC: Checksum Insertion Control These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6. This field is valid when the First Segment control bit (TDES0[28]) is set. <ul style="list-style-type: none"> • 0x0 = Do nothing. Checksum Engine bypassed. • 0x1 = Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 0x2 = Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4. • 0x3 = Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. The TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.
21	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
20	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a "don't care" value. TDES0[21] takes precedence over TDES0[20].

Table 15-2. Enhanced Transmit Descriptor 0 (TDES0) (continued)

Bit	Description
19:18	<p>VLIC: VLAN Insertion Control</p> <p>When set, these bits request the MAC to perform VLAN tagging or untagging before transmitting the frames. If the frame is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes.</p> <p>The values of this field are as follows:</p> <ul style="list-style-type: none"> 0x0 = Do not add a VLAN tag 0x1 = Remove the VLAN tag from the frames before transmission. 0x2 = Insert a VLAN tag with the tag value programmed in the Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP) register, offset 0x584. 0x3 = Replace the VLAN tag in frame with the tag value programmed in the EMACVLNINCREP register. This field is valid when the First Segment control bit (TDES0[28]) is set.
17	<p>TTSS:TX Timestamp</p> <p>This status bit indicates that a timestamp has been captured for the corresponding transmit frame. When this bit is set, TDES6 and TDES7 have timestamp values that were captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES0[29]) in a descriptor is set.</p>
16	<p>IHE: IP Header Error</p> <p>When set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when TX Checksum Offload is enabled. Otherwise, it is reserved. If the Checksum Offload Engine detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.</p>
15	<p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> TDES0[16]: IP Header Error TDES0[14]: Jabber Timeout TDES0[13]: Frame Flush TDES0[12]: Payload Checksum Error TDES0[11]: Loss of Carrier TDES0[10]: No Carrier TDES0[9]: Late Collision TDES0[8]: Excessive Collision TDES0[2]: Excessive Deferral TDES0[1]: Underflow error
14	<p>JT: Jabber Timeout</p> <p>When set, this bit indicates that the MAC transmitter has experienced a jabber time-out. This bit can only be set when the Jabber Disabled (JD) bit of the EMACCFG register is clear.</p>
13	<p>FF: Frame Flushed</p> <p>When set, this bit indicates that the DMA flushed the frame because of a software flush command given by the CPU.</p>
12	<p>IPE: IP Payload Error</p> <p>When set, this bit indicates that MAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.</p>
11	<p>LC: Loss of Carrier</p> <p>When set, this bit indicates that Loss of Carrier occurred during frame transmission. This is valid only for the frames transmitted without collision and when the MAC operates in half-duplex mode.</p>
10	<p>NC: No Carrier</p> <p>When set, this bit indicates that the carrier sense signal from the PHY was not asserted during transmission.</p>
9	<p>LC: Late Collision</p> <p>When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in MII Mode). Not valid if Underflow Error (bit 1) is set.</p>
8	<p>Excessive Collision</p> <p>When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the Disable Retry (DR) bit in EMACCFG register is set, this bit is set after the first collision and the transmission of the frame is aborted.</p>
7	<p>VF: VLAN Frame</p> <p>When set, this bit indicates that the transmitted frame was a VLAN-type frame.</p>
6:3	<p>CC: Collision Count</p> <p>This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collision bit (TDES0[8]) is set.</p>

Table 15-2. Enhanced Transmit Descriptor 0 (TDES0) (continued)

Bit	Description
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24288 bit times (155680 bits times when Jumbo Frame is enabled). This bit is dependent on the Deferral Check (DC) bit being enabled in the EMACCFG register.
1	UF: Underflow Error When set, this bit indicates that the MAC aborted the frame because the data arrived late from system memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both Transmit Underflow (UNF) and Transmit Interrupt (TI) bit in the EMACDMARIS register.
0	DB: Deferred Bit This bit indicates the deferral mechanism is active and that the transmit state machine sends a JAM pattern to defer reception when it senses a carrier before a normal transmission is scheduled. This bit is only valid in half-duplex mode.

Table 15-3. Enhanced Transmit Descriptor 1 (TDES1)

Bit	Description
31:29	SAIC: SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet frame with the value given in the MAC Address 0 register. If the Source Address field is modified in a frame, the MAC automatically recalculates and replaces the CRC bytes. The Bit 31 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of bits [30:29]: <ul style="list-style-type: none"> 0x0 = Do not include the source address. 0x1 = Insert the source address. For reliable transmission, the application must provide frames without source addresses. 0x2 = Replace the source address. For reliable transmission, the application must provide frames with source addresses. 0x3 = Reserved These bits are valid when the First Segment control bit (TDES0[28]) is set.
28:16	TBS2: Transmit Buffer 2 Size This field indicates the second data buffer size in bytes. This field is not valid if TDES0[20] is set.
15:13	Reserved
12:0	TBS1: Transmit Buffer 1 Size These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]).

Table 15-4. Enhanced Transmit Descriptor 2 (TDES2)

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. Note that the buffers are stored in SRAM.

Table 15-5. Enhanced Transmit Descriptor 3 (TDES3)

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES0[20]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES0[20] is set. Note that the buffers are stored in SRAM.

Table 15-6. Enhanced Transmit Descriptor 6 (TDES6)

Bit	Description
31:0	TTSL: Transmit Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment bit (LS), TDES0[29], in the descriptor is set and Timestamp status (TTSS) bit, TDES0[17], is set.

Table 15-7. Enhanced Transmit Descriptor 7 (TDES7)

Bit	Description
31:0	TTSH: Transmit Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field has the timestamp only if the Last Segment bit (LS), TDES0[29], in the descriptor is set and Timestamp status (TTSS) bit, TDES0[17], is set.

15.3.3.5.2 Enhanced Receive Descriptor

The DMA requires at least two descriptors when receiving a frame. The DMA always attempts to acquire an extra descriptor in anticipation of an incoming frame. Before the DMA closes a descriptor, it attempts to acquire the next descriptor even if no frames are received. In a single descriptor (receive) system, the subsystem generates a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. [Figure 15-6](#) shows the enhanced receive descriptor. This descriptor is used when Advanced Timestamp or the Checksum Offload Engine is enabled.

NOTE: When the Advanced Timestamp or Checksum Offload Engine features are enabled, software should set the ATDS bit of the Ethernet MAC DMA Bus Mode (EMACDMABUSMOD) register, offset 0xC00, so that the DMA operates with extended descriptor size. When this control bit is reset to the default (0), the TDES4-TDES7 descriptor space is not valid and only Alternate Descriptors are available, with a default size of 16 bytes (4 words).



Table 15-8. Enhanced Receive Descriptor 0 (RDES0)

Copyright © 2017–2018, Texas Instruments Incorporated

Table 15-8. Enhanced Receive Descriptor 0 (RDES0) (continued)

Bit	Description
15	<p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • RDES0[14]: Descriptor Error • RDES0[11]: Overflow Error • RDES0[7]: IPC Checksum (Type 2) or Giant Frame • RDES0[6]: Late Collision • RDES0[4]: Watchdog Timeout • RDES0[3]: Receive Error • RDES0[1]: CRC Error • RDES[4:3]: IP Header or Payload Error <p>This field is valid only when the Last Descriptor (RDES0[8]) is set.</p>
14	<p>DE: Descriptor Error</p> <p>When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.</p>
13	<p>SAF: Source Address Filter Fail</p> <p>When set, this bit indicates that the SA field of frame failed the SA Filter in the MAC.</p>
12	<p>LE: Length Error</p> <p>When set, this bit indicates that the actual length of the frame received and the Length/Type field do not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. Length error status is not valid when CRC error is present.</p>
11	<p>OE: Overflow Error</p> <p>When set, this bit indicates that the received frame is damaged because of buffer overflow in RX FIFO.</p> <hr/> <p>NOTE: This bit is set only when the DMA transfers a partial frame to the application. This happens only when the RX FIFO is operating in the threshold mode. In the store-and-forward mode, all partial frames are dropped completely in RX FIFO.</p> <hr/>
10	<p>VLAN: VLAN Tag</p> <p>When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the MAC. The VLAN tagging depends on checking VLAN fields of the received frame configured in the Ethernet MAC VLAN Tag (EMACVLANTG) register, offset 0x01C</p>
9	<p>FS: First Descriptor</p> <p>When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.</p>
8	<p>LS: Last Descriptor</p> <p>When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame.</p>
7	<p>Timestamp Available or Giant Frame</p> <p>When Advanced Timestamp feature is enabled, this bit indicates that a snapshot of the Timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set.</p> <p>Otherwise, this bit, when set, indicates the Giant Frame Status. Giant frames are larger than 1518-byte (or 1522-byte for VLAN or 2000-byte when Bit 27 of MAC Configuration register is set) normal frames and larger than 9018-byte (9022-byte for VLAN) frame when Jumbo Frame processing is enabled.</p>
6	<p>LC: Late Collision</p> <p>When set, this bit indicates that a late collision has occurred while receiving the frame in half-duplex mode.</p>
5	<p>FT: Frame Type</p> <p>When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 1536). When this bit is reset, it indicates that the received frame is an IEEE 802.3 frame. This bit is not valid for Runt frames less than 14 bytes. In addition when the IPC bit is set in the EMACCFG register, this bit conveys different information. See Table 15-9.</p>
4	<p>RWT: Receive Watchdog Timeout</p> <p>When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.</p>

Table 15-8. Enhanced Receive Descriptor 0 (RDES0) (continued)

Bit	Description
3	RE: Receive Error When set, this bit indicates that an error occurred during frame reception.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor bit (RDES0[8]) is set.
0	Extended Status Available/RX MAC Address When set, this bit indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. This bit is invalid when Bit 30 is set.

Table 15-9 lists the frame information conveyed in bits 7, 5, and 0 of RDES0 when the Checksum Offload Engine is enabled and disabled through the IPC bit in the EMACCFG register.

Table 15-9. RDES0 Checksum Offload Bits

Bit 5: Frame Type	Bit 7: IPC Checksum Error	Bit 0: Payload Checksum Error	IPC Bit Value in EMACCFG Register	Frame Status
0	0	0	X	IEEE 802.3 Type frame (Length field value is less than 1536). This status definition is valid even when the Checksum Offload engine is disabled.
1	0	0	0	IPv4/IPv6 Type frame in which no checksum error is detected.
1	0	0	1	The frame is an IEEE 802.3 Type frame (Length field value is greater than or equal to 1536).
1	0	1	1	IPv4/IPv6 Type frame with a payload checksum error detected
1	1	1	1	IPv4/IPv6 Type frame with both IP header and payload checksum errors detected
0	0	1	1	IPv4/IPv6 Type frame with no IP header checksum error and the payload check bypassed, due to an unsupported payload
0	1	1	1	A Type frame that is neither IPv4 or IPv6 (the Checksum Offload engine bypasses checksum completely.)
0	1	0	X	Reserved

Table 15-10. Enhanced Receive Descriptor 1 (RDES1)

Bit	Description
31	Disable Interrupt on Completion When set, this bit prevents the setting of the Receive Interrupt (RI) bit in the EMACDMARIS register and prevents the receive interrupt from being asserted.
30:29	Reserved
28:16	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size. The buffer size must be a multiple of 4, even if the value of RDES3 (buffer 2 address pointer) is not aligned to the bus width. When the buffer size is not a multiple of 4, the resulting behavior is undefined. This field is not valid if RCH bit (RDES1[14]) is set.
15	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.
14	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14].
13	Reserved

Table 15-10. Enhanced Receive Descriptor 1 (RDES1) (continued)

Bit	Description
12:0	<p>RBS1: Receive Buffer 1 Size</p> <p>These bits indicate the first data buffer size in bytes. The buffer size must be a multiple of 4 even if the value of RDES2 (buffer 1 address pointer) is not aligned to the bus width. When the buffer size is not a multiple of 4, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor depending on the value of RCH (Bit 14).</p>

Table 15-11. Enhanced Receive Descriptor 2 (RDES2)

Bit	Description
31:0	<p>Buffer 1 Address Pointer</p> <p>These bits indicate the physical address of Buffer 1. The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. The DMA performs a write operation with the RDES2[1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored. Note that buffers should be word-aligned.</p>

Table 15-12. Enhanced Receive Descriptor 3 (RDES3)

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[14]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[14] is set, the buffer (Next Descriptor) address pointer must be bus word-aligned (RDES3[1:0] = 0). However, when RDES1[14] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3 [1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

Table 15-13. Enhanced Received Descriptor 4 (RDES4)

Bit	Description
31:15	Reserved
14	<p>Timestamp Dropped</p> <p>When set, this bit indicates that the timestamp was captured for this frame but got dropped in the RX FIFO because of overflow.</p>
13	<p>PTP Version</p> <p>When set, this bit indicates that the received PTP message uses the IEEE 1588 version 2 format. When reset, it uses the version 1 format.</p>
12	<p>PTP Frame Type</p> <p>When set, this bit indicates that the PTP message is sent directly over Ethernet. When this bit is clear and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information about IPv4 or IPv6 can be obtained from Bits 6 and 7.</p>

Table 15-13. Enhanced Received Descriptor 4 (RDES4) (continued)

Bit	Description
11:8	<p>Message Type</p> <p>These bits are encoded to give the type of the message received:</p> <ul style="list-style-type: none"> • 0x0 = No PTP message received • 0x1 = SYNC (all clock types) • 0x2 = Follow_Up (all clock types) • 0x3 = Delay_Req (all clock types) • 0x4 = Delay_Resp (all clock types) • 0x5 = Pdelay_Req (in peer-to-peer transparent clock) • 0x6 = Pdelay_Resp (in peer-to-peer transparent clock) • 0x7 = Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 0x8 = Announce • 0x9 = Management • 0xA = Signaling • 0xB to 0xE = Reserved • 0xF = PTP packet with Reserved message type
7	<p>IPv6 Packet Received</p> <p>When set, this bit indicates that the received packet is an IPv6 packet. This bit is updated only when the IPC bit of the EMACCFG register is set.</p>
6	<p>IPv4 Packet Received</p> <p>When set, this bit indicates that the received packet is an IPv4 packet. This bit is updated only when the IPC bit of the EMACCFG register is set.</p>
5	<p>IP Checksum Bypassed</p> <p>When set, this bit indicates that the checksum offload engine is bypassed.</p>
4	<p>IP Payload Error</p> <p>When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. This bit is valid when either Bit 7 or Bit 6 is set.</p>
3	<p>IP Header Error</p> <p>When set, this bit indicates that either the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or the IP datagram version is not consistent with the Ethernet Type value. This bit is valid when either Bit 7 or Bit 6 is set.</p>
2:0	<p>IP Payload Type</p> <p>These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE sets this field to 0x0 if it does not process the IP datagram's payload due to an IP header error or fragmented IP packet.</p> <ul style="list-style-type: none"> • 0x0 = Unknown or did not process IP payload • 0x1 = UDP • 0x2 = TCP • 0x3 = ICMP • 0x4 to 0x7 = Reserved <p>This bit is valid when either Bit 7 or Bit 6 is set.</p>

Table 15-14. Enhanced Receive Descriptor 6 (RDES6)

Bit	Description
31:0	<p>RTSL: Receive Frame Timestamp Low</p> <p>This field is updated by the DMA with the least significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by the DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).</p>

Table 15-15. Enhanced Receive Descriptor 7 (RDES7)

Bit	Description
31:0	RTSH: Receive Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

15.3.3.6 DMA Transmission Operation

The following sections describe the modes of the transmit operation.

15.3.3.6.1 TX DMA Default Operation

The TX DMA engine in default mode operates as follows:

1. The CPU configures the transmit descriptor (TDES0-TDES3) and sets the OWN bit (TDES0[31]) after setting up the corresponding data buffers with the Ethernet frame.
2. When the ST bit of the Ethernet MAC DMA Operation Mode (EMACDMAOPMODE) register, offset 0xC18, is set, the DMA enters the RUN state.
3. While in RUN state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the CPU (TDES0[31] = 0), or if an error condition occurs, transmission is suspended and both the Transmit Buffer Unavailable (TU) bit and the Normal Interrupt Summary (NIS) bit are set in the Ethernet MAC DMA Interrupt Status (EMACDMARIS) register, offset 0xC14. The transmit engine then proceeds to Step 9.
4. If the acquired descriptor is flagged as owned by the DMA (TDES0[31] = 1), the DMA decodes the transmit data buffer address from the acquired descriptor.
5. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
6. The DMA fetches the transmit data from system memory and transfers the data to the TX/RX Controller for transmission.
7. If the Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Step 3, Step 4, and Step 5 are repeated until the end of the Ethernet frame data is transferred to the TX/RX Controller.
8. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value is written to the transmit descriptor (TDES6 and TDES7) that contains the end-of-frame buffer. The status information is then written to transmit descriptor TDES0. Because the OWN bit is cleared during this step, the CPU now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES6 and TDES7.
9. The Transmit Interrupt (TI) bit is set in the EMACDMARIS register after transmission completion of a frame that has Interrupt on Completion set in its last descriptor. The Interrupt on Completion bit resides in TDES0[30]. The DMA engine then returns to Step 3.
10. In the suspend state, the DMA tries to reacquire the descriptor (and thereby return to Step 3) when it receives a transmit poll demand in the Ethernet MAC Transmit Poll Demand (EMACTXPOLL) register, and the Underflow Interrupt Status (UNF) bit is cleared in the EMACDMARIS register. If the CPU stopped the DMA by clearing the ST bit of the EMACDMAOPMODE register, the DMA enters the STOP state.

Figure 15-7 shows the flow for the TX DMA default operation.

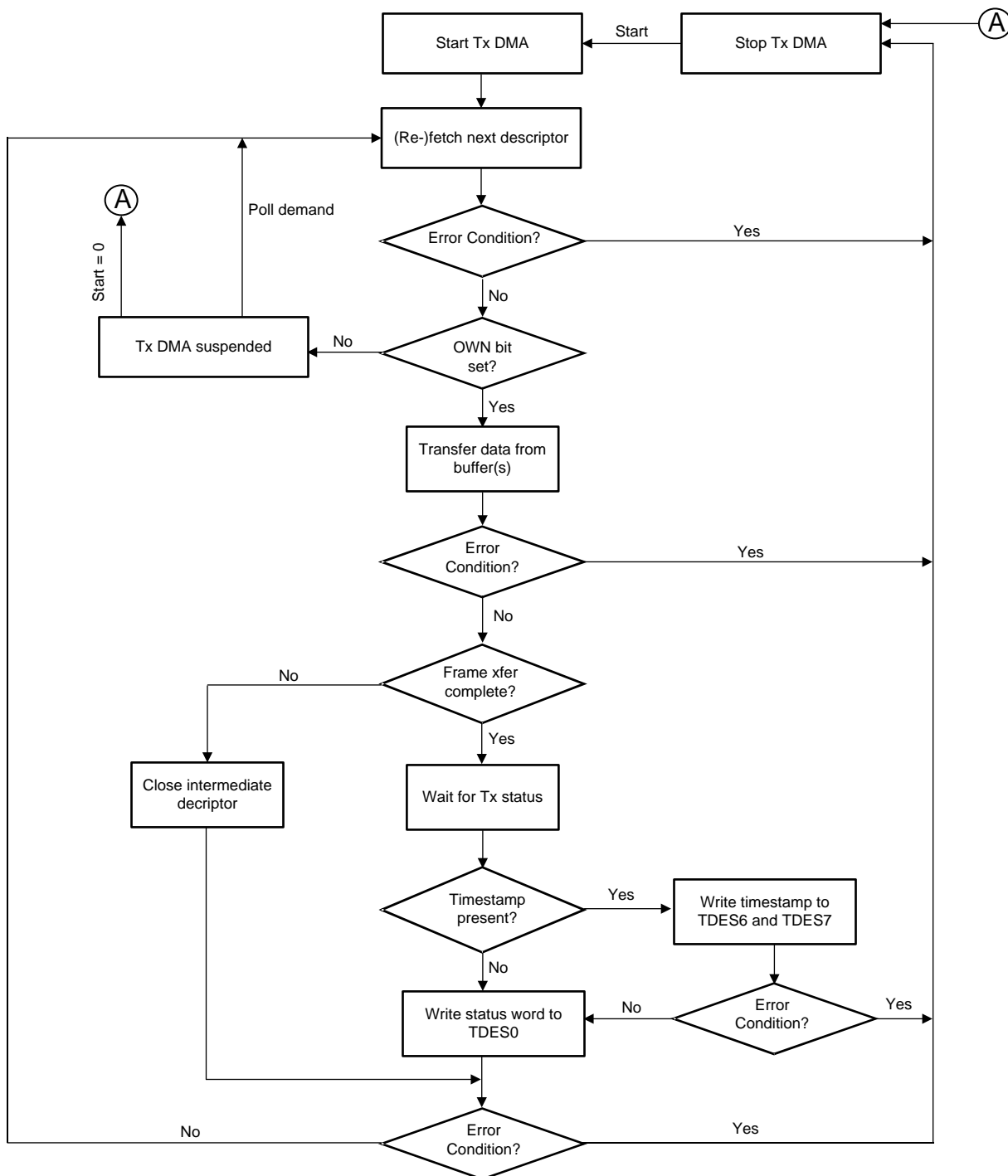


Figure 15-7. TX DMA Default Operation Using Descriptors

15.3.3.6.2 TX DMA OSF Mode Operation

While in the RUN state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first if the OSF bit of the EMACDMAOPMODE register is set. As the transmit process finishes transferring the first frame, it immediately polls the transmit descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In Operate on Second Frame (OSF) mode, the RUN state TX DMA operates in the following sequence:

1. The DMA operates as described in Step 1 to Step 6 of [Section 15.3.3.6.1](#).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into SUSPEND mode and skips to Step 7.
4. The DMA fetches the Transmit frame from system memory and transfers the frame until the end-of-frame data is reached. It closes the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's transmission status and timestamp. When the status is available, the DMA writes the timestamp to TDES6 and TDES7, if such timestamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared OWN bit, to the corresponding TDES0, thus closing the descriptor. If timestamping was not enabled for the previous frame, the DMA does not alter the contents of TDES6 and TDES7.
6. If enabled, the Transmit interrupt (TI) bit is set in the EMACDMARIS register, the DMA fetches the next descriptor and then proceeds to Step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into SUSPEND mode.
7. In SUSPEND mode, if a pending status and timestamp are received, the DMA writes the timestamp (if enabled for the current frame) to TDES6 and TDES7, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to SUSPEND mode.
8. The DMA can exit SUSPEND mode and enter the RUN state only after receiving the Transmit Poll demand in the EMAXXPOLL register.

NOTE: The DMA fetches the next descriptor in advance before closing the current descriptor. Therefore, the descriptor chain should have more than two different descriptors for correct and proper operation.

[Figure 15-8](#) shows the flow for the TX DMA Operate-On-Second-Frame (OSF) operation.

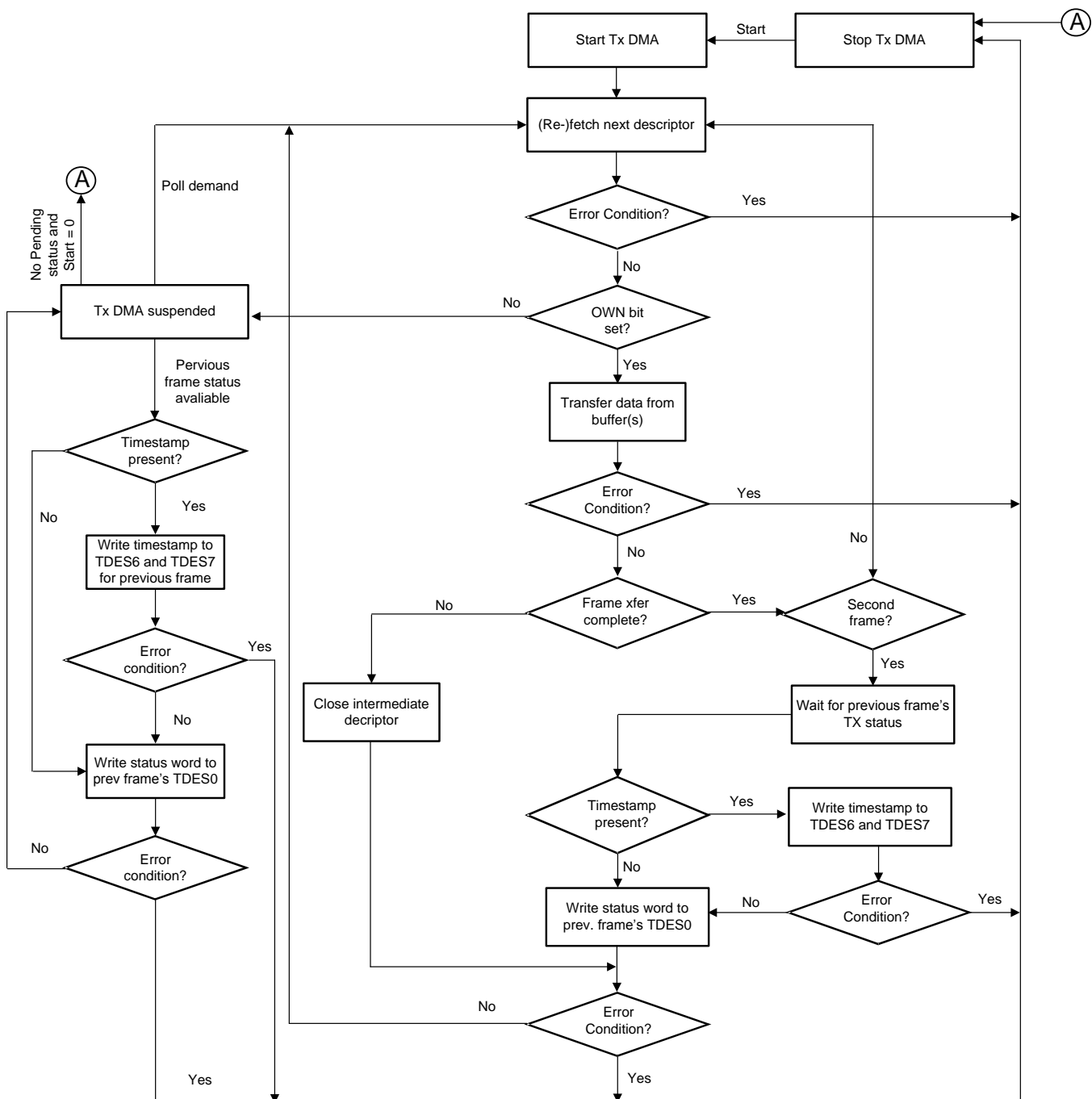


Figure 15-8. TX DMA OSF Mode Operation Using Descriptors

15.3.3.6.3 Transmit Frame Processing

The TX DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and Frame Check Sequence (FCS) fields. The Destination Address (DA), Source Address (SA), and Type/Length fields must contain valid data. If the Transmit Descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes. Frames can be data-chained and can span several buffers. Frames must be delimited by the First Segment Descriptor and the Last Segment Descriptor, respectively. The First Descriptor bit is located at TDES0[28] and the Last Descriptor is located at TDES0[29].

As transmission starts, the First Descriptor must have TDES0[28] set. When this occurs, frame data transfers from the host memory buffer to the TX FIFO. Concurrently, if the current frame has the Last Segment Descriptor (TDES0[29]) clear, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES0[28] clear. If TDES0[29] is clear, it indicates an intermediary buffer. If TDES0[29] is set, it indicates the last buffer of the frame. After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor word of the descriptor that has the last segment bit set in Transmit Descriptor. At this time, if Interrupt on Completion (IC) bit is set, the TI bit in the EMACDMARIS register is set, the next descriptor is fetched and the process repeats.

The actual frame transmission begins after the TX FIFO has reached either the transmit threshold as configured by the TTC bit field of the EMACDMAOPMODE register, or a full frame is contained in the TX FIFO. To wait until there is a full frame in the TX FIFO the TSF bit in the EMACDMAOPMODE register must be set. Descriptors are released (OWN bit in the TDES0[31] cleared) when the DMA finishes transferring the frame.

NOTE: To ensure proper transmission of a frame and the next frame, the transmit descriptor that has the Last Descriptor bit (TDES0[29]) set, must specify a non-zero buffer size.

15.3.3.6.4 Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the CPU (TDES0[31] = 0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command.
- A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate Transmit Descriptor 0 (TDES0) bit is set.

If the DMA goes into SUSPEND state because of the first condition, then both the Normal Interrupt Summary (NIS) bit and the Transmit Buffer Unavailable (TU) bit are set in the EMACDMARIS register. If the second condition occurs, the Abnormal Interrupt Summary (AIS) bit and the Transmit Underflow (UNF) bit of the EMACDMARIS register are set and the information is written to Transmit Descriptor 0, causing the suspension.

In both cases, the position in the Transmit list is retained. The retained position is that of the descriptor following the last descriptor closed by the DMA. The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension case.

15.3.3.7 DMA Receive Operation

The following section describes the receive operation process.

15.3.3.7.1 Default Receive Operation

The RX DMA engine's reception sequence is as follows:

1. The host sets up receive descriptors (RDES0 to RDES3) and sets the OWN bit (RDES0[31]).
2. When the SR bit of the EMACDMAOPMODE register is set, the DMA enters the RUN state. While in the RUN state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the CPU), the DMA enters the Suspend state and jumps to Step 9.
3. The DMA decodes the receive data buffer address from the acquired descriptors.
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the RX DMA engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to Step 7. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error (DE) bit in RDES0 (unless flushing is disabled through the DFF bit in the EMACDMAOPMODE register). The DMA closes the current descriptor (clears the OWN bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value. If flushing is not disabled,

then the DMA would mark it as the last descriptor. In either case, the DMA proceeds to Step 8. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to Step 4.

7. If IEEE 1588 timestamping is enabled, the DMA writes the timestamp to the current descriptor's RDES6 and RDES7. It then takes the receive frame's status and writes the status word to the current descriptor's RDES0, with the OWN bit cleared and the Last Segment (LS) bit set. If the host stopped the RX DMA by clearing the SR bit of the EMACDMAOPMODE register, DMA goes to the STOP state, otherwise the RX DMA proceeds to Step 8.
8. The RX DMA engine checks the last descriptor's OWN bit. If the CPU owns the descriptor (OWN bit is 0), the RU bit of the EMACDMARIS register is set and the DMA RX engine enters the SUSPEND state. If the DMA owns the descriptor, the engine returns to Step 4 and awaits the next frame.
9. Before the RX DMA engine enters the SUSPEND state, partial frames are flushed from the RX FIFO. Flushing can be controlled through the DFF bit of the EMACDMAOPMODE register.
10. The RX DMA enters the STOP state if the CPU has cleared the SR bit of the EMACDMAOPMODE register. Otherwise, it exits the SUSPEND state when a Receive Poll Demand is given or the start of the next frame is available from the RX FIFO. The DMA engine proceeds to Step 2 and fetches the next descriptor again.

The DMA does not acknowledge accepting status from the TX/RX Controller until it has completed the timestamp write-back and is ready to perform status write-back to the descriptor.

If software has enabled timestamping through the Ethernet MAC Timestamp Control (EMACTIMSTCTRL) register, offset 0x700, when a valid timestamp is not available for the frame (for example, because the receive FIFO was full before the timestamp could be written to it), the DMA writes all ones to RDES6 and RDES7. Otherwise if timestamping is not enabled, RDES6 and RDES7 remain unchanged.

[Figure 15-9](#) shows the flow of a RX DMA Operation.

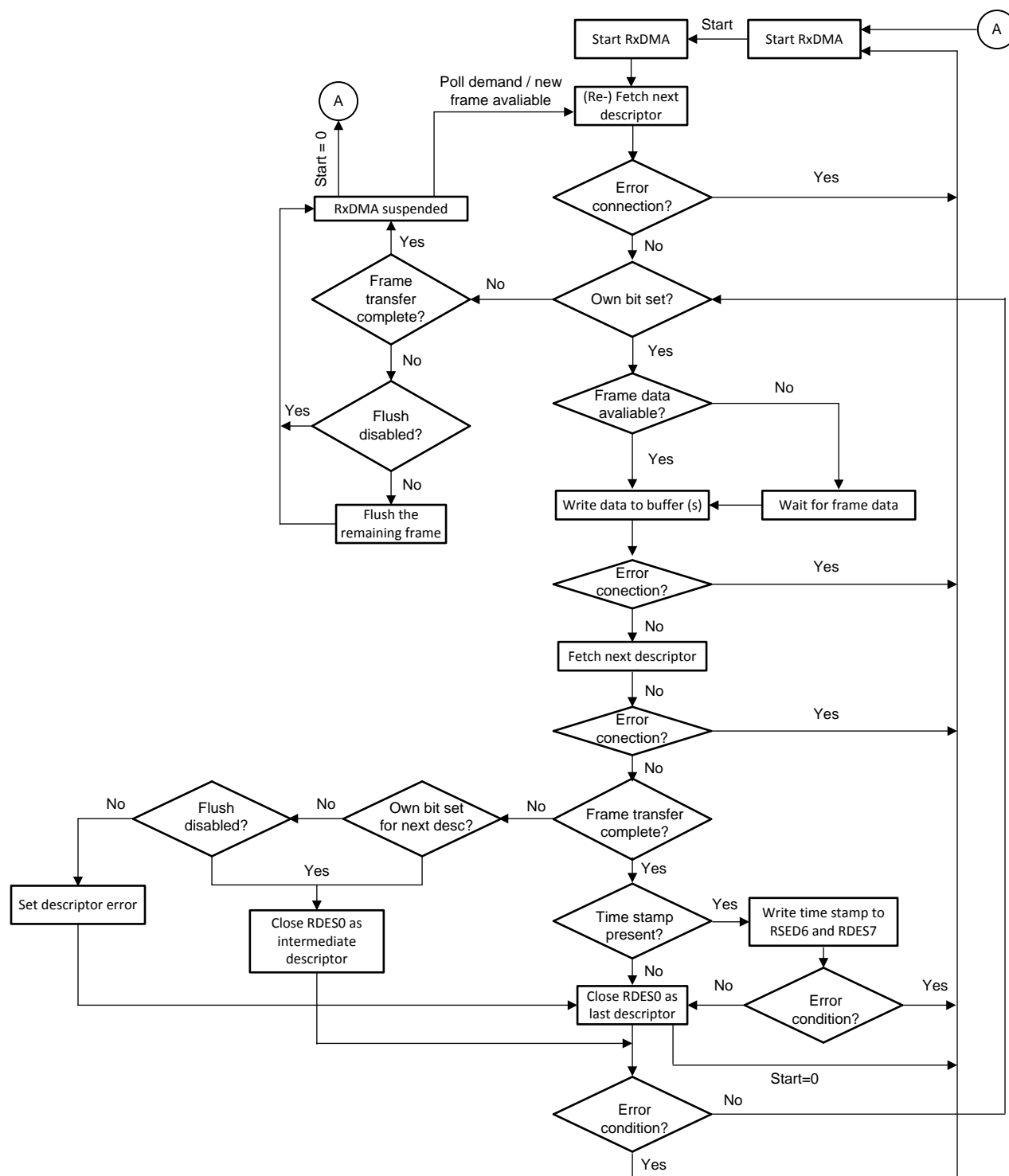


Figure 15-9. RX DMA Operation Flow

15.3.3.7.2 Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- The SR bit of the EMACDMAOPMODE register has been set immediately after being placed in the RUN state.
- The data buffer or current descriptor is full before the frame ends for the current transfer.
- The controller has completed frame reception, but the current receive descriptor is not yet closed.
- The receive process has been suspended because of a host-owned buffer (RDES0[31] = 0) and a new frame is received.
- A receive poll demand has been issued.

15.3.3.7.3 Receive Frame Processing

The MAC transfers the received frames to the system memory only when the frame passes the address filter and the frame size is greater than or equal to configurable threshold bytes set for the RX FIFO, or when the complete frame is written to the RX FIFO in store-and-forward mode.

If the frame fails the address filtering, it is dropped in the MAC block unless the Receive All (RA) bit is set in the Ethernet MAC Frame Filter (EMACFRAMEFLTR), offset 0x004. If the RA bit is set, then the MAC passes all received frames. Frames that are shorter than 64 bytes, because of collision or premature termination, can be removed from the RX FIFO if the DFF bit is clear in the EMACDMAOPMODE register.

After 64 bytes have been received, the TX/RX Controller requests the DMA block to begin transferring the frame data to the receive buffer pointed by the current descriptor. The DMA sets the First Descriptor (RDES0[9]) bit to delimit the frame after the DMA Interface becomes ready to receive a data transfer (if DMA is not fetching transmit data from the system memory). The descriptors are released when the OWN (RDES0[31]) bit is reset to 0, either as the data buffer fills up or as the last segment of the frame is transferred to the receive buffer. If the frame is contained in a single descriptor, both Last Descriptor (RDES0[8]) and First Descriptor (RDES0[9]) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES0[8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets the RI bit of the EMACDMARIS register. The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host. If this occurs, the receive process sets the RU bit of the EMACDMARIS and enters the SUSPEND state. The position in the receive list is retained.

15.3.3.7.4 Receive Process Suspend

If a new receive frame arrives while the receive process is in SUSPEND state, the DMA refetches the current descriptor in the system memory. If the descriptor is now owned by the DMA, the receive process re-enters the RUN state and starts frame reception. If the descriptor is still owned by the CPU, by default, the DMA discards the current frame at the top of the RX FIFO and increments the missed frame counter. If more than one frame is stored in the RX FIFO, the process repeats. The discarding or flushing of the frame at the top of the RX FIFO can be prevented by disabling flushing through the DFF bit of the EMACDMAOPMODE register. In such conditions, the receive process sets the Receive Buffer Unavailable (RU) status and returns to the SUSPEND state.

15.3.3.8 DMA Interrupts

Interrupts can be generated as a result of various transfer events. The current status of interrupts can be read from the EMACDMARIS register and are enabled to trigger an interrupt through the programming of the Ethernet MAC DMA Interrupt Mask (EMACDMAIM) register. There are two groups of transfer event interrupts: Normal and Abnormal. The following lists the two groups:

- Normal Interrupts:
 - Transmit Interrupt (TI, bit 0): Indicates that frame transmission is complete.
 - Transmit Buffer Unavailable (TU, bit 2): Indicates the CPU owns the next descriptor in the transmit list and the DMA cannot acquire it.
 - Receive Interrupt (RI, bit 6): Indicates the frame reception is complete.

- Early Receive Interrupt (ERI, bit 14): Indicates the DMA has filled the first half of the data buffer of the packet
- Abnormal Interrupts:
 - Transmit Process Stopped (TPS, bit 1): Indicates transmission is stopped.
 - Transmit Jabber Timeout (TJT, bit 3): Indicates the Transmit Jabber Timer expired.
 - Receive FIFO Overflow (OVF, bit 4): Indicates the receive buffer had an overflow during frame reception.
 - Transmit Underflow (UNF, bit 5): Indicates the transmit buffer had an underflow during frame transmission.
 - Receive Buffer Unavailable (RU, bit): Indicates the CPU owns the next descriptor in the receive list and the DMA cannot acquire it.
 - Receive Process Stopped (RPS, bit 8): Indicates the receive process entered the STOP state.
 - Receive Watchdog Timeout (RWT, bit 9): Indicates a frame length greater than 2KB is received (10240 when Jumbo Frame is enabled).
 - Early Transmit Interrupt (ETI, bit 10): Indicates a frame to be transmitted is fully transferred to the TX FIFO.
 - Fatal Bus Error (FBI, bit 13): Indicates a bus error occurred.

Any of the interrupts in the Normal Interrupt group that are enabled in the EMACDMAIM register are ORed together to create the Normal Interrupt Summary (NIS) bit in the EMACDMARIS register. Any of the interrupts in the Abnormal Interrupt group that are enabled in the EMACDMAIM register are ORed together to create the Abnormal Interrupt Summary (AIS) bit in the EMACDMARIS register. Interrupts are cleared by writing a 1 to the corresponding bit position in the EMACDMARIS register. When all enabled interrupts within a group are cleared, the corresponding summary bit is cleared.

Interrupts are not queued and if the interrupt event occurs again before the driver has responded to it, no additional interrupts are generated. An interrupt is only generated once for simultaneous, multiple events. The driver must read the EMACDMARIS register for the cause of the interrupt.

The Ethernet MAC Receive Interrupt Watchdog Timer (EMACRXINTWDT) register, offset 0xC24 can be used to control the Receive Interrupt (RI) assertion. If the RDES1[31] bit (Receive Interrupt) bit has not been set in the receive descriptor and the EMACRXINTWDT register is programmed with a non-zero value, it gets activated as soon as the RX DMA completes a transfer of a received frame to system memory without asserting the receive interrupt. When this counter runs out as per the programmed value, the RI bit is set in the EMACDMARIS register and the interrupt is asserted if the corresponding RI bit is enabled in the EMACDMAIM register. This counter gets disabled before it runs out if a frame is transferred to memory and the RI bit is set because it is enabled for that descriptor.

15.3.3.9 DMA Bus Error

If an internal bus error occurs during a DMA transfer, the fatal bus error (FBI) interrupt is set in the EMACDMARIS register and the Access Error status (AE) bit field in the EMACDMARIS register indicates the type of error that caused the bus error. The DMA controller can resume operation only after soft resetting the Ethernet MAC and the re-initializing the DMA.

15.3.4 TX/RX Controller

The TX/RX Controller consists of a FIFO memory which buffers and regulates the frames between the system memory and the MAC. It also controls the data transferred between clock domains. Both the transmit and receive data paths are 32-bits wide. The TX FIFO and RX FIFO are each 2KB in depth.

At reset, the TX/RX Controller is configured and ready to manage data flow to and from the DMA to the MAC. Note that the DMA and MAC must be initialized by the application out of reset.

15.3.4.1 Transmit (TX) Control Path

The DMA controller is used for all Ethernet transmissions. The Ethernet frames are read from memory and transferred to the TX FIFO by the DMA. When the MAC is available, the frame is transferred from the FIFO. When the end-of-frame (EOF) is transferred, the MAC notifies the DMA the status of the transmission.

The TX FIFO has a depth of 2KB. The FIFO fill level has the capability of triggering the DMA to initiate a burst transfer. The DMA also transfers start-of-frame (SOF), end-of-frame (EOF), CRC and pad-insertion information to the TX/RX Controller so that this information can be passed to the MAC when it is ready for transmission from the TX FIFO.

Data can be transmitted to the MAC in threshold mode or store-and-forward mode. If the TTC field is configured in the Ethernet MAC DMA Operation Mode (EMACDMAOPMODE) register at offset 0xC18 and the TSF bit in the same register is 0x0, then the TX Controller is operating in threshold mode. In this mode, the data is transferred to the MAC when the number of bytes in the FIFO crossed the value configured in the TTC bit field or when the end-of-frame is written before the threshold is crossed. In store-and-forward mode, the TTC bit field is configured and the TSF bit is set. Data is transferred to the MAC only when one or more of the following conditions are true:

- A complete frame is stored in the FIFO
- The TX FIFO becomes almost full
- The TX FIFO does not have space to accommodate the requested burst length

With these conditions, the TX Controller continues store-and-forward mode even if the Ethernet frame length is bigger than the TX FIFO size.

The TX FIFO can be flushed of all contents by setting the FTF bit in the EMACDMAOPMODE register. This bit is self-clearing and initializes the FIFO pointers to the default state. If the FTF bit is set during a frame transfer from the TX Controller to the MAC, then the TX Controller stops further transfer. Early termination of the transfer causes a underflow event and this status is communicated to the DMA.

15.3.4.1.1 Transmit Operation

During a transmit, single-packet or double-packets can reside in the buffer. The following describes the details of each:

- Single-packet transmit: During single packet transmission, the DMA controller fetches data from the CPU memory and forwards it to the TX FIFO and continues to receive data until the end-of-frame is transferred. The data is transmitted from the TX FIFO to the MAC by the TX/RX Controller when the threshold level is crossed or a full packet of data is received into the TX FIFO. When the TX/RX Controller receives acknowledgment from the MAC that it has received the EOF, it notifies the DMA so another transmit can begin.
- Two-packet transmit: Because the DMA must update the descriptor status before releasing it to the CPU, there can be at most two frames inside a transmit FIFO. The second frame is fetched by the DMA and put into the TX FIFO only if the OSF bit is set in the EMACDMAOPMODE register at offset 0xC18. If this bit is not set, the next frame is fetched from memory only after the MAC has completely processed the frame and the DMA has released the descriptors.
- If the OSF bit is set, the DMA starts fetching the second frame immediately after completing the transfer of the first frame to the FIFO. It does not wait for the status to be updated. The TX/RX Controller receives the second frame into the FIFO while transmitting the first frame. As soon as the first frame has been transferred and the status is received from the MAC, the TX/RX Controller sends the acknowledgment to the DMA. If the DMA has already completed sending the second packet to the TX/RX Controller, it must wait for the status of the first packet before proceeding to the next frame.

15.3.4.1.2 Collision and Retransmission

If a collision occurs at the MAC application interface while the TX/RX Controller is transferring data to the MAC, the transmission is aborted and the MAC indicates a retry attempt by giving a collision status before the EOF is transferred to the TX/RX Controller from the DMA. This enables the TX/RX Controller to retry transmission of the frame data from the FIFO.

After more than 96 bytes are transferred to the MAC, the FIFO controller clears space in the FIFO and makes it available to the DMA to transfer more data. Retransmission is not possible after this threshold is crossed or when the MAC indicates a late collision event.

When a frame transmission is aborted because of underflow and a collision event follows, which initiates a retry, then the retry has higher priority than the abort.

15.3.4.1.3 TX FIFO Flush Operation

The TX FIFO can be flushed by setting the FTF bit in the EMACDMAOPMODE register. The flush operation is immediate and the TX/RX Controller clears the TX FIFO and the corresponding pointers to the initial state even if it is in the middle of transferring a frame to the MAC. The data which is already accepted by the MAC transmitter is not flushed. This data is scheduled for transmission and results in an underflow event because the TX FIFO did not complete the transfer or the rest of the frame. As in all underflow conditions, a runt frame is transmitted and observed on the line. The status of such a frame is marked with both underflow and frame flush events in the Transmit Descriptor 0 (TDES0) word.

The TX/RX Controller also stops accepting any data from the DMA during the flush operation. It generates and transfers Transmit Status Words to the application for the frames that are flushed inside the FIFO, including partial frames. Frames that are completely flushed in the TX/RX Controller are identified by setting the Flush Status (FF) bit in the Transmit Descriptor 0 (TDES0) word. The TX/RX Controller completes the flush operation when the DMA accepts all of the status words for the frames that were flushed and then clears the TX FIFO Flush control (FTF) bit in the EMACDMAOPMODE register. At this point, the TX/RX Controller starts accepting new frames from the DMA.

15.3.4.1.4 Transmit Status Word

At the end of the transfer of the Ethernet frame to the MAC and after the MAC completes the transmission of the frame, the TX/RX delivers a transmit status word (TDES0) to the application. If IEEE timestamping is enabled, the TX/RX Controller returns the specific frame's 64-bit timestamp, along with the transmit status word. The fields for the Transmit Descriptors are described in [Section 15.3.3.5](#).

15.3.4.2 Receive (RX) Control Path

TX/RX Controller receives frames from the MAC and pushes them into the RX FIFO. When the fill level of the RX FIFO crosses the programmed RX Threshold, the DMA is notified.

15.3.4.2.1 Receive Operation

During a receive operation the TX/RX Controller is a slave to the MAC. The steps of the receive operation are as follows:

1. The MAC receives a frame. This data, along with SOF, EOF and byte enable information is sent to the TX/RX Controller. The TX/RX Controller accepts the data and pushes it into the RX FIFO. After the EOF is transferred, the MAC drives the status word, which is also pushed in to the RX FIFO.
2. When timestamp is enabled by setting the TSEN bit in the Ethernet MAC Timestamp Control (EMACTIMSTCTRL) register, at offset 0x700, and the 64-bit timestamp is present with the receive status, it is appended to the frame and received by the MAC and pushed into the TX FIFO before the corresponding receive status word is written. Thus, two additional locations per frame are taken for storing timestamp in the RX FIFO.
3. Data can be sent to the TX/RX Controller in cut-through mode or store-and-forward mode. When the RTC bit field of the EMACDMAOPMODE register is set to 0x0 and cut-through mode is enabled (RSF = 0), the TX/RX Controller indicates availability to transfer to the DMA when 64 bytes are in the RX FIFO or a full packet of data has been received into the RX FIFO. When the DMA initiates transfers to system memory, the TX/RX Controller continues to transfer data from the RX FIFO until a complete packet has been transferred. When EOF has occurred, the TX/RX Controller sends the status word to the DMA.

NOTE: The timestamp transfer takes two clock cycles and the lower 32-bits of the timestamp are sent first when timestamping is enabled.

4. When the RSF bit is set in the EMACDMAOPMODE register, RX FIFO store-and-forward mode is enabled and a frame is read by the DMA only after it is completely written into the RX FIFO. In this mode, only valid frames are read and forwarded to the application. In cut-through mode, some error frames are not dropped because the error status is received at the end of the frame and by that time the start of that frame has already been read out of the RX FIFO.

The TX/RX Controller is capable of storing any number of frames in the RX FIFO as long as it is not full.

15.3.4.2.2 Error Handling

If the RX FIFO is full before it receives the EOF data from the MAC, an overflow is declared, the entire frame (including the status word) is dropped and the Ethernet MAC Missed Frame and Buffer Overflow Counter (EMACMFBOC) register is incremented. These error actions occur even if the FEF bit is set in the EMACDMAOPMODE register. If the start address of such a frame has already been transferred to the TX/RX Controller, the rest of the frame is dropped and a dummy EOF is written to the FIFO along with its status word. The descriptor status indicates a partial frame because of overflow. In such frames, the Frame Length (FL) field in the receive descriptor is invalid. If the RX FIFO is configured to operate in the store-and-forward mode and if the length of the received frame is more than the FIFO size, overflow occurs and all such frames are dropped. During error handling, the DMA flushes the error frame currently being read.

The Receive control logic can filter error and undersized frames if enabled through configuring the FEF or FUF bit of the EMACDMAOPMODE register. Filtering must be set before the start address of the frame has been transferred to the TX/RX controller for it to take effect.

15.3.4.2.3 Receive Word Status

At the end of an Ethernet frame transfer to the system memory, the TX/RX Controller sends a receive status word, RDES0, to the application. Until the end of the frame transfer, the TX/RX Controller stores the status and frame length in an asynchronous status FIFO whose depth is determined by the size of the RX FIFO (2K) and the minimum size of the frame. If the frame size is 64, then the asynchronous FIFO depth is $2048/64 = 32$ bytes in length. Note that when the status of a partial frame (because of overflow) is sent to the application, the Frame Length field of RDES0 is not valid and is set to zero.

NOTE: When the timestamp feature is enabled, the receive status field is greater than 32-bits. An extended status bit-field [63:32] provides information about the received Ethernet payload when it is carrying PTP packets or TCP/UDP/ICMP over IP packets. Since the data bus is 32 bits, the status is transferred over two clock cycles.

15.3.4.3 MAC Flow Control

Flow control mechanisms can be enabled for both the TX and RX FIFO data path, depending on the configurations in the Ethernet MAC Flow Control (EMACFLOWCTL) register at offset 0x018 and the DUPM bit configuration in the Ethernet MAC Configuration (EMACCFG) register at offset 0x000.

Table 15-16. TX MAC Flow Control

TFE Bit in EMACFLOWCTL	DUPM Bit in EMACCFG	Description
0	X	The MAC transmitter does not perform the flow control or backpressure operation.
1	0	The MAC transmitter performs backpressure when the FCBBPA bit in the Ethernet MAC Flow Control (EMACFLOWCTL) register is set.
1	1	The MAC transmitter sends a pause frame when the FCBBPA bit in the Ethernet MAC Flow Control (EMACFLOWCTL) register is set.

Table 15-17. RX MAC Flow Control

TFE Bit in EMACFLOWCTL	DUPM Bit in EMACCFG	Description
0	X	The MAC receiver does not detect the received Pause frames.

Table 15-17. RX MAC Flow Control (continued)

TFE Bit in EMACFLOWCTL	DUPM Bit in EMACCFG	Description
1	0	The MAC receiver does not detect the received Pause frames but recognizes such frames as Control frames.
1	1	The MAC receiver detects or processes the Pause frames and responds to such frames by stopping the MAC transmitter.

15.3.5 MAC Operation

The MAC module enables the CPU to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard. The MAC supports the interface to the PHY and is comprised of a receive and transmit module whose features are described in the following sections.

15.3.5.1 MAC Transmit Module

MAC transmission is initiated when the TX/RX Controller transmits data with the start of frame (SOF) signal asserted. When the SOF signal is detected, the MAC accepts the data and begins transmitting to the PHY. The time required to transmit the frame data after the application initiates transmission varies, depending on delay factors like inter-frame gap (IFG) delay, time to transmit preamble or start of frame data (SFD), and any back-off delays for half-duplex mode. Until then, the MAC does not accept data received from the TX/RX Controller.

After the end-of-frame (EOF) is transferred to the MAC, the MAC completes the normal transmission and gives the status of transmission to the TX/RX Controller. If a normal collision (in half-duplex mode) occurs during transmission, the MAC conveys the transmit status to the TX/RX Controller. It then accepts and drops all further data until the next SOF is received. The TX/RX Controller should retransmit the same frame from SOF on observing a retry request (in the transmit status word) from the MAC. The MAC issues an underflow status if the TX/RX Controller is not able to provide the data continuously during the transmission. During the normal transfer of a frame from the TX/RX Controller, if the MAC receives an SOF without getting an EOF for the previous frame, it ignores the SOF and considers the new frame as a continuation of the previous frame.

If the number of bytes received from memory is less than 60 bytes, zeros are automatically appended to the transmitting frame to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE 802.3.

The transmit engine controls the operation of Ethernet frame transmission. Some of the functions of the transmit module include:

- Output of (32-bit) Transmit Status (TDES0) to the application at the end of normal transmission or collision
- Generating preamble and Start of Frame Data (SFD)
- Generating jam pattern in half-duplex mode
- Supporting Jabber time-out
- Supporting flow control
- Generating timestamp information for transmission
- Scheduling frame transmission to satisfy inter-frame gap (IFG) and back-off delays
- Generating CRC and FCS field for Ethernet Frame
- Generating pause frames as necessary in full duplex mode

When a new frame transmission is requested, the MAC transmit module sends out a preamble and SFD, followed by the data in the TX FIFO. The preamble is defined as 7 bytes of 0xAA and the SFD is defined as 1 byte of 0xAB pattern.

The collision window is defined as 1 slot time (512-bit times). If a collision occurs any time from the beginning of the frame to the end of the CRC field, the MAC transmit module sends a 32-bit jam pattern of 0x5555.5555 to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the MAC transmit module completes the transmission of the preamble and SFD, and then sends the jam pattern. If the collision occurs after the collision window and before the end of the FCS field, the MAC transmit module sends a 32-bit jam pattern and sets the late collision bit in the transmit frame status. The jam pattern generation is applicable only to half-duplex mode.

A jabber timer is enabled by default at reset in the MAC module. If the JD bit in the EMACCFG register at offset 0x000 is clear and the MAC module has transferred more than 2KB, the jabber timer causes the MAC module to stop transmission of Ethernet frames. The time-out is changed to 10KB when the Jumbo Frame Enable (JFEN) bit is enabled in the EMACCFG register.

The MAC transmit module uses a deferral mechanism for flow control (back pressure) in half-duplex mode. When the application requests to stop receiving frames, the transmit module sends a jam pattern of 32 bytes whenever it senses the reception of a frame, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a frame to be transmitted, the frame is scheduled and transmitted even when backpressure is activated. If the backpressure is activated for a long time (more than 16 consecutive collision events occur) then the remote stations abort their transmissions because of excessive collisions.

If IEEE 1588 timestamping is enabled for the transmit frame, the MAC transmit module takes a snapshot of the system time when the start-of-frame data is output on the bus.

15.3.5.2 MAC Transmit Module CRC Generator

The Transmit CRC Generator is used to generate the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 \quad (48)$$

The calculated CRC is valid on the next clock after the data is received.

15.3.5.3 MAC Receive Module

A receive operation is initiated when the MAC detects start-of-frame data (SFD). The MAC strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field is used to verify the CRC for the frame. The received frame is stored in a buffer until the address filtering is performed. The frame is dropped in the MAC if it fails the address filter.

Some of the functions of the MAC receive module are as follows:

- Stripping the preamble, SFD and carrier extension of the Ethernet frame
- Providing IEEE 1588 timestamping whenever an SFD is detected
- Converting receive nibble data into bytes in MII mode
- Decoding Length/Type field of the Ethernet frame
- Auto-CRC/pad stripping if enabled in the EMACCFG register
- Frame data and status buffering for received frames
- Data path conversion of 8-bit data to 32-bit data
- Frame filtering
- Attaching calculated IP checksum input to frame
- Detecting receiving pause frames and pausing the frame transmission for the delay specified withing the received pause frame
- Receive IP Checksum Verification
- Performing destination and source address checking functions on all received frames and reporting the address filtering status to the receive frame controller module.

15.3.5.4 MAC Receive Module CRC Generator

The receive CRC Generator is used to generate the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (49)$$

The calculated CRC is valid on the next clock after the data is received.

15.3.5.5 MAC Receive Frame Controller

If the RA bit is set in the Ethernet MAC Frame Filter (EMACFRAMEFLTR) register, offset 0x004, the MAC Receive Frame Controller initiates the data transfer to the RX FIFO as soon as four bytes of Ethernet data are received. At the end of the data transfer, the received frame status that includes the frame filter bits (SA or DA filter fail) and status are also sent. These bits indicate to the application whether the received frame has passed the filter controls. This module does not drop any frame on its own in this mode.

If the RA bit is clear, the MAC Receive Frame Controller performs frame filtering based on the destination/source address (the application still needs to perform another level of filtering if it decides not to receive any bad frames like runt, CRC error frames, etc.) After receiving the destination or source address bytes, the MAC Receive Frame Controller checks the filter-fail sign for an address match. On detecting a filter-fail, the frame is dropped and not transferred to the application.

NOTE: When the PMT module is configured for power-down mode, all received frames are dropped and not forwarded to the application.

15.3.6 IEEE 1588 and Advanced Timestamp Function

The MAC module supports the IEEE 1588-2002 Timestamp Precision Time Protocol (PTP) and the IEEE 1588-2008 Advanced Timestamp features. PTP enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by a local area network supporting multicast messaging. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into master and slave nodes for distributing the timing and clock information. [Figure 15-10](#) shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

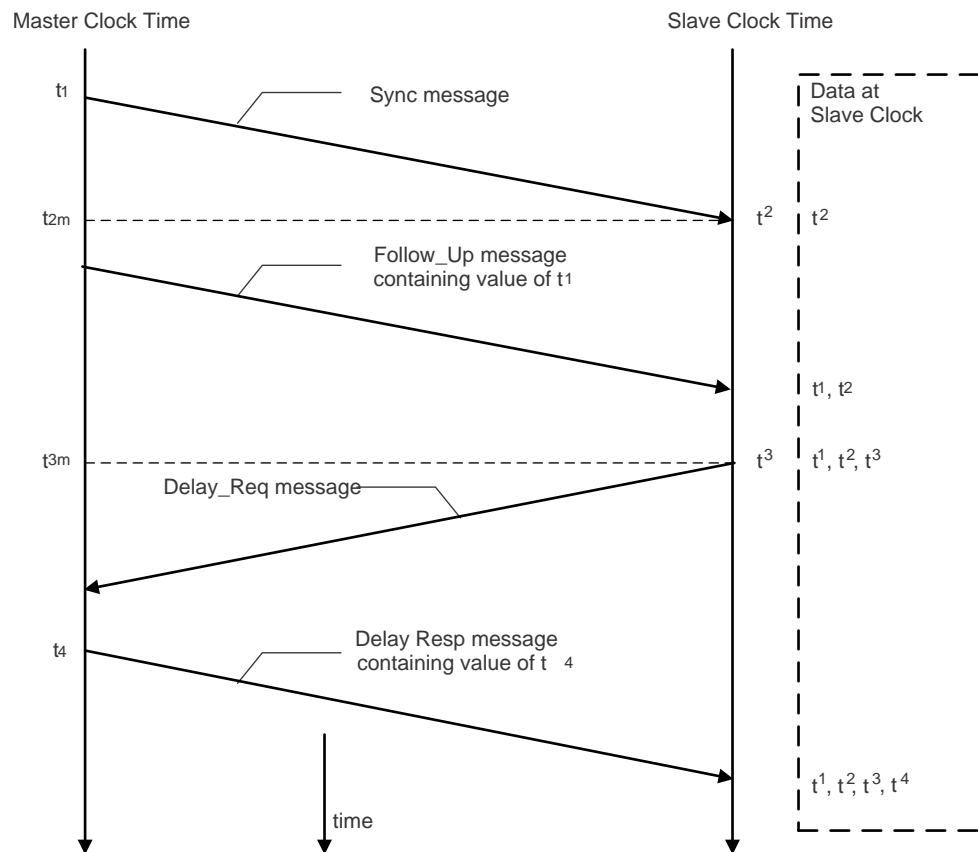


Figure 15-10. Networked Time Synchronization

As shown in [Figure 15-10](#), the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t_1 . This time is captured at the MII interface.
2. The slave receives the Sync message and also captures the exact time, t_2 , using its timing reference.
3. The master sends a Follow_Up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master, noting the exact time, t_3 , at which this frame leaves the MAC.
5. The master receives the message, capturing the exact time, t_4 , at which it enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1 , t_2 , t_3 , and t_4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet MAC. This timing information is captured and returned to the software for the proper implementation of PTP with high accuracy.

15.3.6.1 System Time Module

The System Time module maintains a 64-bit time and is updated using the MOSC clock source as the PTP clock reference. This time is the source for taking snapshots (timestamps) of the Ethernet frames being transmitted or received. Two methods of updating the system time counter are implemented. The counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the MAC System Time - Seconds Update (EMACTIMSECU) register along with the MAC System Time - Nanoseconds Update (EMACTIMNANOU) register. For initialization the system time counter is written with the value in these registers, while for system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, the slave clock frequency drift with respect to the master clock is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the EMACTIMADD register (see Figure 15-11). The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

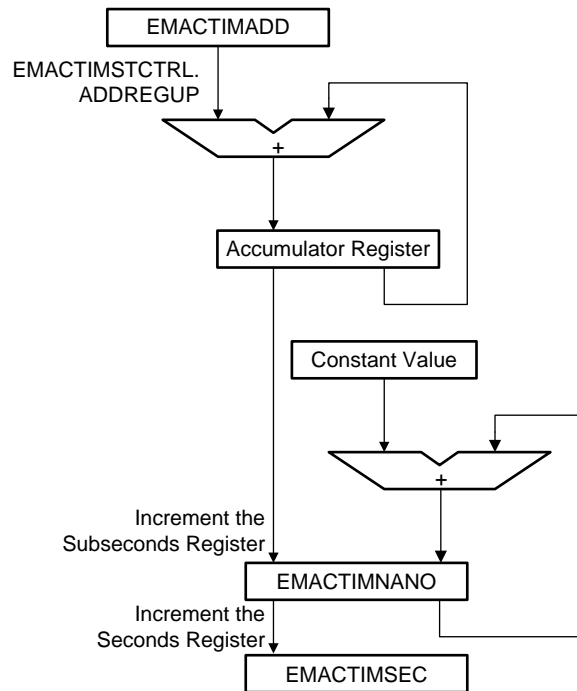


Figure 15-11. System Time Update Using Fine Correction Method

NOTE: If the Ethernet Controller is configured to use the MII/RMII interface to an external PHY, then the MOSC clock that feeds the PTP reference clock to the System Time Module has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. For coarse correction methods, the value of MOSC can be anywhere within this range, but for the fine correction method, a 25-MHz MOSC crystal should be used for the best accuracy. If the Ethernet Controller is configured to use the MII interface connected to the integrated PHY, then the MOSC clock that feeds the PTP reference clock to the System Time Module must be 25 MHz, because it also clocks the integrated PHY module.

Initially, the Ethernet slave clock (from the MOSC) is adjusted with a compensation value (as described in the previous paragraph) which is written to the Timestamp Addend Register (TSAR) field in the EMACTIMADD register. This value is calculated as:

$$\text{FreqCompensationValue}_0 = \text{TSAR} = 2^{32} / \text{FreqDivisionRatio} \quad (50)$$

The System Time Module requires a 20-MHz PTP reference clock frequency to achieve 50-ns accuracy in the fine correction method. An addend must be written to the Ethernet MAC Time Stamp Addend (EMACTIMADD) register, offset 0x718 to achieve timing synchronization. If the MOSC clock source is 25 MHz, the frequency division ratio (FreqDivisionRatio) of the two is calculated as 25 MHz / 20 MHz = 1.25. Hence, the default addend value to be set in the register is $2^{32} / 1.25$ or 0xCCCC.CCD0. If the reference clock drifts lower, to 24 MHz for example, the ratio is 24 / 20, or 1.2 and the value to set in the addend register is $2^{32} / 1.20$, or 0xDFF1.65D2. The software must calculate the drift in frequency based on the Sync messages and update the EMACTIMADD register, at offset 0x718, accordingly.

If the master to slave delay is initially assumed to be the same for consecutive Sync messages, then the following steps can be used to calculate a new TSAR value. The following algorithm calculates the precise master to slave delay value to allow for re-synchronization with the master using the new value:

1. At time MasterSyncTime_n the master sends the slave clock a Sync message. The slave receives this message when its local clock is SlaveClockTime_n and computes MasterClockTime_n as:

$$\text{MasterClockTime}_n = \text{MasterSyncTime}_n + \text{MasterToSlaveDelay}_n \quad (51)$$

2. The master clock count, MasterClockCount_n, for the current Sync cycle is given by:

$$\text{MasterClockCount}_n = \text{MasterClockTime}_n - \text{MasterClockTime}_{n-1} \quad (52)$$

This assumes that the MasterToSlaveDelay is the same for Sync cycles n and n-1.

3. The slave clock count for current Sync cycle, SlaveClockCount_n is given by:

$$\text{SlaveClockCount}_n = \text{SlaveClockTime}_n - \text{SlaveClockTime}_{n-1} \quad (53)$$

4. The difference between the master and slave clock counts for the current Sync cycle, ClockDiffCount_n, is given by:

$$\text{ClockDiffCount}_n = \text{MasterClockCount}_n - \text{SlaveClockCount}_n \quad (54)$$

5. The frequency-scaling factor for the slave clock, FreqScaleFactor_n is given by:

$$\text{FreqScaleFactor}_n = (\text{MasterClockCount}_n + \text{ClockDiffCount}_n) / \text{SlaveClockCount}_n \quad (55)$$

6. The frequency compensation value, FreqCompensationValue, to be written in the TSAR field of the EMACTIMADD register is:

$$\text{FreqCompensationValue}_n = \text{FreqScaleFactor}_n \times \text{FreqCompensationValue}_{n-1} \quad (56)$$

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more Sync cycles.

15.3.6.2 Transmit Timestamping

The MAC captures a timestamp when the Start Frame Delimiter (SFD) of a frame is sent. The transmit frames are marked to indicate whether a timestamp should be captured for that frame and written to the extended transmit descriptors that support timestamping. The MAC returns the timestamp automatically to the corresponding transmit descriptor, thus connecting the timestamp with the specific PTP frame. The 64-bit timestamp information is written to the TDES6 and TDES7 fields.

15.3.6.3 Receive Timestamping

The MAC captures the timestamp of all received frames. The MAC does not process the received frames to identify the PTP frames in default timestamping mode (when Advanced Timestamp is disabled). The MAC gives the timestamp and the corresponding status to the TX/RX Controller along with the EOF data. The TX/RX Controller validates and indicates the availability of the timestamp so that the DMA can return the timestamp to the corresponding receive descriptor. The 64-bit timestamp information is written to the RDES6 and RDES7 fields. The timestamp is written only to the receive descriptor for which the Last Descriptor status field has been set to 1 (the EOF marker). When the timestamp is not available (for example, because of an RX FIFO overflow), an all '1s' pattern is written to the descriptors (RDES6 and RDES7), indicating that the timestamp is not correct. If timestamping is disabled, the DMA does not alter RDES6 or RDES7. RDES0[7] indicates whether the timestamp is updated in RDES6 and RDES7.

15.3.6.4 Timestamp Error Margin

According to the IEEE 1588 specifications, a timestamp must be captured at the SFD of the transmitted and received frames at the MAC interface. Because the reference timing source, MOSC, is taken as different from MAC reference clocks, a small error margin is introduced, because of the transfer of information across asynchronous clock domains. In the transmit path, the captured and reported timestamp has a maximum error margin of 2 PTP (MOSC) clocks. It means that the captured timestamp has the reference timing source (MOSC) value that is given within 2 clocks after the SFD has been transmitted to the PHY. Similarly, in the receive path, the error margin is 3 MAC reference clocks, plus up to 2 PTP clocks. The error margin of the three MAC reference clocks can be ignored by assuming that this constant delay is present in the system (or link) before the SFD data reaches the interface of MAC.

NOTE: When the Ethernet Controller is configured to use the MII interface to an external PHY, the MII clock is provided by the external PHY through EN0RXCK and EN0TXCK. When the Ethernet Controller is connected to the integrated PHY, the reference clock must be 25 MHz because it is also the source to the PHY.

NOTE: When IEEE 1588 timestamping is enabled with internal timestamp, use a PTP clock frequency that is greater than 5 MHz. This is because the SSINC field in the EMACSUBSECINC register limits the PTP frequency that can be used to approximately 4 MHz.

15.3.6.5 IEEE 1588-2008 Advanced Timestamps

In addition to the basic timestamp features mentioned in IEEE 1588-2002 Timestamps, the Ethernet Controller supports the following advanced timestamp features defined in the IEEE 1588-2008 standard:

- Supports the IEEE 1588-2008 (version 2) timestamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end, and peer-to-peer.
- Provides an option to select the node to be a master or slave clock.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to measure subsecond time in digital or binary format.

15.3.6.5.1 Peer-to-Peer Transparent Clock Message Support

The IEEE 1588-2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. [Figure 15-12](#) shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

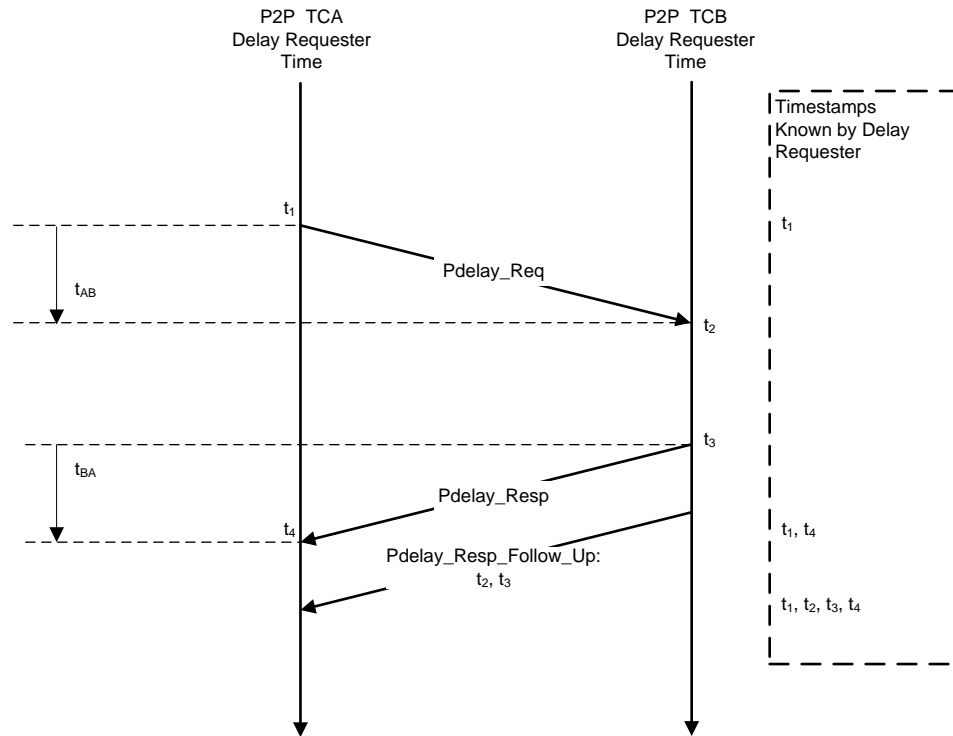


Figure 15-12. Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction

As shown in [Figure 15-12](#), the propagation delay is calculated in the following way:

1. Port-1 issues a Pdelay_Req message and generates a timestamp, t_1 , for the Pdelay_Req message.
2. Port-2 receives the Pdelay_Req message and generates a timestamp, t_2 , for this message.
3. Port-2 returns a Pdelay_Resp message and generates a timestamp, t_3 , for this message.

To minimize errors because of any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The Port-2 returns any one of the following:

- a. The difference between the timestamps t_2 and t_3 in the Pdelay_Resp message.
- b. The difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message.
- c. The timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.
4. Port-1 generates a timestamp, t_4 , on receiving the Pdelay_Resp message.
5. Port-1 uses all four timestamps to compute the mean link delay.

15.3.6.5.2 Advanced Timestamp Supported Clock Types

The Advance Timestamp Module supports an ordinary clock as defined by the IEEE 1588-2008 standard. The characteristics of this clock is as follows:

- Sending and receiving of PTP messages. The timestamp snapshot can be controlled through the Ethernet MAC Timestamp Control (EMACTIMSTCTRL) register, offset 0x700. Timestamp snapshots can be for Sync messages.
- Maintaining the data sets such as timestamp values.

For an ordinary clock, snapshots can be taken of either version 1 or version 2 PTP types but not both. Selecting between the two is controlled by the PTPVER2 bit of the EMACTIMSTCTRL register.

15.3.6.5.3 PTP Processing and Control

There are fields in an Ethernet Payload that the Ethernet Controller can use to detect the PTP packet type and control the snapshot to be taken. These fields are different depending on whether the PTP frames are:

- PTP frames over IPv4
- PTP frames over IPv6
- PTP frames over Ethernet

The PTPIPv4, PTPIPv6, or PTPETH bits can be set depending on which PTP processing is required.

15.3.6.5.4 Reference Timing Source

The MAC supports the following reference timing source features defined in the IEEE 1588-2008 standard:

- 80-bit timestamp
- Fixed Pulse-Per-Second PPS0 Output
- Flexible Pulse-Per-Second PPS0 Output

15.3.6.5.4.1 80-Bit Timestamp

The MAC supports an 80-bit timestamp with a lengthened seconds integer portion which is 48-bits wide. The Ethernet MAC System Time - Seconds (EMACTIMSEC) register at offset 0x708 and the Ethernet MAC System Time-Higher Word Seconds (EMACHWORDSEC) register at offset 0x724 comprise the seconds count. The 32-bit Ethernet MAC System Time - Nanoseconds (EMACTIMNANO) register contains the fractional portion of the timestamp units of nanoseconds. The nanoseconds register supports the following two modes:

- Digital rollover mode: In digital rollover mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF ($10^9 - 1$) nanoseconds.
- Binary rollover mode: In binary rollover mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is approximately 0.466 ns per bit.

Digital or binary rollover mode can be selected by programming the DGTLBIN bit of the EMACTIMSTCTRL register. Note that the timestamp maintained in the MAC is still 64 bits wide, but that the overflow to the upper 16-bits seconds register happens once in 130 years.

15.3.6.5.4.2 Fixed Pulse-Per-Second Output

The Ethernet MAC module supports a pulse-per-second output feature such that the signal, EN0PPS, can indicate one second intervals. The frequency of the EN0PPS output can be changed by setting the PPSCCTRL bit field of the Ethernet MAC PPS Control (EMACPPSCTRL) register, offset 0x72C.

15.3.6.5.4.3 Flexible Pulse-Per-Second Output

The Ethernet MAC can control the following features of the EN0PPS output:

- Start or stop time
- The start point of a single pulse and the start and stop points of the pulse train in terms of a 64-bit system time. The EMACTIMESEC and EMACTIMNANO registers are used to program the start and stop times.
- The stop time can be programmed in advance of starting.
- The signal width. The rising edge and the corresponding falling edge of the EN0PPS output can be programmed in terms of number of units of subsecond increment value programmed in the Ethernet MAC Sub-Second Increment (EMACSUBSECINC) register, offset 0x704.
- The signal interval. The time between the rising edges of EN0PPS signal, in terms of number of units of subsecond increment value can be programmed in the Ethernet MAC PPS0 Interval (EMACPPS0INTVL) register, offset 0x760. You can program the interval between pulses from 1 to $2^{32}-1$ units of subsecond increment value.

- Cancellation of the programmed EN0PPS start or stop request.
- Error indication if the start or stop time being programmed has already passed.

The start time can be programmed in the EMACTARGSEC and EMACTARGNANO registers. The TRGTBUSY bit in the EMACTARGNANO register indicates when the value is synchronized to the PTP clock domain. When this bit is clear, a new start time can be programmed, even before the earlier start time has elapsed. The start or stop time should be programmed with advanced system time to ensure proper EN0PPS signal output. If the application programs a start or stop time that has already elapsed, then the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

For a flexible EN0PPS output, the EMACPPS0INTVL and EMACPPS0WIDTH registers can be configured. The PPS0WIDTH and PPS0INT fields are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to have a EN0PPS pulse width of 80 ns and interval of 120ns, with the PTP reference clock of 25MHz, you should program the width and interval to values 1 and 2, respectively. Note that the PPS0WIDTH and PPS0INT value must be programmed as one less than the required interval. Before giving the command to trigger a pulse or pulse train on the EN0PPS output, the interval and width of the PPS signal output should be programmed or updated.

15.3.6.5.5 Advanced Timestamp Transmit Path Functions

The only aspect of the transmit path that changes with advanced timestamp is the descriptor, which extends to 32-bytes long.

When you enable the advanced timestamp feature, the structure of the descriptor changes. The advanced timestamp feature is supported only through the enhanced descriptors format. The descriptor is 32-bytes long (eight words) and the snapshot of the timestamp is written in descriptor TDES6 and TDES7.

15.3.6.5.6 Advanced Timestamp Receive Path Functions

When the advanced timestamp feature is enabled, the MAC processes the received frames to identify valid PTP frames. The snapshot of the time sent to the application can be configured to:

- Enable snapshot for all frames
- Enable snapshot for IEEE 1588 version 2 or version 1 timestamp
- Enable snapshot for PTP frames transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received frame for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type. This controls the type of messages for which snapshots are taken

The MAC provides the timestamp, along with EOF to the TX/RX Controller. The DMA returns the timestamp inside the corresponding receive descriptor.

15.3.7 Frame Filtering

The following types of filtering are available for receive frames:

- Source Address (SA) or Destination Address (DA) filtering
- VLAN Filtering

The frame filtering supports a sequence where the packet is not forwarded to VLAN filtering if it does not pass the SA or DA filtering first.

15.3.7.1 VLAN Filtering

The Ethernet MAC provides VLAN Tag Perfect Filtering and VLAN Tag Hash Filtering. In VLAN tag perfect filtering, the MAC compares the VLAN tag of the received frame and provides the VLAN frame status to the application. Based on the programmed mode of the ETV bit in the EMACVLANTG register, the MAC compares the lower 12 bits or all 16 bits of the received VLAN tag to determine the perfect match. If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged frames along with VLAN tag match status and drops the VLAN frames that do not match. Inverse matching for VLAN frames can also be enabled by setting the VTIM bit of the Ethernet MAC VLAN Tag (EMACVLANTG) register, offset 0x01C. In addition, matching of S-VLAN tagged frames along with the default C-VLAN tagged frames can be enabled by setting the ESVL bit of the EMACVLANTG register. The VLAN frame status bit (Bit 10 of RDES0) indicates the VLAN tag match status for the matched frames.

NOTE: The Source or Destination Address filter has precedence over the VLAN tag filters. A frame which fails the Source or Destination Address filter is dropped irrespective of the VLAN tag filter results.

The MAC provides VLAN tag hash filtering with a 16-bit Hash table. The MAC performs the VLAN hash matching based on the VTHM bit of the EMACVLANTG register. If the VTHM bit is set, the most significant four bits of VLAN tag's CRC-32 are used to index the content of the Ethernet MAC VLAN Hash Table (EMACVLANTHASH) register, offset 0x588. A value of 1 in the EMACVLANTHASH register, corresponding to the index, indicates that the VLAN tag of the frame matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged frame should be dropped. The MAC also supports the inverse matching of the VLAN frames. In the inverse matching mode, when the VLAN tag of a frame matches the perfect or hash filter, the packet should be dropped. If the VLAN perfect and VLAN hash match are enabled, a frame is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and hash filters indicate mismatch. [Table 15-18](#) shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of the EMACFRAMEFLTR register is set, all frames are received and the VLAN match status is indicated in Bit 10 of Receive Descriptor word 0 (RDES0). When the RA bit is not set and the VTFE bit in the EMACFRAMEFLTR register is set, the frame is dropped if the final VLAN match status is fail. In [Table 15-18](#), value X means that this column can have any value. When the VL field is programmed to 0x0 in EMACVLANTG register, all VLAN-tagged frames are considered as perfect matched but the status of the VLAN hash match depends on the VLAN hash enable (VTHM) bit and VLAN inverse filter (VTIM) bit.

Table 15-18. VLAN Match Status

VLAN ID (VL Field)	VLAN Perfect Filter Match Status (VPF)	VLAN HASH Enable Bit (HPF Bit in EMACFRAMEFLTR)	VLAN Hash Filter Match Status (VTHM)	VLAN Inverse Filter Bit (VTIM)	Final VLAN Match Status
VL = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VL != 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

15.3.8 Source Address, VLAN, and CRC Insertion, Replacement or Deletion

The MAC supports the following functions for transmit frames:

- Source address insertion or replacement
- VLAN insertion, replacement, or deletion
- CRC replacement

15.3.8.1 Source Address Insertion or Replacement

Software can use the SA insertion or replacement feature to instruct the MAC to do the following for transmit frames:

- Insert the content of the MAC Address Registers in the SA field.
- Replace the content of the SA field with the content of the MAC Address Registers.

The software can enable the SA insertion or replacement for all transmitted frames or selective frames:

- To enable SA insertion or replacement feature for all frames, program the SADDR field of the Ethernet MAC Configuration (EMACCFG) register.
- To enable SA insertion or replacement for selective frames, program the SA Insertion Control field (TDES1 bits [31:29]) in the first transmit descriptor of the frame. When Bit 31 of TDES1 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When Bit 31 of TDES1 is reset, it indicates insertion or replacement by MAC Address 0 registers.

When SA insertion is enabled, the application should ensure that the frames that are sent to the MAC do not have the SA field. The MAC does not check the presence of SA field in the transmit frame and just inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application should ensure that the frames that are sent to the MAC have the SA field. The MAC just replaces the six bytes, following the Destination Address field in the transmit frame, with the content of the MAC Address Registers.

15.3.8.2 VLAN Insertion, Replacement or Deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for transmit frames:

- Insert or replace the VLAN Type field (C-VLAN or S-VLAN indicated by the CSVL bit of the Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP), MAC offset 0x584) and the VLAN Tag field in the transmit frame with the VLT field of the EMACVLNINCREP register.
- Delete the VLAN Type and VLAN Tag fields in the transmit frame.

The software can enable the VLAN insertion, replacement, or deletion feature for all transmitted frames or selective frames. To enable this function for all transmit frames, configure the VLT field in the EMACVLNINCREP register.

When VLAN replacement or deletion is enabled, the MAC checks the presence of the VLAN Type field (0x8100 or 0x88A8), after the Destination address (DA) and SA fields, in the transmit frame. The replace or delete operation does not occur if the VLAN Type field is not detected in the two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the transmit frame and just inserts the VLAN Type and VLAN Tag fields.

15.3.8.3 CRC Replacement

The software can use the CRC replacement feature to instruct the MAC to replace the FCS field in the transmit frame with the CRC computed by the MAC. This feature works on a per-frame basis. The CRC replacement control field in the Transmit Descriptor Word 0 (TDES0) can be programmed to enable this for a frame. This feature is valid only when the Disable CRC control (Bit 27 of TDES0) is enabled. If SA or VLAN insertion control is enabled, the MAC appends or replaces the FCS field with the computed CRC when Disable CRC Control is enabled or disabled, respectively.

Table 15-19. CRC Replacement Based on Bit 27 and Bit 24 of TDES0

Bit 27 (DC)	Bit 24 (CRCR)	Description
0	X	Append CRC When DC = 0, the MAC appends the computed CRC irrespective of the CRCR setting.
1	1	Replace CRC
1	0	No operation (user has appended CRC)

15.3.9 Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the MAC Checksum Offload Engine (COE) supports checksum calculation and insertion in the transmit path, and error detection in the receive path.

15.3.9.1 Transmit Checksum Offload Engine

The checksum for TCP, UDP, or ICMP is calculated over a complete frame, and then inserted into its corresponding header field. Because of this requirement this function is enabled only when the TX FIFO is configured for the store-and-forward mode (the TSF bit is set in the EMACDMAOPMODE register).

NOTE: The TX FIFO must be deep enough to store a complete frame before the frame is transferred to the MAC when the checksum offload is being used. If space is not available to accept the programmed burst length of data, the TX/RX Controller starts reading to avoid deadlock. When reading starts, the checksum offload fails and the consequently all succeeding frames may be corrupted because of improper recovery. Therefore, checksum insertion must only be enabled in frames that are less than $2048 - ((PBL + 3) \times 4)$ bytes in size, where PBL is the Programmable Burst Length field in the EMACDMAOPMODE register.

15.3.9.2 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The checksum offload engine detects an IPv4 datagram when the Ethernet frame's Type field has the value 0x0800 and the IP datagram's Version field has the value 0x4. The input frame's checksum field is ignored during calculation and replaced with the calculated value. IPv6 headers do not have a checksum field. Therefore, the checksum offload does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 16 in TDES0). This status bit is set whenever the values of the Ethernet Type field and the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
 - The received Ethernet type is 0x0800, but the IP header's Version field is not equal to 0x4.
 - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
 - The total frame length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
 - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
 - The frame ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

If the checksum offload engine detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

15.3.9.3 Receive Checksum Offload Engine

Both IPv4 and IPv6 frames in the received Ethernet frames are detected and processed for data integrity. The receive checksum feature can be enabled by setting the IPC bit of the Ethernet MAC Configuration (EMACCFG) register. The EMAC receiver identifies IPv4 or IPv6 frames by checking for value 0x0800 or 0x86DD, respectively, in the received Ethernet frames' Type field. This identification also applies to single VLAN-tagged frames. The offline receive checksum engine calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the Length field of the IPv4 header or when fewer than 20 bytes are available in an IPv4 or IPv6 header. This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

15.3.10 MAC Management Counters

The MAC Management Counters (MMC) module maintains a set of registers for gathering statistics on the received and transmitted frames. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (one for receive and one for transmit), and two 32-bit registers containing masks for the Interrupt register (one for receive and one for transmit). The MMC counters are free running and start counting when a corresponding frame is received or transmitted. The MMC counter registers provided are as follows:

- Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB)
- Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL)
- Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL)
- Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG)
- Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB)
- Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR)
- Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR)
- Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI)

15.3.11 Power Management Module

The power management (PMT) module supports the reception of network remote wake-up frames and AMD Magic Packet frames. The PMT module does not perform the clock gate function, but generates interrupts for remote wake-up frames and magic packets that the MAC receives.

When the application enables the power-down mode in the PMT module by setting the PWRDWN bit in the Ethernet MAC PMT Control and Status Register (EMACPMTCTLSTAT) register, MAC offset 0x02C, the MAC drops all received frames and does not forward any frame to the TX/RX Controller RxFIFO or the application. The MAC comes out of the power-down mode only when a magic packet or a remote wake-up frame is received and the corresponding detection is enabled.

15.3.11.1 Remote Wake-Up

The Remote Wake-Up register bank is made up of eight 32-bit registers. It is loaded by writing the Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF) register eight times. To load values in the EMACRWUFF register, the entire register must be written. The first write is assigned to register 0 of the bank, then register 1 and so on. The Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF) register is read the same way. The current pointer value of the bank is updated in the Remote Wake-Up FIFO Pointer (RWKPTR) field of the Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT) register.

Filter 0 Byte Mask							
Filter 1 Byte Mask							
Filter 2 Byte Mask							
Filter 3 Byte Mask							
RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command
Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
Filter 1 CRC - 16				Filter 0 CRC - 16			
Filter 3 CRC - 16				Filter 2 CRC - 16			

Figure 15-13. Wake-Up Frame Filter Register Bank

15.3.11.1.1 Filter n Byte Mask

The Filter n Byte Mask registers of the Remote Wake-Up register define the bytes of the frame that are examined by filter n (0, 1, 2, and 3) in order to determine whether or not a frame is a remote wake-up frame. The most significant bit (bit 31) of each mask must be zero. Bits [30:0] are the Byte Mask. If bit j of the Byte Mask is set, then the CRC block processes the Filter n Offset + j of the incoming frame; otherwise Filter n Offset + j is ignored.

15.3.11.1.2 Filter n Command

The 4-bit Filter n Command field controls the Filter n operation in the following way:

- Filter n Command Bit 3 specifies the address type of the pattern. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame.
- Filter n Command Bit 2 and Bit 1 are reserved.
- Filter n Command Bit 0 is the enable for Filter n. If bit 0 is not set, filter n is disabled.

15.3.11.1.3 Filter n Offset

The Filter n Offset register defines the offset (within the frame) from which the filter n examines the frames. This 8-bit pattern offset is the offset for the filter n first byte to be examined. The minimum allowed offset is 12, which refers to the 13th byte of the frame. The offset value 0 refers to the first byte of the frame.

15.3.11.1.4 Filter n CRC-16

The Filter n CRC-16 register contains the CRC_16 value calculated from the pattern and the byte mask programmed to the wake-up filter register block.

15.3.11.2 Remote Wake-Up Frame Detection

When the MAC is in sleep mode and the remote wake-up frame enable bit, WUPFREN, is set in the Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT) register, the normal operation is resumed after a remote wake-up frame is received. The application writes all eight wake-up filter registers, by performing eight sequential writes to the Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF) register. The Power Management (PMT) block supports four programmable filters that allow support of

different receive frame patterns. If the incoming frame passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, then the MAC identifies the frame as wake-up frame. The Filter Offset determines the offset from which the frame is to be examined. The Filter Byte Mask determines which bytes of the frame must be examined. The 31st bit of Byte Mask must be set to zero. The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The remote wake-up frame is checked only for length error, FCS error, dribble bit error, MII error, and collision. In addition, the remote wake-up frame is checked to ensure that it is not a runt frame. Even if the remote wakeup frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. The remote wake-up frame detection is updated in the Ethernet MAC PMT Control and Status (EMACPMCTLSTAT) register for every remote wake-up frame received. If the PMT interrupt is enabled in the Ethernet MAC Interrupt Mask (EMACIM) register, a PMT interrupt is asserted and the EMACPMCTLSTAT register can be read to determine reception of a remote wake-up frame.

15.3.11.3 Magic Packet Detection

The magic packet frame is based on a method that uses Advanced Micro Device's magic packet technology to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network. The MAC checks only those magic packets that are addressed to the MAC or a broadcast address to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or broadcast) are checked to determine whether they meet the remote wake-up frame data format of 6 bytes of all ones followed by a MAC Address appearing 16 times. The application enables the magic packet wake-up by setting the magic packet enable bit, MGKPKTEN, of the Ethernet MAC PMT Control and Status (EMACPMCTLSTAT) register. The power management block constantly monitors each frame addressed to the node for a specific magic packet pattern. Each frame received is checked for a 0xFFFF.FFFF.FFFF pattern following the destination and source address field. The power management block then checks the frame for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 0xFFFF.FFFF.FFFF pattern in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (0xFFFF.FFFF.FFFF). The device can also accept a multicast frame, as long as the 16 duplications of the MAC address are detected. If the MAC address of a node is 0x0011.2233.4455, then the MAC scans for the following data sequence:

Destination Address Source Address.. FF FF FF FF FF FF

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 CRC

The magic packet detection is updated in the EMACPMCTLSTAT register for the received magic packet. If the PMT interrupt is enabled in the Ethernet MAC Interrupt Mask (EMACIM) register, a PMT interrupt is asserted and the EMACPMCTLSTAT register can be read to determine whether a magic packet frame has been received.

15.3.11.4 Power Management Interrupts

The PMT interrupt signal can be asserted when a valid remote wake-up frame or magic packet is received. The PMT interrupt signal restores the application clock and TX clock to the MAC. When the Ethernet MAC PMT Control and Status (EMACPMCTLSTAT) register is read, the PMT interrupt is cleared in the EMACRIS register at least after four clock cycles of RX clock. When software resets the PWRDWN bit in the Ethernet MAC PMT Control and Status (EMACPMCTLSTAT) register, the MAC comes out of the power-down mode, but this event does not generate a PMT interrupt.

15.3.11.5 Power-Down and Wake-Up Sequence

The recommended power-down and wake-up sequence is as follows:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. These transmissions can be detected when TI is set in the Ethernet MAC DMA Interrupt Status (EMACDMARIS) register.

2. Disable the MAC transmit and receive state machine by clearing the TE and RE bits in the Ethernet MAC Configuration (EMACCFG) register.
3. Wait until the RX DMA empties all the frames from the Rx FIFO to system memory by polling the RXF field of the Ethernet MAC Status (EMACSTATUS) register.
4. Enable a power management mode by setting the magic packet, global unicast or remote wake-up enable bit in the EMACPMTCTLSTAT register.
5. Enable the MAC receive state machine in the EMACCFG register and enter the Power-Down mode by setting the PWRDWN bit in the EMACPMTCTLSTAT register.
6. On receiving a valid remote wake-up frame, the PMT interrupt is set in the EMACRIS register and the Ethernet MAC exits the Power-Down mode.
7. Read the EMACPMTCTLSTAT register to clear the PMT interrupt, then enable the other modules in the system and resume normal operation.

15.3.12 Serial Management Interface

The Ethernet MAC has the ability to read or write to the PHY registers through the EN0MDIO and EN0MDC signals of the Serial Management Interface defined by the IEEE 802.3 standard. The internal EN0MDIO and EN0MDC signals connect to the integrated PHY as well as to the external EN0MDIO and EN0MDC pins. The EN0MDC signal is a 2.5-MHz clock that is sourced from System Clock (SYSCLK) and then divided down to the required frequency by programming the CR field in the Ethernet MAC MII Address (EMACMIIADDR) register. To access the integrated PHY, the PLA field in the EMACMIIADDR register must be 0x0. The available addresses for external PHYs are 0x01 to 0x1F.

15.3.13 Reduced Media Independent Interface (RMII)

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Ethernet MACs. According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces such as switches, the number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port — a 62.5% decrease in pin count.

The RMII module has the following features:

- Supports both 10-Mbps and 100-Mbps operating rates
- Provides independent, two-bit wide transmit and receive paths

Each nibble is transmitted on the RMII two bits at a time. For a nibble {D3, D2, D1, D0}, the data is transferred as {D1, D0} followed by {D3, D2}.

15.3.14 Interrupt Configuration

Interrupts can be generated from the MAC as a result of various events in the MAC and submodules. MAC interrupts are enabled or disabled in the Ethernet MAC Interrupt Mask (EMACIM) register, MAC offset 0x03C. Each interrupt event can be masked by setting the corresponding mask bit in the EMACIM register.

The interrupt register bits in the Ethernet MAC Raw Interrupt Status (EMACRIS) register only indicate the submodule from which the event is reported. The application must read the corresponding status registers to clear the interrupt.

15.4 Ethernet PHY

The integrated PHY supports 10Base-T and 100Base-TX signaling. It integrates all the physical-layer functions needed to transmit and receive data on standard twisted-pair cables. The PHY directly interfaces to the integrated Media Access Controller (MAC).

The Ethernet PHY uses mixed-signal processing to perform equalization, data recovery, and error correction to achieve robust operation over CAT5 twisted-pair wiring. It not only meets the requirements of IEEE 802.3, but maintains high margins in terms of alien cross-talk. The following highlights the features of the PHY module:

- Cable Diagnostics

- Programmable Fast Link Down Modes
- Auto-MDIX for 10/100Mbps
- Energy Detection Mode
- Serial Management Interface
- IEEE 802.3u Auto-Negotiation and Parallel Detection
- IEEE 802.3u ENDEC, 10Base-T Transceivers and Filters
- IEEE 802.3u PCS, 100Base-TX Transceivers
- Integrated ANSI X3.263 Compliant TP-PMD Physical Sublayer with Adaptive Equalization and Baseline Wander Compensation
- Three programmable LEDs that support detection of Link OK, 10/100Mbps activity, TX/RX transfers, collisions and full duplex mode

15.4.1 Integrated PHY Block Diagram

Figure 15-14 shows the internal PHY integration. Note that the reference clock input comes from an external 25 MHz ± 50 ppm crystal or oscillator connected to the MOSC signals.

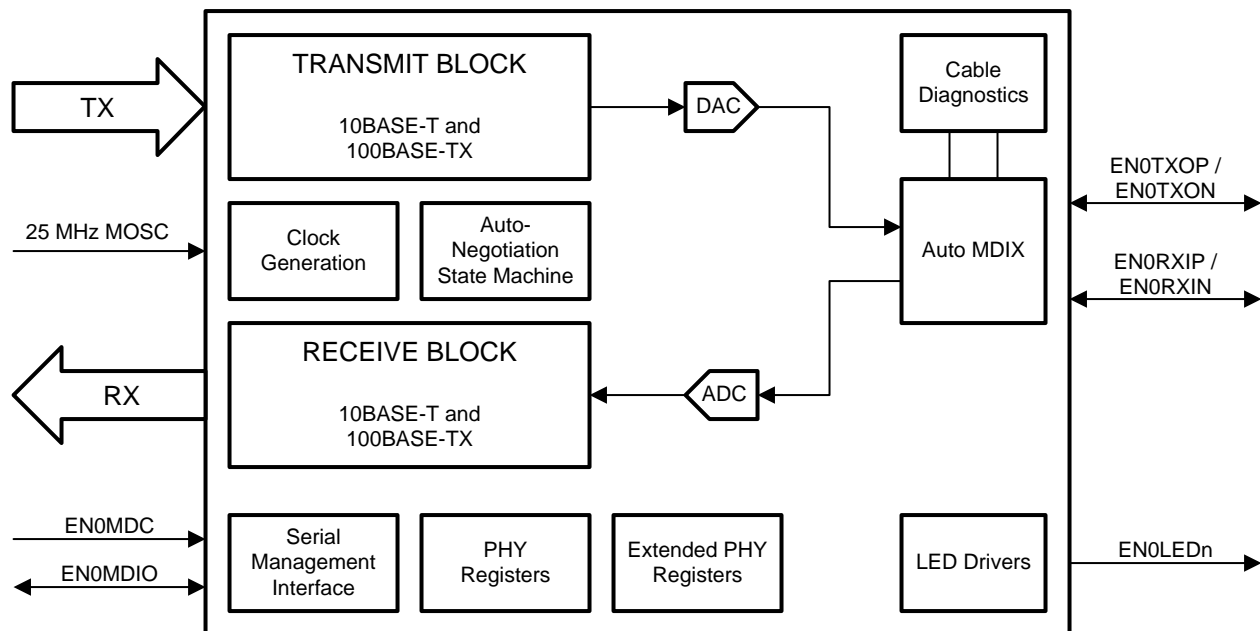


Figure 15-14. Integrated PHY Diagram

15.4.2 Functional Description

The following sections describe the functional characteristics of the integrated PHY.

15.4.2.1 Auto-Negotiation

The Ethernet PHY can auto-negotiate to operate in 10Base-T or 100Base-TX. When the Ethernet PHY is enabled or on the deassertion of software reset, the Ethernet MAC Peripheral Configuration Register (EMACPC) register, offset 0xFC4, is configured such that auto-negotiation is enabled and the auto negotiation mode (ANMODE) bit field is programmed to 10Base-T, Half/Full-Duplex 100Base-TX, Half/Full-Duplex. With auto-negotiation enabled, the PHY negotiates with the link partner to determine the speed and duplex mode with which to operate. If the link partner is unable to auto-negotiate, the PHY goes into parallel-detect mode to determine the speed of the link partner. Under parallel-detect mode, the duplex mode is fixed at half-duplex. The PHY supports four different Ethernet protocols (10Mbps Half-Duplex, 10Mbps Full-Duplex, 100Mbps Half-Duplex, and 100Mbps Full-Duplex). Auto-Negotiation selects the

highest performance protocol based on the advertised ability of the Link Partner. If a different auto-negotiation configuration is required other than the reset initialization values, the application can customize the configuration of the ANEN bit and ANMODE bit field as described in [Section 15.5.1.2](#). The values of ANEN and ANMODE determine whether the PHY is forced into a specific mode, or if Auto-negotiation advertises a specific ability (or abilities) as listed in [Table 15-20](#) and [Table 15-21](#):

Table 15-20. Forced Mode Configurations

ANEN Value	ANMODE	Forced Mode
0	0x0	10Base-T, Half-Duplex
0	0x1	10Base-T, Full-Duplex
0	0x2	100Base-TX, Half-Duplex
0	0x3	100Base-TX, Full-Duplex

Table 15-21. Advertised Mode Configurations

ANEN Value	ANMODE	Advertised Mode
1	0x0	10Base-T, Half/Full-Duplex
1	0x1	100Base-TX, Half/Full-Duplex
1	0x2	10Base-T, Half Duplex 100Base-TX, Half Duplex
1	0x3	10Base-T, Half/Full-Duplex 100Base-TX, Half/Full-Duplex

15.4.2.2 Auto-MDIX

The PHY automatically determines whether or not it needs to cross over between pairs so that an external crossover cable is not required. If the PHY communicates with a device that implements MDI/MDIX crossover, a random algorithm as described in IEEE 802.3 determines which device performs the crossover. Auto-MDIX is enabled by default at reset. If a different auto-MDIX configuration is required other than the reset initialization, the application can customize the configuration as described in [Section 15.5.1.2](#). Neither Auto-Negotiation nor Auto-MDIX is required to be enabled in forcing crossover of the MDI pairs. Auto-MDIX can be used in the forced 100Base-TX mode. Because in modern networks all the nodes are 100Base-TX, having the Auto-MDIX working in the forced 100Base-TX mode resolves the link faster without the need for the long Auto-Negotiation period.

15.4.2.3 Isolate Mode

The PHY can be put into Isolate mode by writing ISOLATE bit of the Ethernet PHY Basic Mode Control - MR0 (EPHYBMCR) register, address 0x000. When in the Isolate mode, the PHY does not respond to packet data present at the internal interface to the Ethernet MAC. When in isolate mode, the PHY continues to respond to all management transactions and the PMD output pair does not transmit packet data, but continues to source 100Base-TX scrambled idles or 10Base-T normal link pulses. The PHY can auto-negotiate or parallel detect on the receive signal at the PMD input pair. A valid link can be established for the receiver even when the PHY is in Isolate mode.

15.4.2.4 LED Interface

The PHY supports three configurable light emitting diode (LED) pins to indicate link status of a port. The Ethernet PHY LED Configuration - MR37 (EPHYLEDCFG) register, address 0x025 can be used to assign different functions to each LED. Each LED can be configured to be active during one of these events:

- Link OK (0x0)
- RX/TX Activity (0x1)
- TX Activity (0x2)
- RX Activity (0x3)
- Collision (0x4)
- 100-BASE TX speed (0x5)

- 10-Base TX speed (0x6)
- Full Duplex (0x7)
- Link OK/Blink on TX/RX Activity (0x8)

At reset, LED0 is initialized to display Link OK and LED1 and LED 2 are initialized to the RX/TX activity encoding. The blink rate of the LEDs can be set by programming the BLINKRATE bit field of the Ethernet PHY LED Control - MR24 (EPHYLEDCCR) register, address 0x018.

15.4.2.5 PHY Address

The integrated PHY address is 0x00. To access the integrated PHY registers through the internal MAC MDIO interface, the PLA bit field of the Ethernet MAC MII Address (EMACMIIADDR) register must be 0x00 and the MII field should be programmed to the desired address value.

15.4.2.6 Serial Management Interface

The Ethernet MAC module has the capability of programming the integrated PHY registers and extended registers through an internal Serial Management Interface (SMI). The SMI is compatible with IEEE 802.3-2002 clause 22.

The SMI includes an MDC management clock input and management MDIO data pin to the Ethernet PHY. The internal MDC clock is sourced by the system clock and then divided down to the required 2.5-MHz maximum clock frequency by setting the CR bit in the Ethernet MAC MII Address (EMACMIIADDR) register. The MDC is not expected to be continuous, and can be turned off by the external management entity when the bus is idle. The MDIO is sourced by the integrated MAC and by the PHY. The data on the MDIO pin is latched on the rising edge of the MDC clock. The PHY's physical address is 0x00.

15.4.2.7 Extended Address Space Access

The PHY SMI function supports read/write accesses to the extended register set using Ethernet PHY Register Control - MR13 (EPHYREGCTL) register and the Ethernet PHY Address or Data - MR14 (EPHYADDR) registers. Accessing the standard register set, MDIO registers 0 to 31, can be performed using the normal direct MDIO access or the indirect method, except for the EPHYREGCTL (0x00D) and the EPHYADDR (0x00E) which can be accessed only using the normal MDIO transaction. The SMI function ignores indirect accesses to these registers.

The EPHYREGCTL register is the MDIO Manageable MMD access control. In general, register EPHYREGCTL[4:0] is the device address DEVAD that directs any accesses of EPHYADDR register to the appropriate MMD. Specifically, the PHY uses the vendor specific DEVAD[4:0] = 0xF for accesses. All accesses through registers EPHYREGCTL and EPHYADDR registers should use this DEVAD. Transactions with other DEVAD are ignored. The EPHYREGCTL[15:14] register holds the access function:

- EPHYREGCTL[15:14] = 0x0: A write to EPHYADDR modifies the extended register set address register. This address register must be initialized in order to access any of the registers within the extended register set.
- EPHYREGCTL[15:14] = 0x1: A read/write to EPHYADDR operates on the register within the extended register set selected (pointed to) by the value in the address register. The address register contents (pointer) remain unchanged.
- EPHYREGCTL[15:14] = 0x2: A read/write to EPHYADDR operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for both reads and writes, the value in the address register is incremented.
- EPHYREGCTL[15:14] = 0x3: A read/write to EPHYADDR operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for write accesses only, the value in the address register is incremented. For read accesses, the value of the address register remains unchanged.

The following sections describe how to perform operations on the extended register set using the EPHYREGCTL and EPHYADDR registers.

15.4.2.7.1 Write to PHY Registers

The following describes the steps to write to a PHY register.

1. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
2. Write the data to be written to the PHY register in the EMACMIIDATA register.
3. Initiate write by programming the EMACMIIADDR register fields as follows:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Address of the PHY register to be written.
 - CR: Clock Reference for the MDIO interface.
 - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.

15.4.2.7.2 Write to Extended PHY Registers

The following describes the steps to write to an extended PHY register.

1. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
2. The EMACMIIDATA register should be written with the value to be passed into the EPHYREGCTL register. The EPHYREGCTL register is used for extended PHY register accesses. The DEVAD field of the EPHYREGCTL register identifies the device address, which is 0x1F, for the integrated PHY. The FUNC field of the EPHYREGCTL register should be set to 0x0 to indicate a write to an extended register address.
3. Initiate write by programming the EMACMIIADDR register fields as follows:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Address of the PHY register to be written. In this case, it should be the address of the EPHYREGCTL register, 0xD.
 - CR: Clock Reference for the MDIO interface.
 - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.
4. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
5. The EMACMIIDATA register should be written with the address of the extended register to be accessed. This value is written to the EPHYADDR register.
6. Initiate write by writing the EMACMIIADDR register fields:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Address of the PHY register to be written. In this case, it should be the address of the EPHYADDR register, 0xE.
 - CR: Clock Reference for the MDIO interface.
 - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.
7. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
8. The EMACMIIDATA register should be written with the value to be passed into the EPHYREGCTL register. The DEVAD field of the EPHYREGCTL register identifies the device address, which is 0x1F, for the integrated PHY. The FUNC field of the EPHYREGCTL register should be set to 0x1 to indicate

a write to an extended register address with no increment.

9. Initiate write by writing the EMACMIIADDR register fields:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Address of the PHY register to be written. In this case, it should be the address of the EPHYADDR register, 0xD.
 - CR: Clock Reference for the MDIO interface.
 - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.
10. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
11. The EMACMIIDATA register should be programmed with the data to be written to the EPHYADDR register which is transferred to the previously selected extended PHY register.
12. Initiate write by writing the EMACMIIADDR register fields:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Address of the PHY register to be written. In this case, it should be the address of the EPHYADDR register, 0xD.
 - CR: Clock Reference for the MDIO interface.
 - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.

15.4.2.7.3 Read from PHY Registers

The following describes the steps to read from an extended PHY register.

1. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to read to the PHY registers.
2. Initiate the read by writing the EMACMIIADDR register fields:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Register address of PHY register to be written.
 - CR: Clock Reference for the MDIO interface.
 - MIIW: Write/Read Initiation. This bit is programmed to a 0 to indicate that a read operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a read operation. The EMAC clears this bit when the write has been transmitted.
3. Wait for the MII interface to complete the read by polling the MIIB bit.
4. When the MIIB is clear, read the contents of the EMACMIIDATA register.

15.4.2.7.4 Read from Extended PHY Registers

The following describes the steps to read from an extended PHY register.

1. Check the MIIB bit in the EMACMIIADDR register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to read to the PHY registers.
2. Initiate the read by writing the EMACMIIADDR register fields:
 - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
 - MII: Register address of PHY register to be written.
 - CR: Clock Reference for the MDIO interface.

- MIIW: Write/Read Initiation. This bit is programmed to a 0 to indicate that a read operation is to be executed.
 - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a read operation. The EMAC clears this bit when the write has been transmitted.
3. Wait for the write to complete by polling the MIIB bit.
 4. When the MIIB is clear, read the contents of the EMACMIIDATA register.

15.4.3 Interface Configuration

Figure 15-15 shows the proper method for interfacing the Ethernet Controller to a 10/100BASE-T Ethernet jack.

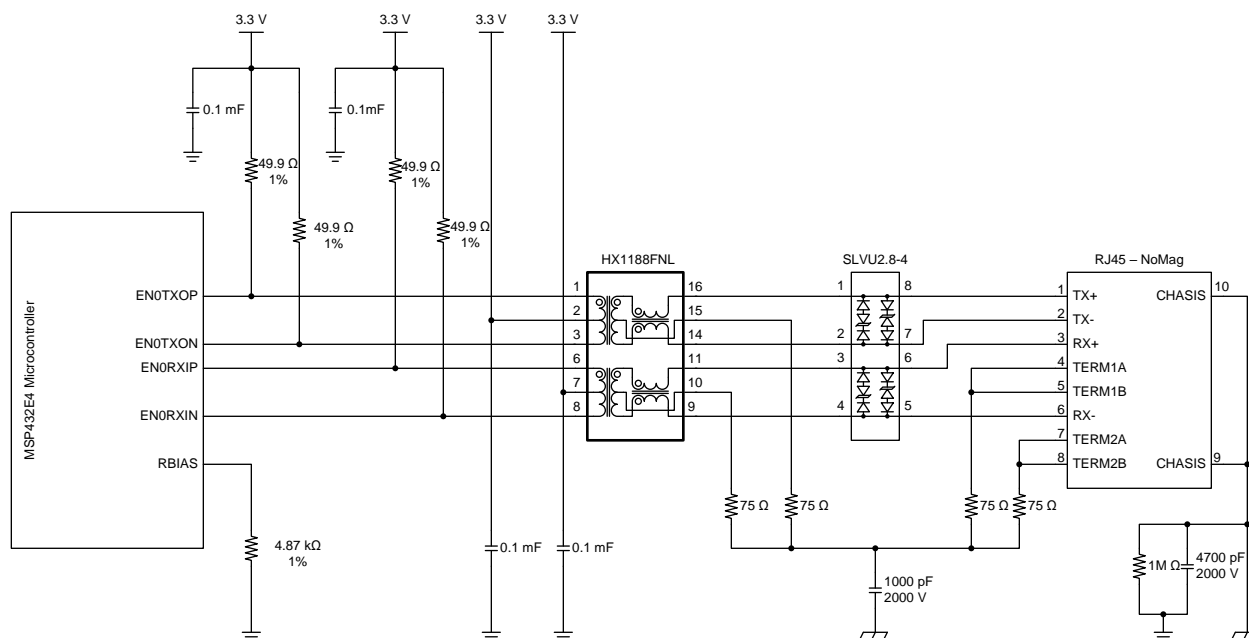


Figure 15-15. Interface to Ethernet Jack

The Ethernet PHY is designed to work with transformers that meet the IEEE 802.3 standard. To utilize the Auto-MDIX (AMDIIX) capability of the Ethernet PHY, a symmetrical transformer is recommended. The Pulse HX1188 transformer has been tested and is known to successfully interface to the Ethernet PHY.

NOTE: If the PHY is not to be used, then the EN0RXIN / EN0TXON EN0RXIP / EN0TXOP pair should be left unconnected and the RBIAS pin should be unconnected as well.

NOTE: When entering hibernation in VDD3ON mode, the supply rails to the Ethernet resistors R1, R2, R3, R4 found in Figure 15-15 must be switched off.

15.5 Initialization and Configuration

The MAC module and registers are enabled and powered at reset. When reset has completed, the application should enable the clock to the Ethernet MAC by setting the R0 bit in the Ethernet Controller Run Mode Clock Gating Control (RCGCEMAC) register at System Control Module offset 0x69C. When the PREMAC register, at System Control offset 0xA9C reads as 0x0000.0001, the EMAC registers are ready to be accessed.

The EMAC interface defaults to MII mode. If RMII mode is required, follow these steps:

1. Enable the external clock source input to the RMII interface signal EN0RREF_CLK by setting both the ECEXT and CLKEN bit in the in the Ethernet Clock Configuration (EMACCC) register at offset 0xFC8.

The external clock source must be 50 MHz with a frequency tolerance of 50 PPM.

2. Select the RMII interface by programming the PINTFS bit field to 0x4 in the Ethernet Peripheral Configuration (EMACPC) register at offset 0xFC4.
3. Reset the Ethernet MAC to latch the new RMII configuration by setting the SWR bit in the EMACDMABUSMOD register. This bit resets the Ethernet MAC registers in addition to configuring the RMII interface. Software must poll the SWR bit to determine when the new configuration has been registered.

NOTE: After this configuration is active, if the Ethernet MAC is reset by setting the R0 bit in the Ethernet MAC Software Reset (SREMAC) register in the System Control Module, then the interface is set back to its default MII configuration. In this case, the steps listed above must be repeated to return to an RMII interface.

The Initialization for the DMA for the Ethernet MAC is as follows:

1. Write to the Ethernet MAC DMA Bus Mode (EMACDMABUSMOD) register to set Host bus parameters.
2. Write to the Ethernet MAC DMA Interrupt Mask Register (EMACDMAIM) register to mask unnecessary interrupt causes.
3. Create the transmit and receive descriptor lists and then write to the Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR) register and the Ethernet MAC Transmit Descriptor List Address (EMACTXDLADDR) register providing the DMA with the starting address of each list.
4. Write to the Ethernet MAC Frame Filter (EMACFRAMEFLTR) register, the Ethernet MAC Hash Table High (EMACHASHTBLH) and the Ethernet MAC Hash Table Low (EMACHASHTBLL) for desired filtering options.
5. Write to the Ethernet MAC Configuration Register (EMACCFG) to configure the operating mode and enable the transmit operation.
6. Program Bit 15 (PS) and Bit 11 (DM) of the EMACCFG register based on the line status received or read from the PHY status register after auto-negotiation.
7. Write to the Ethernet MAC DMA Operation Mode (EMACDMAOPMODE) register to set Bits 13 and 1 to start transmission and reception.
8. Write to the EMACCFG register to enable the receive operation.

The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

15.5.1 Ethernet PHY Initialization

After reset, when the EMAC is powered and enabled, the EMACPC register default reset value may be sampled and used to configure the PHY. The results of this configuration can also be read in the following EPHY registers:

- EPHYBMCR register (MR0)
- EPHYCFG1 register (MR9)
- EPHYCFG2 register (MR10)
- EPHYCFG3 register (MR11)
- EPHYCTL register (MR25)

The mappings of the EMACPC register bits to the PHY register and bits are as follows:

Table 15-22. EMACPC to PHY Register Mapping

EMACPC Register Bit	Corresponding PHY Register	Corresponding PHY Bit (Bit No.)
PHYEXT	N/A	N/A
DIGRESTART	N/A	N/A

Table 15-22. EMACPC to PHY Register Mapping (continued)

EMACPC Register Bit	Corresponding PHY Register	Corresponding PHY Bit (Bit No.)
NIBDETDIS	EPHYCFG2	ODDNDETDIS (1)
RXERIDLE	EPHYCFG2	RXERRIDLE (2)
ISOMILL	EPHYCFG2	ISOMILL (3)
LRR	EPHYCFG1	LLR (7)
TDRRUN	EPHYCFG1	TDRAR (8)
FASTLDMODE	EPHYCFG3	FLDWNM (4:0)
POLSWAP	EPHYCFG3	POLSWAP (7)
MDISWAP	EPHYCFG3	MDIMDIXS (6)
RBSTMDIX	EPHYCFG1	RAMDIX (5)
FASTMDIX	EPHYCFG1	FAMDIX (6)
MDIXEN	EPHYCTL	AUTOMDI (15)
FASTRXDV	EPHYCFG1	FRXDVDDET (1)
FASTLUPD	EPHYCFG2	FLUPPD (6)
EXTFD	EPHYCFG2	EXTFD (5)
FASTANEN	EPHYCFG1	FASTANEN (4)
FASTANSEL	EPHYCFG1	FANSEL (3:2)
ANEN	EPHYBMCR	ANEN
ANMODE	N/A	N/A
PHYHOLD	N/A	N/A

The MAC module and registers are enabled and powered at reset. When reset has completed and the clock to the Ethernet MAC is enabled by setting the R0 bit in the Ethernet Controller Run Mode Clock Gating Control (RCGCEMAC) register at System Control Module offset 0x69C, the application has the option to enable the PHY with its default interface configuration (as defined by the Ethernet MAC Peripheral Configuration Register (EMACPC) register) or with a custom configuration.

15.5.1.1 Default Configuration

To enable the Ethernet PHY with its default configuration, the steps are as follows:

1. To hold the Ethernet PHY from transmitting energy on the line during configuration, set the PHYHOLD bit to 1 in the EMACPC register.
2. Enable the clock to the PHY module by writing 0x0000.0001 to the Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY) register at offset 0x630. When the R0 bit reads as 1 in the PREPHY register at System Control offset 0xA30, continue initialization.
3. Enable power to the Ethernet PHY by setting the P0 bit in the PCEPHY register at System Control offset 0x930. When the R0 bit reads as 1 in the PREPHY register at System Control offset 0xA30, the PHY registers are ready for programming.

15.5.1.2 Custom Configuration

If a custom configuration of the Ethernet PHY is required, the application can program the configuration registers after reset. The steps for custom configuration are as follows:

1. To hold the PHY from transmitting energy on the line during configuration, set the PHYHOLD bit to 1 in the EMACPC register.
2. Enable the clock to the PHY module by writing 0x0000.0001 to the Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY) register at offset 0x630. When the R0 bit reads as 1 in the PREPHY register at System Control offset 0xA30, continue initialization.
3. Enable power to the Ethernet PHY by setting the P0 bit in the PCEPHY register at System Control offset 0x930. When the R0 bit reads as 1 in the PREPHY register at System Control offset 0xA30, the PHY registers are ready for programming.

4. Once the Ethernet PHY Peripheral Ready (PREPHY) register reads 0x0000.0001, software can write the EMACPC register with the required value.
5. After software configuration is complete, the application must set the DONE bit in the Ethernet PHY Configuration 1 (EPHYCFG1) register at offset 0x009.

NOTE: If a software reset is asserted to the PHY afterwards through the SREPHY register, the custom configuration is lost and the steps described above must be repeated.

15.6 EMAC Registers

[Table 15-23](#) lists the memory-mapped registers for the EMAC. All register offset addresses not listed in [Table 15-23](#) should be considered as reserved locations and the register contents should not be modified.

For the MAC registers, the offsets are relative to the MAC base address of 0x400EC000.

PHY registers are accessed through the EMACMIIADDR register thus the base address is N/A (not applicable). The Ethernet MAC MII Address (EMACMIIADDR) register is used to access MII Management registers on the external PHY device. The PLA field in the EMACMIIADDR register supports PHY addresses 1 to 31. See [Section 15.7](#) for details of the PHY registers.

Table 15-23. EMAC Registers

Offset	Acronym	Register Name	Section
0x0	EMACCFG	Ethernet MAC Configuration	Section 15.6.1
0x4	EMACFRAMEFLTR	Ethernet MAC Frame Filter	Section 15.6.2
0x8	EMACHASHTBLH	Ethernet MAC Hash Table High	Section 15.6.3
0xC	EMACHASHTBLL	Ethernet MAC Hash Table Low	Section 15.6.4
0x10	EMACMIIADDR	Ethernet MAC MII Address	Section 15.6.5
0x14	EMACMIIDATA	Ethernet MAC MII Data Register	Section 15.6.6
0x18	EMACFLOWCTL	Ethernet MAC Flow Control	Section 15.6.7
0x1C	EMACVLANTG	Ethernet MAC VLAN Tag	Section 15.6.8
0x24	EMACSTATUS	Ethernet MAC Status	Section 15.6.9
0x28	EMACRWUFF	Ethernet MAC Remote Wake-Up Frame Filter	Section 15.6.10
0x2C	EMACPMTCTLSTAT	Ethernet MAC PMT Control and Status	Section 15.6.11
0x30	EMACLPICLSTAT	LPI Control and Status	Section 15.6.12
0x34	EMACLPITIMERCTRL	LPI Timers Control	Section 15.6.13
0x38	EMACRIS	Ethernet MAC Raw Interrupt Status	Section 15.6.14
0x3C	EMACIM	Ethernet MAC Interrupt Mask	Section 15.6.15
0x40	EMACADDR0H	Ethernet MAC Address 0 High	Section 15.6.16
0x44	EMACADDR0L	Ethernet MAC Address 0 Low Register	Section 15.6.17
0x48	EMACADDR1H	Ethernet MAC Address 1 High	Section 15.6.18
0x4C	EMACADDR1L	Ethernet MAC Address 1 Low	Section 15.6.19
0x50	EMACADDR2H	Ethernet MAC Address 2 High	Section 15.6.20
0x54	EMACADDR2L	Ethernet MAC Address 2 Low	Section 15.6.21
0x58	EMACADDR3H	Ethernet MAC Address 3 High	Section 15.6.22
0x5C	EMACADDR3L	Ethernet MAC Address 3 Low	Section 15.6.23
0xDC	EMACWDOGTO	Ethernet MAC Watchdog Time-out	Section 15.6.24
0x100	EMACMMCCTRL	Ethernet MAC MMC Control	Section 15.6.25
0x104	EMACMMCRXRIS	Ethernet MAC MMC Receive Raw Interrupt Status	Section 15.6.26
0x108	EMACMMCTXRIS	Ethernet MAC MMC Transmit Raw Interrupt Status	Section 15.6.27
0x10C	EMACMMCRXIM	Ethernet MAC MMC Receive Interrupt Mask	Section 15.6.28
0x110	EMACMMCTXIM	Ethernet MAC MMC Transmit Interrupt Mask	Section 15.6.29
0x118	EMACTXCNTGB	Ethernet MAC Transmit Frame Count for Good and Bad Frames	Section 15.6.30
0x14C	EMACTXCNTSCOL	Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision	Section 15.6.31
0x150	EMACTXCNTMCOL	Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions	Section 15.6.32
0x164	EMACTXOCTCNTG	Ethernet MAC Transmit Octet Count Good	Section 15.6.33
0x180	EMACRXCNTGB	Ethernet MAC Receive Frame Count for Good and Bad Frames	Section 15.6.34
0x194	EMACRXCNTCRCERR	Ethernet MAC Receive Frame Count for CRC Error Frames	Section 15.6.35

Table 15-23. EMAC Registers (continued)

Offset	Acronym	Register Name	Section
0x198	EMACRXCNTALGNERR	Ethernet MAC Receive Frame Count for Alignment Error Frames	Section 15.6.36
0x1C4	EMACRXCNTGUNI	Ethernet MAC Receive Frame Count for Good Unicast Frames	Section 15.6.37
0x584	EMACVLNINCREP	Ethernet MAC VLAN Tag Inclusion or Replacement	Section 15.6.38
0x588	EMACVLNHASH	Ethernet MAC VLAN Hash Table	Section 15.6.39
0x700	EMACTIMSTCTRL	Ethernet MAC Timestamp Control	Section 15.6.40
0x704	EMACSUBSECINC	Ethernet MAC Sub-Second Increment	Section 15.6.41
0x708	EMACTIMSEC	Ethernet MAC System Time - Seconds	Section 15.6.42
0x70C	EMACTIMNANO	Ethernet MAC System Time - Nanoseconds	Section 15.6.43
0x710	EMACTIMSECU	Ethernet MAC System Time - Seconds Update	Section 15.6.44
0x714	EMACTIMNANOU	Ethernet MAC System Time - Nanoseconds Update	Section 15.6.45
0x718	EMACTIMADD	Ethernet MAC Timestamp Addend	Section 15.6.46
0x71C	EMACTARGSEC	Ethernet MAC Target Time Seconds	Section 15.6.47
0x720	EMACTARGNANO	Ethernet MAC Target Time Nanoseconds	Section 15.6.48
0x724	EMACHWORDSEC	Ethernet MAC System Time-Higher Word Seconds	Section 15.6.49
0x728	EMACTIMSTAT	Ethernet MAC Timestamp Status	Section 15.6.50
0x72C	EMACPPSCTRL	Ethernet MAC PPS Control	Section 15.6.51
0x760	EMACPPS0INTVL	Ethernet MAC PPS0 Interval	Section 15.6.52
0x764	EMACPPS0WIDTH	Ethernet MAC PPS0 Width	Section 15.6.53
0xC00	EMACDMABUSMOD	Ethernet MAC DMA Bus Mode	Section 15.6.54
0xC04	EMACTXPOLLDD	Ethernet MAC Transmit Poll Demand	Section 15.6.55
0xC08	EMACRXPOLLD	Ethernet MAC Receive Poll Demand	Section 15.6.56
0xC0C	EMACRXDLADDR	Ethernet MAC Receive Descriptor List Address	Section 15.6.57
0xC10	EMACTXDLADDR	Ethernet MAC Transmit Descriptor List Address	Section 15.6.58
0xC14	EMACDMARIS	Ethernet MAC DMA Interrupt Status	Section 15.6.59
0xC18	EMACDMAOPMODE	Ethernet MAC DMA Operation Mode	Section 15.6.60
0xC1C	EMACDMAIM	Ethernet MAC DMA Interrupt Mask Register	Section 15.6.61
0xC20	EMACMFBOC	Ethernet MAC Missed Frame and Buffer Overflow Counter	Section 15.6.62
0xC24	EMACRXINTWDT	Ethernet MAC Receive Interrupt Watchdog Timer	Section 15.6.63
0xC48	EMACHOSTXDESC	Ethernet MAC Current Host Transmit Descriptor	Section 15.6.64
0xC4C	EMACHOSRXDESC	Ethernet MAC Current Host Receive Descriptor	Section 15.6.65
0xC50	EMACHOSTXBA	Ethernet MAC Current Host Transmit Buffer Address	Section 15.6.66
0xC54	EMACHOSRXBA	Ethernet MAC Current Host Receive Buffer Address	Section 15.6.67
0xFC0	EMACPP	Ethernet MAC Peripheral Property Register	Section 15.6.68
0xFC4	EMACPC	Ethernet MAC Peripheral Configuration Register	Section 15.6.69
0xFC8	EMACCC	Ethernet MAC Clock Configuration Register	Section 15.6.70
0xFD0	EPHYRIS	Ethernet PHY Raw Interrupt Status	Section 15.6.71
0xFD4	EPHYIM	Ethernet PHY Interrupt Mask	Section 15.6.72
0xFD8	EPHYMISC	Ethernet PHY Masked Interrupt Status and Clear	Section 15.6.73

Complex bit access types are encoded to fit into small table cells. [Table 15-24](#) shows the codes that are used for access types in this section.

Table 15-24. EMAC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

15.6.1 EMACCFG Register (Offset = 0x0) [reset = 0x8000]

Ethernet MAC Configuration (EMACCFG)

The EMACCFG register establishes receive and transmit operating modes. The TWOKPEN bit is only applicable when the JFEN bit is 0.

EMACCFG is shown in [Figure 15-16](#) and described in [Table 15-25](#).

Return to [Summary Table](#).

Figure 15-16. EMACCFG Register

31	30	29	28	27	26	25	24
RESERVED	SADDR			TWOKPEN	RESERVED	CST	RESERVED
R-0x0	R/W-0x0			R/W-0x0	R-0x0	R/W-0x0	R-0x0
23	22	21	20	19	18	17	16
WDDIS	JD	RESERVED	JFEN	IFG		DISCRS	
R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0		R/W-0x0	
15	14	13	12	11	10	9	8
PS	FES	DRO	LOOPBM	DUPM	IPC	DR	RESERVED
R-0x1	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0
7	6	5	4	3	2	1	0
ACS	BL		DC	TE	RE	PRELEN	
R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	

Table 15-25. EMACCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	
30-28	SADDR	R/W	0x0	<p>Source Address Insertion or Replacement Control. Bit 30 specifies whether MAC address 0 or 1 registers are used during insertion or replacement for all transmitted frames. Thus for encodings 0x2-0x3, where the most significant bit is 0, the Ethernet MAC Address 0 registers are used. For encodings 0x6-0x7, the Ethernet MAC Address 1 registers are used. Bits [29:28] indicate insertion or replacement. If the value is 0x2 insertion is indicated and if the value is 0x3 replacement is indicated. Changes in this field take effect only on the start of a frame. If a write of this field occurs while a frame is being transmitted, only the subsequent frame can use the updated value, and the current frame does not.</p> <p>0x0 = Reserved</p> <p>0x1 = Reserved</p> <p>0x2 = The Ethernet MAC inserts the content of the Ethernet MAC Address 0 (EMACADDR0x) registers in the SA field of all transmitted frames.</p> <p>0x3 = The Ethernet MAC replaces the content of the Ethernet MAC Address 0 (EMACADDR0x) registers in the SA field of all transmitted frames.</p> <p>0x4 = Reserved</p> <p>0x5 = Reserved</p> <p>0x6 = The MAC inserts the content of the Ethernet MAC Address 1 (EMACADDR1x) registers in the source address (SA) field for all transmitted frames.</p> <p>0x7 = The MAC replaces the content of the Ethernet MAC Address 1(EMACADDR1x) registers in the source address (SA) field for all transmitted frames.</p>

Table 15-25. EMACCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
27	TWOKPEN	R/W	0x0	IEEE 802.3as Support for 2K Packets. When set, the MAC considers all frames, up to 2,000 bytes in length, as normal packets. This bit is only valid when the JFEN bit is set to 0. When JFEN is set, configuring TWOKPEN has no effect on Giant Frame status. 0x0 = If the JFEN bit is clear, the MAC considers all received frames larger than 1,518 bytes (1522 bytes tagged) as Giant Frames. 0x1 = Frames up to 2 KB are considered normal packets. If the JFEN bit is clear, the MAC considers all received frames larger than 2K bytes as Giant frames.
26	RESERVED	R	0x0	
25	CST	R/W	0x0	CRC Stripping for Type Frames. When set, the last four bytes (Frame Check Sequence FCS) of all frames of Ether type ((Length/Type field greater than or equal to 0x0600) are removed before forwarding the frame to the application. 0x0 = No bytes are removed. 0x1 = The last four bytes are removed before forwarding.
24	RESERVED	R	0x0	
23	WDDIS	R/W	0x0	Watchdog Disable. When this bit is set, the MAC disables the internal watchdog counter on the receiver. The MAC can receive frames of up to 16,384 bytes. When this bit is cleared, the MAC does not allow more than 2,048 bytes (10,240 if JFEN is set to 1) of the frame being received. The MAC cuts off any bytes received after 2,048 bytes. 0x0 = Watchdog counter enabled. 0x1 = Watchdog counter disabled.
22	JD	R/W	0x0	Jabber Disable. When this bit is set, the MAC disables the jabber counter on the transmitter. The MAC can transfer frames of up to 16,384 bytes. When this bit is clear, the MAC stops transmission if the application sends out more than 2,048 bytes of data (10,240 if JFEN is set to 1). 0x0 = Jabber counter enabled. 0x1 = Jabber counter disabled.
21	RESERVED	R	0x0	
20	JFEN	R/W	0x0	Jumbo Frame Enable. When this bit is set, the MAC allows jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status. 0x0 = Jumbo frames create giant frame error. 0x1 = Jumbo frames allowed without error.
19-17	IFG	R/W	0x0	Inter-Frame Gap (IFG). These bits control the minimum IFG between frames during transmission. In half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 0x4). Lower values are not considered. 0x0 = 96 bit times 0x1 = 88 bit times 0x2 = 80 bit times 0x3 = 72 bit times 0x4 = 64 bit times 0x5 = 56 bit times 0x6 = 48 bit times 0x7 = 40 bit times
16	DISCRS	R/W	0x0	Disable Carrier Sense During Transmission. When this bit is set, the MAC transmit module ignores carrier sense in half-duplex mode. Thus, errors are not generated when there is a loss of carrier or no carrier during transmission. When this bit is clear, the MAC transmitter generates errors because of carrier sense and can even abort the transmissions. 0x0 = Generate errors for carrier sense errors. 0x1 = Ignore carrier sense errors.

Table 15-25. EMACCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15	PS	R	0x1	Port Select. This bit indicates that a 10/100 Mbps interface is supported on this device. This is a read-only bit.
14	FES	R/W	0x0	Speed. This bit indicates the speed of the interface. 0x0 = 10 Mbps 0x1 = 100 Mbps
13	DRO	R/W	0x0	Disable Receive Own. When this bit is set, the MAC disables the reception of frames while transmitting in half-duplex mode. When this bit is clear, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in full-duplex mode. 0x0 = All packets are received by MAC. 0x1 = Disable reception of frames.
12	LOOPBM	R/W	0x0	Loopback Mode. When this bit is set, the MAC operates in the loopback mode at the MII. The MII Receive clock input, EN0RXCK, is required for the loopback to work properly, because the Transmit clock is not looped-back internally. 0x0 = MAC does not operate in loopback mode. 0x1 = MAC operates in loopback mode.
11	DUPM	R/W	0x0	Duplex Mode. When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously. 0x0 = MAC does not operate in full-duplex mode. 0x1 = MAC operates in full-duplex mode.
10	IPC	R/W	0x0	Checksum Offload. 0x0 = The checksum offload function in the receiver is disabled and the corresponding PCE and IP HCE status bits in the frame status are always cleared. 0x1 = Checksum Offload EnableSetting this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking.
9	DR	R/W	0x0	Disable Retry. When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current frame transmission and reports a frame abort with excessive collision error in the transmit frame status. When this bit is cleared, the MAC attempts retries based on the settings of the BL field (Bits [6:5]). This bit is only applicable in half-duplex mode. 0x0 = MAC retries transmissions based on BL bit field. 0x1 = Only one transmission is attempted by the MAC.
8	RESERVED	R	0x0	
7	ACS	R/W	0x0	Automatic Pad or CRC Stripping. When this bit is set, the MAC strips the Pad or Frame Check Sequence (FCS) field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is cleared, the MAC passes all incoming frames, without modifying them, to the Host. 0x0 = All frames are passed to host unmodified. 0x1 = MAC strips FCS field if value of length field is less than 1,536 bytes.

Table 15-25. EMACCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-5	BL	R/W	0x0	<p>Back-Off Limit. The Back-Off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. The random integer r takes the value in the range $0 \leq r < 2^k$. The value of k is programmed in the encodings below and is dependent on the retransmission attempt number, n. This bit is applicable only in the half-duplex mode.</p> <p>0x0 = $k = \min(n, 10)$, where k is the lowest value when evaluating n or 10.</p> <p>0x1 = $k = \min(n, 8)$, where k is the lowest value when evaluating n or 8.</p> <p>0x2 = $k = \min(n, 4)$, where k is the lowest value when evaluating n or 4.</p> <p>0x3 = $k = \min(n, 1)$, where k is the lowest value when evaluating n or 1.</p>
4	DC	R/W	0x0	<p>Deferral Check. When this bit is set, the deferral check function is enabled in the MAC. When the transmit state machine is deferred for more than 24,288 bit times, the MAC issues a Frame Abort status, and sets the excessive deferral error bit (EXDEFF) in the MAC MMC Transmit Interrupt (EMACMMCTXRIS) Register. If the Jumbo frame mode (JFEN) is enabled, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active carrier sense signal (CRS) on the MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and then the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0 and it is restarted. When this bit is clear, the deferral check function is disabled and the EMAC defers until the CRS signal goes inactive. This bit is only applicable in half-duplex mode.</p> <p>0x0 = Deferral check function is disabled.</p> <p>0x1 = Deferral check function is enabled.</p>
3	TE	R/W	0x0	<p>Transmitter Enable. When this bit is set, the transmit state machine of the MAC is enabled for transmission on the MII. When this bit is clear, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.</p> <p>0x0 = MAC transmit state machine is disabled.</p> <p>0x1 = MAC transmit state machine is enabled.</p>
2	RE	R/W	0x0	<p>Receiver Enable. When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the MII. When this bit is clear, the MAC receive state machine is disabled after completion of the reception of the current frame, and does not receive any further frames from the MII.</p> <p>0x0 = MAC receive state machine is disabled.</p> <p>0x1 = MAC receive state machine is enabled.</p>
1-0	PRELEN	R/W	0x0	<p>Preamble Length for Transmit Frames. These bits control the number of preamble bytes that are added to the beginning of every Transmit frame. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.</p> <p>0x0 = 7 bytes of preamble</p> <p>0x1 = 5 bytes of preamble</p> <p>0x2 = 3 bytes of preamble</p> <p>0x3 = Reserved</p>

15.6.2 EMACFRAMEFLTR Register (Offset = 0x4) [reset = 0x0]

Ethernet MAC Frame Filter (EMACFRAMEFLTR)

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

EMACFRAMEFLTR is shown in [Figure 15-17](#) and described in [Table 15-26](#).

Return to [Summary Table](#).

Figure 15-17. EMACFRAMEFLTR Register

31	30	29	28	27	26	25	24
RA	RESERVED						
R/W-0x0	R-0x0						
23	22	21	20	19	18	17	16
RESERVED							VTFE
R-0x0							R/W-0x0
15	14	13	12	11	10	9	8
RESERVED					HPF	SAF	SAIF
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
PCF	DBF	PM	DAIF	HMC	HUC	PR	
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 15-26. EMACFRAMEFLTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RA	R/W	0x0	Receive All. When this bit is set, the MAC Receiver module passes all received frames, irrespective of whether they pass the address filter or not, to the application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is clear, the Receiver module passes only those frames to the application that pass the SA or DA address filter. 0x0 = MAC RX module only passes frames that pass the SA or DA address filter. 0x1 = MAC RX module passes all received frames.
30-17	RESERVED	R	0x0	
16	VTFE	R/W	0x0	VLAN Tag Filter Enable. 0x0 = MAC forwards all frames regardless of match status of VLAN Tag. 0x1 = MAC drops VLAN tagged frames that do not match VLAN tag comparison.
15-11	RESERVED	R	0x0	
10	HPF	R/W	0x0	Hash or Perfect Filter. When this bit is set, it configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by the HMC or HUC bits in this register. When this bit is clear and the HUC or HMC bit is set, the frame is passed only if it matches the Hash filter. 0x0 = Address filter passes a frame if it matches perfect filtering or hash filtering. 0x1 = Address filter passes a frame only if it matches in the hash filter.

Table 15-26. EMACFRAMEFLTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	SAF	R/W	0x0	Source Address Filter Enable. When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SA match bit of Receive Status Word is set. When SA Match bit is set and the SA filter fails, the MAC drops the frame. When this bit is clear, the MAC forwards the received frame to the application with the updated SA Match bit of the Receive Status Word depending on the SA address comparison. 0x0 = Source address filter disabled. 0x1 = Source address filter enabled.
8	SAIF	R/W	0x0	Source Address (SA) Inverse Filtering. When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA address filter. When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter. 0x0 = Frames whose SA does not match the SA registers are marked as failing. 0x1 = Frames whose SA matches the SA registers are marked as failing.
7-6	PCF	R/W	0x0	Pass Control Frames. . These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). The following conditions should be true for the PAUSE control frames processing: Condition 1: The MAC is in full-duplex mode and flow control is enabled by setting the RFE bit of MAC Flow Control Register (EMACFLOWCTL). Condition 2: The destination address (DA) of the received frame matches the special multicast address or the MAC Address 0 (EMACADDR0x) Register when the UP bit of the (EMACFLOWCTL) is set. Condition 3: The Type field of the received frame is 0x8808 and the OPCODE field is 0x0001. This PCF field should be set to 0x1 only when Condition 1 is true; that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled. Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 0x2 or 0x3 (as required by the application). 0x00 = The MAC filters all control frames from reaching application. 0x1 = MAC forwards all control frames except PAUSE control frames to application even if they fail the address filter. 0x2 = MAC forwards all control frames to application even if they fail the address Filter. 0x3 = MAC forwards control frames that pass the address Filter.
5	DBF	R/W	0x0	Disable Broadcast Frames. When this bit is set, the address filtering module (AFM) filters all incoming broadcast frames. In addition, it overrides all other filter settings. 0x0 = Address filtering module passes all received broadcast frames. 0x1 = Address filtering module filters all incoming broadcast frames.
4	PM	R/W	0x0	Pass All Multicast. When set, this bit indicates that all received frames with a multicast destination address (DA) (first bit in the destination address field is 1) are passed. 0x0 = Filtering of multicast frame depends on HMC bit in this register. 0x1 = All received frames with multicast DA are passed.
3	DAIF	R/W	0x0	Destination Address (DA) Inverse Filtering. When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. 0x0 = Normal filtering of frames is performed. 0x1 = Inverse filtering mode is enabled for DA.

Table 15-26. EMACFRAMEFLTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	HMC	R/W	0x0	Hash Multicast. 0x0 = MAC performs a perfect destination address (DA) filtering for multicast frames. It compares the DA field with the values programmed in DA registers. 0x1 = MAC performs destination address filtering of received multicast frames according to the hash table.
1	HUC	R/W	0x0	Hash Unicast. 0x0 = MAC performs a perfect destination address filtering for unicast frames. It compares the DA field with the values programmed in DA registers. 0x1 = MAC performs destination address filtering of unicast frames according to the hash table.
0	PR	R/W	0x0	Promiscuous Mode. When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set. 0x0 = Incoming frames are filtered. 0x1 = All incoming frames are passed.

15.6.3 EMACHASHTBLH Register (Offset = 0x8) [reset = 0x0]

Ethernet MAC Hash Table High (EMACHASHTBLH)

The 64-bit Hash table is used for group address filtering. For hash filtering, the contents of the Destination Address (DA) in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (Ethernet MAC Hash Table High (EMACHASHTBLH) or Ethernet MAC Hash Table Low (EMACHASHTBL)), and the other five bits determine which bit within the register. A hash value of 5b'00000 selects bit 0 of the selected register, and a value of 5b'11111 selects bit 31 of the selected register.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bit-wise reversal for the value obtained in Step 1.
3. Take the upper 6 bits from the value obtained in Step 2.

For example, if the DA of the incoming frame is received as 0x1F52419CB6AF, then the internally calculated 6-bit Hash value is 0x2C and Bit 12 of EMACHASHTBLH register is checked for filtering. If the DA of the incoming frame is received as 0xA00A98000045, then the calculated 6-bit Hash value is 0x07 and Bit 7 of Hash Table Low register is checked for filtering.

If the corresponding bit value of the register is 0x1, the frame is accepted. Otherwise, it is rejected. If the PM (Pass All Multicast) bit is set in the Ethernet MAC Frame Filter (EMACFRAMEFLTR) register, then all multicast frames are accepted regardless of the multicast hash values.

EMACHASHTBLH is shown in [Figure 15-18](#) and described in [Table 15-27](#).

Return to [Summary Table](#).

Figure 15-18. EMACHASHTBLH Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTH																															
R/W-0x0																															

Table 15-27. EMACHASHTBLH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HTH	R/W	0x0	Hash Table High. This field contains the upper 32 bits of the hash table.

15.6.4 EMACHASHTBLL Register (Offset = 0xC) [reset = 0x0]

Ethernet MAC Hash Table Low (EMACHASHTBLL)

The MAC Hash Table Low (EMACHASHTBLL) register contains the lower 32 bits of the hash table.

EMACHASHTBLL is shown in [Figure 15-19](#) and described in [Table 15-28](#).

Return to [Summary Table](#).

Figure 15-19. EMACHASHTBLL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTL																															
R/W-0x0																															

Table 15-28. EMACHASHTBLL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HTL	R/W	0x0	Hash Table Low. This field contains the lower 32 bits of the hash table.

15.6.5 EMACMIIADDR Register (Offset = 0x10) [reset = 0x0]

Ethernet MAC MII Address (EMACMIIADDR)

The Ethernet MAC MII address (EMACMIIADDR) register controls the management cycles to the PHY through the management interface.

EMACMIIADDR is shown in [Figure 15-20](#) and described in [Table 15-29](#).

Return to [Summary Table](#).

Figure 15-20. EMACMIIADDR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
PLA				MII			
R/W-0x0				R/W-0x0			
7	6	5	4	3	2	1	0
MII		CR			MIIW		MIIB
R/W-0x0		R/W-0x0			R/W-0x0		R/W-0x0

Table 15-29. EMACMIIADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-11	PLA	R/W	0x0	Physical Layer Address. This field gives the PHY address of the MII module. Values 1- 31 (0x1 to 0x1F) are available for external PHY addresses. The integrated PHY's address is 0x00. To access the integrated PHY registers, the PLA bits[15:11] must be zeros.
10-6	MII	R/W	0x0	MII Register. These bits select the desired MII registers in the selected PHY device.
5-2	CR	R/W	0x0	Clock Reference Frequency Selection. The clock that is sent to the MAC Clock and Status (CSR) registers is the gated System Clock (SYSCLK). The SYSCLK is divided down through the CR field to produce an MDC clock that is between approximately 1.0 MHz and 2.5 MHz. The application must program the appropriate CR field based on the System Clock input. 0x4-0xF = Reserved 0x0 = The frequency of the System Clock is 60 to 100 MHz providing a MDIO clock of SYSCLK/42. 0x1 = The frequency of the System Clock is 100 to 150 MHz providing a MDIO clock of SYSCLK/62. 0x2 = The frequency of the System Clock is 20 to 35 MHz providing a MDIO clock of System Clock/16. 0x3 = The frequency of the System Clock is 35 to 60 MHz providing a MDIO clock of System Clock/26.
1	MIIW	R/W	0x0	MII Write. 0x0 = Read operation is active and read data is placed in the MII Data register (EMACMIIDATA). 0x1 = PHY is notified that this is a write operation using the MII Data Register (EMACMIIDATA).

Table 15-29. EMACMIADDR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	MIIB	R/W	0x0	<p>MII Busy. Indicates whether the MII is busy with a read or write access. This bit should read logic 0 before writing to the EMACMIADDR register or the EMACMIIDATA register. During a PHY register access, the software sets this bit to indicate that a read or write access is in progress. The EMACMIIDATA register is invalid until this bit is cleared by the MAC. Therefore, EMACMIIDATA should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of EMACMIIDATA are not valid until this bit is cleared. The subsequent read or write operations should happen only after the previous operation is complete.</p> <p>0x0 = EMACMIADDR and EMACMIIDATA are available for reads and writes.</p> <p>0x1 = Read or write access is in progress.</p>

15.6.6 EMACMIIDATA Register (Offset = 0x14) [reset = 0x0]

Ethernet MAC MII Data Register (EMACMIIDATA)

The Ethernet MAC MII Data (EMACMIIDATA) register holds data that is written to and read from the PHY register located at the address specified by the PLA and MII bit fields of the Ethernet MAC MII Address (EMACMIADDR) register.

EMACMIIDATA is shown in [Figure 15-21](#) and described in [Table 15-30](#).

Return to [Summary Table](#).

Figure 15-21. EMACMIIDATA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DATA															
R-0x0																R/W-0x0															

Table 15-30. EMACMIIDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	DATA	R/W	0x0	MII Data. This field contains the 16-bit data value read from the PHY after a management read operation or the 16-bit data value to be written to the PHY before a management write operation.

15.6.7 EMACFLOWCTL Register (Offset = 0x18) [reset = 0x0]

Ethernet MAC Flow Control (EMACFLOWCTL)

The Ethernet MAC Flow Control (EMACFLOWCTL) register controls the generation and reception of the control (pause command) frames by the MAC's Flow control module. A write to a register with the FCBBPA (bit 0) set to 1 triggers the Flow Control block to generate a pause control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the pause time (PT) value from this register is used in the pause time field of the control frame. The FCBBPA bit is cleared by the hardware once the control frame is transferred onto the cable. The Host must make sure that the busy bit is clear before writing to the register.

EMACFLOWCTL is shown in [Figure 15-22](#) and described in [Table 15-31](#).

Return to [Summary Table](#).

Figure 15-22. EMACFLOWCTL Register

31	30	29	28	27	26	25	24
PT							
R/W-0x0							
23	22	21	20	19	18	17	16
PT							
R/W-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
DZQP	RESERVED			UP	RFE	TFE	FCBBPA
R/W-0x0	R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 15-31. EMACFLOWCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	PT	R/W	0x0	Pause Time. This field holds the value to be used in the pause time field in the transmit control frame. For example, if these bits are set to 0x0100 then 256 slot times are used in the Pause Time field in the transmit control frame.
15-8	RESERVED	R	0x0	
7	DZQP	R/W	0x0	Disable Zero-Quanta Pause. When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the deassertion of the flow-control signal from the FIFO layer. When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled. 0x0 = Automatic Zero-Quanta Pause Control generation is enabled. 0x1 = Automatic Zero-Quanta Pause Control generation is disabled.
6-4	RESERVED	R	0x0	
3	UP	R/W	0x0	Unicast Pause Frame Detect. 0x0 = MAC detects only a Pause frame with the unique multicast address specified in the 802.3x standard. 0x1 = The MAC detects the pause frames with the station's unicast address specified in the EMACADDR0H and EMACADDR0L register, in addition to detecting pause frames with the unique multicast address.
2	RFE	R/W	0x0	Receive Flow Control Enable. 0x0 = The decode function of the pause frame is disabled. 0x1 = The MAC decodes the received pause frame and disables its transmitter for a specified (pause) time.

Table 15-31. EMACFLOWCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	TFE	R/W	0x0	<p>Transmit Flow Control Enable.</p> <p>0x0 = In full duplex mode, the flow control operation in the MAC is disabled, and the MAC does not transmit any pause frames. In half duplex mode, the back-pressure feature is disabled.</p> <p>0x1 = In the full-duplex mode, the MAC enables the flow control operation to transmit pause frames. In half-duplex mode, the MAC enables the back-pressure operation.</p>
0	FCBBPA	R/W	0x0	<p>Flow Control Busy or Back-pressure Activate. In the full-duplex mode, this bit should be read as 0x0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 0x1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 0x0. The EMACFLOWCTL register should not be written to until this bit is cleared.</p> <p>0x0 = No effect</p> <p>0x1 = In the full-duplex mode, a pause control frame is enabled. In half-duplex mode, a back-pressure function is enabled if the TFE bit is set.</p>

15.6.8 EMACVLANTG Register (Offset = 0x1C) [reset = 0x0]

Ethernet MAC VLAN Tag (EMACVLANTG)

The Ethernet MAC VLAN Tag (MACVLANTG) register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with the value 0x8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1518 bytes to 1522 bytes.

EMACVLANTG is shown in [Figure 15-23](#) and described in [Table 15-32](#).

Return to [Summary Table](#).

Figure 15-23. EMACVLANTG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				VTHM	ESVL	VTIM	ETV
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
VL							
R/W-0x0							
7	6	5	4	3	2	1	0
VL							
R/W-0x0							

Table 15-32. EMACVLANTG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	VTHM	R/W	0x0	VLAN Tag Hash Table Match Enable. 0x0 = The VLAN Hash Match operation is not performed. 0x1 = The most significant four bits of the VLAN tag's CRC are used to index the content of the MAC VLAN Hash Table (EMACVLANTHASH) register. A value of 1 in the EMACVLANTHASH register, corresponding to the index, indicates that the frame matched the VLAN hash table. When Bit 16 (ETV) is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison whereas when ETV is reset, the CRC of the 16-bit VLAN tag is used for comparison.
18	ESVL	R/W	0x0	Enable S-VLAN. 0x0 = The MAC transmitter does not recognize S-VLAN frames as valid VLAN tagged frames. 0x1 = The MAC transmitter and receiver considers the S-VLAN (Type = 0x88A8) frames as valid VLAN tagged frames.
17	VTIM	R/W	0x0	VLAN Tag Inverse Match Enable. 0x0 = VLAN perfect matching is enabled. The frames with matched VLAN tag are marked as matched. 0x1 = VLAN tag inverse matching is enabled. The frames that do not have matching VLAN tag are marked as matched.
16	ETV	R/W	0x0	Enable 12-Bit VLAN Tag Comparison. 0x0 = All 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering. 0x1 = A 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering.

Table 15-32. EMACVLANTG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-0	VL	R/W	0x0	VLAN Tag Identifier for Receive Frames. This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field: Bits [15:13]: User PriorityBit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)Bits[11:0]: VLAN tag's VLAN Identifier (VID) field When the ETV bit is set, only the VID field (Bits[11:0]) is used for comparison. If VL [11:0] is all zeros when the ETV is set, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 or 0x88A8 as VLAN frames.

15.6.9 EMACSTATUS Register (Offset = 0x24) [reset = 0x0]

Ethernet MAC Status (EMACSTATUS)

This register gives the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

EMACSTATUS is shown in [Figure 15-24](#) and described in [Table 15-33](#).

Return to [Summary Table](#).

Figure 15-24. EMACSTATUS Register

31	30	29	28	27	26	25	24
RESERVED						TXFF	TXFE
R-0x0						R-0x0	R-0x0
23	22	21	20	19	18	17	16
RESERVED	TWC	TRC		TXPAUSED	TFC		TPE
R-0x0	R-0x0	R-0x0		R-0x0	R-0x0		R-0x0
15	14	13	12	11	10	9	8
RESERVED						RXF	
R-0x0						R-0x0	
7	6	5	4	3	2	1	0
RESERVED	RRC		RWC	RESERVED	RFCFC		RPE
R-0x0	R-0x0		R-0x0	R-0x0	R-0x0		R-0x0

Table 15-33. EMACSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25	TXFF	R	0x0	TX/RX Controller TX FIFO Full Status. 0x0 = The TX/RX Controller TX FIFO is not full. 0x1 = The TX/RX Controller TX FIFO is full. Therefore, the TX/RX Controller cannot accept any more frames for transmission.
24	TXFE	R	0x0	TX/RX Controller TX FIFO Not Empty Status. 0x0 = TX/RX Controller TX FIFO is empty. 0x1 = TX/RX Controller TX FIFO is not empty and some data is left for transmission.
23	RESERVED	R	0x0	
22	TWC	R	0x0	TX/RX Controller TX FIFO Write Controller Active Status. 0x0 = TX/RX Controller's TX FIFO write controller is inactive. 0x1 = TX/RX Controller's TX FIFO write controller is active and transferring data to the TX FIFO.
21-20	TRC	R	0x0	TX/RX Controller's TX FIFO Read Controller Status. This field indicates the state of the TX FIFO read controller: 0x0 = IDLE state 0x1 = READ state (transferring data to MAC transmitter) 0x2 = Waiting for TX Status from MAC transmitter 0x3 = Writing the received TX Status or flushing the TX FIFO
19	TXPAUSED	R	0x0	MAC Transmitter PAUSE. 0x0 = MAC transmitter is not in PAUSE mode. 0x1 = Indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission.

Table 15-33. EMACSTATUS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-17	TFC	R	0x0	MAC Transmit Frame Controller Status. This field indicates the state of the MAC Transmit Frame Controller module: 0x0 = IDLE state 0x1 = Waiting for status of previous frame or IFG or backoff period to be over 0x2 = Generating and transmitting a PAUSE control frame (in the full-duplex mode) 0x3 = Transferring input frame for transmission
16	TPE	R	0x0	MAC MII Transmit Protocol Engine Status. 0x0 = MAC MII transmit protocol engine is not actively transmitting data. 0x1 = MAC MII transmit protocol engine is actively transmitting data and is not in the IDLE state.
15-10	RESERVED	R	0x0	
9-8	RXF	R	0x0	TX/RX Controller RX FIFO Fill-level Status. This field gives the status of the fill-level of the RX FIFO. The FIFO threshold is programmed by the TCC field in the Ethernet MAC DMA Operation Mode (EMACDMAOPMODE) register. 0x0 = RX FIFO Empty 0x1 = RX FIFO fill level is below the flow-control deactivate threshold 0x2 = RX FIFO fill level is above the flow-control activate threshold 0x3 = RX FIFO Full
7	RESERVED	R	0x0	
6-5	RRC	R	0x0	TX/RX Controller Read Controller State. This field gives the state of the RX FIFO read Controller. 0x0 = IDLE state 0x1 = Reading frame data 0x2 = Reading frame status (or timestamp) 0x3 = Flushing the frame data and status
4	RWC	R	0x0	TX/RX Controller RX FIFO Write Controller Active Status. 0x0 = The MTL RX FIFO Write Controller is inactive. 0x1 = MTL RX FIFO Write Controller is active and is transferring a received frame to the FIFO.
3	RESERVED	R	0x0	
2-1	RFCFC	R	0x0	MAC Receive Frame Controller FIFO Status. When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module.
0	RPE	R	0x0	MAC MII Receive Protocol Engine Status. 0x0 = MAC MII receive protocol engine is not actively receiving data. 0x1 = Indicates that the MAC MII receive protocol engine is actively receiving data and not in IDLE state.

15.6.10 EMACRWUFF Register (Offset = 0x28) [reset = 0x0]

Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)

This is the address through which the application writes or reads the remote wake-up frame filter registers. To load values in the Ethernet MAC Wake-up Frame Filter (EMACRWUFF) register, the entire register must be written. The Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF) register is a pointer to eight wake-up frame filter registers. The Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF) register is loaded by sequentially loading the eight register values. Eight sequential writes to this address programs all of the remote wake-up frame filter registers. Similarly, eight sequential reads from the EMACRWUFF register reads all wake-up frame registers.

EMACRWUFF is shown in [Figure 15-25](#) and described in [Table 15-34](#).

Return to [Summary Table](#).

Figure 15-25. EMACRWUFF Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUPFIL																															
R/W-0x0																															

Table 15-34. EMACRWUFF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WAKEUPFIL	R/W	0x0	Remote Wake-Up Frame Filter.

15.6.11 EMACPMCTLSTAT Register (Offset = 0x2C) [reset = 0x0]

Ethernet MAC PMT Control and Status Register (EMACPMCTLSTAT)

The Ethernet MAC PMT Control and Status (EMACPMCTLSTAT) programs and monitors wake-up events.

EMACPMCTLSTAT is shown in [Figure 15-26](#) and described in [Table 15-35](#).

Return to [Summary Table](#).

Figure 15-26. EMACPMCTLSTAT Register

31	30	29	28	27	26	25	24
WUPFRRST	RESERVED				RWKPTR		
R/W-0x0	R-0x0				R/W-0x0		
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						GLBLUCAST	RESERVED
R-0x0						R/W-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED	WUPRX	MGKPRX	RESERVED		WUPFREN	MGKPKTEN	PWRDWN
R-0x0	R-0x0	R-0x0	R-0x0		R/W-0x0	R/W-0x0	R/W-0x0

Table 15-35. EMACPMCTLSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	WUPFRRST	R/W	0x0	Wake-Up Frame Filter Register Pointer Reset. 0x0 = No effect. 0x1 = Resets the MAC Remote Wake-Up Frame Filter (EMACRWUFF) register pointer to 0x0. It is automatically cleared after one clock cycle.
30-27	RESERVED	R	0x0	
26-24	RWKPTR	R/W	0x0	Remote Wake-Up FIFO Pointer. This gives the current value (0 to 7) of the MAC Remote Wake-Up Frame Filter (EMACRWUFF) register pointer. The contents of the EMACRWUFF Register are transferred to the receive clock domain when a write occurs to that register when this pointer value equals 7.
23-10	RESERVED	R	0x0	
9	GLBLUCAST	R/W	0x0	Global Unicast. 0x0 = No Effect. 0x1 = Enables any unicast packet filtered by the MAC Destination Address Filter (DAF) module's address recognition to be a wake-up frame.
8-7	RESERVED	R	0x0	
6	WUPRX	R	0x0	Wake-Up Frame Received. 0x0 = No effect. 0x1 = The power management event is generated because of the reception of a wake-up frame. This bit is cleared whenever the register is read.
5	MGKPRX	R	0x0	Magic Packet Received. 0x0 = No effect. 0x1 = The power management event is generated because of the reception of a magic packet. This bit is cleared whenever the register is read.
4-3	RESERVED	R	0x0	

Table 15-35. EMACPMCTLSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	WUPFREN	R/W	0x0	Wake-Up Frame Enable. 0x0 = Wake-up frame does not affect power management control. 0x1 = Generation of a power management event in response to the reception of a wake-up frame is enabled.
1	MGKPKTEN	R/W	0x0	Magic Packet Enable. 0x0 = Magic packet reception does not affect power management control. 0x1 = Generation of a power management event in response to the reception of a magic packet is enabled.
0	PWRDWN	R/W	0x0	Power Down. When set, the MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. Upon reception, PWRDWN is self-cleared and the power-down mode is disabled. Software can also clear this bit before the expected magic packet or wake-up frame is received. The frames, received by the MAC after this bit is cleared, are forwarded to the application. 0x0 = Frames are forwarded to application. 0x1 = MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame.

15.6.12 EMACLPICLSTAT Register (Offset = 0x30) [reset = 0x0]

LPI Control and Status (EMACLPICLSTAT)

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

EMACRIS is shown in [Figure 15-29](#) and described in [Table 15-38](#).

Return to [Summary Table](#).

Figure 15-27. EMACLPICLSTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				LPITXA	PLSEN	PLS	LPIEN
R-0x0				RW-0x0	R/W-0x0	RW-0x0	RW-0x0
15	14	13	12	11	10	9	8
RESERVED						RLPIST	TLPIST
R-0x0						R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED				RLPIEX	RLPIEN	TLPIEX	TLPIEN
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 15-36. EMACLPICLSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	LPITXA	RW	0x0	<p>LPI TX Automate.</p> <p>This bit controls the behavior of the MAC when it is entering or leaving the LPI mode on transmit. This bit is not functional in the GMAC-CORE configuration in which the TX clock gating is done during the LPI mode.</p> <p>If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding frames (in the core) and pending frames have been transmitted.</p> <p>The MAC comes out of the LPI mode when the application sends any frame for transmission or the application issues a TX FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI mode.</p> <p>If the FTF bit is 1 in the EMACDMAOPMODE register when the MAC is in the LPI mode, the MAC exits the LPI mode. When the FTF bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.</p> <p>0x0 = Disabled 0x1 = Enabled</p>
18	PLSEN	RW	0x0	<p>PHY Link Status Enable.</p> <p>This bit enables the link status received on the RGMII, SGMII, or SMII receive paths to be used for activating the LPI LS timer.</p> <p>When set, the MAC uses the link status bits and the PLS bit for the LPI LS timer trigger. When cleared, the MAC ignores the link-status bits and uses only the PLS bit.</p> <p>This bit is reserved if you have not selected the RGMII, SGMII, or SMII PHY interface.</p> <p>0x0 = MAC ignores link status bits 0x1 = MAC uses link status bits</p>

Table 15-36. EMACLPICLSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	PLS	RW	0x0	PHY Link Status. This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OK) at least for the time indicated by the LPI LS timer. 0x0 = Link down 0x1 = Link up (OK)
16	LPIEN	RW	0x0	LPI Enable. When set, this bit instructs the MAC transmitter to enter the LPI mode. When reset, this bit instructs the MAC to exit the LPI mode and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI mode because of the arrival of a new packet for transmission. 0x0 = MAC transmitter exits LPI mode 0x1 = MAC transmitter enters LPI mode
15-10	RESERVED	R	0x0	
9	RLPIST	R	0x0	Receive LPI Mode. 0x0 = MAC is not receiving LPI pattern 0x1 = MAC is receiving LPI pattern
8	TLPIST	R	0x0	Transmit LPI Mode. 0x0 = MAC is not transmitting LPI pattern 0x1 = MAC is transmitting LPI pattern
7-4	RESERVED	R	0x0	
3	RLPIEX	R	0x0	Receive LPI Exit. This bit is cleared by a read into this register. 0x0 = MAC receiver is receiving LPI patterns 0x1 = MAC receiver has stopped receiving the LPI pattern, exited LPI mode, and resumed normal reception. Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as less than 3 clock cycles of l3_sp_clk.
2	RLPIEN	R	0x0	Receive LPI Entry. This bit is cleared by a read into this register. 0x0 = MAC receiver not in LPI mode 0x1 = MAC receiver received an LPI pattern and entered LPI mode Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than 3 clock cycles of l3_sp_clk.
1	TLPIEX	R	0x0	Transmit LPI Exit. This bit is cleared by a read into this register. 0x0 = MAC transmitter not in LPI mode 0x1 = MAC transmitter exited the LPI mode after the user software has cleared the LPIEN bit and the LPI TW timer has expired
0	TLPIEN	R	0x0	Transmit LPI Entry. This bit is cleared by a read into this register. 0x0 = MAC transmitter not in LPI mode 0x1 = MAC transmitter entered the LPI mode because of the setting of the LPIEN bit

15.6.13 EMACLPITIMERCTRL Register (Offset = 0x34) [reset = 0x0]

LPI Timers Control (EMACLPITIMERCTRL)

This register controls the time-out values in the LPI modes. It specifies the time that the MAC transmits the LPI pattern and also the time that the MAC waits before resuming the normal transmission.

EMACRIS is shown in [Figure 15-29](#) and described in [Table 15-38](#).

Return to [Summary Table](#).

Figure 15-28. EMACLPITIMERCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED						LST									
R-0x0						R/W-0x3E8									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TWT															
R/W-0x0															

Table 15-37. EMACLPITIMERCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-16	LST	R/W	0x3E8	LPI LS Timer. Specifies the minimum time (in milliseconds) that the link status from the PHY should be up (OKAY) before the LPI pattern is transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LST reaches the programmed terminal count. The default value of this bit is 1000 (1 second) as defined in the IEEE standard.
15-0	TWT	R/W	0x0	LPI TW Timer. Specifies the minimum time (in microseconds) that the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after this timer expires.

15.6.14 EMACRIS Register (Offset = 0x38) [reset = 0x0]

Ethernet MAC Raw Interrupt Status (EMACRIS)

This register identifies the events in the MAC that can generate interrupt.

EMACRIS is shown in [Figure 15-29](#) and described in [Table 15-38](#).

Return to [Summary Table](#).

Figure 15-29. EMACRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED					LPI	TS	RESERVED
R-0x0					R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED	MMCTX	MMCRX	MMC	PMT	RESERVED		
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0		

Table 15-38. EMACRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
10	LPI	R	0x0	LPI Interrupt Status. This bit is set for any LPI mode entry or exit in the MAC transmitter or receiver. This bit is cleared on reading the TLPIEN bit of the EMACLPICLSTAT register. In all other modes, this bit is reserved.
9	TS	R	0x0	Timestamp Interrupt Status. This bit is cleared by reading the TSSOVF bit in the MAC timestamp Status Register (EMACTIMSTAT) register. In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved. 0x0 = No timestamp interrupt is present. 0x1 = When the advanced timestamp feature is enabled, this bit indicates one of two conditions is true: The system time value equals or exceeds the value specified in the EMAC Target Time Seconds Register (EMACTARGSEC) and MAC Target Time Nanoseconds (EMACTARGNANO) registers. There is an overflow in the EMAC Target Time Seconds (EMACTARGSEC) register. When default timestamping is enabled, this bit indicates that the system time value is equal to or exceeds the value specified in the EMAC Target Time registers. In this mode, this bit is cleared after the completion of the read of this bit.
8-7	RESERVED	R	0x0	
6	MMCTX	R	0x0	MMC Transmit Interrupt Status. This bit is cleared when all of the bits in the MAC MMC Transmit Interrupt (EMACMMCTXRIS) register are clear. 0x0 = No interrupts exist in the MAC MMC Transmit Interrupt (EMACMMCTXRIS) register. 0x1 = Indicates an interrupt has been generated in the MAC MMC Transmit Interrupt (EMACMMCTXRIS) register.

Table 15-38. EMACRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	MMCRX	R	0x0	<p>MMC Receive Interrupt Status. This bit is cleared when all of the bits in the Ethernet MAC MMC Receive Interrupt (EMACMMCRXRIS) register are clear.</p> <p>0x0 = No interrupts exist in the MAC MMC Receive Interrupt (EMACMMCTXRIS) register.</p> <p>0x1 = Indicates an interrupt has been generated in the MAC MMC Receive Interrupt (EMACMMCRXRIS) register.</p>
4	MMC	R	0x0	<p>MMC Interrupt Status.</p> <p>0x0 = Indicates the MMC-related Interrupt bits [6:5] in this register are clear.</p> <p>0x1 = Indicates that one or more of the MMC-related interrupt bits [6:5] in this register are set.</p>
3	PMT	R	0x0	<p>PMT Interrupt Status.</p> <p>0x0 = This bit is cleared when both Bits[6:5] are cleared because of a read operation to the MAC PMT Control and Status Register (EMACPMTCTRLSTAT) register.</p> <p>0x1 = Indicates a Magic packet or Wake-on-LAN frame is received in the power-down mode (see Bits 5 and 6 in the MACPMTCTRLSTAT register).</p>
2-0	RESERVED	R	0x0	

15.6.15 EMACIM Register (Offset = 0x3C) [reset = 0x0]

Ethernet MAC Interrupt Mask (EMACIM)

The Ethernet MAC Interrupt Mask (EMACIM) Register bits enables the application to mask the interrupt signal caused by the corresponding event in the Ethernet MAC Raw Interrupt Status (EMACRIS) Register.

EMACIM is shown in [Figure 15-30](#) and described in [Table 15-39](#).

Return to [Summary Table](#).

Figure 15-30. EMACIM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					LPI	TSI	RESERVED					PMT	RESERVED		
R-0x0					R/W-0x0	R/W-0x0	R-0x0					R/W-0x0	R-0x0		

Table 15-39. EMACIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
10	LPI	R/W	0x0	LPI Interrupt Mask. 0x0 = The LPI interrupt status bit in the MAC Raw Interrupt Status (EMACRIS) register is not masked and can cause an interrupt. 0x1 = The assertion of the LPI interrupt status bit in the MAC Raw Interrupt Status (EMACRIS) register is masked and does not cause an interrupt.
9	TSI	R/W	0x0	Timestamp Interrupt Mask. 0x0 = The TSI interrupt status bit in the MAC Raw Interrupt Status (EMACRIS) register is not masked and can cause an interrupt. 0x1 = The assertion of the TIS interrupt status bit in the MAC Raw Interrupt Status (EMACRIS) register is masked and does not cause an interrupt.
8-4	RESERVED	R	0x0	
3	PMT	R/W	0x0	PMT Interrupt Mask. 0x0 = The PMT interrupt status bit in the MAC Raw Interrupt Status (EMACRIS) register is not masked and can cause an interrupt. 0x1 = The assertion of the PMT interrupt status bit in the MAC Raw Interrupt Status (EMACRIS) register is masked and does not cause an interrupt.
2-0	RESERVED	R	0x0	

15.6.16 EMACADDR0H Register (Offset = 0x40) [reset = 0x8000FFFF]

Ethernet MAC Address 0 High (EMACADDR0H)

The Ethernet MAC Address 0 High (EMACADDR0H) register holds the upper 16 bits of the first 6-byte MAC address of the station. The first (Destination Address) DA byte that is received on the MII interface corresponds to the least significant byte (Bits [7:0]) of the MAC Address 0 Low Register (EMACADDR0L) register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the EMAC Address 0 (EMACADDR0x) register [47:0] is compared with 0x665544332211.

EMACADDR0H is shown in [Figure 15-31](#) and described in [Table 15-40](#).

Return to [Summary Table](#).

Figure 15-31. EMACADDR0H Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	RESERVED														
R-0x1	R-0x0														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
R/W-0xFFFF															

Table 15-40. EMACADDR0H Register Field Descriptions

Bit	Field	Type	Reset	Description
31	AE	R	0x1	Address Enable. This register is always valid, so this bit is set always set to 1.
30-16	RESERVED	R	0x0	
15-0	ADDRHI	R/W	0xFFFF	MAC Address0 [47:32]. This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

15.6.17 EMACADDR0L Register (Offset = 0x44) [reset = 0xFFFFFFFF]

Ethernet MAC Address 0 Low Register (EMACADDR0L)

The MAC Address 0 Low Register (EMACADDR0L) register holds the lower 32 bits of the first 6-byte MAC address of the station.

EMACADDR0L is shown in [Figure 15-32](#) and described in [Table 15-41](#).

Return to [Summary Table](#).

Figure 15-32. EMACADDR0L Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO																															
R/W-0xFFFFFFFF																															

Table 15-41. EMACADDR0L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRLO	R/W	0xFFFFFFFF	MAC Address0 [31:0]. This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

15.6.18 EMACADDR1H Register (Offset = 0x48) [reset = 0xFFFF]

Ethernet MAC Address 1 High (EMACADDR1H)

The MAC Address 1 High (EMACADDR1H) register holds the upper 16 bits of the second 6-byte MAC address of the station.

EMACADDR1H is shown in [Figure 15-33](#) and described in [Table 15-42](#).

Return to [Summary Table](#).

Figure 15-33. EMACADDR1H Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC						RESERVED							
R/W-0x0	R/W-0x0	R/W-0x0						R-0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
R/W-0xFFFF															

Table 15-42. EMACADDR1H Register Field Descriptions

Bit	Field	Type	Reset	Description
31	AE	R/W	0x0	Address Enable. 0x0 = The address filter module ignores the second address for filtering. 0x1 = The address filter module uses the second address for perfect filtering.
30	SA	R/W	0x0	Source Address. 0x0 = When this bit is reset, MAC Address1[47:0] is used to compare with the DA fields of the received frame. 0x1 = When this bit is set, MAC Address1[47:0] is used to compare with the SA fields of the received frame.
29-24	MBC	R/W	0x0	Mask Byte Control. Mask control bits are provided for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: Bit 29: ADDRHI [15:8] of EMACADDR1H Register Bit 28: ADDRHI [7:0] of EMACADDR1H Bit 27: ADDRLO [31:24] of EMACADDR1L register Bit 26: ADDRLO [23:16] of EMACADDR1L register Bit 25: ADDRLO [15:8] of EMACADDR1L register Bit 24: ADDRLO [7:0] of EMACADDR1L register A group of addresses (known as group address filtering) can be filtered by masking one or more bytes of the address.
23-16	RESERVED	R	0x0	
15-0	ADDRHI	R/W	0xFFFF	MAC Address1 [47:32]. This field contains the upper 16 bits (47:32) of the second 6-byte MAC address.

15.6.19 EMACADDR1L Register (Offset = 0x4C) [reset = 0xFFFFFFFF]

Ethernet MAC Address 1 Low (EMACADDR1L)

The MAC Address 1 Low (EMACADDR1L) register holds the lower 32 bits of the second 6-byte MAC address of the station.

EMACADDR1L is shown in [Figure 15-34](#) and described in [Table 15-43](#).

Return to [Summary Table](#).

Figure 15-34. EMACADDR1L Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO																															
R/W-0xFFFFFFFF																															

Table 15-43. EMACADDR1L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRLO	R/W	0xFFFFFFFF	MAC Address1 [31:0]. This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process.

15.6.20 EMACADDR2H Register (Offset = 0x50) [reset = 0xFFFF]

Ethernet MAC Address 2 High (EMACADDR2H)

The MAC Address 2 High (EMACADDR2H) register holds the upper 16 bits of the third 6-byte MAC address of the station.

EMACADDR2H is shown in [Figure 15-35](#) and described in [Table 15-44](#).

Return to [Summary Table](#).

Figure 15-35. EMACADDR2H Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC						RESERVED							
R/W-0x0	R/W-0x0	R/W-0x0						R-0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
R/W-0xFFFF															

Table 15-44. EMACADDR2H Register Field Descriptions

Bit	Field	Type	Reset	Description
31	AE	R/W	0x0	Address Enable. 0x0 = The address filter module ignores the third address for filtering. 0x1 = The address filter module uses the third address for perfect filtering.
30	SA	R/W	0x0	Source Address. 0x0 = When this bit is reset, the MAC Address2[47:0] is used to compare with the DA fields of the received frame. 0x1 = When this bit is set, the MAC Address2[47:0] is used to compare with the SA fields of the received frame.
29-24	MBC	R/W	0x0	Mask Byte Control. Mask control bits are provided for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: Bit 29: ADDRHI [15:8] of EMACADDR2H register Bit 28: ADDRHI [7:0] of EMACADDR2H register Bit 27: ADDRLO [31:24] of EMACADDR2L register Bit 26: ADDRLO [23:16] of EMACADDR2L register Bit 25: ADDRLO [15:8] of EMACADDR2L register Bit 24: ADDRLO [7:0] of EMACADDR2L register A group of addresses (known as group address filtering) can be filtered by masking one or more bytes of the address.
23-16	RESERVED	R	0x0	
15-0	ADDRHI	R/W	0xFFFF	MAC Address2 [47:32]. This field contains the upper 16 bits [47:32] of the third 6-byte MAC address.

15.6.21 EMACADDR2L Register (Offset = 0x54) [reset = 0xFFFFFFFF]

Ethernet MAC Address 2 Low (EMACADDR2L)

The MAC Address2 Low register holds the lower 32 bits of the third 6-byte MAC address of the station.

EMACADDR2L is shown in [Figure 15-36](#) and described in [Table 15-45](#).

Return to [Summary Table](#).

Figure 15-36. EMACADDR2L Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO																															
R/W-0xFFFFFFFF																															

Table 15-45. EMACADDR2L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRLO	R/W	0xFFFFFFFF	MAC Address2 [31:0]. This field contains the lower 32 bits of the third 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

15.6.22 EMACADDR3H Register (Offset = 0x58) [reset = 0xFFFF]

Ethernet MAC Address 3 High (EMACADDR3H)

The Ethernet MAC Address 3 High (EMACADDR3H) register holds the upper 16 bits of the fourth 6-byte MAC address of the station.

EMACADDR3H is shown in [Figure 15-37](#) and described in [Table 15-46](#).

Return to [Summary Table](#).

Figure 15-37. EMACADDR3H Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC						RESERVED							
R/W-0x0	R/W-0x0	R/W-0x0						R-0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
R/W-0xFFFF															

Table 15-46. EMACADDR3H Register Field Descriptions

Bit	Field	Type	Reset	Description
31	AE	R/W	0x0	Address Enable. 0x0 = The address filter module ignores the third address for filtering. 0x1 = The address filter module uses the third address for perfect filtering.
30	SA	R/W	0x0	Source Address. 0x0 = When this bit is reset, the MAC Address3[47:0] is used to compare with the DA fields of the received frame. 0x1 = When this bit is set, the MAC Address3[47:0] is used to compare with the SA fields of the received frame.
29-24	MBC	R/W	0x0	Mask Byte Control. Mask control bits are provided for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: Bit 29: ADDRHI [15:8] of EMACADDR3H register Bit 28: ADDRHI [7:0] of EMACADDR3H register Bit 27: ADDRLO [31:24] of EMACADDR3L register Bit 26: ADDRLO [23:16] of EMACADDR3L register Bit 25: ADDRLO [15:8] of EMACADDR3L register Bit 24: ADDRLO [7:0] of EMACADDR3L register A group of addresses (known as group address filtering) can be filtered by masking one or more bytes of the address.
23-16	RESERVED	R	0x0	
15-0	ADDRHI	R/W	0xFFFF	MAC Address3 [47:32]. This field contains the upper 16 bits [47:32] of the fourth 6-byte MAC address.

15.6.23 EMACADDR3L Register (Offset = 0x5C) [reset = 0xFFFFFFFF]

Ethernet MAC Address 3 Low (EMACADDR3L)

The Ethernet MAC Address 3 Low (EMACADDR3L) register holds the lower 32 bits of the fourth 6-byte MAC address of the station.

EMACADDR3L is shown in [Figure 15-38](#) and described in [Table 15-47](#).

Return to [Summary Table](#).

Figure 15-38. EMACADDR3L Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO																															
R/W-0xFFFFFFFF																															

Table 15-47. EMACADDR3L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRLO	R/W	0xFFFFFFFF	MAC Address3 [31:0]. This field contains the lower 32 bits of the fourth 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process.

15.6.24 EMACWDOGTO Register (Offset = 0xDC) [reset = 0x0]

Ethernet MAC Watchdog Time-out (EMACWDOGTO)

This register controls the watchdog counter for received frames.

EMACWDOGTO is shown in [Figure 15-39](#) and described in [Table 15-48](#).

Return to [Summary Table](#).

Figure 15-39. EMACWDOGTO Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							PWE
R-0x0							R/W-0x0
15	14	13	12	11	10	9	8
RESERVED		WTO					
R-0x0		R/W-0x0					
7	6	5	4	3	2	1	0
WTO							
R/W-0x0							

Table 15-48. EMACWDOGTO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	PWE	R/W	0x0	Programmable Watchdog Enable. 0x0 = The watchdog time-out for a received frame is controlled by setting the WD and JE bits in the EMACCFG register. 0x1 = When the WD bit of the EMACCFG register is clear, the WTO field is used as a watchdog time-out for a received frame.
15-14	RESERVED	R	0x0	
13-0	WTO	R/W	0x0	Watchdog Time-out. When the PWE bit in the EMACWDOGTO register is set and the WD bit of the EMACCFG register is clear, this field is used as a watchdog time-out value for a received frame. If the length of a received frame exceeds the value of this field, such frame is terminated and declared an error frame. When the PWE bit is set the value in this field should be more than 1522 (0x05F2). Otherwise, valid tagged IEEE 802.3 frames are declared as error frames and are dropped.

15.6.25 EMACMMCCTRL Register (Offset = 0x100) [reset = 0x0]

Ethernet MAC MMC Control (EMACMMCCTRL)

The MMC Control register establishes the operating mode of the management counters.

The CNTRST bit has higher priority than the CNTPRST. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST is not set.

EMACMMCCTRL is shown in [Figure 15-40](#) and described in [Table 15-49](#).

Return to [Summary Table](#).

Figure 15-40. EMACMMCCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							UCDBC
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
RESERVED		CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTPRO	CNTRST
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 15-49. EMACMMCCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	UCDBC	R/W	0x0	Update MMC Counters for Dropped Broadcast Frames. 0x0 = MMC Counters are not updated for dropped broadcast frames. 0x1 = MAC updates all related MMC counters for broadcast frames dropped due to setting of DBF bit (Disable Broadcast Frames) of the MAC Frame Filter(EMACFRAMEFLTR) register.
7-6	RESERVED	R	0x0	
5	CNTPRSTLVL	R/W	0x0	Full/Half Preset Level Value. This bit, along with CNTPRST, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0. 0x0 = If CNTPRST is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0xFFFF.F800 (half value - 2 KB) and all frame-counters get preset to 0xFFFF.FFF0 (half value - 16). 0x1 = If CNTPRST is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF.F800 (full value - 2 KB) and all frame-counters gets preset to 0xFFFF.FFF0 (full value - 16).
4	CNTPRST	R/W	0x0	Counters Preset. This bit, along with CNTPRSTLVL, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. 0x0 = Normal operation. 0x1 = All counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. The CNTPRST bit is cleared automatically after 1 clock cycle.

Table 15-49. EMACMMCCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	CNTFREEZ	R/W	0x0	<p>MMC Counter Freeze. When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset, no MMC counter is updated because of any transmitted or received frame. If any MMC counter is read with the RSTONRD bit set, then that counter is also cleared in this mode.</p> <p>0x0 = MMC counters are updated when a frame is transmitted or received.</p> <p>0x1 = When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received frame.</p>
2	RSTONRD	R/W	0x0	<p>Reset on Read.</p> <p>0x0 = No effect.</p> <p>0x1 = MMC counters are reset to zero after a read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.</p>
1	CNTSTPRO	R/W	0x0	<p>Counters Stop Rollover.</p> <p>0x0 = No effect.</p> <p>0x1 = After reaching maximum value, the MMC counters do not roll over to zero.</p>
0	CNTRST	R/W	0x0	<p>Counters Reset.</p> <p>0x0 = No effect</p> <p>0x1 = All MMC counters are reset. This bit is cleared automatically after one clock cycle.</p>

15.6.26 EMACMMCRXRIS Register (Offset = 0x104) [reset = 0x0]

Ethernet MAC MMC Receive Raw Interrupt Status (EMACMMCRXRIS)

The MAC MMC Receive Interrupt (EMACMMCRXRIS) register maintains the interrupts that are generated when the following happens:

- Receive statistic counters reach half of their maximum values (0x80000000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFFFFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the Counter Stop Rollover (CNTSTPRO bit) in the MAC MMC Control (EMACMMCCTRL) register is set, interrupts are set but the counter remains at all-ones. The EMACMMCRXRIS register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

EMACMMCRXRIS is shown in [Figure 15-41](#) and described in [Table 15-50](#).

Return to [Summary Table](#).

Figure 15-41. EMACMMCRXRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						UCGF	RESERVED
R-0x0						R-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	ALGNERR	CRCERR	RESERVED				GBF
R-0x0	R-0x0	R-0x0	R-0x0				R-0x0

Table 15-50. EMACMMCRXRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	UCGF	R	0x0	MMC Receive Unicast Good Frame Counter Interrupt Status. 0x0 = The receive Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI) register has not reached half of the maximum value or the maximum value. 0x1 = The receive Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI) register has reached half of the maximum value or the maximum value.
16-7	RESERVED	R	0x0	
6	ALGNERR	R	0x0	MMC Receive Alignment Error Frame Counter Interrupt Status. 0x0 = The Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR) register has reached half of the maximum value or the maximum value.

Table 15-50. EMACMMCRXRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	CRCERR	R	0x0	MMC Receive CRC Error Frame Counter Interrupt Status. 0x0 = The Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR) register has reached half of the maximum value or the maximum value.
4-1	RESERVED	R	0x0	
0	GBF	R	0x0	MMC Receive Good Bad Frame Counter Interrupt Status. 0x0 = The Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB) register has reached half of the maximum value or the maximum value.

15.6.27 EMACMMCTXRIS Register (Offset = 0x108) [reset = 0x0]

Ethernet MAC MMC Transmit Raw Interrupt Status (EMACMMCTXRIS)

The MAC MMC Transmit Interrupt (EMACMMCTXRIS) register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the CNTSTPRO bit is set in the MAC MMC Control (EMACMMCTRL) register, interrupts are set but the counter remains at all-ones. The EMACMMCTXRIS register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

EMACMMCTXRIS is shown in [Figure 15-42](#) and described in [Table 15-51](#).

Return to [Summary Table](#).

Figure 15-42. EMACMMCTXRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED			OCTCNT	RESERVED			
R-0x0			R-0x0	R-0x0			
15	14	13	12	11	10	9	8
MCOLLGF	SCOLLGF	RESERVED					
R-0x0	R-0x0	R-0x0					
7	6	5	4	3	2	1	0
RESERVED						GBF	RESERVED
R-0x0						R-0x0	R-0x0

Table 15-51. EMACMMCTXRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	RESERVED	R	0x0	
20	OCTCNT	R	0x0	Octet Counter Interrupt Status. 0x0 = The Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG) register has reached half of the maximum value or the maximum value.
19-16	RESERVED	R	0x0	
15	MCOLLGF	R	0x0	MMC Transmit Multiple Collision Good Frame Counter Interrupt Status. 0x0 = The Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL) register has reached half of the maximum value or the maximum value.
14	SCOLLGF	R	0x0	MMC Transmit Single Collision Good Frame Counter Interrupt Status. 0x0 = The Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL) register has reached half of the maximum value or the maximum value.
13-2	RESERVED	R	0x0	

Table 15-51. EMACMMCTXRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	GBF	R	0x0	MMC Transmit Good Bad Frame Counter Interrupt Status. 0x0 = The Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB) register has not reached half of the maximum value or the maximum value. 0x1 = The Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB) register has reached half of the maximum value or the maximum value.
0	RESERVED	R	0x0	

15.6.28 EMACMMCRXIM Register (Offset = 0x10C) [reset = 0x0]

Ethernet MAC MMC Receive Interrupt Mask (EMACMMCRXIM)

The MAC MMC Receive Interrupt Mask (EMACMMCRXIM) register maintains the masks for the interrupts generated when the receive statistic counters reach half of their maximum value, or maximum value. This register is 32-bits wide.

EMACMMCRXIM is shown in [Figure 15-43](#) and described in [Table 15-52](#).

Return to [Summary Table](#).

Figure 15-43. EMACMMCRXIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						UCGF	RESERVED
R-0x0						R/W-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	ALGNERR	CRCERR	RESERVED				GBF
R-0x0	R/W-0x0	R/W-0x0	R-0x0				R/W-0x0

Table 15-52. EMACMMCRXIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	UCGF	R/W	0x0	MMC Receive Unicast Good Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the UCGF bit in the EMACMMCRXRIS register is set. 0x1 = The UCGF interrupt is suppressed and not sent to the interrupt controller.
16-7	RESERVED	R	0x0	
6	ALGNERR	R/W	0x0	MMC Receive Alignment Error Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the ALGNERR bit in the EMACMMCRXRIS register is set. 0x1 = The ALGNERR interrupt is suppressed and not sent to the interrupt controller.
5	CRCERR	R/W	0x0	MMC Receive CRC Error Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the CRCERR bit in the EMACMMCRXRIS register is set. 0x1 = The CRCERR interrupt is suppressed and not sent to the interrupt controller.
4-1	RESERVED	R	0x0	
0	GBF	R/W	0x0	MMC Receive Good Bad Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the GBF bit in the EMACMMCRXRIS register is set. 0x1 = The GBF interrupt is suppressed and not sent to the interrupt controller.

15.6.29 EMACMMCTXIM Register (Offset = 0x110) [reset = 0x0]

Ethernet MAC MMC Transmit Interrupt Mask (EMACMMCTXIM)

The MAC MMC Transmit Interrupt Mask (EMACMMCTXIM) register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

EMACMMCTXIM is shown in [Figure 15-44](#) and described in [Table 15-53](#).

Return to [Summary Table](#).

Figure 15-44. EMACMMCTXIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED			OCTCNT	RESERVED			
R-0x0			R/W-0x0	R-0x0			
15	14	13	12	11	10	9	8
MCOLLGF	SCOLLGF	RESERVED					
R/W-0x0	R/W-0x0	R-0x0					
7	6	5	4	3	2	1	0
RESERVED						GBF	RESERVED
R-0x0						R/W-0x0	R-0x0

Table 15-53. EMACMMCTXIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	RESERVED	R	0x0	
20	OCTCNT	R/W	0x0	MMC Transmit Good Octet Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the OCTCNT bit in the EMACMMCTXRIS register is set. 0x1 = The OCTCNT interrupt is suppressed and not sent to the interrupt controller.
19-16	RESERVED	R	0x0	
15	MCOLLGF	R/W	0x0	MMC Transmit Multiple Collision Good Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the MCOLLGF bit in the EMACMMCTXRIS register is set. 0x1 = The MCOLLGF interrupt is suppressed and not sent to the interrupt controller.
14	SCOLLGF	R/W	0x0	MMC Transmit Single Collision Good Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the SCOLLGF bit in the EMACMMCTXRIS register is set. 0x1 = The SCOLLGF interrupt is suppressed and not sent to the interrupt controller.
13-2	RESERVED	R	0x0	
1	GBF	R/W	0x0	MMC Transmit Good Bad Frame Counter Interrupt Mask. 0x0 = An interrupt is sent to the interrupt controller when the GBF bit in the EMACMMCTXRIS register is set. 0x1 = The GBF interrupt is suppressed and not sent to the interrupt controller.
0	RESERVED	R	0x0	

15.6.30 EMAXCNTGB Register (Offset = 0x118) [reset = 0x0]

Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMAXCNTGB)

The MAC Transmit Frame Count for Good and Bad Frames (EMAXCNTGB) register maintains the number of good and bad frames transmitted, exclusive of retried frames.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCTRL).

EMAXCNTGB is shown in [Figure 15-45](#) and described in [Table 15-54](#).

Return to [Summary Table](#).

Figure 15-45. EMAXCNTGB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXFRMGB																															
R-0x0																															

Table 15-54. EMAXCNTGB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TXFRMGB	R	0x0	This field indicates the number of good and bad frames transmitted, exclusive of retried frames.

15.6.31 EMAXCNTSCOL Register (Offset = 0x14C) [reset = 0x0]

Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMAXCNTSCOL)

This register maintains the number of successfully transmitted frames after a single collision in the half-duplex mode.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCCTRL).

EMAXCNTSCOL is shown in [Figure 15-46](#) and described in [Table 15-55](#).

Return to [Summary Table](#).

Figure 15-46. EMAXCNTSCOL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSNGLCOLG																															
R-0x0																															

Table 15-55. EMAXCNTSCOL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TXSNGLCOLG	R	0x0	This field indicates the number of successfully transmitted frames after a single collision in the half-duplex mode.

15.6.32 EMAXCNTMCOL Register (Offset = 0x150) [reset = 0x0]

Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMAXCNTMCOL)

This register maintains the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMAXMMCCTRL).

EMAXCNTMCOL is shown in [Figure 15-47](#) and described in [Table 15-56](#).

Return to [Summary Table](#).

Figure 15-47. EMAXCNTMCOL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMULTCOLG																															
R-0x0																															

Table 15-56. EMAXCNTMCOL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TXMULTCOLG	R	0x0	This field indicates the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

15.6.33 EMAXOCTCNTG Register (Offset = 0x164) [reset = 0x0]

Ethernet MAC Transmit Octet Count Good (EMAXOCTCNTG)

This register maintains the number of bytes transmitted, exclusive of preamble, only in good frames.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCCTRL).

EMAXOCTCNTG is shown in [Figure 15-48](#) and described in [Table 15-57](#).

Return to [Summary Table](#).

Figure 15-48. EMAXOCTCNTG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXOCTG																															
R-0x0																															

Table 15-57. EMAXOCTCNTG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TXOCTG	R	0x0	This field indicates the number of bytes transmitted, exclusive of preamble, in good frames.

15.6.34 EMACRXCNTGB Register (Offset = 0x180) [reset = 0x0]

Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB)

This register maintains the number of received good and bad frames.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCCTRL).

EMACRXCNTGB is shown in [Figure 15-49](#) and described in [Table 15-58](#).

Return to [Summary Table](#).

Figure 15-49. EMACRXCNTGB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFRMGB																															
R-0x0																															

Table 15-58. EMACRXCNTGB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RXFRMGB	R	0x0	This field indicates the number of received good and bad frames.

15.6.35 EMACRXCNTCRCERR Register (Offset = 0x194) [reset = 0x0]

Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR)

This register maintains the number of frames received with CRC error.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCCTRL).

EMACRXCNTCRCERR is shown in [Figure 15-50](#) and described in [Table 15-59](#).

Return to [Summary Table](#).

Figure 15-50. EMACRXCNTCRCERR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR																															
R-0x0																															

Table 15-59. EMACRXCNTCRCERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RXCRCERR	R	0x0	This field indicates the number of frames received with CRC error.

15.6.36 EMACRXCNTALGNERR Register (Offset = 0x198) [reset = 0x0]

Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR)

This register maintains the number of frames received with alignment (dribble) error.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCCTRL).

EMACRXCNTALGNERR is shown in [Figure 15-51](#) and described in [Table 15-60](#).

Return to [Summary Table](#).

Figure 15-51. EMACRXCNTALGNERR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR																															
R-0x0																															

Table 15-60. EMACRXCNTALGNERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RXALGNERR	R	0x0	This field indicates the number of frames received with alignment (dribble) error.

15.6.37 EMACRXCNTGUNI Register (Offset = 0x1C4) [reset = 0x0]

Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI)

This register maintains the number of received good unicast frames.

NOTE: This counter is reset to all zeros by setting the CNTRST bit in the Ethernet MAC MMC Control (EMACMMCCTRL).

EMACRXCNTGUNI is shown in [Figure 15-52](#) and described in [Table 15-61](#).

Return to [Summary Table](#).

Figure 15-52. EMACRXCNTGUNI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG																															
R-0x0																															

Table 15-61. EMACRXCNTGUNI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RXUCASTG	R	0x0	This field indicates the number of received good unicast frames.

15.6.38 EMACVLNINCREP Register (Offset = 0x584) [reset = 0x0]

Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP)

The MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP) register contains the VLAN tag for insertion or replacement in the transmit frames.

EMACVLNINCREP is shown in [Figure 15-53](#) and described in [Table 15-62](#).

Return to [Summary Table](#).

Figure 15-53. EMACVLNINCREP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				CSVL	VLP	VLC	
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	
15	14	13	12	11	10	9	8
VLT							
R/W-0x0							
7	6	5	4	3	2	1	0
VLT							
R/W-0x0							

Table 15-62. EMACVLNINCREP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	CSVL	R/W	0x0	C-VLAN or S-VLAN. 0x0 = C-VLAN type (0x8100) is inserted or replaced in the transmitted frames. 0x1 = S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted frames.
18	VLP	R/W	0x0	VLAN Priority Control. 0x0 = The control input from the transmit descriptor is used, and the VLC bit field (bits [17:16]) is ignored. 0x1 = The VLC bit field is used for VLAN deletion, insertion, or replacement.
17-16	VLC	R/W	0x0	VLAN Tag Control in Transmit Frames. Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value. 0x0 = No VLAN tag deletion, insertion, or replacement 0x1 = VLAN tag deletionThe MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted frames with VLAN tags. 0x2 = VLAN tag insertionThe MAC inserts VLT in bytes 15 and 16 of the frame after inserting the Type value (0x8100/0x88a8) in bytes 13 and 14. This operation is performed on all transmitted frames, irrespective of whether they already have a VLAN tag. 0x3 = VLAN tag replacementThe MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted frames (Bytes 13 and 14 are 0x8100/0x88a8).
15-0	VLT	R/W	0x0	VLAN Tag for Transmit Frames. This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority, Bit 12 is the CFI/DEI, and Bits[11:0] are the VLAN tag's VID field.

15.6.39 EMACVLANTHASH Register (Offset = 0x588) [reset = 0x0]

Ethernet MAC VLAN Hash Table (EMACVLANTHASH)

The 16-bit hash table is used for group address filtering based on VLAN tag when the VTHM bit of the EMACVLANTG register is set. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (enabled by ETV in the EMACVLANTG register) in the incoming frame is passed through the CRC logic and the upper four bits of the calculated CRC are used to index the contents of the EMACVLANTHASH register. For example, a hash value of 0x8 selects bit 8 of the VLAN Hash table. The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper four bits from the value obtained in Step 2.

If the corresponding bit value of the register is 0x1, the frame is accepted. Otherwise, it is rejected.

EMACVLANTHASH is shown in [Figure 15-54](#) and described in [Table 15-63](#).

Return to [Summary Table](#).

Figure 15-54. EMACVLANTHASH Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VLHT															
R-0x0																R/W-0x0															

Table 15-63. EMACVLANTHASH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	VLHT	R/W	0x0	VLAN Hash Table. This field contains the 16-bit VLAN Hash Table.

15.6.40 EMACTIMSTCTRL Register (Offset = 0x700) [reset = 0x2000]

Ethernet MAC Timestamp Control (EMACTIMSTCTRL)

This register controls the operation of the system time generator and the processing of Precision Time Protocol (PTP) packets for time-stamping in the receiver.

EMACTIMSTCTRL is shown in [Figure 15-55](#) and described in [Table 15-64](#).

Return to [Summary Table](#).

Figure 15-55. EMACTIMSTCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED					PTPFLTR	SELPTP	
R-0x0					R/W-0x0	R/W-0x0	
15	14	13	12	11	10	9	8
TSMAS	TSEVNT	PTPIPV4	PTPIPV6	PTPETH	PTPVER2	DGTLBIN	ALLF
R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED		ADDREGUP	INTTRIG	TSUPDT	TSINIT	TSFCUPDT	TSEN
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 15-64. EMACTIMSTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0x0	
18	PTPFLTR	R/W	0x0	Enable MAC address for PTP Frame Filtering. 0x0 = No effect. 0x1 = The Destination Address (DA) MAC address, that matches any MAC Address register is used to filter PTP frames when PTP is directly sent over Ethernet.
17-16	SELPTP	R/W	0x0	Select PTP packets for Taking Snapshots. These bits along with Bits 15 and 14 decide the set of PTP packet types for which a snapshot needs to be taken.
15	TSMAS	R/W	0x0	Enable Snapshot for Messages Relevant to Master. 0x0 = The snapshot is taken for the messages relevant to the slave node. 0x1 = The snapshot is taken only for the messages relevant to the master node.
14	TSEVNT	R/W	0x0	Enable Timestamp Snapshot for Event Messages. 0x0 = The snapshot is taken for all messages except Announce, Management, and Signaling. 0x1 = The timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp).
13	PTPIPV4	R/W	0x1	Enable Processing of PTP Frames Sent over IPv4-UDP. 0x0 = The MAC ignores PTP transported over UDP-IPv4 packets. This bit is set by default. 0x1 = The MAC receiver processes PTP packets encapsulated in UDP over IPv4 packets.
12	PTPIPV6	R/W	0x0	Enable Processing of PTP Frames Sent Over IPv6-UDP. 0x0 = The MAC ignores PTP transported over UDP-IPv6 packets. 0x1 = The MAC receiver processes PTP packets encapsulated in UDP over IPv6 packets.

Table 15-64. EMACTIMSTCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	PTPETH	R/W	0x0	Enable Processing of PTP Over Ethernet Frames. 0x0 = The MAC ignores PTP over Ethernet packets. 0x1 = The MAC receiver processes PTP packets encapsulated directly in Ethernet frames.
10	PTPVER2	R/W	0x0	Enable PTP Packet Processing For Version 2 Format. 0x0 = PTP packets are processed using the IEEE 1588 version 1 format. 0x1 = PTP packets are processed using the IEEE 1588 version 2 format.
9	DGTLBIN	R/W	0x0	Timestamp Digital or Binary Rollover Control. 0x0 = The TTSLO field of the EMACTARGNANO register, rolls over after 0x7FFF_FFFF and increments the EMACTARGSEC register. The sub-second increment has to be programmed correctly depending on the MOSC frequency and the value of this bit. 0x1 = The EMACTARGNANO register rolls over after 0x3B9A.C9FF value (that is, 1 nanosecond accuracy) and increments the EMACTARGSEC register.
8	ALLF	R/W	0x0	Enable Timestamp For All Frames. 0x0 = No effect. 0x1 = Timestamp snapshots are enabled for all frames received by the MAC.
7-6	RESERVED	R	0x0	
5	ADDREGUP	R/W	0x0	Addend Register Update. 0x0 = No effect. 0x1 = When set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it.
4	INTTRIG	R/W	0x0	Timestamp Interrupt Trigger Enable. 0x0 = No effect. 0x1 = The timestamp interrupt is generated when the System Time becomes greater than the value written in the Ethernet MAC Target Time Seconds/Nanoseconds (EMACTARGSEC/EMACTARGNANO) registers. This bit is reset after the generation of the Timestamp Trigger Interrupt.
3	TSUPDT	R/W	0x0	Timestamp Update. This bit should be read zero before updating it. This bit is reset when the update is completed in hardware. The Timestamp Higher Word register is not updated. 0x0 = No effect. 0x1 = When set, the system time is updated (added or subtracted) with the value specified in EMACTIMSECU (System Time - Seconds Update Register) and EMACTIMNANOU (System Time - Nanoseconds Update Register).
2	TSINIT	R/W	0x0	Timestamp Initialize. This bit should read zero before updating it. 0x0 = This bit is reset when the initialization is complete. 0x1 = The system time is initialized (overwritten) with the value specified in the Ethernet MAC System Time-Seconds Update (EMACTIMSECU) and the Ethernet MAC System Time-Nanoseconds Update (EMACTIMNANOU) registers.
1	TSFCUPDT	R/W	0x0	Timestamp Fine or Coarse Update. 0x0 = Indicates the system timestamp update should be done using the coarse method. 0x1 = Indicates that the system times update should be done using the fine update method.

Table 15-64. EMACTIMSTCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	TSEN	R/W	0x0	<p>Timestamp Enable. The EMACTIMSEC and the EMACTIMNANO registers must be initialized after enabling this mode. On the receive side, the MAC processes 1588 frames only if this bit is set.</p> <p>0x0 = The timestamp is not added for the transmit and receive frames and the timestamp generator module is also suspended.</p> <p>0x1 = The timestamp is added for the transmit and receive frames</p>

15.6.41 EMACSUBSECINC Register (Offset = 0x704) [reset = 0x0]

Ethernet MAC Sub-Second Increment (EMACSUBSECINC)

In the Coarse Update mode (enabled by the TSCFUPDT bit in the MAC Timestamp Control (EMACTIMSTCTRL) register), the value in the EMACSUBSECINC register is added to the system time every clock cycle of slave clock reference, MOSC. In the Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

EMACSUBSECINC is shown in [Figure 15-56](#) and described in [Table 15-65](#).

Return to [Summary Table](#).

Figure 15-56. EMACSUBSECINC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								SSINC							
R-0x0																								R/W-0x0							

Table 15-65. EMACSUBSECINC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	SSINC	R/W	0x0	Sub-second Increment Value. The value programmed in this field is accumulated every clock cycle (MOSC) with the contents of the sub-second register. For example, when MOSC is 25 MHz (period is 40 ns), SSINC should be programmed with the value 40 (0x28) when the Ethernet MAC System Time - Nanoseconds (EMACTIMNANO) register has an accuracy of 1 ns (DGTLBIN bit is set in EMACTIMSTCTRL). When DGTLBIN bit is clear, the EMACTIMNANO register has a resolution of ~0.465ns. In this case, a value of 86 (0x56), that is derived by 40ns/0.465, should be programmed in the SSINC field.

15.6.42 EMACTIMSEC Register (Offset = 0x708) [reset = 0x0]

Ethernet MAC System Time - Seconds (EMACTIMSEC)

The MAC System Time - Seconds (EMACTIMSEC) register, along with the MAC System Time - Nanoseconds (EMACTIMNANO) register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies.

EMACTIMSEC is shown in [Figure 15-57](#) and described in [Table 15-66](#).

Return to [Summary Table](#).

Figure 15-57. EMACTIMSEC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS																															
R-0x0																															

Table 15-66. EMACTIMSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TSS	R	0x0	Timestamp Second. The value in this field indicates the current value in seconds of the system time maintained by the MAC.

15.6.43 EMACTIMNANO Register (Offset = 0x70C) [reset = 0x0]

Ethernet MAC System Time - Nanoseconds (EMACTIMNANO)

The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When DGTLBIN in the EMACTIMSTCTRL register is set, each bit represents 1 ns and the maximum value is 0x3B9AC9FF, after which it rolls-over to zero.

EMACTIMNANO is shown in [Figure 15-58](#) and described in [Table 15-67](#).

Return to [Summary Table](#).

Figure 15-58. EMACTIMNANO Register

31	30	29	28	27	26	25	24
RESERVED	TSSS						
R-0x0	R-0x0						
23	22	21	20	19	18	17	16
TSSS							
R-0x0							
15	14	13	12	11	10	9	8
TSSS							
R-0x0							
7	6	5	4	3	2	1	0
TSSS							
R-0x0							

Table 15-67. EMACTIMNANO Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	
30-0	TSSS	R	0x0	Timestamp Sub-Seconds. The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When DGTLBIN in the EMACTIMSTCTRL register is set, each bit represents 1 ns and the maximum value is 0x3B9A.C9FF, after which it rolls over to zero.

15.6.44 EMACTIMSECU Register (Offset = 0x710) [reset = 0x0]

Ethernet MAC System Time - Seconds Update (EMACTIMSECU)

The MAC System Time - Seconds Update (EMACTIMSECU) register, along with the MAC System Time - Nanoseconds Update (EMACTIMNANOU) register, initializes or updates the system time maintained by the MAC. Both of these register must be written before setting the TSINIT or TSUPDT bits in the EMACTIMSTCTRL register.

EMACTIMSECU is shown in [Figure 15-59](#) and described in [Table 15-68](#).

Return to [Summary Table](#).

Figure 15-59. EMACTIMSECU Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS																															
R/W-0x0																															

Table 15-68. EMACTIMSECU Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TSS	R/W	0x0	Timestamp Second. The value in this field indicates the time in seconds to be initialized or added to the system time.

15.6.45 EMACTIMNANOU Register (Offset = 0x714) [reset = 0x0]

Ethernet MAC System Time - Nanoseconds Update (EMACTIMNANOU)

This register along with the EMACTIMSECU register initializes or updates the system time maintained by the MAC. Both of these register must be written before setting the TSINIT or TSUPDT bits in the EMACTIMSTCTRL register.

EMACTIMNANOU is shown in [Figure 15-60](#) and described in [Table 15-69](#).

Return to [Summary Table](#).

Figure 15-60. EMACTIMNANOU Register

31	30	29	28	27	26	25	24
ADDSUB	TSSS						
R/W-0x0	R/W-0x0						
23	22	21	20	19	18	17	16
TSSS							
R/W-0x0							
15	14	13	12	11	10	9	8
TSSS							
R/W-0x0							
7	6	5	4	3	2	1	0
TSSS							
R/W-0x0							

Table 15-69. EMACTIMNANOU Register Field Descriptions

Bit	Field	Type	Reset	Description
31	ADDSUB	R/W	0x0	Add or subtract time. 0x0 = The time value is added with the contents of the update register. 0x1 = The time value is subtracted with the contents of the update register.
30-0	TSSS	R/W	0x0	Timestamp Sub-Second. The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When DGTLBIN in the EMACTIMSTCTRL register is set, each bit represents 1 ns and the programmed value should not exceed 0x3B9A.C9FF.

15.6.46 EMACTIMADD Register (Offset = 0x718) [reset = 0x0]

Ethernet MAC Timestamp Addend (EMACTIMADD)

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit is set in the EMACTIMSTCTRL register). This register content is added to a 32-bit accumulator every slave clock cycle (MOSC source) and the system time is updated whenever the accumulator overflows.

EMACTIMADD is shown in [Figure 15-61](#) and described in [Table 15-70](#).

Return to [Summary Table](#).

Figure 15-61. EMACTIMADD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR																															
R/W-0x0																															

Table 15-70. EMACTIMADD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TSAR	R/W	0x0	Timestamp Addend Register. This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

15.6.47 EMACTARGSEC Register (Offset = 0x71C) [reset = 0x0]

Ethernet MAC Target Time Seconds (EMACTARGSEC)

The MAC Target Time Seconds (EMACTARGSEC) register, along with the MAC Target Time Nanoseconds (EMACTARGNANO) register, is used to schedule an interrupt event.

EMACTARGSEC is shown in [Figure 15-62](#) and described in [Table 15-71](#).

Return to [Summary Table](#).

Figure 15-62. EMACTARGSEC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTR																															
R/W-0x0																															

Table 15-71. EMACTARGSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TSTR	R/W	0x0	Target Time Seconds Register. This register stores the time in seconds. When the timestamp value matches or exceeds both the MAC Target Time Seconds (EMACTARGSEC) and MAC Target Time Nanoseconds (EMACTARGNANO) registers, then based on the TRGMODS0 bit field in the MAC PPS Control (EMACPPSCTRL), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled).

15.6.48 EMACTARGNANO Register (Offset = 0x720) [reset = 0x0]

Ethernet MAC Target Time Nanoseconds (EMACTARGNANO)

The MAC Target Time Seconds (EMACTARGSEC) register, along with the MAC Target Time Nanoseconds (EMACTARGNANO) register, is used to schedule an interrupt event.

EMACTARGNANO is shown in [Figure 15-63](#) and described in [Table 15-72](#).

Return to [Summary Table](#).

Figure 15-63. EMACTARGNANO Register

31	30	29	28	27	26	25	24
TRGTBUSY	TTSLO						
R/W-0x0	R/W-0x0						
23	22	21	20	19	18	17	16
TTSLO							
R/W-0x0							
15	14	13	12	11	10	9	8
TTSLO							
R/W-0x0							
7	6	5	4	3	2	1	0
TTSLO							
R/W-0x0							

Table 15-72. EMACTARGNANO Register Field Descriptions

Bit	Field	Type	Reset	Description
31	TRGTBUSY	R/W	0x0	Target Time Register Busy. This bit is set and cleared by the MAC. 0x0 = The Ethernet MAC Target Time Seconds/Nanoseconds (EMACTARGSEC/EMACTARGNANO) registers are not busy. 0x1 = The Ethernet MAC Target Time Seconds/Nanoseconds (EMACTARGSEC/EMACTARGNANO) registers are busy. This bit is set when the PPSCTRL field in the EMACPPSCTRL register is programmed to 0x2 or 0x3 and the MAC is instructed to synchronize the EMACTARGSEC/EMACTARGNANO registers to the PTP clock domain. The EMACTARGSEC and EMACTARGNANO registers must not be updated when this bit is read as 1.
30-0	TTSLO	R/W	0x0	Target Timestamp Low Register. This register stores the time in (signed) nanoseconds. When the value of the timestamp matches both EMACTARGx registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGMODS0 field in the MAC PPS Control (EMACPPSCTRL) register. This value should not exceed 0x3B9A.C9FF when DGTLBIN is set in the EMACTIMSTCTRL register. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value.

15.6.49 EMACHWORDSEC Register (Offset = 0x724) [reset = 0x0]

Ethernet MAC System Time-Higher Word Seconds (EMACHWORDSEC)

This register is used to support the most significant 16-bits of the timestamp seconds value.

EMACHWORDSEC is shown in [Figure 15-64](#) and described in [Table 15-73](#).

Return to [Summary Table](#).

Figure 15-64. EMACHWORDSEC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TSHWR															
R-0x0																R/W-0x0															

Table 15-73. EMACHWORDSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	TSHWR	R/W	0x0	Target Timestamp Higher Word Register. This field contains the most significant 16-bits of the timestamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the System Time - Seconds register.

15.6.50 EMACTIMSTAT Register (Offset = 0x728) [reset = 0x0]

Ethernet MAC Timestamp Status (EMACTIMSTAT)

EMACTIMSTAT is shown in [Figure 15-65](#) and described in [Table 15-74](#).

Return to [Summary Table](#).

Figure 15-65. EMACTIMSTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						TSTARGET	TSSOVF
R-0x0						R-0x0	R-0x0

Table 15-74. EMACTIMSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	TSTARGET	R	0x0	Timestamp Target Time Reached. 0x0 = Target Time has not been reached. 0x1 = Indicates that the value of system time is greater or equal to the value specified in EMACTARGSEC (Target Time Seconds Register) and EMACTARGNANO (Target Time Nanoseconds Register).
0	TSSOVF	R	0x0	Timestamp Seconds Overflow. 0x0 = No timestamp overflow has occurred. 0x1 = Indicates the seconds value of the timestamp has overflowed beyond 0xFFFF.FFFF.

15.6.51 EMACPPSCTRL Register (Offset = 0x72C) [reset = 0x0]

Ethernet MAC PPS Control (EMACPPSCTRL)

This register is used to control the EN0PPS signal output.

NOTE: The PTP reference clock referred to below is MOSC clock in course update mode and in fine correction mode, is the clock tick at which the system time gets updated.

EMACPPSCTRL is shown in [Figure 15-66](#) and described in [Table 15-75](#).

Return to [Summary Table](#).

Figure 15-66. EMACPPSCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	TRGMODS0		PPSEN0	PPSCTRL			
R-0x0	R-0x0		R-0x0	R/W-0x0			

Table 15-75. EMACPPSCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6-5	TRGMODS0	R	0x0	Target Time Register Mode for PPS0 Output. This field indicates the Target Time registers (EMACTARGSEC and EMACTARGNANO) mode for the EN0PPS output signal: 0x0 = Indicates that the Target Time registers are programmed only for generating the interrupt event. 0x1 = Reserved 0x2 = Indicates that the Target Time registers are programmed for generating the interrupt event and starting or stopping the generation of the EN0PPS output signal. 0x3 = Indicates that the Target Time registers are programmed only for starting or stopping the generation of the EN0PPS output signal. No interrupt is asserted.
4	PPSEN0	R	0x0	Flexible PPS Output Mode Enable. 0x0 = Bits[3:0] function as PPS output frequency control (PPSCTRL). 0x1 = Bits[3:0] function as PPS Command (PPSCMD).

Table 15-75. EMACPPSCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	PPSCTRL	R/W	0x0	<p>EN0PPS Output Frequency Control (PPSCTRL) or Command Control (PPSCMD).</p> <p>This bit field has two different functions depending on how the PPSEN0 bit is set. See the values in Table 15-76.</p> <p>If the PPSEN0 bit is set to 0x0, this field functions as a PPS0 output frequency control (PPSCTRL), which controls the frequency of the output signal, EN0PPS. The default value of this field is 0x0 and the PPS output is one pulse every second.</p> <p>If the PPSEN0 bit is 0x1, this field functions as a flexible output command control for the EN0PPS signal. Programming the PPSCTRL bit field with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits are cleared automatically. Software should ensure that these bits are programmed only when they are "all-zeros."</p> <p>In the binary rollover mode, the EN0PPS signal has a duty cycle of 50 percent with these frequencies.</p> <p>In the digital rollover mode, the EN0PPS signal frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:</p> <p>When PPSCTRL = 0x1, EN0PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.</p> <p>When PPSCTRL = 0x2, EN0PPS (2 Hz) is a sequence of:</p> <ul style="list-style-type: none"> - One clock of 50 percent duty cycle and 537 ms period - Second clock of 463 ms period (268 ms low and 195 ms high) <p>When PPSCTRL = 0x3, EN0PPS (4 Hz) is a sequence of:</p> <ul style="list-style-type: none"> - Three clocks of 50 percent duty cycle and 268 ms period - Fourth clock of 195 ms period (134 ms low and 61 ms high) <p>This signaling behavior is because of the non-linear toggling of bits in the digital rollover mode in the Ethernet MAC System Time - Nanoseconds (EMACTIMNANO) register.</p>

Table 15-76. PPSCTRL Bit Field Values

Value	Description
0x0	<p>When the PPSEN0 bit = 0x0, the EN0PPS signal is 1 pulse of the PTP reference clock. every second.</p> <p>When the PPSEN0 bit = 0x1, this encoding indicates no command.</p>
0x1	<p>When the PPSEN0 bit = 0x0, the binary rollover is 2 Hz, and the digital rollover is 1 Hz.</p> <p>When the PPSEN0 bit = 0x1, START single pulse. This command generates a single pulse rising at the start point defined in the Ethernet MAC Target Time Second/Nanoseconds (EMACTARGX) registers (MAC offsets 0x71C and 0x720) and a duration defined in the Ethernet MAC PPS0 Width (EMACPPS0WIDTH) register (offset 0x764).</p>
0x2	<p>When the PPSEN0 bit = 0x0, the binary rollover is 4 Hz, and the digital rollover is 2 Hz.</p> <p>When the PPSEN0 bit = 0x1, START pulse train. This command generates the train of pulses rising at the start point defined in the Ethernet MAC Target Time Second/Nanoseconds (EMACTARGX) registers (MAC offsets 0x71C and 0x720) and repeated at an interval defined in the Ethernet MAC PPS0 Width (EMACPPS0WIDTH) register (offset 0x764). By default, the EN0PPS pulse train is free-running unless stopped by STOP pulse train at a time or STOP pulse train immediately command.</p>
0x3	<p>When the PPSEN0 bit = 0x0, the binary rollover is 8 Hz, and the digital rollover is 4 Hz,</p> <p>When the PPSEN0 bit = 0x1, cancel START. This command cancels the START single pulse and START pulse train commands if the system time has not crossed the programmed start time.</p>
0x4	<p>When the PPSEN0 bit = 0x0, the binary rollover is 16 Hz, and the digital rollover is 8 Hz.</p> <p>When the PPSEN0 bit = 0x1, STOP pulse train at time. This command stops the train of puses initiated by the START pulse train command after the time programmed in the Ethernet MAC Target Time Second/Nanoseconds (EMACTARGX) registers (MAC offsets 0x71C and 0x720).</p>
0x5	<p>When the PPSEN0 bit = 0x0, the binary rollover is 32 Hz, and the digital rollover is 16 Hz.</p> <p>When the PPSEN0 bit = 0x1, STOP pulse train immediately. This command immediately stops the train of pulses initiated by the START pulse train command.</p>

Table 15-76. PPSCCTRL Bit Field Values (continued)

Value	Description
0x6	When the PPSEN0 bit = 0x0, the binary rollover is 64 Hz, and the digital rollover is 32 Hz. When the PPSEN0 bit = 0x1, cancel STOP pulse train. This command cancels the STOP pulse train at the time command if the programmed stop time has not elapsed. The EN0PPS pulse train becomes free-running on the successful execution of this command.
0x7	When the PPSEN0 bit = 0x0, the binary rollover is 128 Hz, and the digital rollover is 64 Hz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0x8	When the PPSEN0 bit = 0x0, the binary rollover is 256 Hz, and the digital rollover is 128 Hz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0x9	When the PPSEN0 bit = 0x0, the binary rollover is 512 Hz, and the digital rollover is 256 Hz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0xA	When the PPSEN0 bit = 0x0, the binary rollover is 1.024 kHz, and the digital rollover is 512 Hz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0xB	When the PPSEN0 bit = 0x0, the binary rollover is 2.048 kHz, and the digital rollover is 1.024 kHz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0xC	When the PPSEN0 bit = 0x0, the binary rollover is 4.096 kHz, and the digital rollover is 2.048 kHz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0xD	When the PPSEN0 bit = 0x0, the binary rollover is 8.192 kHz, and the digital rollover is 4.096 kHz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0xE	When the PPSEN0 bit = 0x0, the binary rollover is 16.384 kHz, and the digital rollover is 8.092 kHz. When the PPSEN0 bit = 0x1, this encoding is reserved.
0xF	When the PPSEN0 bit = 0x0, the binary rollover is 32.768 kHz, and the digital rollover is 16.384 kHz. When the PPSEN0 bit = 0x1, this encoding is reserved.

15.6.52 EMACPPS0INTVL Register (Offset = 0x760) [reset = 0x0]

Ethernet MAC PPS0 Interval (EMACPPS0INTVL)

The MAC PPS0 Interval (EMACPPS0INTVL) register contains the number of units of sub-second increment value between the rising edges of EN0PPS signal output.

NOTE: The PTP reference clock referred to below is MOSC clock in course update mode and in fine correction mode, is the clock tick at which the system time gets updated.

EMACPPS0INTVL is shown in [Figure 15-67](#) and described in [Table 15-77](#).

Return to [Summary Table](#).

Figure 15-67. EMACPPS0INTVL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPS0INT																															
R/W-0x0																															

Table 15-77. EMACPPS0INTVL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PPS0INT	R/W	0x0	<p>PPS0 Output Signal Interval.</p> <p>These bits store the interval between the rising edges of the EN0PPS signal output in terms of units of sub-second increment value.</p> <p>It must be programmed one value less than the required interval. For example, if the PTP reference clock is 25 MHz (period of 40 ns), and desired interval between rising edges of EN0PPS signal output is 120 ns (that is, three units of sub-second increment value), then you should program value 2 (3 -1) in this register.</p>

15.6.53 EMACPPS0WIDTH Register (Offset = 0x764) [reset = 0x0]

Ethernet MAC PPS0 Width (EMACPPS0WIDTH)

The MAC PPS0 Width (EMACPPS0WIDTH) register contains the number of units of sub-second increment value between the rising and corresponding falling edges of the EN0PPS output signal.

NOTE: The PTP reference clock referred to below is MOSC clock in course update mode and in fine correction mode, is the clock tick at which the system time gets updated.

EMACPPS0WIDTH is shown in [Figure 15-68](#) and described in [Table 15-78](#).

Return to [Summary Table](#).

Figure 15-68. EMACPPS0WIDTH Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPS0WIDTH																															
R/W-0x0																															

Table 15-78. EMACPPS0WIDTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PPS0WIDTH	R/W	0x0	<p>EN0PPS Output Signal Width.</p> <p>These bits store the width between the rising edges and corresponding falling edge of the of the EN0PPS signal output in terms of units of sub-second increment value.</p> <p>It must be programmed one value less than the required interval. For example, if the PTP reference clock is 25 MHz (period of 40 ns), and the desired interval between rising edges of EN0PPS signal output is 80 ns (that is, two units of sub-second increment value), then you should program value 1 (2 -1) in this register.</p> <p>The value programmed in this register must be less than the value programmed in the Ethernet MAC PPS0 Interval (EMACPPS0INTVL) register.</p>

15.6.54 EMACDMABUSMOD Register (Offset = 0xC00) [reset = 0x00020101]

Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)

The Ethernet MAC DMA Bus Mode (EMACDMABUSMODE) register establishes the operation modes for the DMA.

EMACDMABUSMOD is shown in [Figure 15-69](#) and described in [Table 15-79](#).

Return to [Summary Table](#).

Figure 15-69. EMACDMABUSMOD Register

31	30	29	28	27	26	25	24
RIB	RESERVED			TXPR	MB	AAL	8xPBL
R/W-0x0	R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
USP	RPBL					FB	
R/W-0x0	R/W-0x1					R/W-0x0	
15	14	13	12	11	10	9	8
PR		PBL					
R/W-0x0		R/W-0x1					
7	6	5	4	3	2	1	0
ATDS	DSL				DA		SWR
R/W-0x0	R/W-0x0				R/W-0x0		R/W-0x1

Table 15-79. EMACDMABUSMOD Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RIB	R/W	0x0	Rebuild Burst. 0x0 = During a retry, split or loss of bus, the DMA rebuilds the pending beats of any burst transfer with a continuous, uninterrupted burst until the last word, which is a single burst. 0x1 = During a retry, split or loss of bus, the DMA rebuilds the pending beats of any burst transfer initiated with a defined fixed burst of 1, 4, 8, or 16.
30-28	RESERVED	R	0x0	
27	TXPR	R/W	0x0	Transmit Priority. 0x0 = The RX DMA has higher priority than the TX DMA during arbitration for the system bus. 0x1 = The TX DMA has higher priority than the RX DMA during arbitration for the system bus.
26	MB	R/W	0x0	Mixed Burst. 0x0 = Mixed burst is not enabled. 0x1 = If the FB bit is 0, the DMA starts all bursts of length more than 16 with a continuous undefined burst. For bursts less than 16, fixed and single bursts are used.
25	AAL	R/W	0x0	Address Aligned Beats. 0x0 = Address aligned transfers are not enabled. 0x1 = If the FB bit is set, the internal bus interface generates all bursts aligned to the start address least significant bits. If the FB bit is 0, the first burst is not aligned but subsequent bursts are aligned to the address.
24	8xPBL	R/W	0x0	8 x Programmable Burst Length (PBL) Mode. 0x0 = 8 x PBL mode is inactive. 0x1 = Bit field RPBL and bit field PBL are multiplied 8 times. Therefore, the DMA transfers the data in bursts of 8, 16, 32, 64, 128, and 256 words.

Table 15-79. EMACDMABUSMOD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
23	USP	R/W	0x0	Use Separate Programmable Burst Length (PBL). 0x0 = The PBL value in bits[13:8] is applicable for both the RX and TX DMA engines. 0x1 = RX DMA is uses the RPBL bit field as its defined programmable burst length and TX DMA uses the PBL bit field as its defined programmable burst length.
22-17	RPBL	R/W	0x1	RX DMA Programmable Burst Length (PBL). When the USP bit is 1, this field is used to indicate the maximum number of words to be transferred in one RX DMA transaction. This is the maximum value that is used in a single block read or write. The RX DMA always attempts to burst as specified in the RPBL bit each time it starts a burst transfer on the system bus. The application can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high.
16	FB	R/W	0x0	Fixed Burst. This bit defines if burst is used during burs transfer operations. 0x0 = The DMA bursts the entire length during burst transfers except for the last word, which is a single transfer. 0x1 = The DMA uses only single, or fixed bursts incremented by 4, 8, or 16 during normal bus transfers.
15-14	PR	R/W	0x0	Priority Ratio. These bits control the priority ratio in the weighted round-robin arbitration between the RX DMA and TX DMA. These bits are valid only when the DA bit in this register is clear. The priority ratio is RX:TX or TX:RX depending on whether the TXPR bit in this register is clear or set. 0x0 = The Priority Ratio is 1:1. 0x1 = The Priority Ratio is 2:1. 0x2 = The Priority Ratio is 3:1. 0x3 = The Priority Ratio is 4:1.
13-8	PBL	R/W	0x1	Programmable Burst Length. These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block read or write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable only for TX DMA transactions. If the number of beats to be transferred is more than 32, then perform the following steps: 1. Set the 8xPBL mode. 2. Set the PBL value. For example, if the maximum number of beats to be transferred is 64, then first set 8xPBL to 1 and then set PBL to 0x8.
7	ATDS	R/W	0x0	Alternate Descriptor Size. 0x0 = Descriptor size reverts back to four words 0x1 = Alternate descriptors, each eight words in length, are used.
6-2	DSL	R/W	0x0	Descriptor Skip Length. This bit specifies the number of words to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA in Ring mode.

Table 15-79. EMACDMABUSMOD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	DA	R/W	0x0	<p>DMA Arbitration Scheme. This bit specifies the arbitration scheme between the transmit and receive paths of the DMA channel.</p> <p>0x0 = Weighted round-robin with RX:TX or TX:RX. The priority between the paths is according to the priority specified in the PR bit field and priority weights specified in the TXPR bit.</p> <p>0x1 = Fixed priority. The transmit path has priority over receive path when the TXPR bit is set. Otherwise, receive path has priority over the transmit path.</p>
0	SWR	R/W	0x1	<p>DMA Software Reset. The software reset function is driven by this bit. The reset operation is completed only when all resets in all active clock domains are deasserted. It is essential that all the PHY input clocks are present for the software reset completion. The time of the software reset operation depends on the frequency of the slowest active clock.</p> <p>0x0 = Reset is completed. A value of 0 should be read before reprogramming any MAC registers after a reset</p> <p>0x1 = MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the MAC clock domains.</p>

15.6.55 EMAXXPOLLDD Register (Offset = 0xC04) [reset = 0x0]

Ethernet MAC Transmit Poll Demand (EMAXXPOLLDD)

The MAC Transmit Poll Demand (EMAXXPOLLDD) register enables the TX DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the TX DMA if it is in the suspend mode. The TX DMA can go into the SUSPEND mode because of an underflow error in a transmitted frame or the unavailability of descriptors owned by it. This Transmit Poll Demand command can be given at any time and the TX DMA resets this command when it again starts fetching the current descriptor from host memory.

EMAXXPOLLDD is shown in [Figure 15-70](#) and described in [Table 15-80](#).

Return to [Summary Table](#).

Figure 15-70. EMAXXPOLLDD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPD																															
W-0x0																															

Table 15-80. EMAXXPOLLDD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TPD	W	0x0	Transmit Poll Demand. When these bits are written with any value, the DMA reads the current descriptor pointed to by the MAC Current Host Transmit Descriptor (EMACHOSTXDESC) register. If that descriptor is not available (owned by the Host), the transmission returns to the SUSPEND state and the TU bit of the EMACDMARIS is asserted. If the descriptor is available, the transmission resumes.

15.6.56 EMACRXPOLLD Register (Offset = 0xC08) [reset = 0x0]

Ethernet MAC Receive Poll Demand (EMACRXPOLLD)

The Ethernet MAC Receive Poll Demand (EMACRXPOLLD) register enables the RX DMA to check for new descriptors. This command is used to wake up the RX DMA from the SUSPEND state. The RX DMA can go into the SUSPEND state only because of the unavailability of descriptors it owns.

EMACRXPOLLD is shown in [Figure 15-71](#) and described in [Table 15-81](#).

Return to [Summary Table](#).

Figure 15-71. EMACRXPOLLD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPD																															
W-0x0																															

Table 15-81. EMACRXPOLLD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RPD	W	0x0	Receive Poll Demand. When these bits are written with any value, the DMA reads the current descriptor pointed to by MAC Current Host Receive Descriptor (EMACHOSRXDESC) register. If that descriptor is not available (owned by the Host), the reception returns to the SUSPEND state and the RU bit of EMACDMARIS is not asserted. If the descriptor is available, the RX DMA returns to the active state.

15.6.57 EMACRXDLADDR Register (Offset = 0xC0C) [reset = 0x0]

Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR)

The Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR) register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be word-aligned. The DMA internally converts it to a 32-bit aligned address by making the two least significant bits zero. Writing to this register is permitted only when the DMA receive transaction has stopped (SR = 0 in the MAC DMA Operation Mode (EMACDMAOPMODE) register). When stopped, this register must be written with a new descriptor list address before the receive Start command is given. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA uses the already existing descriptor address.

EMACRXDLADDR is shown in [Figure 15-72](#) and described in [Table 15-82](#).

Return to [Summary Table](#).

Figure 15-72. EMACRXDLADDR Register

31	30	29	28	27	26	25	24
STRXLIST							
R/W-0x0							
23	22	21	20	19	18	17	16
STRXLIST							
R/W-0x0							
15	14	13	12	11	10	9	8
STRXLIST							
R/W-0x0							
7	6	5	4	3	2	1	0
STRXLIST						RESERVED	
R/W-0x0						R-0x0	

Table 15-82. EMACRXDLADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	STRXLIST	R/W	0x0	Start of Receive List. This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits[1:0] are ignored and internally taken as zero by the DMA. Therefore, bits[1:0] are read-only (R).
1-0	RESERVED	R	0x0	

15.6.58 EMAXTXDLADDR Register (Offset = 0xC10) [reset = 0x0]

Ethernet MAC Transmit Descriptor List Address (EMAXTXDLADDR)

The MAC Transmit Descriptor List Address (EMAXTXDLADDR) register points to the start of the transmit descriptor list. The descriptor lists reside in the host's physical memory space and must be word-aligned.

This register can only be written when the DMA transmit has stopped (ST = 0 in the MAC DMA Operation Mode (EMACDMAOPMODE) register). When the ST bit is set, the DMA takes the uses the newly programmed descriptor base address.

If this register is not changed when the ST bit is set to 0, then the DMA uses the already existing descriptor address.

EMAXTXDLADDR is shown in [Figure 15-73](#) and described in [Table 15-83](#).

Return to [Summary Table](#).

Figure 15-73. EMAXTXDLADDR Register

31	30	29	28	27	26	25	24
TXDLADDR							
R/W-0x0							
23	22	21	20	19	18	17	16
TXDLADDR							
R/W-0x0							
15	14	13	12	11	10	9	8
TXDLADDR							
R/W-0x0							
7	6	5	4	3	2	1	0
TXDLADDR						RESERVED	
R/W-0x0						R-0x0	

Table 15-83. EMAXTXDLADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	TXDLADDR	R/W	0x0	Start of Transmit List Base Address. This field contains the base address of the first descriptor in the transmit descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (R).
1-0	RESERVED	R	0x0	

15.6.59 EMACDMARIS Register (Offset = 0xC14) [reset = 0x0]

Ethernet MAC DMA Interrupt Status (EMACDMARIS)

The MAC DMA Interrupt Status (EMACDMARIS) register contains all status bits that the DMA reports to the host. The software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing a 1 to any bit in the lower half-word of this register clears the bit and writing a 0 has no effect. Status bits [16:0] can be masked by enabling the appropriate mask in the MAC DMA Interrupt Mask Register (EMACDMAIM) register.

EMACDMARIS is shown in [Figure 15-74](#) and described in [Table 15-84](#).

Return to [Summary Table](#).

Figure 15-74. EMACDMARIS Register

31	30	29	28	27	26	25	24
RESERVED	TT	PMT	MMC	RESERVED	AE		
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	
23	22	21	20	19	18	17	16
AE	TS			RS			NIS
R-0x0	R-0x0			R-0x0			R/W1C-0x0
15	14	13	12	11	10	9	8
AIS	ERI	FBI	RESERVED	ETI	RWT	RPS	
R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0
7	6	5	4	3	2	1	0
RU	RI	UNF	OVF	TJT	TU	TPS	TI
R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 15-84. EMACDMARIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0x0	
29	TT	R	0x0	Timestamp Trigger Interrupt Status. Software can read the Ethernet MAC Timestamp Status (EMACTIMSTAT) register for the exact cause of interrupt and clear its source to reset this bit to 0. When this bit is 1, the interrupt signal from the MAC subsystem is high. 0x0 = No Timestamp interrupt has occurred. 0x1 = An interrupt event in the Timestamp module has occurred.
28	PMT	R	0x0	MAC PMT Interrupt Status. Software can read the PMT Control and Status (EMACPMTCTLSTAT) register for the exact cause of the interrupt and clear its source to reset this bit to 0. When this bit is 1, the interrupt signal from the MAC subsystem is high. 0x0 = No PMT interrupt has occurred. 0x1 = An interrupt event in the PMT module has occurred.
27	MMC	R	0x0	MAC MMC Interrupt. This bit reflects an interrupt event in the MMC module. Software must read the corresponding EMACMMCTXRIS/EMACMMCRXRIS register to determine the cause of the interrupt and then clear its source to reset this bit to 0. 0x0 = No MMC interrupt has occurred. 0x1 = An interrupt in the MMC module has occurred.
26	RESERVED	R	0x0	

Table 15-84. EMACDMARIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
25-23	AE	R	0x0	<p>Access Error. This field indicates the type of error that caused a bus error, for example, error response on the internal bus interface. This field is valid only when bit[13] (FBI) is set. This field does not generate an interrupt.</p> <p>0x0 = Error during RX DMA Write Data Transfer 0x1 = Reserved 0x2 = Reserved 0x3 = Error during TX DMA Read Data Transfer 0x4 = Error during RX DMA Descriptor Write Access 0x5 = Error during TX DMA Descriptor Write Access 0x6 = Error during RX DMA Descriptor Read Access 0x7 = Error during TX DMA Descriptor Read Access</p>
22-20	TS	R	0x0	<p>Transmit Process State. This field indicates the Transmit DMA state. This field does not generate an interrupt.</p> <p>0x0 = Stopped; Reset or Stop transmit command processed 0x1 = Running; Fetching transmit transfer descriptor 0x2 = Running; Waiting for status 0x3 = Running; Reading data from host memory buffer and queuing it to transmit buffer (TX FIFO) 0x4 = Writing Timestamp 0x5 = Reserved 0x6 = Suspended; Transmit descriptor unavailable or transmit buffer underflow 0x7 = Running; Closing transmit descriptor</p>
19-17	RS	R	0x0	<p>Received Process State. This field indicates the Receive DMA state. This field does not generate an interrupt.</p> <p>0x0 = Stopped: Reset or stop receive command issued 0x1 = Running: Fetching receive transfer descriptor 0x2 = Reserved 0x3 = Running: Waiting for receive packet 0x4 = Suspended: Receive descriptor unavailable 0x5 = Running: Closing receive descriptor 0x6 = Writing Timestamp 0x7 = Running: Transferring the receive packet data from receive buffer to host memory</p>
16	NIS	R/W1C	0x0	<p>Normal Interrupt Summary.</p> <p>Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in EMACDMAIM register:</p> <ul style="list-style-type: none"> • EMACDMARIS register, bit [0]: Transmit Interrupt • EMACDMARIS register, bit[2]: Transmit Buffer Unavailable • EMACDMARIS register, bit[6]: Receive Interrupt • EMACDMARIS register, bit[14]: Early Receive Interrupt <p>Only unmasked bits (interrupts for which interrupt enable is set in the EMACDMAIM register) affect the Normal Interrupt Summary bit.</p> <p>This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes NIS to be set, is cleared.</p>

Table 15-84. EMACDMARIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15	AIS	R/W1C	0x0	<p>Abnormal Interrupt Summary.</p> <p>Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the EMACDMAIM register:</p> <ul style="list-style-type: none"> • EMACDMARIS register, bit[1]: Transmit Process Stopped • EMACDMARIS register, bit[3]: Transmit Jabber Timeout • EMACDMARIS register, bit[4]: Receive FIFO Overflow • EMACDMARIS register, bit[5]: Transmit Underflow • EMACDMARIS register, bit[7]: Receive Buffer Unavailable • EMACDMARIS register, bit[8]: Receive Process Stopped • EMACDMARIS register, bit[9]: Receive Watchdog Timeout • EMACDMARIS register, bit[10]: Early Transmit Interrupt • EMACDMARIS register, bit[13]: Fatal Bus Error <p>Only unmasked bits affect the Abnormal Interrupt Summary bit. This bit must be cleared each time a corresponding bit, which causes AIS to be set, is cleared.</p>
14	ERI	R/W1C	0x0	<p>Early Receive Interrupt.</p> <p>0x0 = No early receive event has occurred.</p> <p>0x1 = The DMA has filled the first data buffer of the packet. This bit is cleared when software writes a 1 to this bit or if bit[6] (RI) of this register is set.</p>
13	FBI	R/W1C	0x0	<p>Fatal Bus Error Interrupt.</p> <p>0x0 = No bus error has occurred.</p> <p>0x1 = A bus error has occurred, as described in the Error Bit field (EB [25:23]). When this bit is set, the corresponding DMA engine disables all of its bus accesses. This bit is cleared by writing a 1 to it.</p>
12-11	RESERVED	R	0x0	
10	ETI	R/W1C	0x0	<p>Early Transmit Interrupt.</p> <p>0x0 = No early transmit has occurred.</p> <p>0x1 = A frame to be transmitted has been fully transferred to the TX/RX Controller Transmit FIFO. This bit is cleared by writing a 1 to it.</p>
9	RWT	R/W1C	0x0	<p>Receive Watchdog Time-out.</p> <p>0x0 = No watchdog time-out event has occurred.</p> <p>0x1 = Indicates a frame with length greater than 2,048 bytes is received (10, 240 when Jumbo Frame mode is enabled). This bit is cleared by writing a 1 to it.</p>
8	RPS	R/W1C	0x0	<p>Receive Process Stopped.</p> <p>0x0 = No receive process stopped event has occurred.</p> <p>0x1 = Indicates the receive process has entered the stopped state. This bit is cleared by writing a 1 to it.</p>

Table 15-84. EMACDMARIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	RU	R/W1C	0x0	Receive Buffer Unavailable. To resume processing receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the receive process resumes when the next recognized incoming frame is received. This bit is set only when the previous receive descriptor is owned by the DMA. 0x0 = No receive buffer unavailable event has occurred. 0x1 = Indicates the host owns the next descriptor in the receive list and the DMA cannot acquire it. The receive process is suspended. This bit is cleared by writing a 1 to it.
6	RI	R/W1C	0x0	Receive Interrupt. 0x0 = No frame reception complete event has occurred. 0x1 = A frame reception is complete. When reception is complete, bit[31] of RDES1 (disable interrupt on completion) is reset in the last descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state. This bit is cleared by writing a 1 to it.
5	UNF	R/W1C	0x0	Transmit Underflow. 0x0 = No transmit underflow event has occurred. 0x1 = Indicates the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set. This bit is cleared by writing a 1 to it.
4	OVF	R/W1C	0x0	Receive Overflow. 0x0 = No receive overflow event has occurred. 0x1 = The receive buffer had an overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11]. This bit is cleared by writing a 1 to it.
3	TJT	R/W1C	0x0	Transmit Jabber Time-out. 0x0 = No transmit jabber time-out event has occurred. 0x1 = The Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when Jumbo frame is enabled). When the Jabber Time-out occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Time-out TDES0[14] flag to assert. This bit is cleared by writing a 1 to it.
2	TU	R/W1C	0x0	Transmit Buffer Unavailable. To resume processing Transmit descriptors, the host should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command. 0x0 = No transmit buffer unavailable event has occurred. 0x1 = Indicates the host owns the next descriptor in the transmit list and the DMA cannot acquire it. Transmission is suspended. The transmit process state bits (TS [22:20]) explain the transmit process state transitions. This bit is cleared by writing a 1 to it.
1	TPS	R/W1C	0x0	Transmit Process Stopped. 0x0 = Interrupt is inactive. 0x1 = Indicates transmission is stopped.
0	TI	R/W1C	0x0	Transmit Interrupt. This bit indicates that frame transmission is complete. When transmission is complete, the Bit 31 (Interrupt on Completion) of TDES1 is reset in the first descriptor, and the specific frame status information is updated in the descriptor. 0x0 = Frame transmission completion event has not occurred. 0x1 = Frame transmission is complete. This bit is cleared by writing a 1 to it.

15.6.60 EMACDMAOPMODE Register (Offset = 0xC18) [reset = 0x0]

Ethernet MAC DMA Operation Mode (EMACDMAOPMODE)

The MAC DMA Operation Mode (EMACDMAOPMODE) register establishes the Transmit and Receive operating modes and commands. This register should be the last register to be written as part of the DMA initialization.

EMACDMAOPMODE is shown in [Figure 15-75](#) and described in [Table 15-85](#).

Return to [Summary Table](#).

Figure 15-75. EMACDMAOPMODE Register

31	30	29	28	27	26	25	24
RESERVED					DT	RSF	DFF
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
RESERVED		TSF	FTF	RESERVED		TTC	
R-0x0		R/W-0x0	R/W-0x0	R-0x0		R/W-0x0	
15	14	13	12	11	10	9	8
TTC		ST	RESERVED				
R/W-0x0		R/W-0x0	R-0x0				
7	6	5	4	3	2	1	0
FEF	FUF	DGF	RTC		OSF	SR	RESERVED
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0	R-0x0

Table 15-85. EMACDMAOPMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0x0	
26	DT	R/W	0x0	Disable Dropping of TCP/IP Checksum Error Frames. 0x0 = All error frames are dropped if the FEF bit is reset. 0x1 = The MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload.
25	RSF	R/W	0x0	Receive Store and Forward. 0x0 = The RX FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits. 0x1 = The TX/RX Controller reads a frame from the RX FIFO only after the complete frame has been written to it, ignoring the RTC bits.
24	DFF	R/W	0x0	Disable Flushing of Received Frames. 0x0 = RX DMA flushes frames based on receive descriptors or buffers. 0x1 = The RX DMA does not flush any frames because of the unavailability of receive descriptors or buffers.
23-22	RESERVED	R	0x0	
21	TSF	R/W	0x0	Transmit Store and Forward. 0x0 = Transmission starts according to TTC bit field. 0x1 = Transmission starts when a full frame resides in the TX/RX Controller Transmit FIFO. Additionally, the TTC values specified in TTC bits[16:14] are ignored. This bit should be changed only when the transmission is stopped.

Table 15-85. EMACDMAOPMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
20	FTF	R/W	0x0	<p>Flush Transmit FIFO. This bit is cleared internally when the flushing operation is completed. This register should not be written to until the FTF bit is cleared. The data which has already been accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission. The flush operation is complete only when the TX FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host.</p> <p>0x0 = This bit indicated normal operation or that the flushing operation has completed.</p> <p>0x1 = The transmit FIFO controller logic is reset to its default values and thus all data in the TX FIFO is lost or flushed. This bit is cleared internally when the flushing operation is complete.</p>
19-17	RESERVED	R	0x0	
16-14	TTC	R/W	0x0	<p>Transmit Threshold Control. These bits control the threshold level of the TX/RX Controller Transmit FIFO. Transmission starts when the frame size within the TX/RX Controller Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset.</p> <p>0x0 = 64 bytes 0x1 = 128 bytes 0x2 = 192 bytes 0x3 = 256 bytes 0x4 = 40 bytes 0x5 = 32 bytes 0x6 = 24 bytes 0x7 = 16 bytes</p>
13	ST	R/W	0x0	<p>Start or Stop Transmission Command.</p> <p>When this bit is set, transmission is placed in the running state. The DMA attempts to acquire the descriptor from the Transmit Descriptor List. Descriptor acquisition is attempted from the current position in the list, which is the Transmit List Base Address set by Transmit Descriptor List Address (EMACTXDLADDR) register, or from the position retained when transmission was stopped previously.</p> <p>If the DMA does not own the current descriptor, transmission enters the suspended state and bit[2] (Transmit Buffer Unavailable, TU) of the MAC DMA Raw Interrupt Status Register (EMACDMARIS) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting EMACTXDLADDR, then the DMA behavior is unpredictable.</p> <p>When this bit is cleared, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program EMACTXDLADDR with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.</p> <p>0x0 = Transmission process is placed in the stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted.</p> <p>0x1 = Transmission is placed in the running state, and the DMA checks the transmit list at the current position for a frame to be transmitted.</p>
12-8	RESERVED	R	0x0	

Table 15-85. EMACDMAOPMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	FEF	R/W	0x0	<p>Forward Error Frames. When this bit is reset, the RX FIFO drops frames with error status (CRC error, collision error, MII_ER, giant frame, watchdog time-out, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the RX controller side (in Threshold mode), then the frame is not dropped. When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If Bit 25, Receive Store and Forward (RSF), is set and the RX FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the RSF is reset and the RX FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA.</p> <p>0x0 = The Receive FIFO drops frames with error status 0x1 = All frames except runt error frames are forwarded to the DMA.</p>
6	FUF	R/W	0x0	<p>Forward Undersized Good Frames.</p> <p>0x0 = The Receive FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold (RTC) bit field (for example, RTC = 0x1). 0x1 = The Receive FIFO forwards undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC.</p>
5	DGF	R/W	0x0	<p>Drop Giant Frame Enable.</p> <p>0x0 = The MAC does not drop the giant frames in the RX FIFO. 0x1 = The MAC drops received frames larger than the computed giant frame limit in the RX FIFO.</p>
4-3	RTC	R/W	0x0	<p>Receive Threshold Control. These two bits control the threshold level of the RX FIFO. Transfer (request) to DMA starts when the frame size within the RX FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically. These bits are valid only when the RSF bit (bit 25) of the EMACDMAOPMODE is zero, and are ignored when the RSF bit is set to 1.</p> <p>0x0 = 64 bytes 0x1 = 32 bytes 0x2 = 96 bytes 0x3 = 128 bytes</p>
2	OSF	R/W	0x0	<p>Operate on Second Frame.</p> <p>0x0 = DMA processes frames normally. 0x1 = DMA processes second frame of the Transmit data even before the status for the first frame is obtained.</p>
1	SR	R/W	0x0	<p>Start or Stop Receive.</p> <p>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by the Receive Descriptor List Address Register (EMACRXDLADDR) or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and Bit 7 (Receive Buffer Unavailable, RU) of the MAC DMA Interrupt Status Register (EMACDMARIS) is set. The Start Receive command is effective only when the reception has stopped. If the command is issued before setting EMACRXDLADDR, the DMA behavior is unpredictable.</p> <p>When this bit is cleared, the receive DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.</p> <p>0x0 = The receive DMA operation is stopped after the transfer of the current frame 0x1 = The Receive process is placed in the Running state.</p>

Table 15-85. EMACDMAOPMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RESERVED	R	0x0	

15.6.61 EMACDMAIM Register (Offset = 0xC1C) [reset = 0x0]

Ethernet MAC DMA Interrupt Mask Register (EMACDMAIM)

The Interrupt Enable register enables the interrupts reported by the MAC DMA Interrupt Status Register (EMACDMARIS). Setting a bit to 0x1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

EMACDMAIM is shown in [Figure 15-76](#) and described in [Table 15-86](#).

Return to [Summary Table](#).

Figure 15-76. EMACDMAIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							NIE
R-0x0							R/W-0x0
15	14	13	12	11	10	9	8
AIE	ERE	FBE	RESERVED		ETE	RWE	RSE
R/W-0x0	R/W-0x0	R/W-0x0	R-0x0		R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RUE	RIE	UNE	OVE	TJE	TUE	TSE	TIE
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 15-86. EMACDMAIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	NIE	R/W	0x0	Normal Interrupt Summary Enable. This bit enables/masks the ERI, RI, TU, and TI bits in MAC DMA Interrupt Status Register (EMACDMARIS) 0x0 = Normal interrupt summary is masked. 0x1 = Normal interrupt summary is enabled.
15	AIE	R/W	0x0	Abnormal Interrupt Summary Enable. This bit enables/masks the TPS, TJT, OVF, UNF, RU, RPS, RWT, ETI and FBI bits in MAC DMA Interrupt Status Register (EMACDMARIS) 0x0 = Abnormal interrupt summary is disabled. 0x1 = Abnormal interrupt summary is enabled.
14	ERE	R/W	0x0	Early Receive Interrupt Enable. 0x0 = Early receive interrupt is disabled. 0x1 = Early receive interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 16) must also be set to 0x1.
13	FBE	R/W	0x0	Fatal Bus Error Enable. 0x0 = Fatal Bus Error Enable Interrupt is disabled. 0x1 = Fatal Bus Error Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
12-11	RESERVED	R	0x0	
10	ETE	R/W	0x0	Early Transmit Interrupt Enable. 0x0 = Early Transmit Interrupt is disabled. 0x1 = Early Transmit Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
9	RWE	R/W	0x0	Receive Watchdog Time-out Enable. 0x0 = The Receive Watchdog Time-out Interrupt is disabled. 0x1 = The Receive Watchdog Time-out Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.

Table 15-86. EMACDMAIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	RSE	R/W	0x0	Receive Stopped Enable. 0x0 = Receive Stopped Interrupt is disabled. 0x1 = Receive Stopped Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
7	RUE	R/W	0x0	Receive Buffer Unavailable Enable. 0x0 = The Receive Buffer Unavailable Interrupt is disabled. 0x1 = The Receive Buffer Unavailable Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
6	RIE	R/W	0x0	Receive Interrupt Enable. 0x0 = The Receive Interrupt is disabled. 0x1 = The Receive Interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 15) must also be set to 0x1.
5	UNE	R/W	0x0	Underflow Interrupt Enable. 0x0 = Transmit Underflow Interrupt is disabled. 0x1 = The Transmit Underflow Interrupt is enabled Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
4	OVE	R/W	0x0	Overflow Interrupt Enable. 0x0 = The Overflow Interrupt is disabled. 0x1 = The Receive Overflow Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
3	TJE	R/W	0x0	Transmit Jabber Time-out Enable. 0x0 = Transmit Jabber Time-out Interrupt is disabled. 0x1 = Transmit Jabber Time-out Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
2	TUE	R/W	0x0	Transmit Buffer Unavailable Enable. 0x0 = Transmit Buffer Unavailable Interrupt is disabled. 0x1 = Transmit Buffer Unavailable Interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 15) must also be set to 0x1.
1	TSE	R/W	0x0	Transmit Stopped Enable. 0x0 = Transmission Stopped Interrupt is disabled. 0x1 = Transmission Stopped Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1.
0	TIE	R/W	0x0	Transmit Interrupt Enable. 0x0 = Transmit Interrupt is disabled. 0x1 = Transmit Interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 15) must also be set to 0x1.

15.6.62 EMACMFOBC Register (Offset = 0xC20) [reset = 0x0]

Ethernet MAC Missed Frame and Buffer Overflow Counter (EMACMFOBC)

The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counters. The counter is used for diagnostic purposes. The MISFRMCNT field indicates missed frames because of the host buffer being unavailable. The OVFFRMCNT field indicates missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

EMACMFOBC is shown in [Figure 15-77](#) and described in [Table 15-87](#).

Return to [Summary Table](#).

Figure 15-77. EMACMFOBC Register

31	30	29	28	27	26	25	24
RESERVED			OVFCNTOVF	OVFFRMCNT			
R-0x0			R-0x0	R-0x0			
23	22	21	20	19	18	17	16
OVFFRMCNT							MISCNTOVF
R-0x0							R-0x0
15	14	13	12	11	10	9	8
MISFRMCNT							
R-0x0							
7	6	5	4	3	2	1	0
MISFRMCNT							
R-0x0							

Table 15-87. EMACMFOBC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0x0	
28	OVFCNTOVF	R	0x0	Overflow Bit for FIFO Overflow Counter. This bit is set every time the overflow frame counter (bits [27:17]) overflows; that is, the RX FIFO overflows with the overflow frame counter at maximum value. In such a scenario, the overflow frame counter is reset to all zeros and this bit indicates the rollover happened.
27-17	OVFFRMCNT	R	0x0	Overflow Frame Counter. This field indicates the number of frames missed by the application. This counter is incremented each time the TX/RX Controller indicates overflow. This counter is cleared when the TX/RX Controller accepts data.
16	MISCNTOVF	R	0x0	Overflow bit for Missed Frame Counter. This bit is set every time the Missed Frame Counter (bits [15:0]) overflows; which means the DMA discards an incoming frame because of the Host Receive Buffer being unavailable with the missed frame counter at maximum value. In such a scenario, the Missed Frame Counter is reset to all zeros and this bit indicates that the rollover happened.
15-0	MISFRMCNT	R	0x0	Missed Frame Counter. This field indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. This counter is cleared when the DMA accepts frames.

15.6.63 EMACRXINTWDT Register (Offset = 0xC24) [reset = 0x0]

Ethernet MAC Receive Interrupt Watchdog Timer (EMACRXINTWDT)

This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt, RI (Bit 6), of the EMACDMARIS register at EMAC offset 0xC14.

EMACRXINTWDT is shown in [Figure 15-78](#) and described in [Table 15-88](#).

Return to [Summary Table](#).

Figure 15-78. EMACRXINTWDT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RIWT							
R-0x0																								R/W-0x0							

Table 15-88. EMACRXINTWDT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	RIWT	R/W	0x0	<p>Receive Interrupt Watchdog Timer Count.</p> <p>This field indicates the period in which the receive counter expires. The value in this field is programmed by 256 to calculate the number of system clock periods the timer must count.</p> <p>Watchdog Timer Period = (RIWT * 256) system clocks.</p> <p>When the watchdog timer runs out, the RI bit is set and the timer is stopped.</p> <p>If the RDES[31] bit is clear, the watchdog timer is reset.</p>

15.6.64 EMACHOSTXDESC Register (Offset = 0xC48) [reset = 0x0]

Ethernet MAC Current Host Transmit Descriptor (EMACHOSTXDESC)

The MAC Current Host Transmit Descriptor (EMACHOSTXDESC) register points to the start address of the current Transmit Descriptor read by the DMA.

EMACHOSTXDESC is shown in [Figure 15-79](#) and described in [Table 15-89](#).

Return to [Summary Table](#).

Figure 15-79. EMACHOSTXDESC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTXDESC																															
R-0x0																															

Table 15-89. EMACHOSTXDESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CURTXDESC	R	0x0	Host Transmit Descriptor Address Pointer. This register contains the start address of the current transmit descriptor read by the DMA. This register is cleared on reset. The pointer is updated by the DMA during operation.

15.6.65 EMACHOSRXDESC Register (Offset = 0xC4C) [reset = 0x0]

Ethernet MAC Current Host Receive Descriptor (EMACHOSRXDESC)

The MAC Current Host Receive Descriptor (EMACHOSRXDESC) register points to the start address of the current Receive Descriptor read by the DMA.

EMACHOSRXDESC is shown in [Figure 15-80](#) and described in [Table 15-90](#).

Return to [Summary Table](#).

Figure 15-80. EMACHOSRXDESC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRXDESC																															
R-0x0																															

Table 15-90. EMACHOSRXDESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CURRXDESC	R	0x0	Host Receive Descriptor Address Pointer. This register contains the start address of the current receive descriptor read by the DMA. This register is cleared on reset. The pointer is updated by the DMA during operation.

15.6.66 EMACHOSTXBA Register (Offset = 0xC50) [reset = 0x0]

Ethernet MAC Current Host Transmit Buffer Address (EMACHOSTXBA)

The MAC Current Host Transmit Buffer Address (EMACHOSTXBA) register points to the current Transmit Buffer Address being read by the DMA.

EMACHOSTXBA is shown in [Figure 15-81](#) and described in [Table 15-91](#).

Return to [Summary Table](#).

Figure 15-81. EMACHOSTXBA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTXBUFA																															
R-0x0																															

Table 15-91. EMACHOSTXBA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CURTXBUFA	R	0x0	Host Transmit Buffer Address Pointer. This register contains the current transmit buffer address being read by the DMA. This register is cleared on reset. The pointer is updated by the DMA during operation.

15.6.67 EMACHOSRXBA Register (Offset = 0xC54) [reset = 0x0]

Ethernet MAC Current Host Receive Buffer Address (EMACHOSRXBA)

The MAC Current Host Receive Buffer Address (EMACHOSRXBA) register points to the current receive buffer address being read by the DMA.

EMACHOSRXBA is shown in [Figure 15-82](#) and described in [Table 15-92](#).

Return to [Summary Table](#).

Figure 15-82. EMACHOSRXBA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRXBUFA																															
R-0x0																															

Table 15-92. EMACHOSRXBA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CURRXBUFA	R	0x0	Host Receive Buffer Address Pointer. This register contains the current receive buffer address being read by the DMA. This register is cleared on reset. The pointer is updated by the DMA during operation.

15.6.68 EMACPP Register (Offset = 0xFC0) [reset = 0x103]

Ethernet MAC Peripheral Property Register (EMACPP)

This register defines the Ethernet MAC and PHY type used.

EMACPP is shown in [Figure 15-83](#) and described in [Table 15-93](#).

Return to [Summary Table](#).

Figure 15-83. EMACPP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					MACTYPE			RESERVED					PHYTYPE		
R-0x0					R-0x1			R-0x0					R-0x3		

Table 15-93. EMACPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0x0	
10-8	MACTYPE	R	0x1	Ethernet MAC Type. 0x0 = Reserved 0x1 = MSP432E4 class MAC. 0x2 = Reserved 0x3 = Reserved 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved
7-3	RESERVED	R	0x0	
2-0	PHYTYPE	R	0x3	Ethernet PHY Type. This field specifies the type of PHY provided. 0x0 = Reserved 0x1 = Reserved 0x2 = Reserved 0x3 = MSP432E4 class PHY 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved

15.6.69 EMACPC Register (Offset = 0xFC4) [reset = 0x0080040E]

Ethernet MAC Peripheral Configuration Register (EMACPC)

The Ethernet MAC Peripheral Configuration Register (EMACPC) register configures the MAC and PHY reset and interface parameters.

EMACPC is shown in [Figure 15-84](#) and described in [Table 15-94](#).

Return to [Summary Table](#).

Figure 15-84. EMACPC Register

31	30	29	28	27	26	25	24
PHYEXT	PINTFS			RESERVED		DIGRESTART	NIBDETDIS
R/W-0x0	R/W-0x0			R-0x0		R/W-0x0	R-0x0
23	22	21	20	19	18	17	16
RXERIDLE	ISOMIILL	LRR	TDRRUN	FASTLDMODE			
R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0			
15	14	13	12	11	10	9	8
FASTLDMODE	POLSWAP	MDISWAP	RBSTMDIX	FASTMDIX	MDIXEN	FASTRXDV	FASTLUPD
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x1	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
EXTFD	FASTANEN	FASTANSEL		ANEN	ANMODE		PHYHOLD
R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x1	R/W-0x3		R/W-0x0

Table 15-94. EMACPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PHYEXT	R/W	0x0	PHY Select. This bit is used to select whether the internal or an external PHY is used. 0x0 = Internal PHY 0x1 = External PHY
30-28	PINTFS	R/W	0x0	Ethernet Interface Select. This field selects the PHY interface used by the MAC. This input is sampled during reset and an update to this register field must result in the MAC undergoing a reset event. This field has the following encoded values: 0x0 = MII (default) Used for internal PHY or external PHY connected via MII. 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved 0x4 = RMII: Used for external PHY connected via RMII. 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved
27-26	RESERVED	R	0x0	
25	DIGRESTART	R/W	0x0	PHY Soft Restart. This bit allows the user to restart the PHY. Asserting this bit causes the PHY logic and internal register to reset to initial conditions. This bit does not affect the configuration bits provided by the EMACPC register, which are stored in the PHY following a chip reset. To initiate the soft reset to the PHY, this bit must be written to a 1 and written again to a 0.
24	NIBDETDIS	R	0x0	Odd Nibble TXER Detection Disable. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the ODDNDETDIS bit of the Ethernet PHY Configuration 2 (EPHYCFG2) register, PHY offset 0x00A.
23	RXERIDLE	R/W	0x1	RXER Detection During Idle. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the RXERRIDLE bit of the Ethernet PHY Configuration 2 (EPHYCFG2) register, PHY offset 0x00A.

Table 15-94. EMACPC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
22	ISOMIILL	R/W	0x0	Isolate MII in Link Loss. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the ISOMIILL bit of the Ethernet PHY Configuration 2 (EPHYCFG2) register, PHY offset 0x00A.
21	LRR	R/W	0x0	Link Loss Recovery. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the LLR bit of the Ethernet PHY Configuration 1 (EPHYCFG1) register, PHY offset 0x009.
20	TDRRUN	R/W	0x0	TDR Auto Run. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the TDRAR bit of the Ethernet PHY Configuration 1 (EPHYCFG1) register, PHY offset 0x009.
19-15	FASTLDMODE	R/W	0x0	Fast Link Down Mode. These bits are sampled on the deassertion of the PHY reset signal and are used as the default for the FLDWNM bit field of the Ethernet PHY Configuration 3 (EPHYCFG3) register, PHY offset 0x00B.
14	POLSWAP	R/W	0x0	Polarity Swap. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the POLSWAP bit of the Ethernet PHY Configuration 3 (EPHYCFG3) register, PHY offset 0x00B.
13	MDISWAP	R/W	0x0	MDI Swap. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the MDIMDIXS bit of the Ethernet PHY Configuration 3 (EPHYCFG3) register, PHY offset 0x00B.
12	RBSTMDIX	R/W	0x0	Robust Auto MDI-X. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the RAMDIX bit of the Ethernet PHY Configuration 1 (EPHYCFG1) register, PHY offset 0x009.
11	FASTMDIX	R	0x0	Fast Auto MDI-X. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the FAMDIX bit of the Ethernet PHY Configuration 1 (EPHYCFG1) register, PHY offset 0x009.
10	MDIXEN	R/W	0x1	MDIX Enable. This bit is sampled on the deassertion of the PHY reset signal and is used to determine whether automatic MDI/MDIX crossover is enabled. 0x0 = Disable automatic crossover. 0x1 = Enable automatic crossover.
9	FASTRXDV	R/W	0x0	Fast RXDV Detection. This bit is sampled on the deassertion of the PHY reset signal and is used to select whether fast RXDV detection is enabled in the PHY. 0x0 = Disable fast RXDV detection. 0x1 = Enable fast RXDV detection.
8	FASTLUPD	R/W	0x0	FAST Link-Up in Parallel Detect. This bit is sampled on the deassertion of the PHY reset signal and is used as the default value for the FLUPPD bit of the Ethernet PHY Configuration 2 (EPHYCFG2) register, PHY offset 0x00A.
7	EXTFD	R/W	0x0	Extended Full Duplex Ability. This bit is sampled on the deassertion of the PHY reset signal and is used as the default value for the EXTFD bit of the Ethernet PHY Configuration 2 (EPHYCFG2) register, PHY offset 0x00A.
6	FASTANEN	R/W	0x0	Fast Auto Negotiation Enable. This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the FASTANEN bit of the Ethernet PHY Configuration 1 (EPHYCFG1) register, PHY offset 0x009.
5-4	FASTANSEL	R/W	0x0	Fast Auto Negotiation Select. These bits are sampled on the deassertion of the PHY reset signal and are used as the defaults for the FANSEL bit field of the Ethernet PHY Configuration 1 (EPHYCFG1) register, PHY offset 0x009.

Table 15-94. EMACPC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	ANEN	R/W	0x1	Auto-Negotiation Enable. This bit is sampled on the deassertion of the PHY reset signal and is to select whether auto-negotiation is enabled. 0x0 = Auto-negotiation disabled. 0x1 = Auto-negotiation enabled.
2-1	ANMODE	R/W	0x3	Auto Negotiation Mode. These bits are sampled on the deassertion of the PHY reset signal and are used to determine the auto-negotiation mode of the PHY. 0x0 = When ANEN = 0x0, the mode is 10Base-T, Half-Duplex. When ANEN = 0x1, the mode is 10Base-T, Half/Full-Duplex. 0x1 = When ANEN = 0x0, the mode is 10Base-T, Full-Duplex. When ANEN = 0x1, the mode is 100Base-TX, Half/Full-Duplex. 0x2 = When ANEN = 0x0, the mode is 100Base-TX, Half-Duplex. When ANEN = 0x1, the mode is 10Base-T, Half-Duplex. 0x3 = When ANEN = 0x0, the mode is 100Base-TX, Full-Duplex. When ANEN = 0x1, the mode is 10Base-T, Half/Full-Duplex.
0	PHYHOLD	R/W	0x0	Ethernet PHY Hold. This bit is sampled on the deassertion of the PHY reset signal and is used to keep the PHY from transmitting energy on the line. 0x0 = PHY transmits energy on the line. 0x1 = PHY is held off from transmitting energy on the line.

15.6.70 EMACCC Register (Offset = 0xFC8) [reset = 0x0]

Ethernet MAC Clock Configuration Register (EMACCC)

The following register is used to configure the clocks of the Ethernet Controller.

EMACCC is shown in [Figure 15-85](#) and described in [Table 15-95](#).

Return to [Summary Table](#).

Figure 15-85. EMACCC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED					PTPCEN	POL	CLKEN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							
R-0x0							

Table 15-95. EMACCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0x0	
18	PTPCEN	R/W	0x0	PTP Clock Reference Enable. The PTP clock reference is MOSC. This bit enables the MOSC to drive the PTP clock reference of the Ethernet MAC. 0x0 = PTP clock reference is disabled. 0x1 = PTP clock reference is enabled.
17	POL	R/W	0x0	LED Polarity Control. This bit controls the polarity of the LED outputs coming from the Ethernet PHY. 0x0 = LEDs are active high. 0x1 = LEDs are active low.
16	CLKEN	R/W	0x0	EN0RREF_CLK Signal Enable. When using the RMII interface, this bit must be set to a 1. 0x0 = EN0RREF_CLK signal is disabled 0x1 = EN0RREF_CLK signal is enabled
15-0	RESERVED	R	0x0	

15.6.71 EPHYRIS Register (Offset = 0xFD0) [reset = 0x0]

Ethernet PHY Raw Interrupt Status (EPHYRIS)

The Ethernet PHY Raw Interrupt Status (EPHYRIS) register is used to mask the interrupt from the Ethernet PHY, which is either from the internal integrated PHY or an external PHY.

EPHYRIS is shown in [Figure 15-86](#) and described in [Table 15-96](#).

[Return to Summary Table.](#)

Figure 15-86. EPHYRIS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															INT
R-0x0															R-0x0

Table 15-96. EPHYRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	INT	R	0x0	Ethernet PHY Raw Interrupt Status. 0x0 = No interrupt has triggered. 0x1 = The Ethernet PHY has signaled an interrupt using the EN0INTR input.

15.6.72 EPHYIM Register (Offset = 0xFD4) [reset = 0x0]

Ethernet PHY Interrupt Mask (EPHYIM)

The Ethernet PHY Interrupt Mask (EPHYIM) register is used to mask the interrupt from the Ethernet PHY, which is either from the internal integrated PHY or an external PHY.

EPHYIM is shown in [Figure 15-87](#) and described in [Table 15-97](#).

Return to [Summary Table](#).

Figure 15-87. EPHYIM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															INT
R-0x0															R/W-0x0

Table 15-97. EPHYIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	INT	R/W	0x0	<p>Ethernet PHY Interrupt Mask.</p> <p>0x0 = An Ethernet PHY interrupt is suppressed and not sent to the interrupt controller.</p> <p>0x1 = An Ethernet PHY interrupt is sent to the interrupt controller when the INT bit is set in the EPHYRIS register. This interrupt could be generated from either the integrated PHY or an external PHY through the EN0INTR signal depending on the configuration chosen.</p>

15.6.73 EPHYMISC Register (Offset = 0xFD8) [reset = 0x0]

Ethernet PHY Masked Interrupt Status and Clear (EPHYMISC)

The Ethernet Masked Interrupt Status and Clear (EPHYMISC) register displays the masked interrupt status of the Ethernet PHY, which is either from the internal integrated PHY or an external PHY. This register can be written to clear the EPHYRIS register.

This register is used for clearing the EPHYRIS register bits.

EPHYMISC is shown in [Figure 15-88](#) and described in [Table 15-98](#).

Return to [Summary Table](#).

Figure 15-88. EPHYMISC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														INT	
R-0x0														R/W1 C-0x0	

Table 15-98. EPHYMISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	INT	R/W1C	0x0	Ethernet PHY Status and Clear register. Reading this register provides a result which is the logical AND of the EPHYRIS and EPHYIM registers. A write of 1 to a bit of this register clears the corresponding bit in the EPHYRIS register. The Ethernet MAC interrupt is an OR'd summary of both the masked EMACRIS register output and this register. When an Ethernet MAC interrupt is asserted, software must check both the EMACRIS and EMACIM registers along with this register.

15.7 MII Management (EPHY) Registers

[Table 15-99](#) lists the memory-mapped registers for the EPHY. All register offset addresses not listed in [Table 15-99](#) should be considered as reserved locations and the register contents should not be modified.

PHY registers are accessed through the EMACMIIADDR register thus the base address is not applicable.

The IEEE 802.3 standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers. All addresses given are absolute and are written directly to the MII field of the Ethernet MAC MII Address (EMACMIIADDR) register. The PLA value of the EMACMIIADDR register for the internal PHY is 0x00.

Table 15-99. MII Management (EPHY) Registers

Address	Acronym	Register Name	Section
0x0	EPHYBMCR	Ethernet PHY Basic Mode Control - MR0	Section 15.7.1
0x1	EPHYBMSR	Ethernet PHY Basic Mode Status - MR1	Section 15.7.2
0x2	EPHYID1	Ethernet PHY Identifier Register 1 - MR2	Section 15.7.3
0x3	EPHYID2	Ethernet PHY Identifier Register 2 - MR3	Section 15.7.4
0x4	EPHYANA	Ethernet PHY Auto-Negotiation Advertisement - MR4	Section 15.7.5
0x5	EPHYANLPA	Ethernet PHY Auto-Negotiation Link Partner Ability -MR5	Section 15.7.6
0x6	EPHYANER	Ethernet PHY Auto-Negotiation Expansion - MR6	Section 15.7.7
0x7	EPHYANNPTR	Ethernet PHY Auto-Negotiation Next Page TX - MR7	Section 15.7.8
0x8	EPHYANLNPTR	Ethernet PHY Auto-Negotiation Link Partner Ability Next Page - MR8	Section 15.7.9
0x9	EPHYCFG1	Ethernet PHY Configuration 1 - MR9	Section 15.7.10
0xA	EPHYCFG2	Ethernet PHY Configuration 2 - MR10	Section 15.7.11
0xB	EPHYCFG3	Ethernet PHY Configuration 3 - MR11	Section 15.7.12
0xD	EPHYREGCTL	Ethernet PHY Register Control - MR13	Section 15.7.13
0xE	EPHYADDAR	Ethernet PHY Address or Data - MR14	Section 15.7.14
0x10	EPHYSTS	Ethernet PHY Status - MR16	Section 15.7.15
0x11	EPHYSCR	Ethernet PHY Specific Control - MR17	Section 15.7.16
0x12	EPHYMISR1	Ethernet PHY MII Interrupt Status 1 - MR18	Section 15.7.17
0x13	EPHYMISR2	Ethernet PHY MII Interrupt Status 2 - MR19	Section 15.7.18
0x14	EPHYFCSCR	Ethernet PHY False Carrier Sense Counter - MR20	Section 15.7.19
0x15	EPHYRXERCNT	Ethernet PHY Receive Error Count - MR21	Section 15.7.20
0x16	EPHYBISTCR	Ethernet PHY BIST Control - MR22	Section 15.7.21
0x18	EPHYLEDCR	Ethernet PHY LED Control - MR24	Section 15.7.22
0x19	EPHYCTL	Ethernet PHY Control - MR25	Section 15.7.23
0x1A	EPHY10BTSC	Ethernet PHY 10Base-T Status/Control - MR26	Section 15.7.24
0x1B	EPHYBICSR1	Ethernet PHY BIST Control and Status 1 - MR27	Section 15.7.25
0x1C	EPHYBICSR2	Ethernet PHY BIST Control and Status 2 - MR28	Section 15.7.26
0x1E	EPHYCDCR	Ethernet PHY Cable Diagnostic Control - MR30	Section 15.7.27
0x1F	EPHYRCR	Ethernet PHY Reset Control - MR31	Section 15.7.28
0x25	EPHYLEDCFG	Ethernet PHY LED Configuration - MR37	Section 15.7.29

Complex bit access types are encoded to fit into small table cells. [Table 15-100](#) shows the codes that are used for access types in this section.

Table 15-100. EPHY Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		

Table 15-100. EPHY Access Type Codes (continued)

Access Type	Code	Description
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

15.7.1 EPHYBMCR Register (Address = 0x0) [reset = 0x3100]

Ethernet PHY Basic Mode Control - MR0 (EPHYBMCR)

This register describes the basic mode controls available to the EPHY. The reset state of the ANEN bit is controlled by the EMACPC register.

EPHYBMCR is shown in [Figure 15-89](#) and described in [Table 15-101](#).

Return to [Summary Table](#).

Figure 15-89. EPHYBMCR Register

15	14	13	12	11	10	9	8
MIIRESET	MILOOPBK	SPEED	ANEN	PWRDWN	ISOLATE	RESTARTAN	DUPLEXM
R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1
7	6	5	4	3	2	1	0
COLLTST	RESERVED						
R/W-0x0	R-0x0						

Table 15-101. EPHYBMCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	MIIRESET	R/W	0x0	MII Register reset. Writing a 1 to this bit resets the contents of the MII-related registers: EPHYBMCR (0x000) EPHYANA (0x004) and EPHYANNPTR (0x007). When the reset operation is done, this bit is cleared to 0 automatically. 0x0 = Normal operation. 0x1 = Initiate MII Reset / Reset in Process.
14	MILOOPBK	R/W	0x0	MII Loopback. When MII loopback mode is activated, the transmitter data presented on MII TXD is looped back to MII RXD internally. 0x0 = Normal operation. 0x1 = MII Loopback enabled.
13	SPEED	R/W	0x1	Speed Select. When auto-negotiation is disabled writing to this bit allows the port speed to be selected. 0x0 = 10Mbps 0x1 = 100Mbps
12	ANEN	R/W	0x1	Auto-Negotiate Enable. 0x0 = Auto-Negotiation Disabled. The SPEED bit and the DUPLEXM bit determine the port speed and duplex mode. 0x1 = Auto-Negotiation Enabled. The SPEED bit and the DUPLEXM bit of this register are ignored when this bit is set.
11	PWRDWN	R/W	0x0	Power Down. Setting this bit powers down the PHY. Only minimal register functionality is enabled during the power down condition. 0x0 = Normal operation 0x1 = Power-down modes are enabled: General Power Down Mode, Active Sleep Mode and Passive Sleep Mode (see Ethernet PHY Specific Control (EPHYSCR) register, offset 0x011).
10	ISOLATE	R/W	0x0	Port Isolate. 0x0 = Normal operation. 0x1 = Isolates the Port from the MII with the exception of the serial management.
9	RESTARTAN	R/W	0x0	Restart Auto-Negotiation. 0x0 = Normal operation. 0x1 = Auto-Negotiation process is re-initiated. If Auto-Negotiation is disabled (ANEN = 0), this bit is ignored. This bit is self-clearing and returns a value of 1 until Auto-Negotiation is initiated, whereupon it self-clears. Operation of the Auto-Negotiation process is not affected by the management entity clearing this bit.

Table 15-101. EPHYBMCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	DUPLEXM	R/W	0x1	Duplex Mode. When auto-negotiation is disabled writing to this bit allows the port-duplex capability to be selected. 0x0 = Half Duplex operation. 0x1 = Full Duplex Operation
7	COLLTST	R/W	0x0	Collision Test. When set, this bit causes the EN0COL signal to be asserted in response to the assertion of EN0TXEN within 512 bit times. The EN0COL signal is deasserted within four bit times in response to the deassertion of EN0TXEN. 0x0 = Normal operation 0x1 = Collision test enabled.
6-0	RESERVED	R	0x0	

15.7.2 EPHYBMSR Register (Address = 0x1) [reset = 0x7849]

Ethernet PHY Basic Mode Status - MR1 (EPHYBMSR)

This register reflects the basic mode features that are available in the EPHY.

EPHYBMSR is shown in [Figure 15-90](#) and described in [Table 15-102](#).

Return to [Summary Table](#).

Figure 15-90. EPHYBMSR Register

15	14	13	12	11	10	9	8
RESERVED	100BTXFD	100BTXHD	10BTFD	10BTHD	RESERVED		
R-0x0	R-0x1	R-0x1	R-0x1	R-0x1	R-0x0		
7	6	5	4	3	2	1	0
RESERVED	MFPRESUP	ANC	RFAULT	ANEN	LINKSTAT	JABBER	EXTEN
R-0x0	R-0x1	R-0x0	R-0x0	R-0x1	R-0x0	R-0x0	R-0x1

Table 15-102. EPHYBMSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0x0	
14	100BTXFD	R	0x1	100Base-TX Full Duplex Capable. 0x0 = Device does not support 100Base-TX in full duplex mode. 0x1 = Device supports 100Base-TX in full duplex mode.
13	100BTXHD	R	0x1	100Base-TX Half Duplex Capable. 0x0 = The device does not support 100Base-TX in half duplex mode. 0x1 = The device supports 100Base-TX in half duplex mode.
12	10BTFD	R	0x1	10 Base-T Full Duplex Capable. 0x0 = The device does not support 10Base-T in full duplex mode. 0x1 = Device able to perform 10Base-T in full duplex mode.
11	10BTHD	R	0x1	10 Base-T Half Duplex Capable. 0x0 = Device not able to perform 10Base-T in half duplex mode. 0x1 = Device able to perform 10Base-T in half duplex mode.
10-7	RESERVED	R	0x0	
6	MFPRESUP	R	0x1	Preamble Suppression Capable. 0x0 = The device does not perform management transactions with preambles suppressed. 0x1 = The device performs management transactions with preambles suppressed. For this mode, the 32-bits of preamble are needed only once after reset, invalid opcode or invalid turnaround.
5	ANC	R	0x0	Auto-Negotiation Complete. 0x0 = Auto-Negotiation process not complete (either still in process, disabled, or reset) 0x1 = Auto-Negotiation process complete.
4	RFAULT	R	0x0	Remote Fault. 0x0 = No remote fault condition detected. 0x1 = Remote Fault condition detected (cleared on read or by reset). Fault criteria: Far End Fault Indication or notification from Link Partner of Remote Fault.
3	ANEN	R	0x1	Auto Negotiation Enabled. 0x0 = Device is not able to perform Auto-Negotiation. 0x1 = Device is able to perform Auto-Negotiation.
2	LINKSTAT	R	0x0	Link Status. 0x0 = Link not established. 0x1 = Valid link established (for either 10 or 100Mbps operation).

Table 15-102. EPHYBMSR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	JABBER	R	0x0	Jabber Detect. This bit is implemented with a latching function, such that the occurrence of a jabber condition causes it to set until it is cleared by a read from this register, by the management interface, or by a reset. 0x0 = No Jabber condition detected. 0x1 = Jabber condition detected
0	EXTEN	R	0x1	Extended Capability Enable. 0x0 = Basic register set capabilities only. 0x1 = Extended register capabilities.

15.7.3 EPHYID1 Register (Address = 0x2) [reset = 0x2000]

Ethernet PHY Identifier Register 1 - MR2 (EPHYID1)

The Ethernet PHY Identifier 1 and 2 (EPHYIDn) registers together form a unique identifier. The identifier consists of a concatenation of the Organizationally Unique Identifier (OUI), the vendor's model number and the model revision number. A PHY may return a value of zero in each of the 32 bits of the PHY Identifier if desired. The PHY Identifier is intended to support network management. The Texas Instruments IEEE-assigned OUI is 0x080028. This OUI field is broken into two fields in the EPHYID1 and EPHYID2 register. The most significant OUI field, OUI MSB in the EPHYID1 register, is equal to 0x0002 (the two most significant bits of the OUI are ignored). The least significant OUI field, OUI LSB in the EPHYID2 register, is equal to 0x28.

EPHYID1 is shown in [Figure 15-91](#) and described in [Table 15-103](#).

Return to [Summary Table](#).

Figure 15-91. EPHYID1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUI MSB															
R-0x2000															

Table 15-103. EPHYID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OUI MSB	R	0x2000	OUI Most Significant Bits. OUI[21:6] = 0x200. The most significant two bits of the OUI are ignored (the IEEE standard refers to these as bits 1 and 2).

15.7.4 EPHYID2 Register (Address = 0x3) [reset = 0xA221]

Ethernet PHY Identifier Register 2 - MR3 (EPHYID2)

The Ethernet PHY Identifier 1 and 2 (EPHYIDn) registers together form a unique identifier. The identifier consists of a concatenation of the Organizationally Unique Identifier (OUI), the vendor's model number and the model revision number. A PHY may return a value of zero in each of the 32 bits of the PHY Identifier if desired. The PHY Identifier is intended to support network management. The Texas Instruments IEEE-assigned OUI is 0x080028. This OUI field is broken into two fields in the EPHYID1 and EPHYID2 register. The most significant OUI field, OUI_MSB in the EPHYID1 register, is equal to 0x0002 (the two most significant bits of the OUI are ignored). The least significant OUI field, OUI_LSB in the EPHYID2 register, is equal to 0x28.

EPHYID2 is shown in [Figure 15-92](#) and described in [Table 15-104](#).

Return to [Summary Table](#).

Figure 15-92. EPHYID2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUI_LSB						VND_RMDL						MDL_REV			
R-0x28						R-0x22						R-0x1			

Table 15-104. EPHYID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	OUI_LSB	R	0x28	OUI Least Significant Bits. Bits 19 to 24 of the OUI (0x080028) are mapped from bits 15 to 10 of this register respectively.
9-4	VND_RMDL	R	0x22	Vendor Model Number. The six bits of vendor model number are mapped from bits 9 to 4 (most significant bit to bit 9).
3-0	MDL_REV	R	0x1	Model Revision Number. Four bits of the vendor model revision number are mapped from bits 3 to 0 (most significant bit to bit 3). This field is incremented for all major device changes.

15.7.5 EPHYANA Register (Address = 0x4) [reset = 0x01E1]

Ethernet PHY Auto-Negotiation Advertisement - MR4 (EPHYANA)

This register contains the advertised abilities of this device as they are transmitted to its link partner during Auto-Negotiation.

EPHYANA is shown in [Figure 15-93](#) and described in [Table 15-105](#).

Return to [Summary Table](#).

Figure 15-93. EPHYANA Register

15	14	13	12	11	10	9	8
NP	RESERVED	RF	RESERVED	ASMDUP	PAUSE	100BT4	100BTXFD
R/W-0x0	R-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x1
7	6	5	4	3	2	1	0
100BTX	10BTFD	10BT	SELECT				
R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1				

Table 15-105. EPHYANA Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NP	R/W	0x0	Next Page Indication. 0x0 = Next Page Transfer not desired. 0x1 = Next Page Transfer desired.
14	RESERVED	R	0x0	
13	RF	R/W	0x0	Remote Fault. 0x0 = No Remote Fault detected. 0x1 = Advertises that this device has detected a Remote Fault.
12	RESERVED	R	0x0	
11	ASMDUP	R/W	0x0	Asymmetric PAUSE support for Full Duplex Links. Encoding and resolution of PAUSE bits is defined in IEEE 802.3 Annex 28B, Tables 28B-2 and 28B-3, respectively. Pause resolution status is reported in the Pause Status bits [13:12], of the Ethernet PHY Control (EPHYCTL) register. 0x0 = Asymmetric PAUSE not implemented. 0x1 = Asymmetric PAUSE implemented. Advertise that the MAC has implemented both the optional MAC control sublayer and the pause function as specified in clause 31 and annex 31B of IEEE802.3u.
10	PAUSE	R/W	0x0	PAUSE Support for Full Duplex Links. The PAUSE bit indicates that the device is capable of providing the symmetric PAUSE functions as defined in Annex 31B. Encoding and resolution of PAUSE bits is defined in IEEE 802.3 Annex 28B, Tables 28B-2 and 28B-3, respectively. Pause resolution status is reported in the Ethernet PHY Control (EPHYCTL) register. 0x0 = MAC PAUSE not implemented 0x1 = MAC PAUSE implemented. Advertise that the MAC has implemented both the optional MAC control sub-layer and the pause function as specified in clause 31 and annex 31B of 802.3u.
9	100BT4	R	0x0	100Base-T4 Support. 0x0 = 100Base-T4 not supported by the internal PHY. 0x1 = 100Base-T4 is supported by the internal PHY.
8	100BTXFD	R/W	0x1	100Base-TX Full Duplex Support. 0x0 = 100Base-TX Full Duplex not supported by the internal PHY. 0x1 = 100Base-TX Full Duplex is supported by the internal PHY.
7	100BTX	R/W	0x1	100Base-TX Support. 0x0 = 100Base-TX not supported by the internal PHY. 0x1 = 100Base-TX is supported by the internal PHY.

Table 15-105. EPHYANA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	10BTFD	R/W	0x1	10Base-T Full Duplex Support. 0x0 = 10Base-T Full Duplex not supported by the internal PHY. 0x1 = 10Base-T Full Duplex is supported by the internal PHY.
5	10BT	R/W	0x1	10Base-T Support. 0x0 = 10Base-T not supported by the internal PHY. 0x1 = 10Base-T is supported by the internal PHY.
4-0	SELECT	R/W	0x1	Protocol Selection. These bits contain the binary encoded protocol selector supported by this port. 0x1 indicates that this device supports IEEE 802.3u.

15.7.6 EPHYANLPA Register (Address = 0x5) [reset = 0x0]

Ethernet PHY Auto-Negotiation Link Partner Ability - MR5 (EPHYANLPA)

This register contains the advertised abilities of the Link Partner as received during Auto-Negotiation. The content changes after the successful auto-negotiation if next-pages are supported.

EPHYANLPA is shown in [Figure 15-94](#) and described in [Table 15-106](#).

Return to [Summary Table](#).

Figure 15-94. EPHYANLPA Register

15	14	13	12	11	10	9	8
NP	ACK	RF	RESERVED	ASMDUP	PAUSE	100BT4	100BTXFD
R-0x0	R-0x0	R-0x0	R-0x0	R/W-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
100BTX	10BTFD	10BT	SELECT				
R-0x0	R-0x0	R-0x0	R-0x1				

Table 15-106. EPHYANLPA Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NP	R	0x0	Next Page Indication. 0x0 = Link Partner does not desire Next Page Transfer. 0x1 = Link Partner desires Next Page Transfer.
14	ACK	R	0x0	Acknowledge. 0x0 = Not acknowledged. The Auto-Negotiation state machine will automatically control the this bit based on the incoming FLP bursts. 0x1 = Link Partner acknowledges reception of the ability data word.
13	RF	R	0x0	Remote Fault. 0x0 = No remote fault indicated by link partner. 0x1 = Remote fault indicated by link partner.
12	RESERVED	R	0x0	
11	ASMDUP	R/W	0x0	Asymmetric PAUSE. 0x0 = Asymmetric pause is not supported by the Link Partner. 0x1 = Asymmetric pause is supported by the Link Partner.
10	PAUSE	R	0x0	PAUSE. 0x0 = Pause function is not supported by the Link Partner. 0x1 = Pause function is supported by the Link Partner.
9	100BT4	R	0x0	100Base-T4 Support. 0x0 = 100Base-T4 is not supported by the Link Partner. 0x1 = 100Base-T4 is supported by the Link Partner.
8	100BTXFD	R	0x0	100Base-TX Full Duplex Support. 0x0 = 100Base-TX Full Duplex is not supported by the Link Partner. 0x1 = 100Base-TX Full Duplex is supported by the Link Partner.
7	100BTX	R	0x0	100Base-TX Support. 0x0 = 100Base-TX is not supported by the Link Partner. 0x1 = 100Base-TX is supported by the Link Partner.
6	10BTFD	R	0x0	10Base-T Full Duplex Support. 0x0 = 10Base-T Full Duplex is not supported by the Link Partner. 0x1 = 10Base-T Full Duplex is supported by the Link Partner.
5	10BT	R	0x0	10Base-T Support. 0x0 = 10Base-T is not supported by the Link Partner. 0x1 = 10Base-T is supported by the Link Partner
4-0	SELECT	R	0x1	Protocol Selection. Link Partner's binary encoded protocol selector.

15.7.7 EPHYANER Register (Address = 0x6) [reset = 0x4]

Ethernet PHY Auto-Negotiation Expansion - MR6 (EPHYANER)

This register contains additional local device and link partner status information.

EPHYANER is shown in [Figure 15-95](#) and described in [Table 15-107](#).

Return to [Summary Table](#).

Figure 15-95. EPHYANER Register

15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			PDF	LPNPABLE	NPABLE	PAGERX	LPANABLE
R-0x0			R-0x0	R-0x0	R-0x1	R-0x0	R-0x0

Table 15-107. EPHYANER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0x0	
4	PDF	R	0x0	Parallel Detection Fault. 0x0 = A fault has not been detected. 0x1 = A fault has been detected via the Parallel Detection function.
3	LPNPABLE	R	0x0	Link Partner Next Page Able. 0x0 = Link Partner does not support next page. 0x1 = Link Partner does support next page.
2	NPABLE	R	0x1	Next Page Able. 0x0 = Indicates local device is not able to send additional next pages. 0x1 = Indicates local device is able to send additional next pages
1	PAGERX	R	0x0	Link Code Word Page Received. 0x0 = Link Code Word has not been received. 0x1 = Link Code Word has been received. This bit is cleared when read.
0	LPANABLE	R	0x0	Link Partner Auto-Negotiation Able. 0x0 = Indicates that the Link Partner does not support Auto-Negotiation. 0x1 = Indicates that the Link Partner supports Auto-Negotiation.

15.7.8 EPHYANNPTR Register (Address = 0x7) [reset = 0x2001]

Ethernet PHY Auto-Negotiation Next Page TX - MR7 (EPHYANNPTR)

This register contains the next page information sent by this device to its link partner during Auto-Negotiation.

EPHYANNPTR is shown in [Figure 15-96](#) and described in [Table 15-108](#).

Return to [Summary Table](#).

Figure 15-96. EPHYANNPTR Register

15	14	13	12	11	10	9	8
NP	RESERVED	MP	ACK2	TOGTX	CODE		
R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x1		
7	6	5	4	3	2	1	0
CODE							
R/W-0x1							

Table 15-108. EPHYANNPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NP	R/W	0x0	Next Page Indication. 0x0 = No other next-page transfer desired. 0x1 = Another next page desired.
14	RESERVED	R	0x0	
13	MP	R/W	0x0	Message Page. 0x0 = Unformatted Page. 0x1 = Message Page.
12	ACK2	R/W	0x0	Acknowledge 2. 0x0 = Cannot comply with message. 0x1 = Can comply with message.
11	TOGTX	R	0x0	Toggle. Toggle is used by the arbitration function within auto-negotiation to synchronize with the Link Partner during next page exchange. This bit always takes the opposite value of the toggle bit in the previously exchanged Link Code Word. 0x0 = Value of toggle bit in previously transmitted Link Code Word was 1. 0x1 = Value of toggle bit in previously transmitted Link Code Word was 0.
10-0	CODE	R/W	0x1	Code. This field represents the code field of the next page transmission. If the MP bit is set (bit 13 of this register), then the code is interpreted as a Message Page, as defined in annex 28C of IEEE 802.3u. Otherwise, the code is interpreted as an Unformatted Page, and the interpretation is application specific. The default value of the CODE field represents a Null Page as defined in Annex 28C of IEEE 802.3u.

15.7.9 EPHYANLNPTR Register (Address = 0x8) [reset = 0x0]

Ethernet PHY Auto-Negotiation Link Partner Ability Next Page - MR8 (EPHYANLNPTR)

This register contains the next page information sent by this device to its Link Partner during Auto-Negotiation.

EPHYANLNPTR is shown in [Figure 15-97](#) and described in [Table 15-109](#).

Return to [Summary Table](#).

Figure 15-97. EPHYANLNPTR Register

15	14	13	12	11	10	9	8
NP	ACK	MP	ACK2	TOG	CODE		
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0		
7	6	5	4	3	2	1	0
CODE							
R-0x0							

Table 15-109. EPHYANLNPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NP	R	0x0	Next Page Indication. 0x0 = No other next page transfer desired. 0x1 = Another next page desired.
14	ACK	R	0x0	Acknowledge. The auto-negotiation state machine automatically controls this bit based on the incoming fast-link pulse (FLP) bursts. Software should not attempt to write to this bit. 0x0 = Not acknowledged. 0x1 = Link Partner acknowledges reception of the ability data word.
13	MP	R	0x0	Message Page. 0x0 = Unformatted Page. 0x1 = Message Page.
12	ACK2	R	0x0	Acknowledge 2. Acknowledge2 is used by the next page function to indicate that Local Device has the ability to comply with the message received. 0x0 = Link Partner cannot comply to next-page message. 0x1 = Link Partner has the ability to comply to next-page message.
11	TOG	R	0x0	Toggle. Toggle is used by the Arbitration function within Auto-Negotiation to synchronize with the Link Partner during Next Page exchange. This bit always takes the opposite value of the Toggle bit in the previously exchanged Link Code Word 0x0 = Value of toggle bit in previously transmitted Link Code Word was 1. 0x1 = Value of toggle bit in previously transmitted Link Code Word was 0.
10-0	CODE	R	0x0	Code. This field represents the code field of the next page transmission. If the MP bit is set (bit 13 of this RW register), then the code is interpreted as a Message Page, as defined in annex 28C of IEEE 802.3u. Otherwise, the code is interpreted as an Unformatted Page, and the interpretation is application specific. The default value of the CODE represents a Null Page as defined in Annex 28C of IEEE 802.3u.

15.7.10 EPHYCFG1 Register (Address = 0x9) [reset = 0x0]

Ethernet PHY Configuration 1 - MR9 (EPHYCFG1)

This register configuration for the Ethernet PHY. These configuration values are programmed by the system processor after a POR. The DONE bit in the EPHYCFG1 register is set when configuration is complete. This register is used when the user requires a configuration different from what is provided in the EMACPC register.

EPHYCFG1 is shown in [Figure 15-98](#) and described in [Table 15-110](#).

Return to [Summary Table](#).

Figure 15-98. EPHYCFG1 Register

15	14	13	12	11	10	9	8
DONE	RESERVED						TDRAR
W-0x0	R-0x0						R/W-0x0
7	6	5	4	3	2	1	0
LLR	FAMDIX	RAMDIX	FASTANEN	FANSEL	FRXDVDDET	RESERVED	
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 15-110. EPHYCFG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DONE	W	0x0	Configuration Done. This bit reads as a zero. The application must write a one to this bit to terminate the PHYHOLD mode set in the EMACPC register and wake up the EPHY. 0x0 = Configuration process is not complete. 0x1 = Configuration process is complete, and the PHY can continue and complete its internal reset sequence.
14-9	RESERVED	R	0x0	
8	TDRAR	R/W	0x0	TDR Auto-Run at Link Down. 0x0 = Disable automatic execution of TDR. 0x1 = Enable execution of TDR procedure after link down event.
7	LLR	R/W	0x0	Link Loss Recovery. 0x0 = Normal link loss operation. Link status goes down approximately 250 μ s from signal loss. 0x1 = Enable link loss recovery mechanism. This mode allows recovery from short interference and continues to hold the link up for a period of an additional few microseconds until the short interference is gone and the signal is OK.
6	FAMDIX	R/W	0x0	Fast Auto MDI/MDIX. If both link partners are configured to work in Force 100Base-TX mode (auto-negotiation is disabled), this mode enables automatic MDI/MDIX resolution in a short time. 0x0 = Normal Auto MDI/MDIX mode. 0x1 = Enable Fast Auto MDI/MDIX mode.
5	RAMDIX	R/W	0x0	Robust Auto MDI/MDIX. If link partners are configured to operational modes that are not supported by normal Auto MDI/MDIX mode (like Auto-Neg versus Force 100Base-TX or Force 100Base-TX versus Force 100Base-TX), this Robust Auto MDI/MDIX mode allows MDI/MDIX resolution and prevents deadlock. 0x0 = Normal Auto MDI/MDIX mode. 0x1 = Enable Robust Auto MDI/MDIX resolution.

Table 15-110. EPHYCFG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	FASTANEN	R/W	0x0	<p>Fast Auto Negotiation Enable. Adjusting these bits reduces the time it takes to auto-negotiate between two PHYs. When using this option care must be taken to maintain proper operation of the system. While shortening these timer intervals may not cause problems in normal operation, there are certain situations where this may lead to problems.</p> <p>0x0 = Disable fast auto-negotiation mode. The PHY auto-negotiates using the normal timer setting</p> <p>0x1 = Enable fast auto-negotiation mode. The PHY auto-negotiates using the timer setting according to the FANSEL bits (bits [3:2] of this register)</p>
3-2	FANSEL	R/W	0x0	<p>Fast Auto-Negotiation Select Configuration. This bit is preconfigured at reset by the EMACPC register. For custom configuration see . Adjusting these bits reduces the time it takes to auto-negotiate between two PHYs. In Fast AN mode, both PHYs should be configured to the same configuration. These two bits define the duration for each state of the auto-negotiation process according to the table above. The new duration time must be enabled by setting bit 4 (FASTANEN) of this register. Using this mode in cases where both link partners are not configured to the same fast auto-negotiation configuration might produce scenarios with unexpected behavior.</p> <p>0x0 = Break Link Timer: 80 ms, Link Fail Inhibit Timer: 50 ms, Auto-Negate Wait Timer: 35 ms</p> <p>0x1 = Break Link Timer: 120 ms, Link Fail Inhibit Timer: 75 ms, Auto-Negate Wait Timer: 50 ms</p> <p>0x2 = Break Link Timer: 240 ms, Link Fail Inhibit Timer: 150 ms, Auto-Negate Wait Timer: 100 ms</p> <p>0x3 = Reserved</p>
1	FRXDVDDET	R/W	0x0	<p>FAST RXDV Detection. Enabling this feature allows the PHY to pass data to the MII interface earlier. This bit is set to disabled if using the EMACPC register bits to program the PHY configuration.</p> <p>0x0 = Disable fast RXDV detection. The PHY operates in normal mode where an internally asserted RXDV signal is sent to the MII interface after detection of /J/K/.</p> <p>0x1 = An internal RXDV signal is asserted high on receive packet due to detection of /J/ symbol only. Fast RXDV detection allows the PHY to pass data to the MII interface earlier. If a consecutive /K/ does not appear, RXERR is generated.</p>
0	RESERVED	R	0x0	

15.7.11 EPHYCFG2 Register (Address = 0xA) [reset = 0x4]

Ethernet PHY Configuration 2 - MR10 (EPHYCFG2)

Fields in this register are used to configure the Ethernet PHY. These configuration values are programmed by the system processor after a POR. The DONE bit in the EPHYCFG1 register is set when configuration is complete. This register is used when the user requires a configuration different from what is provided in the EMACPC register.

EPHYCFG2 is shown in [Figure 15-99](#) and described in [Table 15-111](#).

Return to [Summary Table](#).

Figure 15-99. EPHYCFG2 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	FLUPPD	EXTFD	ENLEDLINK	ISOMIILL	RXERRIDLE	ODDNDTDIS	RESERVED
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0	R-0x0

Table 15-111. EPHYCFG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0x0	
6	FLUPPD	R/W	0x0	Fast Link-Up in Parallel Detect Mode. In fast-auto MDI-X and in robust-auto MDI-X modes (bits 6 and 5 in register EPHYCFG1), this bit is automatically set. 0x0 = Normal Parallel Detection link establishment 0x1 = Enable Fast Link-Up time During Parallel Detection
5	EXTFD	R/W	0x0	Extended Full-Duplex Ability. Encodes the type of PHY attached. 0x0 = Disable extended full-duplex ability. Decision to work in full-duplex or half-duplex mode follows IEEE specification. 0x1 = Force full-duplex while working with link partner in forced 100B-TX. When the PHY is set to Auto-Negotiation or Force 100B-TX and the link partner is operated in Force 100B-TX, the link is always full duplex
4	ENLEDLINK	R/W	0x0	Enhanced LED Functionality. 0x0 = LED Link is ON when link is established. 0x1 = LED Link is ON only when link is established in 100B-TX Full Duplex mode.
3	ISOMIILL	R/W	0x0	Isolate MII outputs when Enhanced Link is not Achievable. 0x0 = Normal MII outputs operation 0x1 = When no link established in 100B-TX and Full Duplex conditions, MII outputs are tied low.
2	RXERRIDLE	R/W	0x1	Detection of Receive Symbol Error During IDLE State. 0x0 = Disable detection of Receive symbol error during IDLE state. 0x1 = Enable detection of Receive symbol error during IDLE state.
1	ODDNDTDIS	R/W	0x0	Detection of Transmit Error. 0x0 = Enable detection of deassertion of the internal transmit enable on an odd-nibble boundary. In this case the internal transmit enable is extended by one additional transmit clock cycle and behaves as if the internal transmit error was asserted during that additional cycle. 0x1 = Disable detection of internal transmit error in odd-nibble boundary.
0	RESERVED	R	0x0	

15.7.12 EPHYCFG3 Register (Address = 0xB) [reset = 0x0]

Ethernet PHY Configuration 3 - MR11 (EPHYCFG3)

Fields in this register are used to configure the Ethernet PHY. These configuration values are programmed by the system processor after a POR. The DONE bit in the EPHYCFG1 register is set when configuration is complete. This register is used when the user requires a configuration different from what is provided in the EMACPC register.

EPHYCFG3 is shown in [Figure 15-100](#) and described in [Table 15-112](#).

Return to [Summary Table](#).

Figure 15-100. EPHYCFG3 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
POLSWAP	MDIMDIXS	RESERVED	FLDWNM				
R/W-0x0	R/W-0x0	R-0x0	R/W-0x0				

Table 15-112. EPHYCFG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0x0	
7	POLSWAP	R/W	0x0	Polarity Swap. To enable the port mirror function, set this bit and bit 6 (MDIMDIXS) to 1. 0x0 = MDI pairs normal (Receive on EN0RXIN/EN0RXIP pair, Transmit on EN0TXON/EN0TXOP pair) 0x1 = Inverted polarity on both pairs.
6	MDIMDIXS	R/W	0x0	MDI/MDIX Swap. To enable the port mirroring function, set bit 7 and this bit to 1. 0x0 = MDI pairs normal (Receive on EN0RXIN/EN0RXIP pair, Transmit on EN0TXON/EN0TXOP pair) 0x1 = Swap MDI pairs (Receive on EN0TXON/EN0TXOP pair, Transmit on EN0RXIN/EN0RXIP pair)
5	RESERVED	R	0x0	
4-0	FLDWNM	R/W	0x0	Fast Link Down Modes. The Fast Link Down function is an OR of the following five options. The application can enable combinations of these conditions. Bit 4: Drop the link due to descrambler sync loss. If this bit is enabled, the link is dropped when the receiver loses synchronization of the transmitter. Bit 3: Drop the link based on RX Error count of the MII interface. When a predefined number of 32 RX Error occurrences in a 10 us interval is reached, the link is dropped. Bit 2: Drop the link based on MLT3 Errors count (Violation of the MLT3 coding in the DSP output). When a predefined number of 20 MLT3 Error occurrences in a 10 us interval is reached, the link is dropped. Bit 1: Drop the link based on Low SNR Threshold. When a predefined number of 20 Threshold crossing occurrences in a 10 us interval is reached, the link is dropped. Bit 0: Drop the link based on Signal/Energy loss indication - When the Energy detector indicates Energy Loss, the link is dropped. Typical reaction time is 10 us.

15.7.13 EPHYREGCTL Register (Address = 0xD) [reset = 0x0]

Ethernet PHY Register Control - MR13 (EPHYREGCTL)

EPHYREGCTL (0x00D) and EPHYADDAR (0x00E) registers allow read and write access to the extended register set using indirect addressing. The modes for the FUNC field are as follows:

- EPHYREGCTL[15:14] = 0x0: A write to EPHYADDAR modifies the extended register set address register. This address register must be initialized in order to access any of the registers within the extended register set.
- EPHYREGCTL[15:14] = 0x1: A read/write to EPHYADDAR operates on the register within the extended register set selected (pointed to) by the value in the address register. The address register contents (pointer) remain unchanged.
- EPHYREGCTL[15:14] = 0x2: A read/write to EPHYADDAR operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for both reads and writes, the value in the address register is incremented.
- EPHYREGCTL[15:14] = 0x3: A read/write to EPHYADDAR operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for write accesses only, the value in the address register is incremented. For read accesses, the value of the address register remains unchanged.

This register is the MDIO Manageable Device (MMD) access control. In general, register EPHYREGCTL [4:0] is the device address, DEVAD, that directs any accesses of EPHYADDAR (0x00E) register to the appropriate MMD. It also contains selection bits for auto-increment of the data register. This register contains the device address to be written to access the extended registers. Write 0x1F into bits [4:0] of this register. It also contains selection bits [15:14] for the address auto-increment mode of EPHYADDAR.

EPHYREGCTL is shown in [Figure 15-101](#) and described in [Table 15-113](#).

Return to [Summary Table](#).

Figure 15-101. EPHYREGCTL Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FUNC		RESERVED									DEVAD				
W-0x0		R-0x0									W-0x0				

Table 15-113. EPHYREGCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FUNC	W	0x0	Function. 0x0 = Address 0x1 = Data, no post increment 0x2 = Data, post increment on read and write 0x3 = Data, post increment on write only
13-5	RESERVED	R	0x0	
4-0	DEVAD	W	0x0	Device Address. In general, these bits [4:0] are the device address DEVAD that directs any accesses of EPHYADDAR register (0x00E) to the appropriate MMD. The PHY uses the vendor specific DEVAD [4:0] = 0x1F for accesses. All accesses through registers EPHYREGCR and EPHYADDAR should use this DEVAD. Transactions with other DEVAD are ignored.

15.7.14 EPHYADDAR Register (Address = 0xE) [reset = 0x0]

Ethernet PHY Address or Data - MR14 (EPHYADDAR)

This register is the address/data MMD register. It is used in conjunction with EPHYREGCTL register (PHY offset 0x00D) to provide the access by an indirect read/write mechanism to the extended register set.

EPHYADDAR is shown in [Figure 15-102](#) and described in [Table 15-114](#).

Return to [Summary Table](#).

Figure 15-102. EPHYADDAR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRDATA															
W-0x0															

Table 15-114. EPHYADDAR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ADDRDATA	W	0x0	Address or Data. If the EPHYREGCTL register bits [15:14] = 0x0, this register holds the MDIO Manageable Device (MMD) DEVAD's address register, otherwise holds the MMD DEVAD's data register. See the EPHYREGCTL register for more information about the DEVAD field.

15.7.15 EPHYSTS Register (Address = 0x10) [reset = 0x2]

Ethernet PHY Status - MR16 (EPHYSTS)

This register provides quick access to commonly accessed PHY control status and general information.

EPHYSTS is shown in [Figure 15-103](#) and described in [Table 15-115](#).

Return to [Summary Table](#).

Figure 15-103. EPHYSTS Register

15	14	13	12	11	10	9	8
RESERVED	MDIXM	RXLERR	POLSTAT	FCSL	SD	DL	PAGERX
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
MIIREQ	RF	JD	ANS	MIILB	DUPLEX	SPEED	LINK
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x1	R-0x0

Table 15-115. EPHYSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0x0	
14	MDIXM	R	0x0	MDI-X Mode. This is a read-only status as reported by the Auto-Negotiation state machine. This bit is affected by the settings of the MDIXEN and FORCEMDIX bits in the EPHYCTL register. When MDIX is enabled, but not forced, this bit updates dynamically as the Auto-MDIX algorithm swaps between MDI and MDI-X configurations. 0x0 = MDI pairs normal (Receive on TPRD pair, Transmit on TPTD pair) 0x1 = MDI pairs swapped (Receive on TPTD pair, Transmit on TPRD pair)
13	RXLERR	R	0x0	Receive Error Latch. This bit is cleared on a read of the EPHYRXERCNT register. 0x0 = No receive error event has occurred. 0x1 = Receive error event has occurred since last read of EPHYRXERCNT register (PHY offset 0x015).
12	POLSTAT	R	0x0	Polarity Status. This bit is a duplication of bit 4 (POLSTAT) in the EPHY10BTSC register (PHY offset 0x01A). This bit is cleared upon a read of the EPHY10BTSC register, but not on a read of the EPHYSTS register. 0x0 = Correct Polarity detected. 0x1 = Inverted Polarity detected.
11	FCSL	R	0x0	False Carrier Sense Latch. This bit is cleared on a read of the EPHYFCSR register. 0x0 = No False Carrier event has occurred. 0x1 = False Carrier event has occurred since last read of EPHYFCSR register (0x014).
10	SD	R	0x0	Signal Detect. This bit displays the active high 100Base-TX unconditional Signal Detect indication from the PMD (Physical Layer Medium Dependent). This bit is latched low and held until it is read, based upon the occurrence of the corresponding event.
9	DL	R	0x0	Descrambler Lock. This bit displays the active high 100Base-TX Descrambler Lock indication from PMD. This bit is latched low and held until it is read, based upon the occurrence of the corresponding event.
8	PAGERX	R	0x0	Link Code Page Received. This bit is not cleared upon a read of the EPHYSTS register. 0x0 = Link Code Word Page has not been received. 0x1 = A new Link Code Word Page has been received. This is a duplicate of Page Received (bit 1) in the EPHYANER register and it is cleared on read of the EPHYANER register (0x006).

Table 15-115. EPHYSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	MIIREQ	R	0x0	<p>MI Interrupt Pending.</p> <p>0x0 = No interrupt pending</p> <p>0x1 = Indicates that an internal interrupt is pending. Interrupt source can be determined by reading the EPHYMISR1 Register (PHY offset 0x012). Reading the EPHYMISR1 clears this Interrupt bit indication.</p>
6	RF	R	0x0	<p>Remote Fault.</p> <p>0x0 = No remote fault condition detected.</p> <p>0x1 = Remote Fault condition detected. Criteria for a fault is when there is notification from Link Partner of Remote Fault via Auto-Negotiation. Cleared on read of EPHYBMSR register (PHY offset 0x001) or by reset.</p>
5	JD	R	0x0	<p>Jabber Detect. This bit will not be cleared upon a read of the EPHYSTS register.</p> <p>0x0 = No Jabber.</p> <p>0x1 = Jabber condition detected. This bit has meaning only in 10 Mb/s mode. This bit is a duplicate of the Jabber Detect bit in the EPHYBMSR register (PHY offset 0x001).</p>
4	ANS	R	0x0	<p>Auto-Negotiation Status.</p> <p>0x0 = Auto-Negotiation not complete.</p> <p>0x1 = Auto-Negotiation complete.</p>
3	MIILB	R	0x0	<p>MI Loopback Status.</p> <p>0x0 = Normal operation.</p> <p>0x1 = Loopback active (enabled).</p>
2	DUPLEX	R	0x0	<p>Duplex Status. This bit indicates duplex status and is determined from Auto-Negotiation or Forced Modes. Therefore, it is only valid if Auto-Negotiation is enabled and complete and there is a valid link or if Auto-Negotiation is disabled and there is a valid link.</p> <p>0x0 = Half Duplex Mode</p> <p>0x1 = Full Duplex Mode</p>
1	SPEED	R	0x1	<p>Speed Status. This bit indicates the status of the speed and is determined from Auto-Negotiation or Forced Modes. It is only valid if Auto-Negotiation is enabled and complete and there is a valid link or if Auto-Negotiation is disabled and there is a valid link.</p> <p>0x0 = 100 Mb/s mode.</p> <p>0x1 = 10 Mb/s mode.</p>
0	LINK	R	0x0	<p>Link Status. This bit is not cleared upon a read of the EPHYSTS register.</p> <p>0x0 = Link is not established.</p> <p>0x1 = Valid link is established (for either 10 or 100 Mb/s operation). This bit is a duplicate of the Link Status bit in the EPHYBMSR register (PHY offset 0x001).</p>

15.7.16 EPHYSCR Register (Address = 0x11) [reset = 0x103]

Ethernet PHY Specific Control- MR17 (EPHYSCR)

This register implements the PHY Specific Control register. This register allows access to general functionality inside the PHY to enable operation in reduced power modes and control the interrupt mechanism.

EPHYSCR is shown in [Figure 15-104](#) and described in [Table 15-116](#).

Return to [Summary Table](#).

Figure 15-104. EPHYSCR Register

15	14	13	12	11	10	9	8
DISCLK	PSEN	PSMODE	SBPYASS	RESERVED	LBFIFO		
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1	R-0x0	R/W-0x0		
7	6	5	4	3	2	1	0
RESERVED	COLFDM	RESERVED	TINT	INTEN	RESERVED		
R-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x1	R-0x0		

Table 15-116. EPHYSCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DISCLK	R/W	0x0	Disable CLK. Clocks can be disabled only in IEEE power down mode. 0x0 = Normal mode of operation 0x1 = Disable internal clocks
14	PSEN	R/W	0x0	Power Saving Modes Enable. 0x0 = Normal mode of operation 0x1 = Enable power saving modes
13-12	PSMODE	R/W	0x0	Power Saving Modes. 0x0 = Normal: Normal operation mode. PHY is fully functional 0x1 = IEEE Power DownLow Power mode that shuts down all internal circuitry other than SMI functionality. Power could be dropped further by setting high DISCLK bit in this register and disabling internal clocks circuitries. 0x2 = Active SleepLow Power Active Wake-On-LAN (WOL) mode that shuts down all internal circuitry other than SMI and energy detect functionality. In this mode the PHY sends NLP every 1.4 seconds to wake up the link-partner. Automatic power-up is done when link partner is detected. 0x3 = Passive SleepLow Power WOL mode that shuts down all internal circuitry besides SMI and energy detect functionality. Automatic power-up is done when link partner is detected.
11	SBPYASS	R/W	0x1	Scrambler Bypass. 0x0 = Scrambler bypass disabled. 0x1 = Scrambler bypass enabled.
10	RESERVED	R	0x0	
9-8	LBFIFO	R/W	0x0	Loopback FIFO Depth. This FIFO is used to adjust RX (recovered) clock rate to TX clock rate. FIFO depth must be set based on expected maximum packet size and clock accuracy. Default value sets to 5 nibbles. 0x0 = Four nibble FIFO 0x1 = Five nibble FIFO 0x2 = Six nibble FIFO 0x3 = Eight nibble FIFO
7-5	RESERVED	R	0x0	

Table 15-116. EPHYSCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	COLFDM	R/W	0x0	Collision in Full-Duplex Mode. 0x0 = Disable collision indication in full-duplex mode. Collision indication is active in half-duplex mode only. 0x1 = Enable collision signaling in full-duplex mode.
3	RESERVED	R	0x0	
2	TINT	R/W	0x0	Test Interrupt. Forces the PHY to generate an interrupt to facilitate interrupt testing. Interrupts will continue to be generated as long as this bit remains set. 0x0 = Do not generate interrupt 0x1 = Generate an interrupt
1	INTEN	R/W	0x1	Interrupt Enable. Enables interrupt dependent on the event enables in the EPHYMISR register (0x012). 0x0 = Disable event based interrupts 0x1 = Enable event based interrupts
0	RESERVED	R	0x0	

15.7.17 EPHYMISR1 Register (Address = 0x12) [reset = 0x0]

Ethernet PHY MII Interrupt Status 1 - MR18 (EPHYMISR1)

This register contains events status and enables for the interrupt function. If an event has occurred since the last read of this register, the corresponding status bit is set. If the corresponding enable bit in the register is set, an interrupt is generated if the event occurs. The INTEN bit (Bit 1) in the EPHYSCR register (0x011) must also be set to allow interrupts. The status indications in this register are set even if the interrupt is not enabled.

EPHYMISR1 is shown in [Figure 15-105](#) and described in [Table 15-117](#).

Return to [Summary Table](#).

Figure 15-105. EPHYMISR1 Register

15	14	13	12	11	10	9	8
RESERVED		LINKSTAT	SPEED	DUPLEXM	ANC	FCHF	RXHF
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED		LINKSTATEN	SPEEDEN	DUPLEXMEN	ANCEN	FCHFEN	RXHFEN
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 15-117. EPHYMISR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0x0	
13	LINKSTAT	R	0x0	Change of Link Status Interrupt. Reading this bit clears the interrupt and thus, the status bit. 0x0 = No change of link status. 0x1 = Change of link status interrupt is pending.
12	SPEED	R	0x0	Change of Speed Status Interrupt. Reading this bit clears the interrupt and thus, the status bit. 0x0 = No change of speed status. 0x1 = Change of speed status interrupt is pending.
11	DUPLEXM	R	0x0	Change of Duplex Status Interrupt. Reading this bit clears the interrupt and thus, the status bit. 0x0 = No change of duplex status. 0x1 = Duplex status change interrupt is pending.
10	ANC	R	0x0	Auto-Negotiation Complete Interrupt. Reading this bit clears the interrupt and thus, the status bit. 0x0 = No Auto-negotiation complete event is pending. 0x1 = Auto-negotiation complete interrupt is pending.
9	FCHF	R	0x0	False Carrier Counter Half-Full Interrupt. Reading this bit clears the interrupt and thus, the status bit. 0x0 = False carrier counter half-full event is not pending. 0x1 = False carrier counter (Register EPHYFCSCR) exceeds half-full and an interrupt is pending.
8	RXHF	R	0x0	Receive Error Counter Half-Full Interrupt. Reading this bit clears the interrupt and thus, the status bit. 0x0 = False carrier counter half-full event is not pending. 0x1 = Receive error counter (Register EPHYRXERCNT) exceeds half-full and an interrupt is pending.
7-6	RESERVED	R	0x0	
5	LINKSTATEN	R/W	0x0	Link Status Interrupt Enable. 0x0 = Change of link status interrupt disabled. 0x1 = Enable interrupt on change of link status.

Table 15-117. EPHYMISR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	SPEEDEN	R/W	0x0	Speed Change Interrupt Enable. 0x0 = Change of speed status interrupt disabled. 0x1 = Enable interrupt on change of speed status
3	DUPLEXMEN	R/W	0x0	Duplex Status Interrupt Enable. 0x0 = Change of duplex status interrupt disabled. 0x1 = Enable interrupt on change of duplex status
2	ANCEN	R/W	0x0	Auto-Negotiation Complete Interrupt Enable. 0x0 = Auto-negotiation complete interrupt disabled. 0x1 = Enable interrupt on Auto-negotiation complete event
1	FCHFEN	R/W	0x0	False Carrier Counter Register half-full Interrupt Enable. 0x0 = False Carrier Counter Register half-full interrupt disabled. 0x1 = Enable interrupt on False Carrier Counter Register half-full event
0	RXHFEN	R/W	0x0	Receive Error Counter Register Half-Full Event Interrupt. 0x0 = Receive Error Counter Register half-full interrupt disabled. 0x1 = Enable interrupt on Receive Error Counter Register half-full event

15.7.18 EPHYMISR2 Register (Address = 0x13) [reset = 0x0]

Ethernet PHY MII Interrupt Status 2 - MR19 (EPHYMISR2)

This register contains additional event status and enables for the interrupt function. If an event has occurred since the last read of this register, the corresponding status bit is set. If the corresponding enable bit in the register is set, an interrupt is generated if the event occurs. The INTEN bit (bit 1) in the EPHYSCR register (PHY offset 0x011) must also be set to allow interrupts. The status indications in this register are set even if the interrupt is not enabled.

EPHYMISR2 is shown in [Figure 15-106](#) and described in [Table 15-118](#).

Return to [Summary Table](#).

Figure 15-106. EPHYMISR2 Register

15	14	13	12	11	10	9	8
RESERVED	ANERR	PAGERX	LBFIFO	MDICO	SLEEP	POLINT	JABBER
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED	ANERREN	PAGERXEN	LBFIFOEN	MDICOEN	SLEEPEN	POLINTEN	JABBEREN
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 15-118. EPHYMISR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0x0	
14	ANERR	R	0x0	Auto-Negotiation Error Interrupt. Reading this bit clears the interrupt. 0x0 = No Auto-negotiation error event pending. 0x1 = Auto-negotiation error interrupt is pending.
13	PAGERX	R	0x0	Page Receive Interrupt. Reading this bit clears the interrupt. 0x0 = Page has not been received. 0x1 = Page has been received.
12	LBFIFO	R	0x0	Loopback FIFO Overflow/Underflow Event Interrupt. Reading this bit clears the interrupt. 0x0 = No FIFO Overflow/Underflow event pending. 0x1 = FIFO Overflow/Underflow event interrupt pending.
11	MDICO	R	0x0	MDI/MDIX Crossover Status Changed Interrupt. Reading this bit clears the interrupt. 0x0 = MDI crossover status has not changed. 0x1 = MDI crossover status changed interrupt is pending.
10	SLEEP	R	0x0	Sleep Mode Event Interrupt. Reading this bit clears the interrupt. 0x0 = No sleep mode event pending. 0x1 = Sleep Mode event interrupt is pending.
9	POLINT	R	0x0	Polarity Changed Interrupt. Reading this bit clears the interrupt. 0x0 = No Data polarity event pending. 0x1 = Data polarity changed interrupt pending.
8	JABBER	R	0x0	Jabber Detect Event Interrupt. Reading this bit clears the interrupt. 0x0 = No Jabber detect event pending 0x1 = Jabber detect event interrupt pending.
7	RESERVED	R	0x0	
6	ANERREN	R/W	0x0	Auto-Negotiation Error Interrupt Enable. 0x0 = Auto-Negotiation error event interrupt disabled. 0x1 = Enable interrupt on Auto-Negotiation error event
5	PAGERXEN	R/W	0x0	Page Receive Interrupt Enable. 0x0 = Page Receive error event interrupt disabled. 0x1 = Page receive event interrupt enabled.

Table 15-118. EPHYMISR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	LBFIFOEN	R/W	0x0	Loopback FIFO Overflow/Underflow Interrupt Enable. 0x0 = Loopback FIFO overflow/underflow event interrupt disabled 0x1 = Loopback FIFO overflow/underflow event interrupt enabled.
3	MDICOEN	R/W	0x0	MDI/MDIX Crossover Status Changed Interrupt Enable. 0x0 = MDI/MDIX Crossover Change Status interrupt disabled 0x1 = MDI/MDIX Crossover Change Status interrupt enabled.
2	SLEEPEN	R/W	0x0	Sleep Mode Event Interrupt Enable. 0x0 = Sleep mode interrupt disabled. 0x1 = Sleep Mode event interrupt enabled.
1	POLINTEN	R/W	0x0	Polarity Changed Interrupt Enable. 0x0 = Polarity Change interrupt is disabled. 0x1 = Polarity change interrupt is enabled.
0	JABBEREN	R	0x0	Jabber Detect Event Interrupt Enable. 0x0 = Jabber detect interrupt disabled. 0x1 = Jabber detect interrupt enabled.

15.7.19 EPHYFCSCR Register (Address = 0x14) [reset = 0x0]

Ethernet PHY False Carrier Sense Counter - MR20 (EPHYFCSCR)

This counter provides information required to implement the False Carriers attribute within the MAU managed object class of Clause 30 of the IEEE 802.3u specification.

EPHYFCSCR is shown in [Figure 15-107](#) and described in [Table 15-119](#).

Return to [Summary Table](#).

Figure 15-107. EPHYFCSCR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FCSCNT							
R-0x0								R-0x0							

Table 15-119. EPHYFCSCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0x0	
7-0	FCSCNT	R	0x0	False Carrier Event Counter. This 8-bit counter increments on every false carrier event. This counter stops when it reaches its maximum count (0xFF). When the counter exceeds half full (0x7F), an interrupt event is generated. This register is cleared on read.

15.7.20 EPHYRXERCNT Register (Address = 0x15) [reset = 0x0]

Ethernet PHY Receive Error Count - MR21 (EPHYRXERCNT)

This counter provides information required to implement the Symbol Error During Carrier attribute within the PHY managed object class of Clause 30 of the IEEE 802.3u specification.

EPHYRXERCNT is shown in [Figure 15-108](#) and described in [Table 15-120](#).

[Return to Summary Table.](#)

Figure 15-108. EPHYRXERCNT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXERRCNT															
R-0x0															

Table 15-120. EPHYRXERCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RXERRCNT	R	0x0	Receive Error Count. When a valid carrier is present (while EN0RXDV is active), and there is at least one occurrence of an invalid data symbol, this 16-bit counter increments for each receive error detected. The EPHYRXERCNT counter does not count in MII loopback mode. The counter stops when it reaches its maximum count of 0xFFFF. When the counter exceeds half-full (0x7FFF), an interrupt is generated. This register is cleared on read.

15.7.21 EPHYBISTCR Register (Address = 0x16) [reset = 0x100]

Ethernet PHY BIST Control - MR22 (EPHYBISTCR)

This register is used for Build-In Self Test (BIST) configuration. The BIST functionality provides Pseudo Random Bit Stream (PRBS) mechanism including packet generation generator and checker. Selection of the exact loopback point in the signal chain is also done in this register.

EPHYBISTCR is shown in [Figure 15-109](#) and described in [Table 15-121](#).

Return to [Summary Table](#).

Figure 15-109. EPHYBISTCR Register

15	14	13	12	11	10	9	8
RESERVED	PRBSM	PRBSPKT	PKTEN	PRBSCHKLK	PRBSCHKSYN C	PKTGENSTAT	PWRMODE
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R-0x0	R-0x0	R-0x1
7	6	5	4	3	2	1	0
RESERVED	TXMILB	RESERVED	LBMODE				
R-0x0	R-0x0	R-0x0	R/W-0x0				

Table 15-121. EPHYBISTCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0x0	
14	PRBSM	R/W	0x0	PRBS Single/Continuous Mode. 0x0 = Single mode selected. When BIST Error Counter reaches its max value, the PRBS checker stops counting. 0x1 = Continuous mode selected. When the PRBS counters reach maximum count value, the counter starts counting from zero again.
13	PRBSPKT	R/W	0x0	Generated PRBS Packets. 0x0 = When packet generator is enabled, generate a single packet with constant data. PRBS generation and check is disabled. 0x1 = When packet generator is enabled, generate continuous packets with PRBS data. When packet generator is disabled, PRBS checker is still enabled.
12	PKTEN	R/W	0x0	Packet Generation Enable. 0x0 = Disable packet generator 0x1 = Enable packet generation with PRBS data
11	PRBSCHKLK	R	0x0	PRBS Checker Lock Indication. 0x0 = PRBS checker is not locked 0x1 = PRBS checker is locked and synchronized on received bit stream
10	PRBSCHKSYNC	R	0x0	PRBS Checker Lock Sync Loss Indication. 0x0 = PRBS checker has not lost synchronization. 0x1 = PRBS checker has lost synchronization on received bit stream. This is an error indication.
9	PKTGENSTAT	R	0x0	Packet Generator Status Indication. 0x0 = Packet Generator is disabled. 0x1 = Packet Generator is active and generate packets.
8	PWRMODE	R	0x1	Power Mode Indication. 0x0 = Indicates that the PHY is in one of the sleep modes, either active or passive. 0x1 = Indicates that the PHY is in normal power mode.
7	RESERVED	R	0x0	

Table 15-121. EPHYBISTCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	TXMIILB	R	0x0	Transmit Data in MII Loopback Mode. 0x0 = Data is not transmitted to the line in MII loopback 0x1 = Enable transmission of the data from the MAC received on the TX pins to the line in parallel to the MII loopback to RX pins. This bit may be set only in MII Loopback mode, which is enabled by setting the MIILLOOPBK bit in the EPHYBMCR register, offset EPHY.0x00.
5	RESERVED	R	0x0	
4-0	LBMODE	R/W	0x0	Loopback Mode Select. The PHY provides several options for Loopback that test and verify various functional blocks within the PHY. 0x00 = Reserved 0x0 = Reserved 0x01 = Near-end loopback: PCS Input Loopback 0x02 = Near-end loopback: PCS Output Loopback (In 100Base-TX only) 0x03 = Reserved 0x04 = Near-end loopback: Digital Loopback 0x05 = Reserved 0x6 = Reserved 0x7 = Reserved 0x08 = Near-end loopback: Analog Loopback (requires 100-Ω termination) 0x09 = Reserved 0xA = Reserved 0xB = Reserved 0xC = Reserved 0xD = Reserved 0xE = Reserved 0xF = Reserved 0x10 = Far-end Loopback: Reverse Loopback

15.7.22 EPHYLEDCR Register (Address = 0x18) [reset = 0x400]

Ethernet PHY LED Control - MR24 (EPHYLEDCR)

This register provides the ability to control the blink rate on the LED outputs.

EPHYLEDCR is shown in [Figure 15-110](#) and described in [Table 15-122](#).

Return to [Summary Table](#).

Figure 15-110. EPHYLEDCR Register

15	14	13	12	11	10	9	8
RESERVED					BLINKRATE	RESERVED	
R-0x0					R/W-0x2	R-0x0	
7	6	5	4	3	2	1	0
RESERVED							
R-0x0							

Table 15-122. EPHYLEDCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0x0	
10-9	BLINKRATE	R/W	0x2	LED Blinking Rate (ON/OFF duration):. 0x0 = 20 Hz (50 ms) 0x1 = 10 Hz (100 ms) 0x2 = 5 Hz (200 ms) 0x3 = 2 Hz (500 ms)
8-0	RESERVED	R	0x0	

15.7.23 EPHYCTL Register (Address = 0x19) [reset = 0x8000]

Ethernet PHY Control - MR25 (EPHYCTL)

This register provides the ability to control and set general functionality inside the PHY.

EPHYCTL is shown in [Figure 15-111](#) and described in [Table 15-123](#).

Return to [Summary Table](#).

Figure 15-111. EPHYCTL Register

15	14	13	12	11	10	9	8
AUTOMDI	FORCEMDI	PAUSERX	PAUSETX	MIILNKSTAT	RESERVED		
R/W-0x1	R/W-0x0	R-0x0	R-0x0	R-0x0	R-0x0		
7	6	5	4	3	2	1	0
BYPLEDSTRCH	RESERVED						
R/W-0x0	R-0x0						

Table 15-123. EPHYCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	AUTOMDI	R/W	0x1	Auto-MDIX Enable. 0x0 = Disable Auto- negotiation, Auto-MDIX capability. 0x1 = Enable Auto-negotiation, Auto-MDIX capability.
14	FORCEMDI	R/W	0x0	Force MDIX. 0x0 = Normal operation. (Transmit on TPTD pair, Receive on TPRD pair) 0x1 = Force MDI pairs to cross. (Receive on TPTD pair, Transmit on TPRD pair)
13	PAUSERX	R	0x0	Pause Receive Negotiated Status. 0x0 = No effect. 0x1 = Indicates that pause receive should be enabled in the MAC. This bit is set based on bits [11:10] in the EPHYANA register and in the EPHYANLPAR register settings. This function is enabled according to IEEE 802.3 Annex 28B Table 28B-3, Pause Resolution, only if the Auto-Negotiated Highest Common Denominator is a full-duplex technology.
12	PAUSETX	R	0x0	Pause Transmit Negotiated Status. 0x0 = No effect. 0x1 = Indicates that pause transmit should be enabled in the MAC. This bit is set based on bits [11:10] in the EPHYANA register and in the EPHYANLPAR register settings. This function is enabled according to IEEE 802.3 Annex 28B Table 28B-3, Pause Resolution, only if the Auto-Negotiated Highest Common Denominator is a full-duplex technology.
11	MIILNKSTAT	R	0x0	MII Link Status. 0x0 = No active link of 100BT full-duplex has been established using Auto-Negotiation. 0x1 = 100BT Full-duplex Link is active and was established using Auto-Negotiation.
10-8	RESERVED	R	0x0	
7	BYPLEDSTRCH	R/W	0x0	Bypass LED Stretching. This bypasses LED stretching and allows the LEDs to reflect normal operation values. 0x0 = Normal LED operation. 0x1 = Bypass LED stretching.
6-0	RESERVED	R	0x0	

15.7.24 EPHY10BTSC Register (Address = 0x1A) [reset = 0x0]

Ethernet PHY 10Base-T Status/Control - MR26 (EPHY10BTSC)

This register provides the ability to control and read status of the PHY's internal 10Base-T functionality.

EPHY10BTSC is shown in [Figure 15-112](#) and described in [Table 15-124](#).

Return to [Summary Table](#).

Figure 15-112. EPHY10BTSC Register

15	14	13	12	11	10	9	8
RESERVED		RXTHEN	SQUELCH			RESERVED	
R-0x0		R/W-0x0	R/W-0x0			R-0x0	
7	6	5	4	3	2	1	0
NLPDIS	RESERVED		POLSTAT	RESERVED			JABBERD
R/W-0x0	R-0x0		R-0x0	R-0x0			R-0x0

Table 15-124. EPHY10BTSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0x0	
13	RXTHEN	R/W	0x0	Lower Receiver Threshold Enable. 0x0 = Normal 10Base-T operation. 0x1 = Enable 10Base-T lower receiver threshold to allow operation with longer cables
12-9	SQUELCH	R/W	0x0	Squelch Configuration. Used to set the Peak Squelch ON threshold for the 10Base-T receiver. Every value corresponds to a 50-mV increase. The squelch threshold range is from 200 mV to 600 mV. The default Squelch threshold is set to 200 mV. Values 0xA to 0xF are reserved.
8	RESERVED	R	0x0	
7	NLPDIS	R/W	0x0	Normal Link Pulse (NLP) Transmission Control. 0x0 = Enable transmission of Normal Link Pulses (NLPs). 0x1 = Disable transmission of Normal Link Pulses (NLPs).
6-5	RESERVED	R	0x0	
4	POLSTAT	R	0x0	10 Mb Polarity Status. This bit is a duplication of POLSTAT bit in the EPHYSTS register, offset 0x0010. Both bits are cleared on a read of the Ethernet PHY 10Base-T Status/Control (EPHY10BTSC), but not upon a read of the EPHYSTS register. 0x0 = Correct Polarity detected. 0x1 = Inverted Polarity detected.
3-1	RESERVED	R	0x0	
0	JABBERD	R	0x0	Jabber Disable. This function is applicable only in 10Base-T. 0x0 = Jabber function enabled. 0x1 = Jabber function disabled.

15.7.25 EPHYBICSR1 Register (Address = 0x1B) [reset = 0x7D]

Ethernet PHY BIST Control and Status 1 - MR27 (EPHYBICSR1)

This register provides the total number of error bytes that are received by the PRBS checker and defines the Inter-Packet Gap (IPG) for the packet generator.

EPHYBICSR1 is shown in [Figure 15-113](#) and described in [Table 15-125](#).

Return to [Summary Table](#).

Figure 15-113. EPHYBICSR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRCNT								IPGLENGTH							
R-0x0								R/W-0x7D							

Table 15-125. EPHYBICSR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	ERRCNT	R	0x0	BIST Error Count. This field holds the number of erroneous bytes that were received by the PRBS checker. The value in this register is locked when a write is done to bit 15. When the PRBSM field in the EPHYBISCR register (0x0016) is set to zero, the count stops at 0xFF. See the EPHYBISCR register for further details.
7-0	IPGLENGTH	R/W	0x7D	BIST IPG Length. Inter Packet Gap (IPG) Length defines the size of the gap (in bytes) between any 2 successive packets generated by the BIST. Default value is 0x7D, which is equal to 125 bytes.

15.7.26 EPHYBICSR2 Register (Address = 0x1C) [reset = 0x5EE]

Ethernet PHY BIST Control and Status 2 - MR28 (EPHYBICSR2)

This register allows programming the length of the generated packets in bytes for the BIST mechanism

EPHYBICSR2 is shown in [Figure 15-114](#) and described in [Table 15-126](#).

Return to [Summary Table](#).

Figure 15-114. EPHYBICSR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						PKTLENGTH									
R-0x0						R/W-0x5EE									

Table 15-126. EPHYBICSR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0x0	
10-0	PKTLENGTH	R/W	0x5EE	BIST Packet Length. The value of this register defines the size (in bytes) of every packet that is generated by the BIST. Default value is 0x5EE, which is equal to 1518 bytes.

15.7.27 EPHYCDCCR Register (Address = 0x1E) [reset = 0x102]

Ethernet PHY Cable Diagnostic Control - MR30 (EPHYCDCCR)

This register provides the ability to start cable diagnostics and monitor its status.

EPHYCDCCR is shown in [Figure 15-115](#) and described in [Table 15-127](#).

Return to [Summary Table](#).

Figure 15-115. EPHYCDCCR Register

15	14	13	12	11	10	9	8
START	RESERVED					LINKQUAL	
R/W-0x0	R-0x0					R-0x1	
7	6	5	4	3	2	1	0
RESERVED						DONE	FAIL
R-0x0						R-0x1	R-0x0

Table 15-127. EPHYCDCCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	START	R/W	0x0	Cable Diagnostic Process Start. Diagnostic Start bit is cleared with raise of Diagnostic Done indication. 0x0 = Cable Diagnostic is disabled 0x1 = Start cable diagnostic measurement
14-10	RESERVED	R	0x0	
9-8	LINKQUAL	R	0x1	Link Quality Indication. The value of these bits are valid only when link is active (when the LINK bit in the EPHYSTS register is a 0x1). 0x0 = Reserved 0x1 = Good Quality Link Indication 0x2 = Mid- Quality Link Indication 0x3 = Poor Quality Link Indication
7-2	RESERVED	R	0x0	
1	DONE	R	0x1	Cable Diagnostic Process Done. 0x0 = Diagnostic has not completed 0x1 = Indicates that the cable measurement process completed
0	FAIL	R	0x0	Cable Diagnostic Process Fail. 0x0 = Diagnostic has not failed 0x1 = Indicates that the cable measurement process failed

15.7.28 EPHYRCR Register (Address = 0x1F) [reset = 0x0]

Ethernet PHY Reset Control - MR31 (EPHYRCR)

This register allows the system to reset or restart the PHY by register access.

EPHYRCR is shown in [Figure 15-116](#) and described in [Table 15-128](#).

Return to [Summary Table](#).

Figure 15-116. EPHYRCR Register

15	14	13	12	11	10	9	8
SWRST	SWRESTART	RESERVED					
R/W-0x0	R/W-0x0	R-0x0					
7	6	5	4	3	2	1	0
RESERVED							
R-0x0							

Table 15-128. EPHYRCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SWRST	R/W	0x0	Software Reset. 0x0 = Normal Operation. 0x1 = Soft reset. This mode resets the digital portion of the PHY and all of the registers. This bit self clears after completion.
14	SWRESTART	R/W	0x0	Software Restart. 0x0 = Normal Operation. 0x1 = Restart PHY. This mode resets the digital portion of the PHY but no the registers. This bit self clears after completion of the restart.
13-0	RESERVED	R	0x0	

15.7.29 EPHYLED CFG Register (Address = 0x25) [reset = 0x510]

Ethernet PHY LED Configuration - MR37 (EPHYLED CFG)

This register configures the LEDs provided in the PHY.

EPHYLED CFG is shown in [Figure 15-117](#) and described in [Table 15-129](#).

Return to [Summary Table](#).

Figure 15-117. EPHYLED CFG Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				LED2				LED1				LED0			
R-0x0				R/W-0x5				R/W-0x1				R/W-0x0			

Table 15-129. EPHYLED CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0x0	
11-8	LED2	R/W	0x5	LED2 Configuration. The following encodings are used to program the specific function desired for the LED. 0x0 = Link OK 0x1 = RX/TX Activity 0x2 = TX Activity 0x3 = RX Activity 0x4 = Collision 0x5 = 100-Base TX 0x6 = 10-Base TX 0x7 = Full Duplex 0x8 = Link OK/Blink on TX/RX Activity 0x9 = Reserved 0xA = Reserved 0xB = Reserved 0xC = Reserved 0xD = Reserved 0xE = Reserved 0xF = Reserved
7-4	LED1	R/W	0x1	LED1 Configuration. The following encodings are used to program the specific function desired for the LED. 0x0 = Link OK 0x1 = RX/TX Activity 0x2 = TX Activity 0x3 = RX Activity 0x4 = Collision 0x5 = 100-Base TX 0x6 = 10-Base TX 0x7 = Full Duplex 0x8 = Link OK/Blink on TX/RX Activity 0x9 = Reserved 0xA = Reserved 0xB = Reserved 0xC = Reserved 0xD = Reserved 0xE = Reserved 0xF = Reserved

Table 15-129. EPHYLEDCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	LED0	R/W	0x0	<p>LED0 Configuration. The following encodings are used to program the specific function desired for the LED.</p> <p>0x0 = Link OK</p> <p>0x1 = RX/TX Activity</p> <p>0x2 = TX Activity</p> <p>0x3 = RX Activity</p> <p>0x4 = Collision</p> <p>0x5 = 100-Base TX</p> <p>0x6 = 10-Base TX</p> <p>0x7 = Full Duplex</p> <p>0x8 = Link OK/Blink on TX/RX Activity</p> <p>0x9 = Reserved</p> <p>0xA = Reserved</p> <p>0xB = Reserved</p> <p>0xC = Reserved</p> <p>0xD = Reserved</p> <p>0xE = Reserved</p> <p>0xF = Reserved</p>

External Peripheral Interface (EPI)

The External Peripheral Interface is a high-speed parallel bus for external peripherals or memory. It has several modes of operation to interface gluelessly to many types of external devices. Enhanced capabilities include μ DMA support, clocking control, and support for external FIFO buffers.

Topic	Page
16.1 Introduction	1087
16.2 EPI Block Diagram	1088
16.3 Functional Description	1088
16.4 Initialization and Configuration.....	1090
16.5 EPI Registers.....	1122

16.1 Introduction

The EPI has the following features:

- 8-, 16-, or 32-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from synchronous dynamic random access memory (SDRAM), synchronous random access memory (SRAM), and flash memory
- Blocking and nonblocking reads
- Offloads the processor from the timing details of the parallel interface through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for read and write
 - A read-channel request is asserted by programmable levels on the internal nonblocking read FIFO (NBRFIFO)
 - A write-channel request is asserted by empty on the internal write FIFO (WFIFO)

The EPI supports three primary functional modes: SDRAM mode, traditional host-bus mode, and general-purpose mode. The EPI module also provides a custom GPIO interface that enables fast parallel interfaces using a FIFO with speed control that uses an internal bit clock.

- SDRAM mode
 - Supports x16 (single data rate) SDRAM at up to 60 MHz
 - Supports low-cost SDRAMs up to 64MB (512 megabits)
 - Includes automatic refresh and access to all banks and rows
 - Includes a sleep or standby mode to keep contents active with minimal power draw
 - Multiplexed address/data interface for reduced pin count
- Host-bus mode
 - Traditional x8 and x16 MCU bus interface capabilities
 - Access to SRAM, NOR flash memory, and other devices, with up to 1MB of addressing in nonmultiplexed mode and 256MB in multiplexed mode (512MB in host-bus 16 mode with no byte selects)
 - Support for up to 512Mb PSRAM in quad chip select mode, with dedicated configuration register read and write enable
 - Support of both muxed and demuxed address and data
 - Access to a range of devices supporting the nonaddress FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) empty and full signals
 - Speed controlled, with read and write data wait-state counters
 - Support for read or write burst mode to host bus
 - Multiple chip select modes including single, dual, and quad chip selects, with and without ALE
 - External iRDY signal provided for stall capability of reads and writes
 - Manual chip-enable (or use extra address pins)
- General-purpose mode
 - Wide parallel interfaces for fast communications with CPLDs and FPGAs
 - Data widths up to 32 bits
 - Data rates up to 150 MB/second
 - Optional "address" sizes from 4 bits to 20 bits
 - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
 - 1 to 32 bits, FIFOed with speed control
 - Useful for custom peripherals or for digital data acquisition and actuator controls

16.2 EPI Block Diagram

Figure 16-1 shows the block diagram of the EPI module.

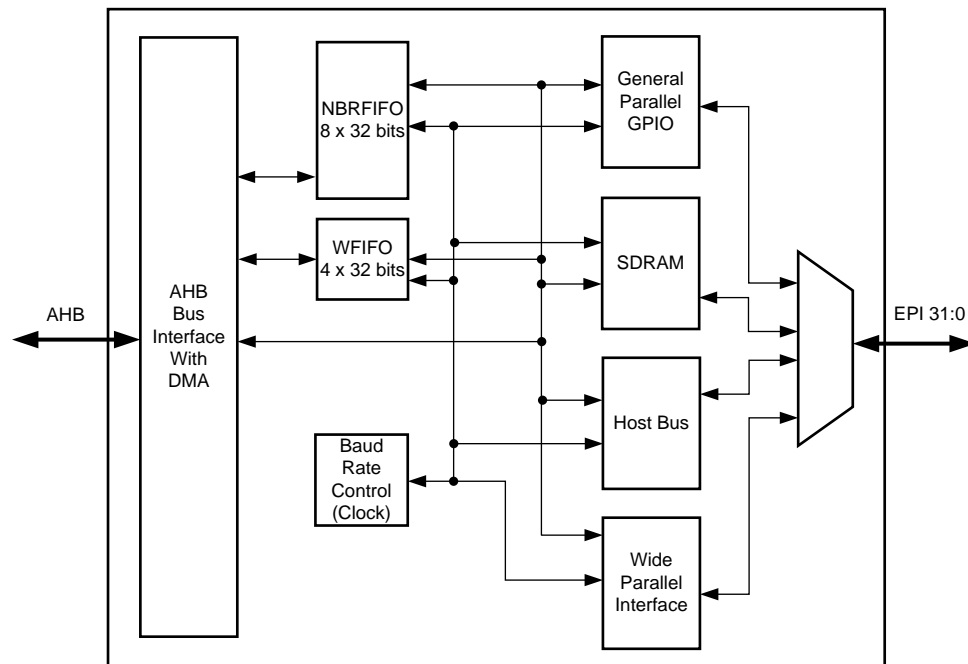


Figure 16-1. EPI Block Diagram

16.3 Functional Description

The EPI controller provides a glueless, programmable interface to a variety of common external peripherals such as SDRAM x16, host bus x8 and x16 devices, RAM, NOR flash memory, CPLDs, and FPGAs. In addition, the EPI controller provides custom GPIOs that can provide a fast speed-controlled parallel interface using either the internal write FIFO (WFIFO) or the nonblocking read FIFO (NBRFIFO) based on the amount of data in the FIFO.

The WFIFO can hold four words of data that are written to the external interface at the rate controlled by the EPI Main Baud Rate (EPIBAUD) registers. The NBRFIFO can hold 8 words of data and samples at the rate controlled by the EPIBAUD register. The EPI controller provides predictable operation and thus has an advantage over regular GPIOs, which have more variable timing due to on-chip bus arbitration and delays across bus bridges. Blocking reads stall the CPU until the transaction completes. Nonblocking reads are performed in the background and allow the processor to continue operation. In addition, write data can also be stored in the WFIFO to allow multiple writes with no stalls.

NOTE: Poll both the WTAV bit field in the EPIWFIFOCNT register and the WBUSY bit in the EPISTAT register to determine if there is a current write transaction from the WFIFO. If both of these bits are clear, then a new bus access may begin.

Main read and write operations can be performed in subsets of the range 0x6000.0000 to 0xDFFF.FFFF. A read from an address-mapped location uses the offset and size to control the address and size of the external operation. When performing a multiple-value load, the read is done as a burst (when available) to maximize performance. A write to an address-mapped location uses the offset and size to control the address and size of the external operation. When performing a multiple-value store, the write is done as a burst (when available) to maximize performance.

16.3.1 Master Access to EPI

These bus masters have access to the EPI:

- CPU
- μ DMA
- LCD

16.3.2 Nonblocking Reads

The EPI Controller supports a special kind of read called a nonblocking read, also referred to as a posted read. Where a normal read stalls the processor or μ DMA until the data is returned, a nonblocking read is performed in the background.

A nonblocking read is configured by writing the start address into a EPIRADDRn register, the size per transaction into a EPIRSIZEn register, and then the count of operations into a EPIRPSTDn register. After each read is completed, the result is written into the NBRFIFO and the EPIRADDRn register is incremented by the size (1, 2, or 4). The three most significant bits of the EPIRADDRn register are only relevant in the host bus multi-chip select mode when they are used to enable the different chip selects.

If the NBRFIFO is filled, then the reads pause until space is made available. The NBRFIFO can be configured to interrupt the processor or trigger the μ DMA based on the amount of data in the FIFO using the EPIFIFOLVL register. By using the trigger and interrupt method, the μ DMA (or processor) can keep space available in the NBRFIFO and allow the reads to continue unimpeded.

When performing nonblocking reads, the SDRAM controller issues two additional read transactions after the burst request is terminated. The data for these additional transfers is discarded. This situation is transparent to the user other than the additional EPI bus activity and can safely be ignored.

Two nonblocking read register sets are available to allow sequencing and ping-pong use. When one completes, the other then activates. So, for example, if 20 words are to be read from 0x100 and 10 words from 0x200, the EPIRPSTD0 register can be set up with the read from 0x100 (with a count of 20), and the EPIRPSTD1 register can be set up with the read from 0x200 (with a count of 10). When EPIRPSTD0 finishes (count goes to 0), the EPIRPSTD1 register then starts its operation. The NBRFIFO has then passed 30 values. When used with the μ DMA, it may transfer 30 values (simple sequence), or the primary/alternate model may be used to handle the first 20 in one way and the second 10 in another. It is also possible to reload the EPIRPSTD0 register when it is finished (and the EPIRPSTD1 register is active); thereby, keeping the interface constantly busy.

To cancel a nonblocking read, the EPIRPSTDn register is cleared. Care must be taken, however if the register set was active to drain away any values read into the NBRFIFO and ensure that any read in progress is allowed to complete.

To ensure that the cancel is complete, the following algorithm is used (using the EPIRPSTD0 register for example):

```
EPIRPSTD0 = 0;
while ((EPISTAT & 0x11) == 0x10)
; // we are active and busy
// if here, then other one is active or interface no longer busy
cnt = (EPIRADDR0 - original_address) / EPIRSIZE0; // count of values read
cnt -= values_read_so_far;
// cnt is now number left in FIFO
while (cnt-->0)
value = EPIREADFIFO; // drain
ISB
```

The preceding algorithm can be optimized in code; however, the important point is to wait for the cancel to complete, because the external interface could have been in the process of reading a value when the cancel came in, and it must be allowed to complete.

16.3.3 DMA Operation

The μ DMA can be used to achieve maximum transfer rates on the EPI through the NBRFIFO and the WFIFO. The μ DMA has one channel for write and one for read. For writes, the EPI DMA Transmit Count (EPIDMATXCNT) register is programmed with the total number of transfers by the μ DMA. An equivalent value is programmed into the DMA Channel Control Word (DMACHCTL) register of the μ DMA at offset 0x008. A μ DMA request is asserted by the EPI WRFIFO when the TXCNT value of the EPIDMATXCNT register is greater than zero and the WTAV bit field of the EPIWFIFOCNT register is less than the programmed threshold trigger, WRFIFO, of the EPIFIFOLVL register. The write channel continues to write data until the TXCNT value in the EPIDMATXCNT register is zero.

NOTE: When the WRFIFO bit in the EPIFIFOLVL register is set to 0x4 and the application bursts four words to an empty FIFO, the WRFIFO trigger may or may not deassert depending on if all four words were written to the WRFIFO or if the first word was passed immediately to the function requiring it. Thus, the application may not see the WRRIS bit in the EPIRIS register clear on a burst of four words.

The nonblocking read channel copies values from the NBRFIFO when the NBRFIFO is at the level specified by the EPIFIFOLVL register. For nonblocking reads, the start address, the size per transaction, and the count of elements must be programmed in the μ DMA. Both nonblocking read register sets can be used, and they fill the NBRFIFO such that one runs to completion, then the next one starts (they do not interleave). Using the NBRFIFO provides the best possible transfer rate.

For blocking reads, the μ DMA software channel (or another unused channel) is used for memory-to-memory transfers (or memory to peripheral, where some other peripheral is used). In this situation, the μ DMA stalls until the read is complete and is not able to service another channel until the read is done. As a result, the arbitration size should normally be programmed to one access at a time. The μ DMA controller can also transfer from and to the NBRFIFO and the WFIFO using the μ DMA software channel in memory mode, however, the μ DMA is stalled once the NBRFIFO is empty or the WFIFO is full. When the μ DMA controller is stalled, the core continues operation. For more information on configuring the μ DMA, see [Chapter 8](#).

The size of the FIFOs must be taken into consideration when configuring the μ DMA to transfer data to and from the EPI. The arbitration size should be 4 or less when writing to EPI address space and 8 or less when reading from EPI address space.

16.4 Initialization and Configuration

To enable and initialize the EPI controller, the following steps are necessary:

1. Enable the EPI module using the RCGCEPI register, see [Section 4.2.89](#).
2. Enable the clock to the appropriate GPIO module using the RCGCGPIO registers (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet.
3. Set the GPIO AFSEL bits for the appropriate pins, see [Section 17.5.10](#). To determine which GPIOs to configure, see the device-specific data sheet.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected, see [Section 17.5.11](#) and [Section 17.5.17](#).
5. Configure the PMCN fields in the GPIOPCTL register to assign the EPI signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).
6. Select the mode for the EPI block to SDRAM, HB8, HB16, or general parallel use, using the MODE field in the EPI Configuration (EPICFG) register. Set the mode-specific details (if needed) using the appropriate mode configuration EPI Host Bus Configuration (EPIHBnCFGn) registers for the desired chip-select configuration. Set the EPI Main Baud Rate (EPIBAUD) and EPI Main Baud Rate 2 (EPIBAUD2) register if the baud rate must be slower than the system clock rate.
7. Configure the address mapping using the EPI Address Map (EPIADDRMAP) register. The selected start address and range is dependent on the type of external device and maximum address (as appropriate). For example, for a 512-megabit SDRAM, program the ERADR field to 0x1 for address 0x6000.0000 or 0x2 for address 0x8000.0000; and program the ERSZ field to 0x3 for 256MB. If using General-Purpose mode and no address at all, program the EPADR field to 0x1 for address

- 0xA000.0000 or 0x2 for address 0xC000.0000; and program the EPSZ field to 0x0 for 256 bytes.
8. To read or write directly, use the mapped address area (configured with the EPIADDRMAP register). Up to 4 or 5 writes can be performed at once without blocking. Each read is blocked until the value is retrieved.
 9. To perform a nonblocking read, see [Section 16.3.2](#).

NOTE: The application should not attempt external access until eight system clock cycles after the EPI has been fully configured.

NOTE: When a MODE field has been programmed in the EPICFG register, the application should reset all configuration registers before programming to a new MODE value.

The following subsections describe the initialization and configuration for each of the modes of operation. Initialize everything properly to ensure correct operation. Control of the GPIO states is also important, as changes may cause the external device to interpret pin states as actions or commands (see). Normally, a pullup or pulldown is needed on the board to at least control the chip-select or chip-enable as the GPIOs come out of reset in high-impedance.

16.4.1 EPI Interface Options

There are a variety of memories and peripherals that can interface to the EPI module. [Table 16-1](#) shows the various configurations with their maximum performance.

Table 16-1. EPI Interface Options

Interface	Maximum Frequency
Single SDRAM	60 MHz
Single SRAM	60 MHz
Single PSRAM without iRDY signal use	55 MHz
Single PSRAM with iRDY signal use	52 MHz
FPGAs, CPLDs, and others using general-purpose mode	60 MHz
Memory configurations with 2 chip selects	40 MHz
Memory configurations with 4 chip selects	20 MHz

16.4.2 SDRAM Mode

When activating the SDRAM mode, it is important to consider a few points:

- Generally, it takes over 100 μ s from when the mode is activated to when the first operation is allowed. The SDRAM controller begins the SDRAM initialization sequence as soon as the mode is selected and enabled via the EPICFG register. It is important that the GPIOs are properly configured before the SDRAM mode is enabled, as the EPI controller is relying on the GPIO block's ability to drive the pins immediately. As part of the initialization sequence, the LOAD MODE REGISTER command is automatically sent to the SDRAM with a value of 0x27, which sets a CAS latency of 2 and a full page burst length.
- The INITSEQ bit in the EPI Status (EPISTAT) register can be checked to determine when the initialization sequence is complete.
- When using a frequency range and/or refresh value other than the default value, it is important to configure the FREQ and RFSH fields in the EPI SDRAM Configuration (EPISDRAMCFG) register shortly after activating the mode. After the 100- μ s startup time, the EPI block must be configured properly to keep the SDRAM contents stable.
- The SLEEP bit in the EPISDRAMCFG register may be configured to put the SDRAM into a low-power self-refreshing state. It is important to note that the SDRAM mode must not be disabled once enabled, or else the SDRAM is no longer clocked and the contents are lost.
- Before entering SLEEP mode, make sure all nonblocking reads and normal reads and writes have

completed. If the system is running at 30 to 50 MHz, wait 2 EPI clock cycles after clearing the SLEEP bit before executing nonblocking reads, or normal reads and writes. If the system is configured to greater than 50 MHz, wait 5 EPI clock cycles before read and write transactions. For all other configurations, wait 1 EPI clock.

The SIZE field of the EPISDRAMCFG register must be configured correctly based on the amount of SDRAM in the system.

The FREQ field must be configured according to the value that represents the range being used. Based on the range selected, the number of external clocks used between certain operations (for example, PRECHARGE or ACTIVATE) is determined. If a higher frequency is given than is used, then the only downside is that the peripheral is slower (uses more cycles for these delays). If a lower frequency is given, incorrect operation occurs.

For timing details for the SDRAM mode, see the device-specific data sheet.

16.4.2.1 External Signal Connections

Table 16-2 defines how EPI module signals should be connected to SDRAMs. The table applies when using a x16 SDRAM up to 512 megabits. The EPI signals must use 8-mA drive when interfacing to SDRAM, see Section 17.5.13. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 16-2. EPI SDRAM x16 Signal Connections

EPI Signal	SDRAM Signal ⁽¹⁾	
EPI0S0	A0	D0
EPI0S1	A1	D1
EPI0S2	A2	D2
EPI0S3	A3	D3
EPI0S4	A4	D4
EPI0S5	A5	D5
EPI0S6	A6	D6
EPI0S7	A7	D7
EPI0S8	A8	D8
EPI0S9	A9	D9
EPI0S10	A10	D10
EPI0S11	A11	D11
EPI0S12	A12 ⁽²⁾	D12
EPI0S13	BA0	D13
EPI0S14	BA1	D14
EPI0S15	D15	
EPI0S16	DQML	
EPI0S17	DQMH	
EPI0S18	CASn	
EPI0S19	RASn	
EPI0S20-EPI0S27	Not used	
EPI0S28	WEn	
EPI0S29	CSn	
EPI0S30	CKE	
EPI0S31	CLK	

⁽¹⁾ If two signals are listed, connect the EPI signal to both pins.

⁽²⁾ Only for 256- or 512-megabit SDRAMs

16.4.2.2 Refresh Configuration

The refresh count is based on the external clock speed and the number of rows per bank as well as the refresh period. The RFSH field represents how many external clock cycles remain before an AUTO-REFRESH is required. The normal formula is:

$$\text{RFSH} \leq (t_{\text{Refresh_us}} / \text{number_rows}) / \text{ext_clock_period_}\mu\text{s}$$

A refresh period is normally 64 ms, or 64000 μs . The number of rows is normally 4096 or 8192. The ext_clock_period is a value expressed in μs and is derived by dividing 1000 by the clock speed expressed in MHz. So, 50 MHz is $1000 / 50 = 20$ ns, or 0.02 μs . A typical SDRAM is 4096 rows per bank if the system clock is running at 50 MHz with an EPIBAUD register value of 0:

$$\text{RFSH} = (64000 / 4096) / 0.02 = 15.625 \mu\text{s} / 0.02 \mu\text{s} = 781.25.$$

The default value in the RFSH field is 750 decimal or 0x2EE to allow for a margin of safety and providing 15 μs per refresh. It is important to note that this number should always be smaller or equal to what is required by the above equation. For example, if running the external clock at 25 MHz (40 ns per clock period), 390 is the highest number that may be used. The external clock may be 25 MHz when the system clock is 25 MHz or when the system clock is 50 MHz and configuring the COUNT0 field in the EPIBAUD register to 1 (divide by 2).

If a number larger than allowed is used, the SDRAM is not refreshed often enough, and data is lost.

16.4.2.3 Bus Interface Speed

The EPI Controller SDRAM interface can operate up to 60 MHz. The COUNT0 field in the EPIBAUD register configures the speed of the EPI clock. For system clock (SysClk) speeds up to 60 MHz, the COUNT0 field can be 0x0000, and the SDRAM interface can run at the same speed as SysClk. However, if SysClk is running at higher speeds, the bus interface can run only as fast as half speed, and the COUNT0 field must be configured to at least 0x0001.

16.4.2.4 Nonblocking Read Cycle

Figure 16-2 shows a nonblocking read cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPIOS[15:0] signals. With the programmed CAS latency of 2, the Read command with the column address on the EPIOS[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPIOS[15:0] signals on every rising clock edge. The Burst Terminate command is issued during the cycle when the next-to-last halfword is read in. The DQMH and DQML signals are deasserted after the last halfword of data is received; the CSn signal deasserts on the following clock cycle, signaling the end of the read cycle. At least one clock period of inactivity separates any two SDRAM cycles.

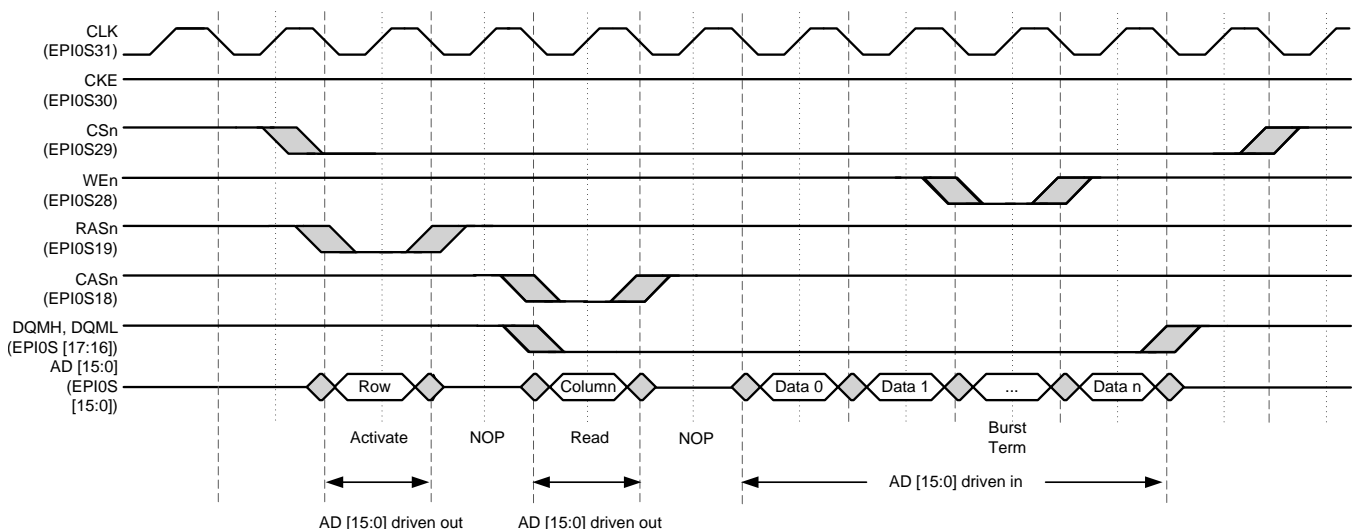


Figure 16-2. SDRAM Nonblocking Read Cycle

16.4.2.5 Normal Read Cycle

Figure 16-3 shows a normal read cycle of n halfwords; n can be 1 or 2. The cycle begins with the Activate command and the row address on the EPIOS[15:0] signals. With the programmed CAS latency of 2, the Read command with the column address on the EPIOS[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPIOS[15:0] signals on every rising clock edge. The DQMH, DQML, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the cycle. At least one clock period of inactivity separates any two SDRAM cycles.

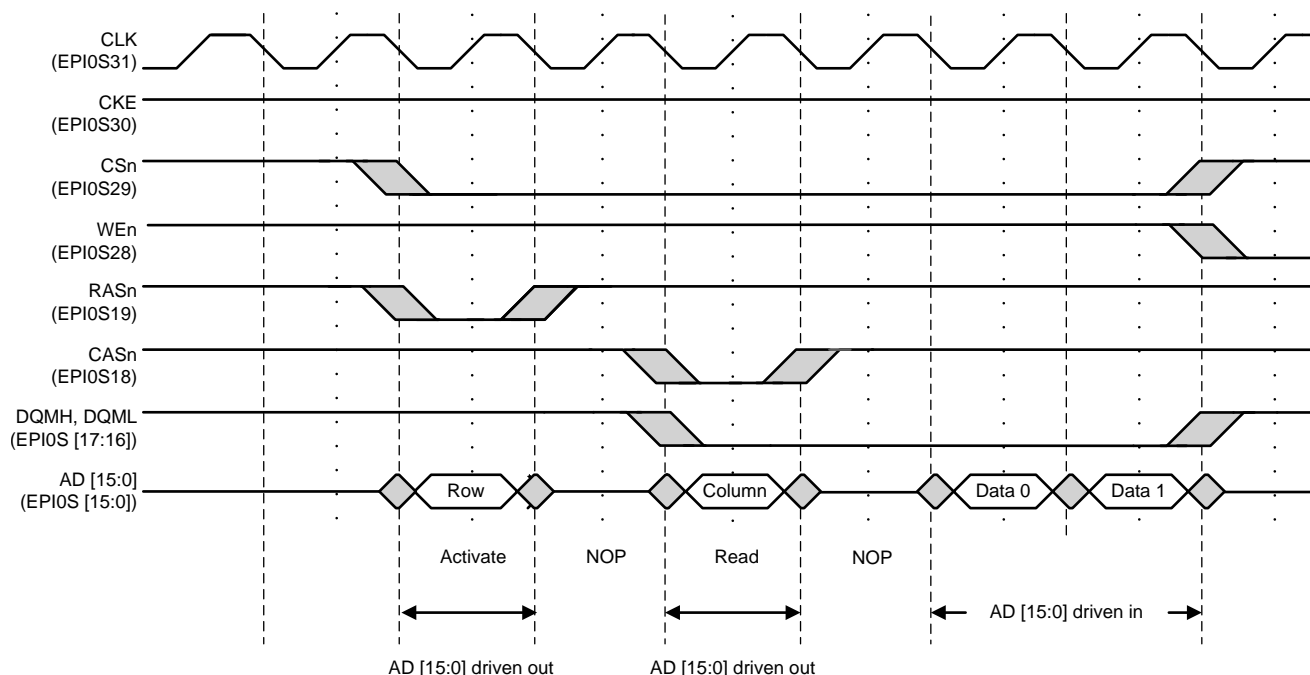


Figure 16-3. SDRAM Normal Read Cycle

16.4.2.6 Write Cycle

Figure 16-4 shows a write cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPIOS[15:0] signals. With the programmed CAS latency of 2, the Write command with the column address on the EPIOS[15:0] signals follows after 2 clock cycles. When writing to SDRAMs, the Write command is presented with the first halfword of data. Because the address lines and the data lines are multiplexed, the column address is modified to be (programmed address -1). During the Write command, the DQMH and DQML signals are high, so no data is written to the SDRAM. On the next clock, the DQMH and DQML signals are asserted, and the data associated with the programmed address is written. The Burst Terminate command occurs during the clock cycle following the write of the last halfword of data. The WEn, DQMH, DQML, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the access. At least one clock period of inactivity separates any two SDRAM cycles.

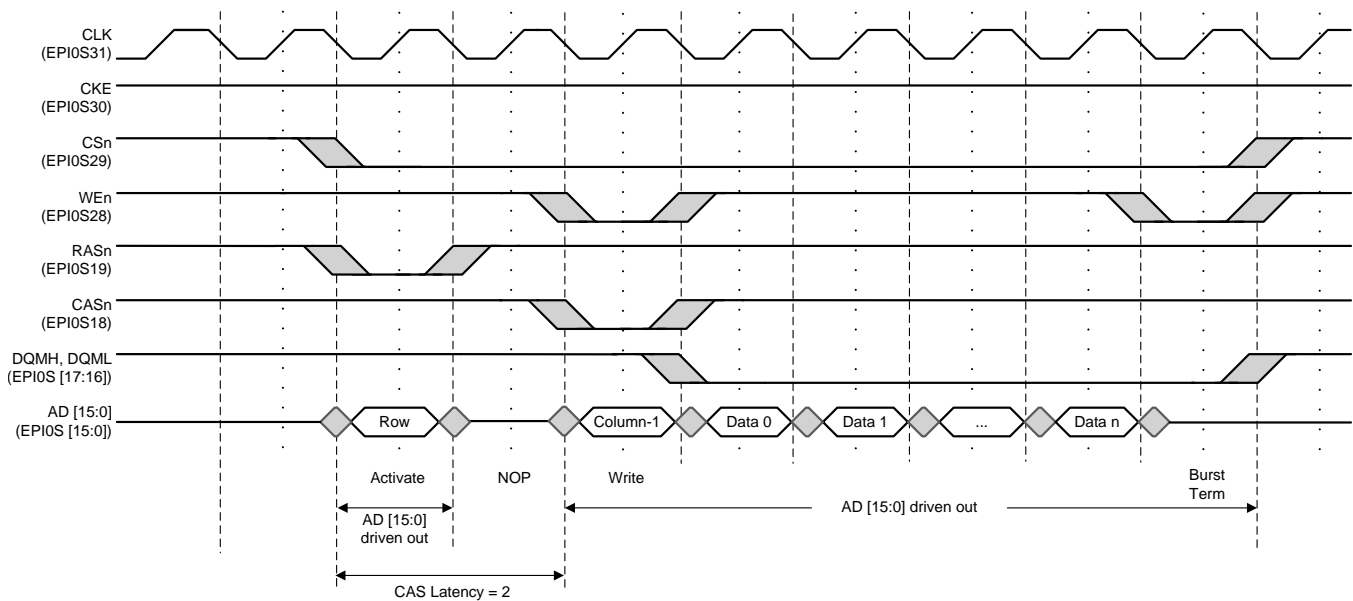


Figure 16-4. SDRAM Write Cycle

16.4.3 Host Bus Mode

Host Bus supports the traditional 8-bit and 16-bit interfaces popularized by the 8051 devices and SRAM devices, as well as PSRAM and NOR Flash memory. This interface is asynchronous and uses strobe pins to control activity. Addressable memory can be doubled using Host Bus-16 mode as it performs halfword accesses. The EPIOS0 is the LSB of the address and is equivalent to the internal Cortex-M4 A1 address. EPIOS0 should be connected to A0 of 16-bit memories.

16.4.3.1 Control Pins

The main three strobes are Address Latch Enable (ALE), Write (WRn), and Read (RDn, sometimes called OEn). The polarity of the read and write strobes can be active-high or active-low by clearing or setting the RDHIGH and WRHIGH bits in the EPI Host-Bus n Configuration (EPIHBnCFGn) register.

The ALE can be changed to an active-low chip select signal, CSn, through the EPIHBnCFGn register. The ALE is best used for Host-Bus muxed mode in which EPI address and data pins are shared. All Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold it until the data phase. The polarity of the ALE can be active High or Low by clearing or setting the ALEHIGH bit in the EPI Host-Bus n Configuration (EPIHBnCFGn) register. CSn is best used for Host-Bus unmuxed mode in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring. Both the ALE and

the CSn modes can be enhanced to access four external devices using settings in the EPIHBnCFGn register. PSRAM accesses must use both ALE and CSn. Wait states can be added to the data phase of the access using the WRWS and RDWS bits in the EPIHBnCFGn register. Additionally, within these wait state options, the WRWSM and RDWSM bit of the EPIHBnTIMEn register can be set to reduce the given wait states by 1 EPI clock cycle for finer granularity.

For FIFO mode, the ALE is not used, and two input holds are optionally supported to gate input and output to what the XFIFO can handle. FIFO mode is only applicable in EPI asynchronous mode.

Host-bus 8 and host-bus 16 modes are very configurable. The user has the ability to connect 1, 2, or 4 external devices to the EPI signals, as well as control whether byte select signals are provided in HB16 mode. These capabilities depend on the configuration of the MODE field in the EPIHBnCFG register, the CSCFG field and the CSCFGEXT bit in the EPIHBnCFGn register, and the BSEL bit in the EPIHB16CFG register. The CSCFGEXT bit extends the chip select configuration possibilities by providing the most significant bit of the CSCFGEXT field. For the possible ALE and chip select options that can be programmed by the combination of the CSCFGEXT and CSCFGEXT bits, see [Table 16-3](#). CSCFGEXT is the most significant bit.

Table 16-3. CSCFGEXT and CSCFG Encodings

Value	Description
0x0	ALE configuration EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the EPIHB8CFG register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.
0x1	CSn Configuration EPI0S30 is used as a chip select (CSn). When using this mode, the address and data are generally not muxed (MODE field in the EPIHB8CFG register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low.
0x2	Dual CSn configuration EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices and when using both an external RAM and an external peripheral.
0x3	ALE with dual CSn configuration EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.
0x4	ALE with single CSn configuration EPI0S30 is used as address latch (ALE) and EPI0S27 is used as CSn.
0x5	Quad CSn configuration EPI0S30 is used as CS0n, EPI0S27 is used as CS1n, EPI0S34 is used as CS2n, and EPI0S33 is used as CS3n.
0x6	ALE with quad CSn configuration EPI0S30 is used as ALE, EPI0S26 is used as CS0n, EPI0S27 is used as CS1n, EPI0S34 is used as CS2n, and EPI0S33 is used as CS3n.
0x7	Reserved

If one of the dual-chip-select modes is selected (CSCFGEXT is 0x0 and CSCFG is 0x2 or 0x3 in the EPIHBnCFGn register), both chip selects can share the peripheral, code, or the memory space, or one chip select can use the peripheral space and the other can use the memory or code space. In the EPIADDRMAP register, if the EPADR field is not 0x0, the ECADR field is 0x0, and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with CS0n being asserted when the MSB of the address range is 0 and CS1n being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0, the ECADR field is 0x0, and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0, and the ECADR field is 0x0 and the EPI is configured for dual-chip selects, then CS0n is asserted for either address range defined by EPADR and CS1n is asserted for either address range defined by ERADR. The two chip selects can also be shared between the code space and memory or peripheral space. If the ECADR field is 0x1, ERADR field is 0x0, and the EPADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by EPADR. If the ECADR field is 0x1, EPADR field is 0x0, and the ERADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by ERADR.

In quad chip select mode (CSCFGEXT is 0x1 and CSCFG is 0x1 or 0x2 in the EPIHBnCFG2 register), both the peripheral and the memory space must be enabled. In the EPIADDRMAP register, the EPADR field is 0x3, the ERADR field is 0x3, and the ECADR field is 0x0. With this configuration, CS0n asserts for the address range beginning at 0x6000.0000, CS1n asserts for 0x8000.0000, CS2n for 0xA000.0000, and CS3n for 0xC000.0000. [Table 16-4](#) gives a detailed explanation of chip select address range mappings based on combinations of enabled peripheral and memory space.

NOTE: Only one memory area can be mapped to a single chip select. Enabling multiple memory areas for one chip select may produce unexpected results.

Table 16-4. Dual- and Quad- Chip Select Address Mappings

Chip Select Mode	ERADR	EPADR	ECADR	CS0 ⁽¹⁾	CS1	CS2	CS3
Dual-chip select	0x0	0x1 or 0x2	0x0	EPADR defined address range (0xA000.000 or 0xC000.0000)	EPADR defined address range (0xA000.000 or 0xC000.0000)	N/A	N/A
Dual-chip select	0x1 or 0x2	0x0	0x0	ERADR defined address range (0x6000.000 or 0x8000.000)	ERADR defined address range (0x6000.000 or 0x8000.000)	N/A	N/A
Dual-chip select	0x1 or 0x2	0x1 or 0x2	0x0	EPADR defined address range (0xA000.000 or 0xC000.0000)	ERADR defined address range (0x6000.000 or 0x8000.000)	N/A	N/A
Dual-chip select	0x0	0x1 or 0x2	0x1	ECADR defined address range (0x1000.000)	EPADR defined address range (0xA000.0000 or 0xC000.0000)	N/A	N/A
Dual-chip select	0x1 or 0x2	0x0	0x1	ECADR defined address range (0x1000.000)	ERADR defined address range (0x6000.000 or 0x8000.000)	N/A	N/A
Quad-chip select	0x3	0x3	0x0	0x6000.0000	0x8000.0000	0xA000.0000	0xC000.0000

⁽¹⁾ When CS0 and CS1 share address space, CS0 asserts when the MSB of the address is 0 and CS1, when the MSB of the address is 1.

The MODE field of the EPIHBnCFGn registers configure the interface for the chip selects, which support ADMUX or ADNOMUX. See [Table 16-5](#) for details on which configuration register controls each chip select. If the CSBAUD bit is clear, all chip selects are configured by the MODE bit field of the EPIHBnCFG register.

If the CSBAUD bit in the EPIHBnCFG2 register is set in Dual-chip select mode, the 2 chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, both chip selects use the clock frequency, wait states, and strobe polarity defined for CS0n. Additionally, if the CSBAUD bit is set, the two chip selects can use different interface modes. If any interface modes are programmed to ADMUX, then dual chip select mode must include the ALE capability. In quad chip select mode, if the CSBAUD bit in the EPIHBnCFG2 register is set, the 4 chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, all chip selects use the clock frequency, wait states, and strobe polarity defined for CS0n. If the CSBAUD bit is set, the four chip selects can use different interface modes.

Table 16-5. Chip Select Configuration Register Assignment

Configuration Register ⁽¹⁾	Corresponding Chip Select
EPIHBnCFG	CS0n
EPIHBnCFG2	CS1n
EPIHBnCFG3	CS2n
EPIHBnCFG4	CS3n

⁽¹⁾ If the CSBAUD bit in the EPIHBnCFG2 register is clear and multiple chip selects are enabled, then all chip selects are configured by the MODE bit field in the EPIHBnCFG register.

Multiple chip select modes do not allow the intermixing of Host-Bus 8 and Host-Bus16 modes.

When BSEL = 1 in the EPIHB16CFG register, byte select signals are provided, so byte-sized data can be read and written at any address, however these signals reduce the available address width by 2 pins. The byte select signals are active Low. BSEL0n corresponds to the LSB of the halfword, and BSEL1n corresponds to the MSB of the halfword.

When BSEL = 0, byte reads and writes at odd addresses only act on the even byte, and byte writes at even addresses write invalid values into the odd byte. As a result, accesses should be made as halfwords (16-bits) or words (32-bits). In C/C++, programmers should use only short int and long int for accesses. Also, because data accesses in HB16 mode with no byte selects are on 2-byte boundaries, the available address space is doubled. For example, 28 bits of address accesses 512MB in this mode. [Table 16-6](#) shows the capabilities of the HB8 and HB16 modes as well as the available address bits with the possible combinations of these bits.

Although the EPIOS31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system.

Table 16-6. Capabilities of Host Bus 8 and Host Bus 16 Modes

Host Bus Type	MODE	CSCFGEXT	CSCFG	Maximum No. of External Devices	BSEL	Byte Access	Available Address	Addressable Memory
HB8	0x0	0	0x0, 0x1	1	N/A	Always	28 bits	256MB
HB8	0x0	0	0x2	2	N/A	Always	27 bits	128MB
HB8	0x0	0	0x3	2	N/A	Always	26 bits	64MB
HB8	0x0	1	0x0	1	N/A	Always	27 bits	128MB
HB8	0x0	1	0x1	4	N/A	Always	27 bits	128MB
HB8	0x0	1	0x2	4	N/A	Always	26 bits	64MB
HB8	0x1	0	0x0, 0x1	1	N/A	Always	20 bits	1MB
HB8	0x1	0	0x2	2	N/A	Always	19 bits	512KB
HB8	0x1	0	0x3	2	N/A	Always	18 bits	256KB
HB8	0x1	1	0x0	1	N/A	Always	19 bits	512KB
HB8	0x1	1	0x1	4	N/A	Always	19 bits	512MB
HB8	0x1	1	0x2	4	N/A	Always	18 bits	256KB
HB8	0x2	0	0x1	1	N/A	Always	20 bits	1MB
HB8	0x3	0	0x1	1	N/A	Always	none	—
HB8	0x3	0	0x3	2	N/A	Always	none	—
HB8	0x3	1	0x0	1	N/A	Always	none	—
HB8	0x3	1	0x1	4	N/A	Always	none	—
HB8	0x3	1	0x2	4	N/A	Always	none	—
HB16	0x0	0	0x0, 0x1	1	0	No	28 bits ⁽¹⁾	512MB
HB16	0x0	0	0x0, 0x1	1	1	Yes	26 bits ⁽²⁾	128MB

⁽¹⁾ If byte selects are not used, data accesses are on 2-byte boundaries. As a result, the available address space is doubled.

⁽²⁾ Two EPI signals are used for byte selects, reducing the available address space by two bits.

Table 16-6. Capabilities of Host Bus 8 and Host Bus 16 Modes (continued)

Host Bus Type	MODE	CSCFGEXT	CSCFG	Maximum No. of External Devices	BSEL	Byte Access	Available Address	Addressable Memory
HB16	0x0	0	0x2	2	0	No	27 bits ⁽¹⁾	256MB
HB16	0x0	0	0x2	2	1	Yes	25 bits ⁽²⁾	64MB
HB16	0x0	0	0x3	2	0	No	26 bits ⁽¹⁾	128MB
HB16	0x0	0	0x3	2	1	Yes	24 bits ⁽²⁾	32MB
HB16	0x0	1	0x0	1	0	No	27 bits ⁽¹⁾	256MB
HB16	0x0	1	0x0	1	1	Yes	25 bits ⁽²⁾	128MB
HB16	0x0	1	0x1	4	0	No	27 bits ⁽¹⁾	256MB
HB16	0x0	1	0x1	4	1	Yes	25 bits ⁽²⁾	64MB
HB16	0x0	1	0x2	4	0	No	26 bits ⁽¹⁾	128MB
HB16	0x0	1	0x2	4	1	Yes	24 bits ⁽²⁾	32MB
HB16	0x1	0	0x0, 0x1	1	0	No	12 bits ⁽¹⁾	8KB
HB16	0x1	0	0x0, 0x1	1	1	Yes	10 bits ⁽²⁾	2KB
HB16	0x1	0	0x2	2	0	No	11 bits ⁽¹⁾	4KB
HB16	0x1	0	0x2	2	1	Yes	9 bits ⁽²⁾	1KB
HB16	0x1	0	0x3	2	0	No	10 bits ⁽¹⁾	2KB
HB16	0x1	0	0x3	2	1	Yes	8 bits ⁽²⁾	512 B
HB16	0x1	1	0x0	1	0	No	11 bits ⁽¹⁾	4KB
HB16	0x1	1	0x0	1	1	Yes	9 bits ⁽²⁾	1KB
HB16	0x1	1	0x1	4	0	No	11 bits ⁽¹⁾	4KB
HB16	0x1	1	0x1	4	1	Yes	9 bits ⁽²⁾	1KB
HB16	0x1	1	0x2	4	0	No	10 bits ⁽¹⁾	2KB
HB16	0x1	1	0x2	4	1	Yes	8 bits ⁽²⁾	512 B
HB16	0x3	0	0x1	1	0	No	none	—
HB16	0x3	0	0x1	1	1	Yes	none	—
HB16	0x3	0	0x3	2	0	No	none	—
HB16	0x3	0	0x3	2	1	Yes	none	—
HB16	0x3	1	0x0	1	0	No	none	—
HB16	0x3	1	0x0	1	1	Yes	none	—
HB16	0x3	1	0x1	4	0	No	none	—
HB16	0x3	1	0x1	4	1	Yes	none	—
HB16	0x3	1	0x2	4	0	No	none	—
HB16	0x3	1	0x2	4	1	Yes	none	—

Table 16-7 shows how the EPI[31:0] signals function while in Host-Bus 8 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the EPIHB8CFGn register and on the chip select configuration selected by the CSCFG and CSCFGEXT field in the EPIHB8CFG2 register.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 16-7. EPI Host-Bus 8 Signal Connections

EPI Signal	CSCFG	HB8 Signal (MODE = ADMUX)	HB8 Signal (MODE = ADNOMUX (Cont. Read))	HB8 Signal (MODE = XFIFO)
EPI0S0	X ⁽¹⁾	AD0	D0	D0
EPI0S1	X	AD1	D1	D1
EPI0S2	X	AD2	D2	D2
EPI0S3	X	AD3	D3	D3
EPI0S4	X	AD4	D4	D4
EPI0S5	X	AD5	D5	D5
EPI0S6	X	AD6	D6	D6
EPI0S7	X	AD7	D7	D7
EPI0S8	X	A8	A0	–
EPI0S9	X	A9	A1	–
EPI0S10	X	A10	A2	–
EPI0S11	X	A11	A3	–
EPI0S12	X	A12	A4	–
EPI0S13	X	A13	A5	–
EPI0S14	X	A14	A6	–
EPI0S15	X	A15	A7	–
EPI0S16	X	A16	A8	–
EPI0S17	X	A17	A9	–
EPI0S18	X	A18	A10	–
EPI0S19	X	A19	A11	–
EPI0S20	X	A20	A12	–
EPI0S21	X	A21	A13	–
EPI0S22	X	A22	A14	–
EPI0S23	X	A23	A15	–
EPI0S24	X	A24	A16	–
EPI0S25	0x0	A25 ⁽²⁾	A17	–
	0x1			CS1n
	0x2			–
	0x3			–
	0x4			–
	0x5			–
	0x6			–

⁽¹⁾ "X" indicates the state of this field is a don't care.

⁽²⁾ When an entry straddles several row, the signal configuration is the same for all rows.

Table 16-7. EPI Host-Bus 8 Signal Connections (continued)

EPI Signal	CSCFG	HB8 Signal (MODE = ADMUX)	HB8 Signal (MODE = ADNOMUX (Cont. Read))	HB8 Signal (MODE = XFIFO)
EPI0S26	0x0	A26	A18	FEMPTY
	0x1			
	0x2			
	0x3	CS0n	CS0n	
	0x4	A26	A18	
	0x5			
	0x6	CS0n	CS0n	
EPI0S27	0x0	A27	A19	FFULL
	0x1			
	0x2			
	0x3	CS1n	CS1n	
	0x4	CS0n	CS0n	
	0x5	CS1n	CS1n	
	0x6			
EPI0S28	X	RDn/OEn	RDn/OEn	RDn
EPI0S29	X	WRn	WRn	WRn
EPI0S30	0x0	ALE	ALE	–
	0x1	CSn	CSn	CSn
	0x2	CS0n	CS0n	CS0n
	0x3	ALE	ALE	–
	0x4			–
	0x5	CS0n	CS0n	–
	0x6	ALE	ALE	–
EPI0S31	X	Clock ⁽³⁾	Clock ⁽³⁾	Clock ⁽³⁾
EPI0S32	X	iRDY	iRDY	iRDY
EPI0S33	0x0	X	X	X
	0x1	X	X	X
	0x2	X	X	X
	0x3	X	X	X
	0x4	X	X	X
	0x5	CS3n	CS3n	X
	0x6			X
EPI0S34	0x0	X	X	X
	0x1	X	X	X
	0x2	X	X	X
	0x3	X	X	X
	0x4	X	X	X
	0x5	CS2n	CS2n	X
	0x6			X

⁽³⁾ The clock signal is not required for this mode.

Table 16-7. EPI Host-Bus 8 Signal Connections (continued)

EPI Signal	CSCFG	HB8 Signal (MODE = ADMUX)	HB8 Signal (MODE = ADNOMUX (Cont. Read))	HB8 Signal (MODE = XFIFO)
EPI0S35	0x0	X	X	X
	0x1	X	X	X
	0x2	X	X	X
	0x3	X	X	X
	0x4	X	X	X
	0x5	CRE	CRE	X
	0x6			X

Table 16-8 shows how the EPI[31:0] signals function while in host-bus 16 mode. The signal configuration changes based on the address/data mode selected by the MODE field in the EPIHB16CFGn register, on the chip select configuration selected by the CSCFG and CSCFGEXT field in the same register, and on whether byte selects are used as configured by the BSEL bit in the EPIHB16CFG register.

Although the EPI0S31 signal can be configured for the EPI clock signal in host-bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 16-8. EPI Host-Bus 16 Signal Connections

EPI Signal	CSCFG	BSEL	HB16 Signal (MODE = ADMUX)	HB16 Signal (MODE = ADNOMUX (Cont. Read))	HB16 Signal (MODE = XFIFO)
EPI0S0	X ⁽¹⁾	X	AD0 ⁽²⁾	D0	D0
EPI0S1	X	X	AD1	D1	D1
EPI0S2	X	X	AD2	D2	D2
EPI0S3	X	X	AD3	D3	D3
EPI0S4	X	X	AD4	D4	D4
EPI0S5	X	X	AD5	D5	D5
EPI0S6	X	X	AD6	D6	D6
EPI0S7	X	X	AD7	D7	D7
EPI0S8	X	X	AD8	D8	D8
EPI0S9	X	X	AD9	D9	D9
EPI0S10	X	X	AD10	D10	D10
EPI0S11	X	X	AD11	D11	D11
EPI0S12	X	X	AD12	D12	D12
EPI0S13	X	X	AD13	D13	D13
EPI0S14	X	X	AD14	D14	D14
EPI0S15	X	X	AD15	D15	D15
EPI0S16	X	X	A16	A0 ⁽²⁾	—
EPI0S17	X	X	A17	A1	—
EPI0S18	X	X	A18	A2	—
EPI0S19	X	X	A19	A3	—
EPI0S20	X	X	A20	A4	—
EPI0S21	X	X	A21	A5	—
EPI0S22	X	X	A22	A6	—

⁽¹⁾ X = don't care

⁽²⁾ In this mode, halfword accesses are used. A0 is the LSB of the address and is equivalent to the internal Cortex-M4 A1 address. This pin should be connected to A0 of 16-bit memories.

Table 16-8. EPI Host-Bus 16 Signal Connections (continued)

EPI Signal	CSCFG	BSEL	HB16 Signal (MODE = ADMUX)	HB16 Signal (MODE = ADNOMUX (Cont. Read))	HB16 Signal (MODE = XFIFO)
EPI0S23	X ⁽³⁾	0	A23	A7	—
		1			
EPI0S24	0x0	0	A24	A8	—
		1			
	0x1	0			
		1			
	0x2	0			
		1			
	0x3	0	BSEL0n	BSEL0n	—
		1			
	0x4	0	A24	A8	—
		1			—
	0x5	0			—
		1			—
EPI0S25	0x0	X	A25	A9	—
	0x1	0	A25	A9	CS1n
	0x2	0	A25	A9	—
		1	BSEL0n	BSEL0n	
	0x3	0	A25	A9	—
		1	BSEL1n	BSEL1n	
	0x4	0	A25	A9	—
		1	BSEL0n	BSEL0n	
	0x5	0	A25	A9	—
		1	BSEL0n	BSEL0n	
EPI0S26	0x0	0	A26	A10	FEMPTY
		1	BSEL0n	BSEL0n	
	0x1	0	A26	A10	
		1	BSEL0n	BSEL0n	
	0x2	0	A26	A10	—
		1	BSEL1n	BSEL1n	
	0x3	X	CS0n	CS0n	—
		0	A26	A10	
	0x4	0	A26	A10	—
		1	BSEL1n	BSEL1n	
	0x5	0	A26	A10	—
		1	BSEL1n	BSEL1n	
	0x6	0	CS0n	CS0n	—
		1			

⁽³⁾ When an entry straddles several rows, the signal configuration is the same for all rows.

Table 16-8. EPI Host-Bus 16 Signal Connections (continued)

EPI Signal	CSCFG	BSEL	HB16 Signal (MODE = ADMUX)	HB16 Signal (MODE = ADNOMUX (Cont. Read))	HB16 Signal (MODE = XFIFO)
EPI0S27	0x0	0	A27	A11	FFULL
		1	BSEL1n	BSEL1n	
	0x1	0	A27	A11	
		1	BSEL1n	BSEL1n	
	0x2	X	CS1n	CS1n	–
	0x3	X	CS1n	CS1n	
	0x4	X	CS0n	CS0n	
	0x5	X	CS1n	CS1n	–
	0x6	X	CS1n	CS1n	–
EPI0S28	X	X	RDn/OEn	RDn/OEn	RDn
EPI0S29	X	X	WRn	WRn	WRn
EPI0S30	0x0	X	ALE	ALE	–
	0x1	X	CSn	CSn	CSn
	0x2	X	CS0n	CS0n	CS0n
	0x3	X	ALE	ALE	–
	0x4	X	ALE	ALE	–
	0x5	X	CS0n	CS0n	–
	0x6	X	ALE	ALE	–
EPI0S31	X	X	Clock ⁽⁴⁾	Clock ⁽⁴⁾	Clock ⁽⁴⁾
EPI0S32	X	X	iRDY	iRDY	iRDY
EPI0S33	X	X	CS3n	CS3n	X
EPI0S34	X	X	CS2n	CS2n	X
EPI0S35	X	X	CRE	CRE	X

⁽⁴⁾ The clock signal is not required for this mode.

The RDYEN in the EPIHBnCFG enables the monitoring of the external iRDY pin to stall accesses. On the rising edge of EPI clock, if iRDY is low, access is stalled. The IRDYDLY can program the number of EPI clock cycles in advance to the stall (1, 2, or 3) (see Figure 16-5). This is a conceptual timing diagram of how the iRDY signal works with different IRDYDLY configurations. When enabled, the iRDY stalls the internal states of the EPI, while IRDYDLY controls the delay pipeline when this stall takes affect. The iRDY signal can be connected to multiple devices with a pullup resistor (see Figure 16-6). When multiple PSRAMs are connected to iRDY, the EPIHPnCFG registers must be programmed to the same iRDY signal polarity through the IRDYINV bit. When connected to a PSRAM, iRDY is used to control the address to data latency.

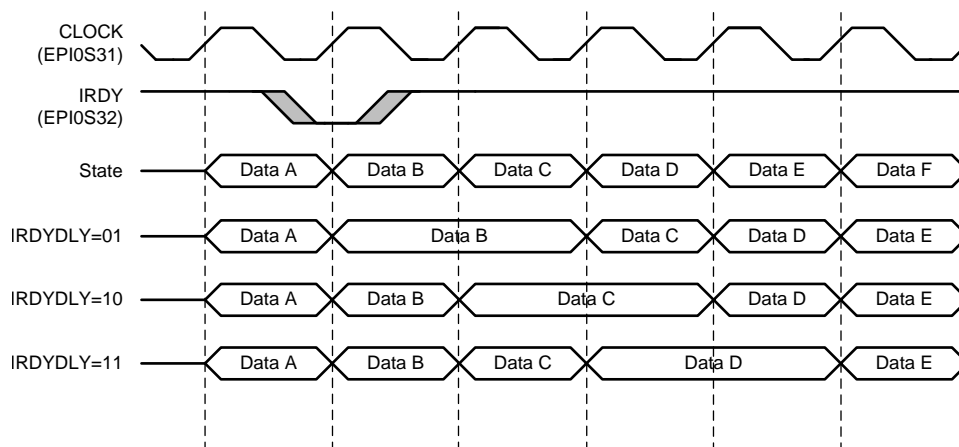


Figure 16-5. iRDY Access Stalls, IRDYDLY = 01, 10, 11

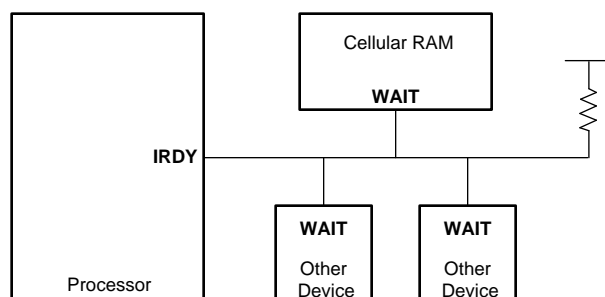


Figure 16-6. iRDY Signal Connection

16.4.3.2 PSRAM Support

The EPI host bus supports both a synchronous and asynchronous interface to PSRAM memory when configured in 16-bit bus multiplexed mode. The EPIHBPSRAM register holds the values for the PSRAM bus configuration (CR) registers. The contents of the EPIHBPSRAM register can be sent to different memories depending on which WRCRE or RDCRE bit is set in the various EPIHB16CFGn registers. For example, if the WRCRE bit is enabled in EPIHB16CFG, then the CRE signal asserts and the contents are sent to the memory enabled by CS0. Enabling the WRCRE or RDCRE bit in EPIHB16CFG2 register activates CS1n during a PSRAM configuration register write or read. The WRCRE and RDCRE bit in EPIHB16CFG3 corresponds to CS2n and EPIHB16CFG4, to CS3n. The WRCRE bit clears when the transfer is done. There must not be any system access or nonblocking read activity during the CRE read or write-enable transfer. During a write to the PSRAM's CR, the configuration data is written out on data pins [20:0] of the EPI bus. For a PSRAM configuration read access, the RDCRE bit in the EPIHB16CFG register is set to signal that the next access is a read of the PSRAM configuration register (CR). The address for the CR is written to bits CR[19:18] of the EPIHBPSRAM register. The read data is returned at CR bits [15:0] of the EPIHBPSRAM register.

NOTE:

- CRE read and write operations may only occur in asynchronous mode. During synchronous mode the CRE bit should be disabled. Setting the CRE bit during synchronous PSRAM accesses can lead to unpredictable behavior.
 - When the chip select is programmed to access the PSRAM, the MODE bit of the EPIHBnCFGn register must be programmed to enable address and data muxed (ADMUX). Page mode accesses are not supported by the EPI.
 - BURST is optimized for word-length bursting for SDRAM and PSRAM accesses.
-

These steps initialize the PSRAM interface:

1. Perform the EPI initialization steps in [Section 16.4](#).
2. Enable host bus 16 mode by setting the MODE bits in the EPICFG register to 0x13. Choose between an integer or formula clock divide for the baud rate by configuring the INTDIV bit in the EPICFG register.
3. Configure the EPIBAUD register to the desired baud rate.
4. Because the EPI module supports only asynchronous programming of the configuration registers, clock gate the EPI clock by programming both the CLKGATE and CLKGATEI bits in the EPIHB16CFG register to 0.
5. Prepare for writing the PSRAM Bus Configuration register by setting the ALEHIGH = 1 and MODE = 0x0 in the EPIHB16CFG register.
6. Program the EPIHBPSRAM register to be loaded into the CR register of the PSRAM by configuring bits [21:0].
 1. CR[20:19] = 0x0, reserved
 2. CR[19:18] = 0x2 to enable configuring of the CR register
 3. CR[15] = 0x1 to enable asynchronous access
 4. CR [14] = 0 if the iRDY signal is used for memory transfers; if the design will not use the iRDY signal CR[14] should be cleared.
 5. CR[13:11] must be programmed to have a matching read and write wait state configuration as is programmed in the EPIHB16CFG and EPIHB16TIME register.
 6. CR[10] configures the polarity of the WAIT signal and should match the configuration of the IRDYINV bit in the EPIHB16CFG register.
 7. CR[8] = 0x1 to configure the appropriate wait configuration of the data
 8. CR[2:0] = 0x7 since the EPI interface in PSRAM mode is a continuous burst access.
7. Set the WRCRE bit in the EPIHB16CFGn register to initiate a write from the EPIHBPSRAM register to the PSRAM CR register.

NOTE: If the PSRAM CR register must be reprogrammed after initialization, the application should allow the previous transfer to complete before beginning configuration to ensure proper PSRAM functionality.

Table 16-9. PSRAM Fixed Latency Wait State Configuration

Latency Counter	Latency in Cycles	RDWS[1:0]/WRWS[1:0]	RWSM/WRWSM
BCR Code 2	3	0x0	0
BCR Code 3	4	0x1	1
BCR Code 4	5	0x1	0
BCR Code 5	6	0x2	1
BCR Code 6	7	0x2	0
BCR Code 8	9	0x3	0

In variable initial latency mode, the memory's WAIT (iRDY) pin guides the EPI module when to read and write. The WAIT (iRDY) pin stalls the access for the duration of the latency and adds cycles if there is a refresh collision. To get the best performance, set CR[13:11] = 0x2, the WRWS field of the EPIHB16CFG register to 0x0, and the WRWSM and RDWSM bit of the EPI16TIMEn register to 0. For the WAIT pin to be recognized correctly, set the IRDYDLY bit in the EPI16TIMEn register to 1 and the CR[8] = 1 in the EPIHBPSRAM register.

NOTE: Wait state latency works differently in PSRAM Burst mode than in other modes. In PSRAM Burst mode, the RDWS and WRWS bit fields define the latency for only the first access of the write or read cycle. Every access after that is a single access.

Figure 16-7 and Figure 16-8 depict a PSRAM burst read and write.

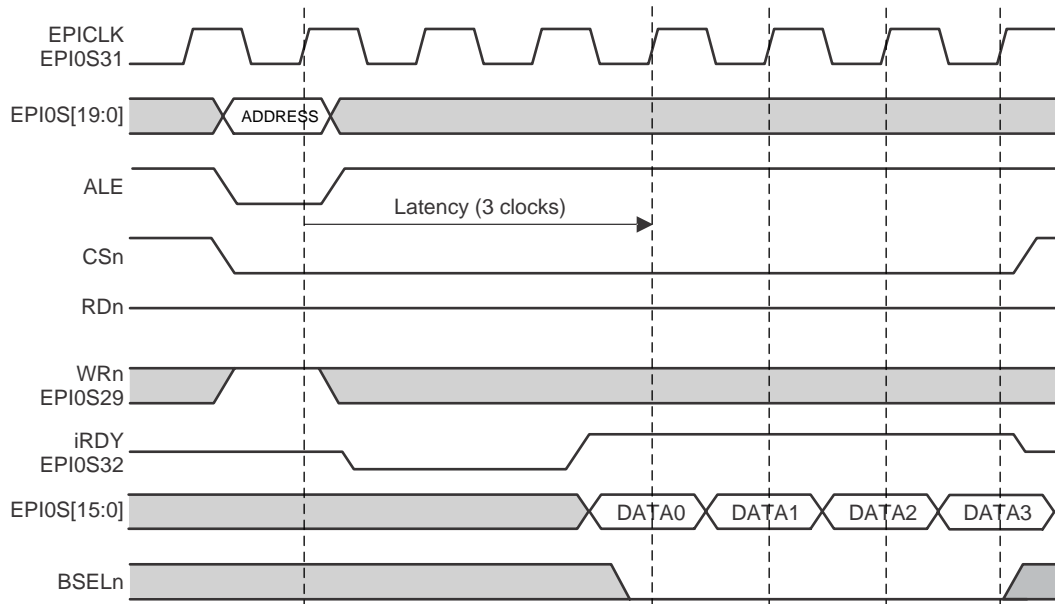
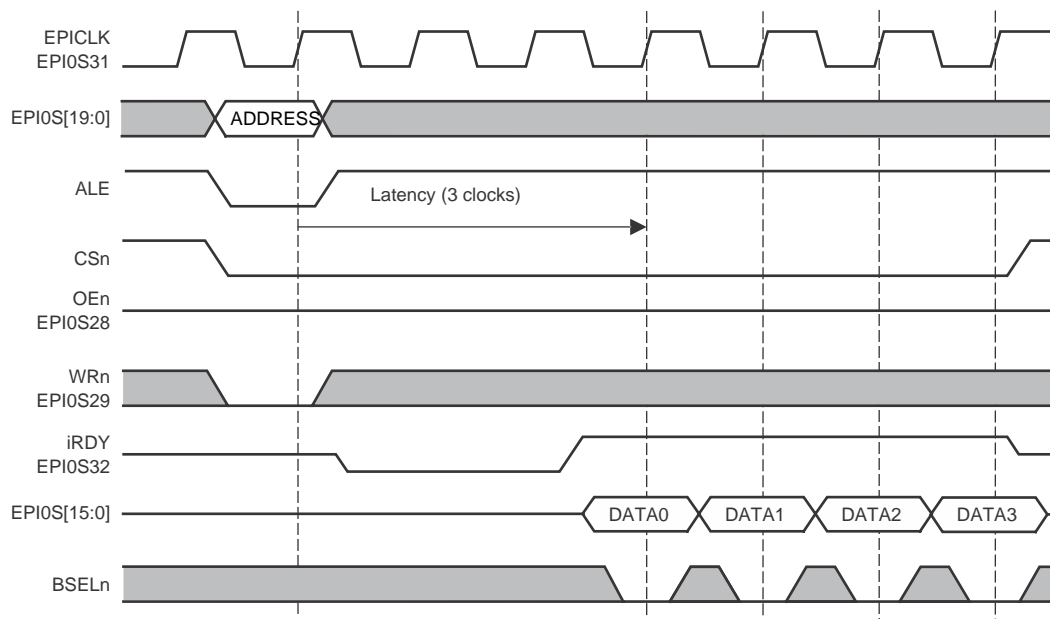
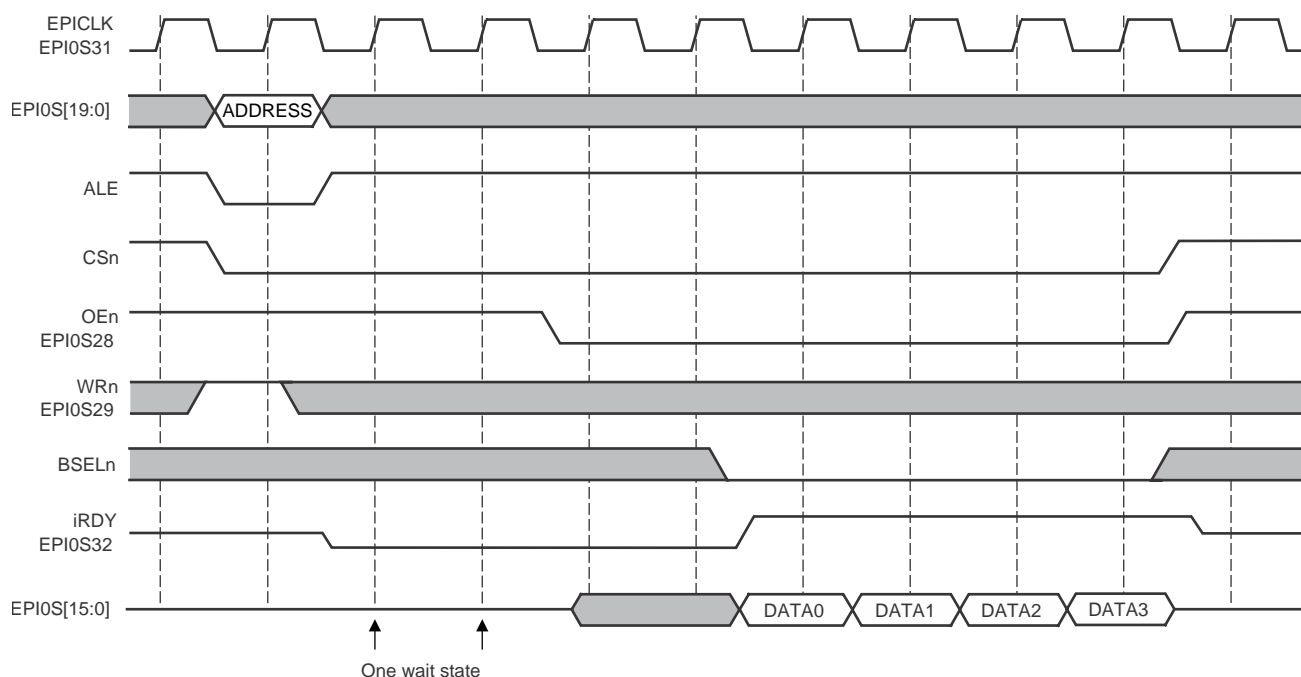


Figure 16-7. PSRAM Burst Read


Figure 16-8. PSRAM Burst Write

If a read or write transfer attempts to begin during a refresh event, the transfer is held off by the assertion of the **iRDY** pin by the memory to the EPI module. [Figure 16-9](#) and [Figure 16-10](#) depict the delay in data transfer during a refresh collision.


Figure 16-9. Read Delay During Refresh Event

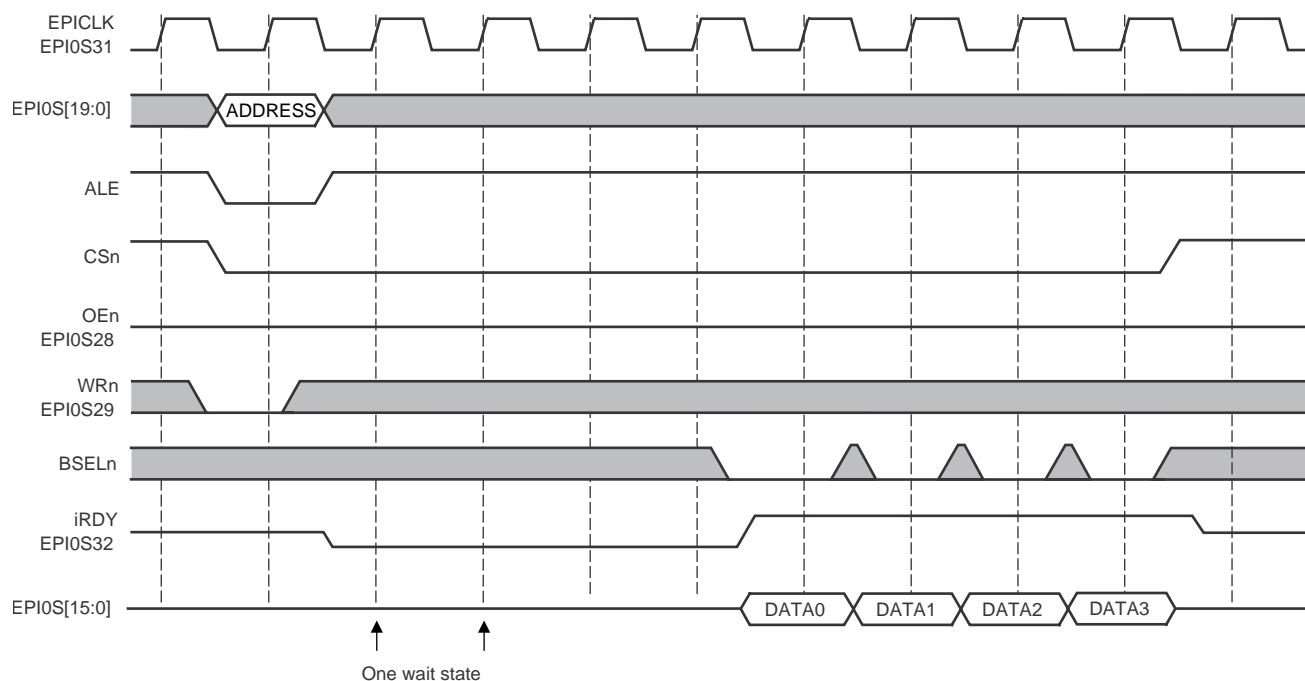


Figure 16-10. Write Delay During Refresh Event

16.4.3.3 Host Bus 16-Bit Muxed Interface

Figure 16-11 shows how to connect the EPI signals to a 16-bit SRAM and a 16-bit flash memory with muxed address and memory using byte selects and dual chip selects with ALE. This schematic is just an example of how to connect the signals; timing and loading have not been analyzed. In addition, not all bypass capacitors are shown.

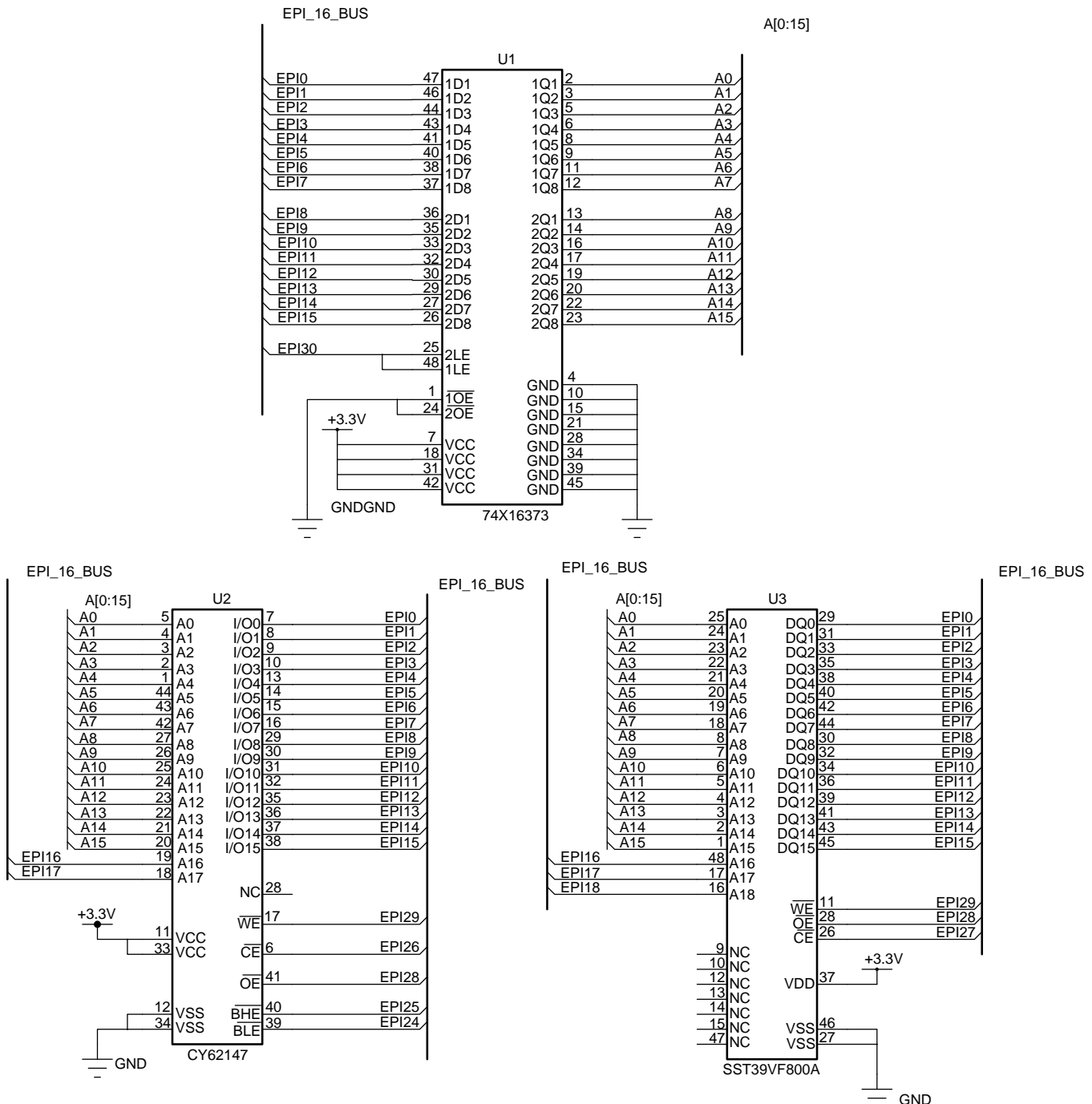


Figure 16-11. Example Schematic for Muxed Host-Bus 16 Mode

16.4.3.4 Speed of Transactions

The COUNT0 field in the EPIBAUD register must be configured to set the main transaction rate based on what the slave device can support (including wiring considerations). The main control transitions are normally 1/2 the baud rate (COUNT0 = 1) because the EPI block forces data versus control to change on alternating clocks. When using dual chip selects, each chip select can access the bus using differing baud rates by setting the CSBAUD bit in the EPIHBnCFG2 register. In this case, the COUNT0 field controls the CS0n transactions, and the COUNT1 field controls the CS1n transactions. When using quad chip select mode, the COUNT0 bit field of the EPIBAUD2 register controls the baud rate of CS2n and the COUNT1 bit field is programmed to control the baud rate of CS3n.

Additionally, the Host-Bus mode provides read and write wait states for the data portion to support different classes of device. These wait states stretch the data period (hold the rising edge of data strobe) and may be used in all four sub-modes. The wait states are set using the WRWS and RDWS bits in the EPI Host-Bus n Configuration (EPIHBnCFGn) register. The WRWS and RDWS bits are enhanced with more precision by WRWSM and RDWSM bits in the EPIHBnTIMEn registers. Note none of the wait state configuration bits can be set concurrently with the BURST bit in the same EPIHBnCFGn register. See [Table 16-10](#) for programming information.

Table 16-10. Data Phase Wait State Programming

RDWS or WRWS Encoding in EPIHBnCFGn Register	RDWSM or WRWSM Encoding in EPIHBnTIMEn Registers	Data Phase Wait States
0x0	1	1 EPI clock cycle
0x0	0	2 EPI clock cycles
0x1	1	3 EPI clock cycles
0x1	0	4 EPI clock cycles
0x2	1	5 EPI clock cycles
0x2	0	6 EPI clock cycles
0x3	1	7 EPI clock cycles
0x3	0	8 EPI clock cycles

The CAPWIDTH bit in EPIHBnTIMEn registers controls the delay between Host-Bus transfers. When the CSBAUD bit is set and multiple chip selects have been configured in the EPIHBnCFG2 registers, delay takes an additional clock cycle to adjust the clock rate of different chip selects.

Word read and write transactions can be enhanced through the enabling of the BURST bit in the EPIHB16CFGn registers.

16.4.3.5 Sub-Modes of Host Bus 8 and 16

The EPI controller supports four variants of the host-bus model using 8 or 16 bits of data in all four cases. The four sub-modes are selected using the MODE bits in the EPIHBnCFG register, and are:

1. Address and data are muxed. This scheme is used by many 8051 devices, some Microchip PIC parts, and some ATmega parts. When used for standard SRAMs, a latch must be used between the microcontroller and the SRAM. This sub-mode is provided for compatibility with existing devices that support data transfers without a latch (that is, CPLDs). In general, the de-muxed sub-mode should normally be used. The ALE configuration should be used in this mode, as all Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold until the data phase. The ALE configuration is controlled by configuring the CSCFG and CSCFGEXT field to be 0x0 in the EPIHBnCFG2 register. The ALE can be enhanced to access two or four external devices with four separate CSn signals. By configuring the CSCFG field to be 0x3 and the CSCFGEXT bit to be 0 in the EPIHBnCFG2 register, EPI0S30 functions as ALE, EPI0S27 functions as CS1n, and EPI0S26 functions as CS0n. When the CSCFG field is set to 0x0 and the CSCFGEXT bit is set to 1 in the EPIHBnCFG2 register, EPI0S30 functions as ALE, EPI0S33 functions as CS3n, EPI0S34 functions as CS2n, EPI0S27 functions as CS1n, and EPI0S26 functions as CS0n. The CSn is best used for Host-Bus unmuxed mode, in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring.

2. Address and data are separate with 8 or 16 bits of data and up to 20 bits of address (1MB). This scheme is used by more modern 8051 devices, as well as some PIC and ATmega parts. This mode is generally used with SRAMs in continuous read modes, many EEPROMs, and many NOR Flash memory devices. There is no hardware command write support for flash memory devices; this mode should only be used for Flash memory devices programmed at manufacturing time. If a Flash memory device must be written and does not support a direct programming model, the command mechanism must be performed in software. The CSn configuration should be used in this mode. The CSn signal indicates when the address and data phases of a read or write access is occurring. The CSn configuration is controlled by configuring the CSCFG field to be 0x1 and the CSCFGEXT bit to be 0 in the EPIHBnCFG2 register.
3. Continuous read mode where address and data are separate. This read sub-mode is used by some SRAMs and can read more quickly by only changing the address (and not using RDn/OEn strobing). In this sub-mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change. For example, to read data from address 0x100 and then 0x101, the EPI controller asserts the output-enable signal and then configures the address pins to 0x100; the EPI controller then captures what is on the data pins and increments A0 to 1 (so the address is now 0x101); the EPI controller then captures what is on the data pins. This mode consumes higher power because the SRAM must continuously drive the data pins. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available. Writes are not permitted in this mode.
4. FIFO mode uses 8 or 16 bits of data, removes ALE and address pins and optionally adds external XFIFO FULL/EMPTY flag inputs. This scheme is used by many devices, such as radios, communication devices (including USB2 devices), and some FPGA configurations (FIFO through block RAM). This sub-mode provides the data side of the normal Host-Bus interface, but is paced by the FIFO control signals. It is important to consider that the XFIFO FULL/EMPTY control signals may stall the interface and could have an impact on blocking read latency from the processor or μ DMA. The EPI FIFO can only be used in asynchronous mode.

For the three modes above (1, 2, and 4) that the Host-Bus 16 mode supports, byte select signals can be optionally implemented by setting the BSEL bit in the EPIHB16CFG register.

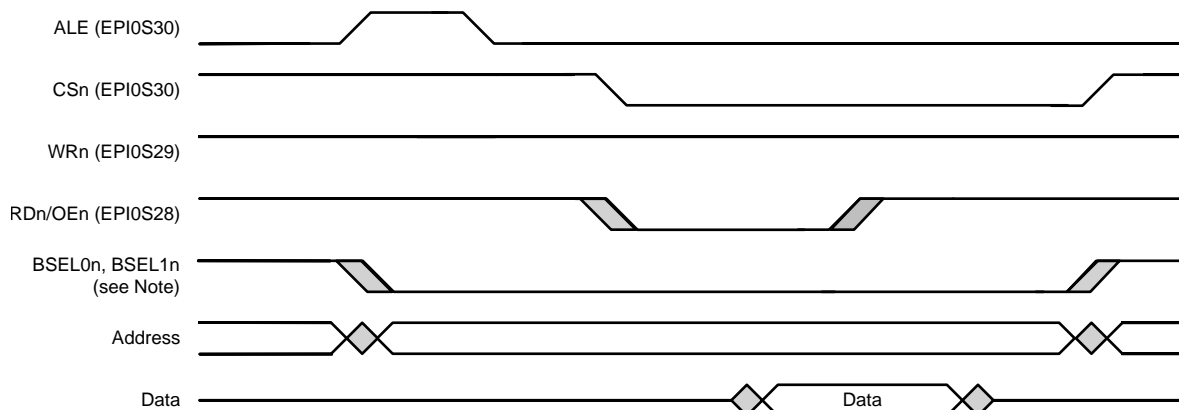
NOTE: Byte accesses should not be attempted if the BSEL bit has not been enabled in Host-Bus 16 Mode.

For timing details for the Host-Bus mode, see the device-specific data sheet.

16.4.3.6 Bus Operation

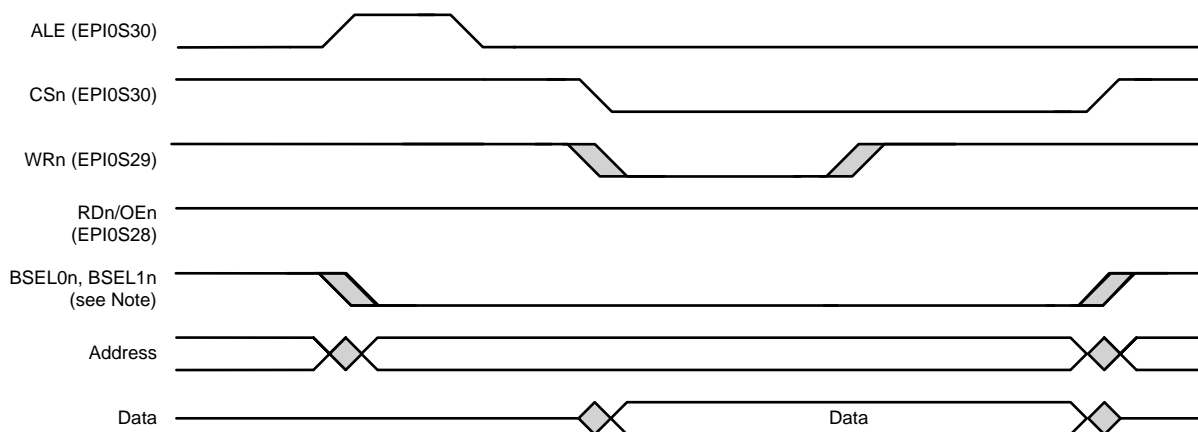
Bus operation is the same in Host-Bus 8 and Host-Bus 16 modes and is asynchronous. Timing diagrams show both ALE and CSn operation. The optional HB16 byte select signals have the same timing as the address signals. If wait states are required in the bus access, they can be inserted during the data phase of the access using the WRWS and RDWS bits in the EPIHBnCFG2 register. Each wait state adds 2 EPI clock cycles to the duration of the WRn or RDn strobe. During idle cycles, the address and muxed address data signals maintain the state of the last cycle.

Figure 16-12 shows a basic Host-Bus read cycle. Figure 16-13 shows a basic Host-Bus write cycle. Both of these figures show address and data signals in the non-multiplexed mode (MODE field ix 0x1 in the EPIHBnCFG register).



NOTE: BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

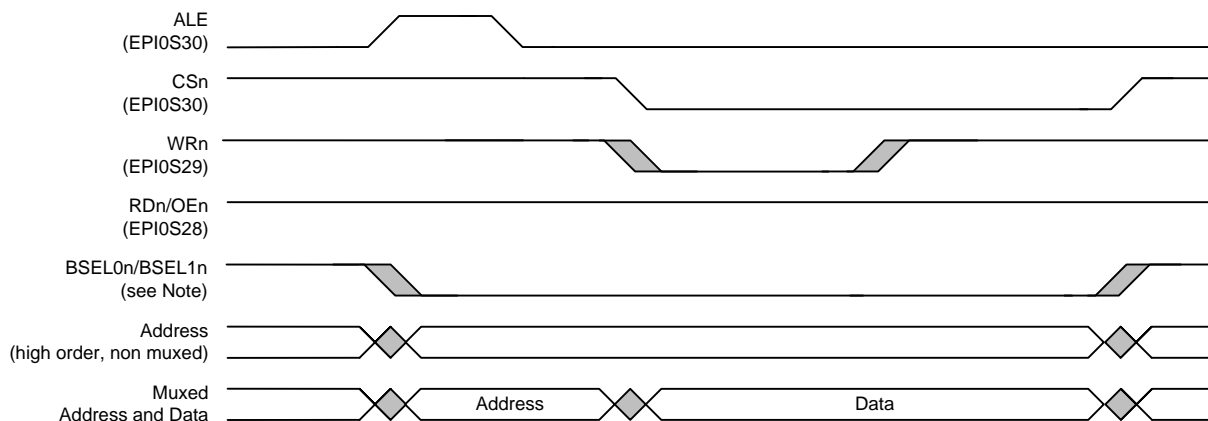
Figure 16-12. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0



NOTE: BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 16-13. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0

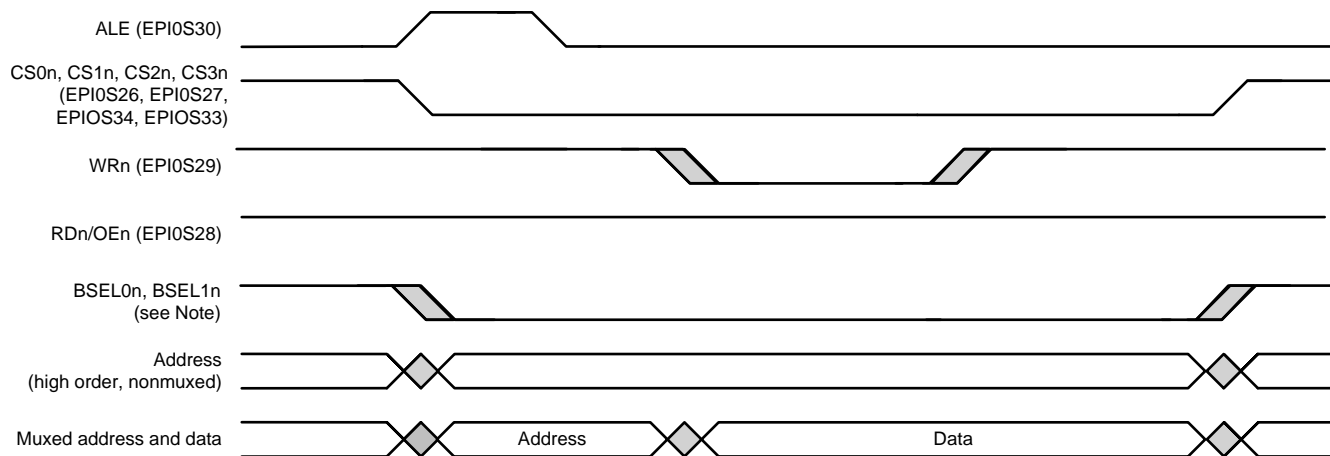
Figure 16-14 shows a write cycle with the address and data signals multiplexed (MODE field is 0x0 in the EPIHBnCFG register). A read cycle would look similar, with the RDn strobe being asserted along with CSn and data being latched on the rising edge of RDn.



NOTE: BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 16-14. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0

When using ALE with dual CSn configuration (CSCFGEXT bit is 0 and the CSCFG field is 0x3 in the EPIHBnCFG2 register) or quad chip select (CSCFGEXT bit is 1 and CSCSFG is 0x2), the appropriate CSn signal is asserted at the same time as ALE, as shown in [Figure 16-15](#).



NOTE: BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 16-15. Host-Bus Write Cycle with Multiplexed Address and Data and ALE With Dual or Quad CSn

[Figure 16-16](#) shows continuous read mode accesses. In this mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change.

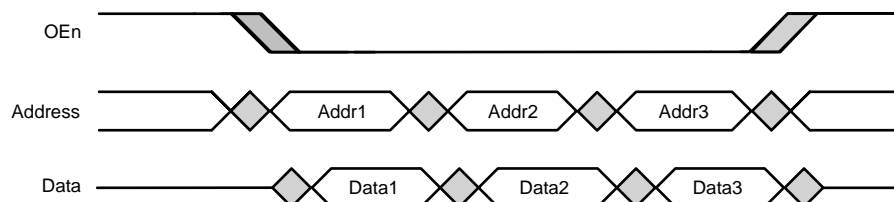


Figure 16-16. Continuous Read Mode Accesses

FIFO mode accesses are the same as normal read and write accesses, except that the ALE signal and address pins are not present. Two input signals can be used to indicate when the XFIFO is full or empty to gate transactions and avoid overruns and underruns. The FFULL and FEMPTY signals are synchronized and must be recognized as asserted by the microcontroller for 2 system clocks before they affect transaction status. The MAXWAIT field in the EPIHBnCFG register defines the maximum number of EPI clock cycles to wait while the FEMPTY or FFULL signal is holding off a transaction. [Figure 16-17](#) shows how the FEMPTY signal should respond to a write and read from the XFIFO. [Figure 16-18](#) shows how the FEMPTY and FFULL signals should respond to 2 writes and 1 read from an external FIFO that contains two entries.

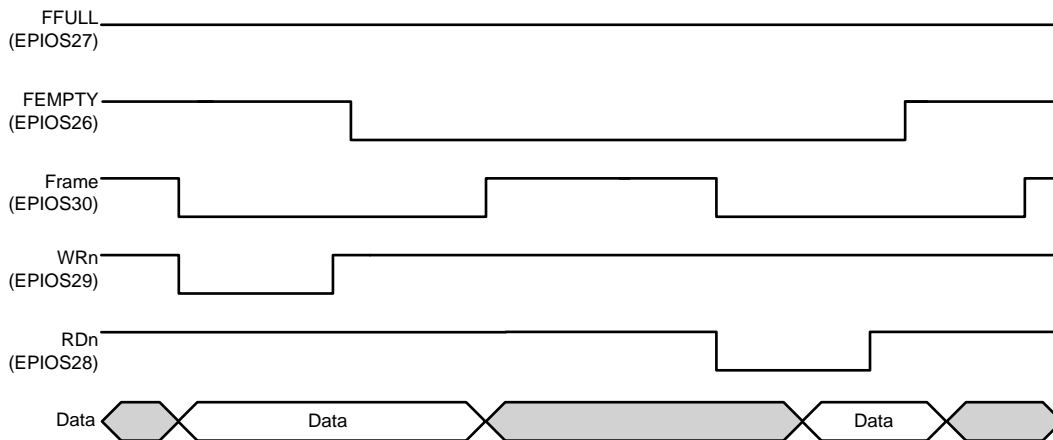


Figure 16-17. Write Followed by Read to External FIFO

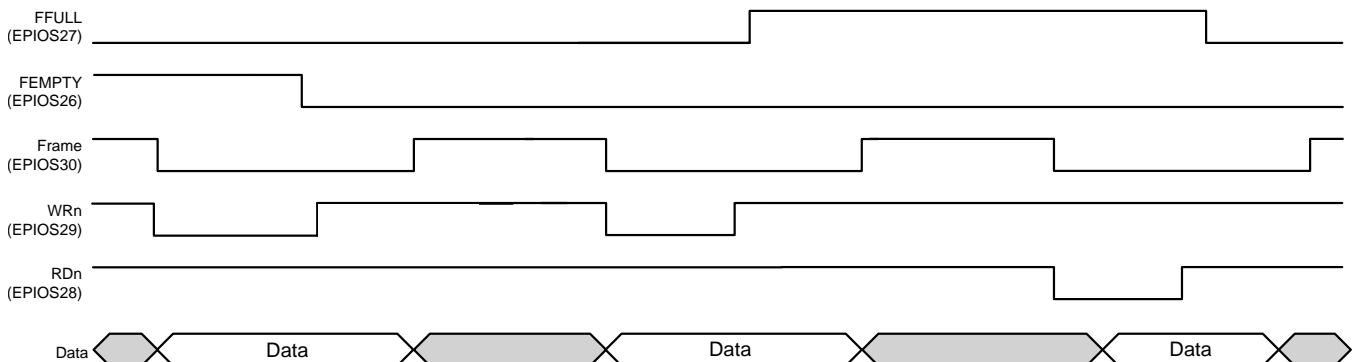


Figure 16-18. Two-Entry FIFO

16.4.4 General-Purpose Mode

The General-Purpose Mode Configuration (EPIGPCFG) register is used to configure the control, data, and address pins, if used. Any unused EPI controller signals can be used as GPIOs or another alternate function. The general-purpose configuration can be used for custom interfaces with FPGAs, CPLDs, and digital data acquisition and actuator control.

General-Purpose mode is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs. Three sizes of data and optional address are supported. Framing and clock-enable functions permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the EPIBAUD register baud rate (when used with the WFIFO and/or the NBRFIFO) or by the rate of accesses from software or μ DMA. Examples of this type of use include:
 - Reading 20 sensors at fixed time periods by configuring 20 pins to be inputs, configuring the COUNT0 field in the EPIBAUD register to some divider, and then using nonblocking reads.
 - Implementing a very wide ganged PWM/PCM with fixed frequency for driving actuators or LEDs.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free-running or gated), a framing signal (with frame size), a ready input (to stretch transactions), an address (of varying sizes), and data (of varying sizes). Additionally, provisions are made for separating data and address phases.

The interface has the following optional features:

- Use of the EPI clock output is controlled by the CLKPIN bit in the EPIGPCFG register. Unclocked uses include general-purpose I/O and asynchronous interfaces (optionally using RD and WR strobes).

Clocked interfaces allow for higher speeds and are much easier to connect to FPGAs and CPLDs (which usually include input clocks).

- EPI clock, if used, may be free running or gated depending on the CLKGATE bit in the EPIGPCFG register. A free-running EPI clock requires another method for determining when data is live, such as the frame pin or RD/WR strobes. A gated clock approach uses a setup-time model in which the EPI clock controls when transactions are starting and stopping. The gated clock is held high until a new transaction is started and goes high at the end of the cycle where RD/WR/FRAME and address (and data if write) are emitted.
- Use of the RD and WR outputs is controlled by the RW bit in the EPIGPCFG register. For interfaces where the direction is known (in advance, related to frame size, or other means), these strobes are not needed. For most other interfaces, RD and WR are used so the external peripheral knows what transaction is taking place, and if any transaction is taking place.
- Separation of address/request and data phases may be used on writes using the WR2CYC bit in the EPIGPCFG register. This configuration allows the external peripheral extra time to act. Address and data phases must be separated on reads. When configured to use an address as specified by the ASIZE field in the EPIGPCFG register, the address is emitted on the with the RD strobe (first cycle) and data is expected to be returned on the next cycle (when RD is not asserted). If no address is used, then RD is asserted on the first cycle and data is captured on the second cycle (when RD is not asserted), allowing more setup time for data.

NOTE: When WR2CYC = 0, write data is valid when the WR strobe is asserted (High). When WR2CYC = 1, write data is valid when the WR strobe is Low after being asserted (High).

For writes, the output may be in one or two cycles. In the two-cycle case, the address (if any) is emitted on the first cycle with the WR strobe and the data is emitted on the second cycle (with WR not asserted). Although split address and write data phases are not normally needed for logic reasons, it may be useful to make read and write timings match. If 2-cycle reads or writes are used, the RW bit is automatically set.

- Address may be emitted (controlled by the ASIZE field in the EPIGPCFG register). The address may be up to 4 bits (16 possible values), up to 12 bits (4096 possible values), or up to 20 bits (1 M possible values). Size of address limits size of data, for example, 4 bits of address support up to 24 bits data. 4-bit address uses EPIOS[27:24]; 12-bit address uses EPIOS[27:16]; 20-bit address uses EPIOS[27:8]. The address signals may be used by the external peripheral as an address, code (command), or for other unrelated uses (such as a chip enable). If the chosen address/data combination does not use all of the EPI signals, the unused pins can be used as GPIOs or for other functions. For example, when using a 4-bit address with an 8-bit data, the pins assigned to EPIS0[23:8] can be assigned to other functions.
- Data may be 8 bits, 16 bits, 24 bits, or 32 bits (controlled by the DSIZE field in the EPIGPCFG register). By default, the EPI controller uses data bits [7:0] when the DSIZE field in the EPIGPCFG register is 0x0; data bits [15:0] when the DSIZE field is 0x1; data bits [23:0] when the DSIZE field is 0x2; and data bits [31:0] when the DSIZE field is 0x3. 32-bit data cannot be used with address or EPI clock or any other signal. 24-bit data can only be used with 4-bit address or no address.
- When using the EPI controller as a GPIO interface, writes are FIFOed (up to 4 can be held at any time), and up to 32 pins are changed using the EPIBAUD clock rate specified by COUNT0. As a result, output pin control can be very precisely controlled as a function of time. By contrast, when writing to normal GPIOs, writes can only occur 8-bits at a time and take up to two clock cycles to complete. In addition, the write itself may be further delayed by the bus due to μ DMA or draining of a previous write. With both GPIO and the EPI controller, reads may be performed directly, in which case the current pin states are read back. With the EPI controller, the nonblocking interface may also be used to perform reads based on a fixed time rule via the EPIBAUD clock rate.

Table 16-11 shows how the EPIOS[31:0] signals function while in General-Purpose mode. Notice that the address connections vary depending on the data-width restrictions of the external peripheral.

Table 16-11. EPI General-Purpose Signal Connections

EPI Signal	General-Purpose Signal (D8, A20)	General-Purpose Signal (D16, A12)	General-Purpose Signal (D24, A4)	General-Purpose Signal (D32)
EPI0S0	D0	D0	D0	D0
EPI0S1	D1	D1	D1	D1
EPI0S2	D2	D2	D2	D2
EPI0S3	D3	D3	D3	D3
EPI0S4	D4	D4	D4	D4
EPI0S5	D5	D5	D5	D5
EPI0S6	D6	D6	D6	D6
EPI0S7	D7	D7	D7	D7
EPI0S8	A0	D8	D8	D8
EPI0S9	A1	D9	D9	D9
EPI0S10	A2	D10	D10	D10
EPI0S11	A3	D11	D11	D11
EPI0S12	A4	D12	D12	D12
EPI0S13	A5	D13	D13	D13
EPI0S14	A6	D14	D14	D14
EPI0S15	A7	D15	D15	D15
EPI0S16	A8	A0 ⁽¹⁾	D16	D16
EPI0S17	A9	A1	D17	D17
EPI0S18	A10	A2	D18	D18
EPI0S19	A11	A3	D19	D19
EPI0S20	A12	A4	D20	D20
EPI0S21	A13	A5	D21	D21
EPI0S22	A14	A6	D22	D22
EPI0S23	A15	A7	D23	D23
EPI0S24	A16	A8	A0 ⁽²⁾	D24
EPI0S25	A17	A9	A1	D25
EPI0S26	A18	A10	A2	D26
EPI0S27	A19	A11	A3	D27
EPI0S28	WR	WR	WR	D28
EPI0S29	RD	RD	RD	D29
EPI0S30	Frame	Frame	Frame	D30
EPI0S31	Clock	Clock	Clock	D31

⁽¹⁾ In this mode, halfword accesses are used. A0 is the LSB of the address and is equivalent to the system A1 address.

⁽²⁾ In this mode, word accesses are used. A0 is the LSB of the address and is equivalent to the system A2 address.

16.4.4.1 Bus Operation

A basic access is 1 EPI clock for write cycles and 2 EPI clock cycles for read cycles. An additional EPI clock can be inserted into a write cycle by setting the WR2CYC bit in the EPIGPCFG register.

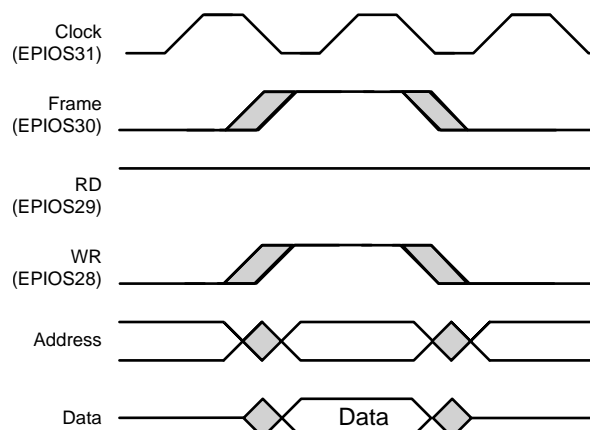


Figure 16-19. Single-Cycle Single Write Access, FRM50 = 0, FRMCNT = 0, WR2CYC = 0

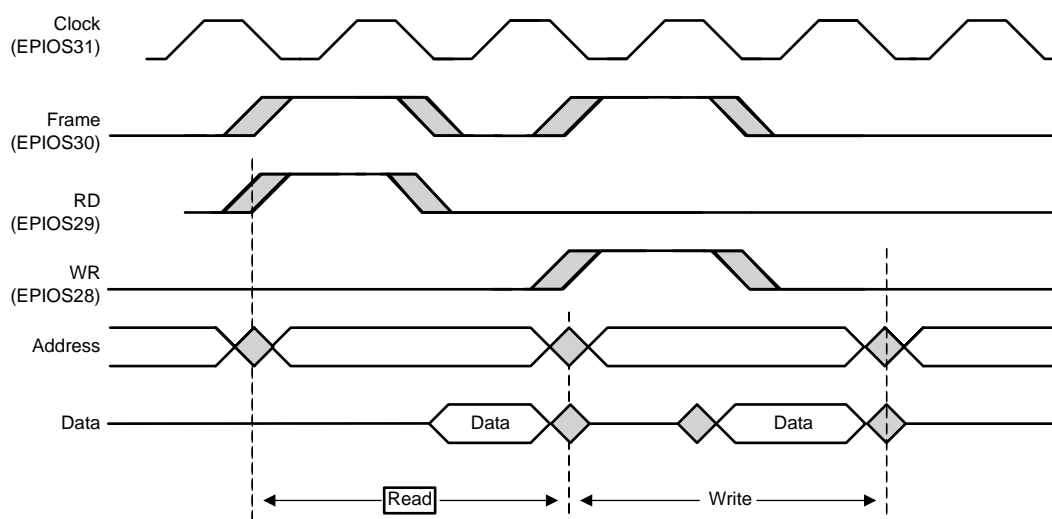


Figure 16-20. Two-Cycle Read, Write Accesses, FRM50 = 0, FRMCNT = 0, WR2CYC = 1

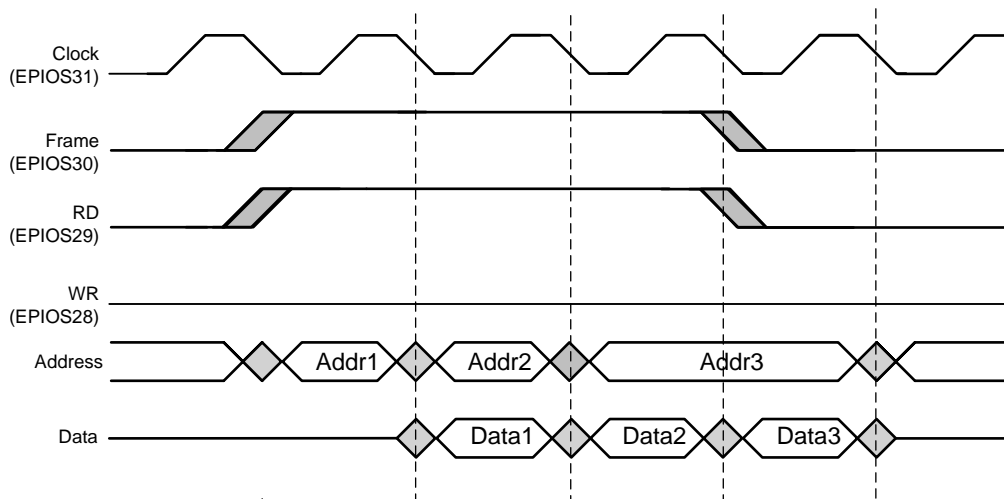


Figure 16-21. Read Accesses, FRM50 = 0, FRMCNT = 0

16.4.4.1.1 FRAME Signal Operation

The operation of the FRAME signal is controlled by the FRMCNT and FRM50 bits. When FRM50 is clear, the FRAME signal is high whenever the WR or RD strobe is high. When FRMCNT is clear, the FRAME signal is simply the logical OR of the WR and RD strobes so the FRAME signal is high during every read or write access, see [Figure 16-22](#).

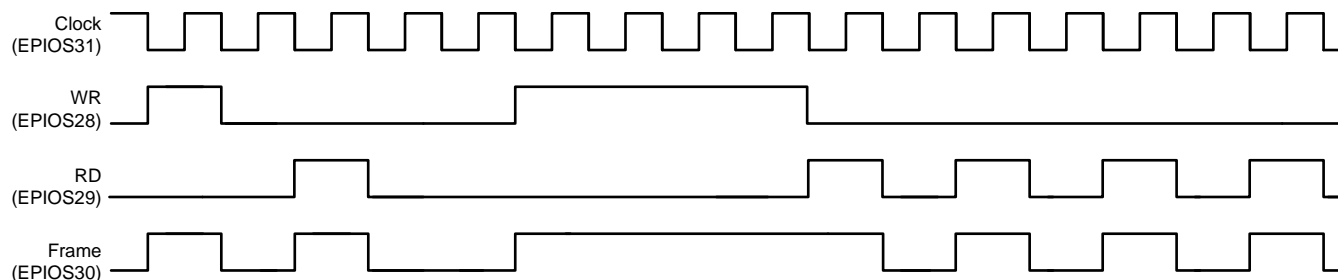


Figure 16-22. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 0

If the FRMCNT field is 0x1, then the FRAME signal pulses high during every other read or write access, see [Figure 16-23](#).

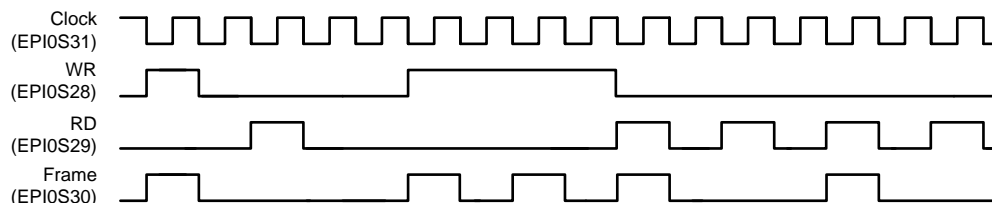


Figure 16-23. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 1

If the FRMCNT field is 0x2 and FRM50 is clear, then the FRAME signal pulses high during every third access, and so on for every value of FRMCNT, see [Figure 16-24](#).

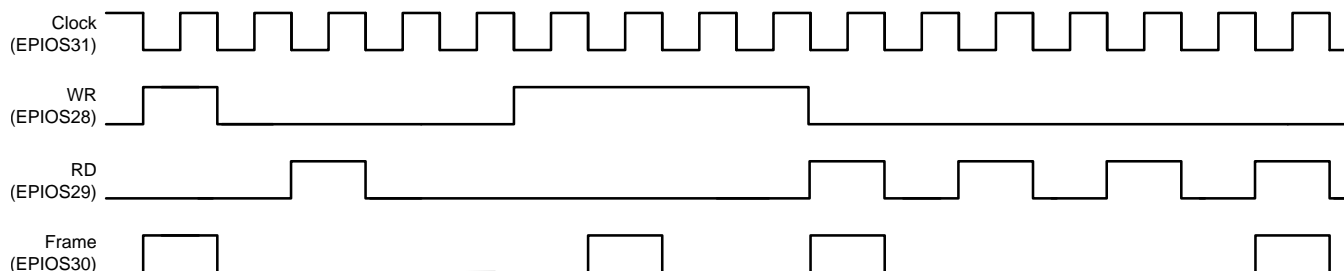


Figure 16-24. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 2

When FRM50 is set, the FRAME signal transitions on the rising edge of either the WR or RD strobes. When FRMCNT = 0, the FRAME signal transitions on the rising edge of WR or RD for every access, see [Figure 16-25](#).

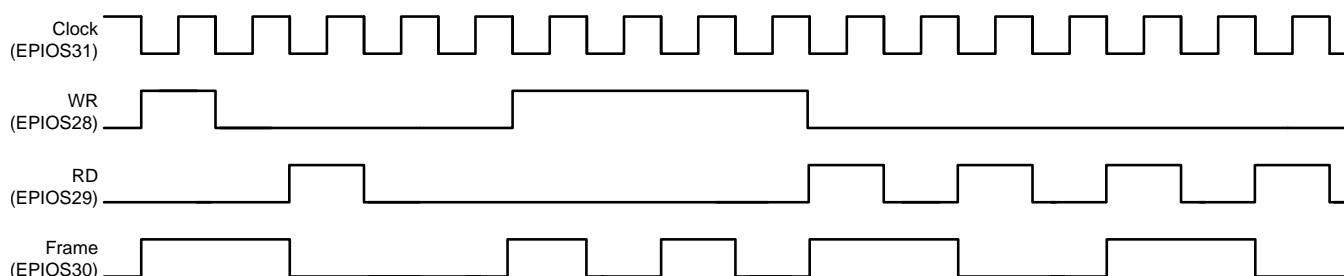


Figure 16-25. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 0

When FRMCNT = 1, the FRAME signal transitions on the rising edge of the WR or RD strobes for every other access, see [Figure 16-26](#).

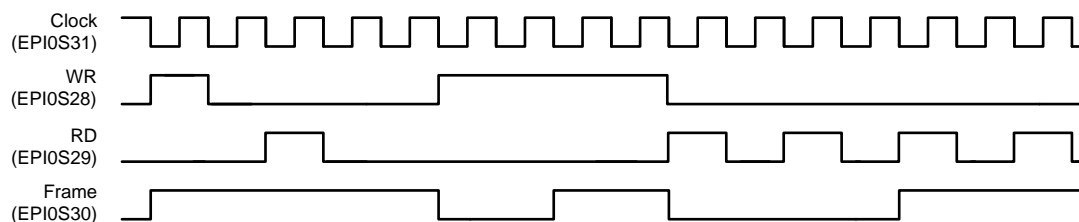


Figure 16-26. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 1

When FRMCNT = 2, the FRAME signal transitions the rising edge of the WR or RD strobes for every third access, and so on for every value of FRMCNT, see [Figure 16-27](#).

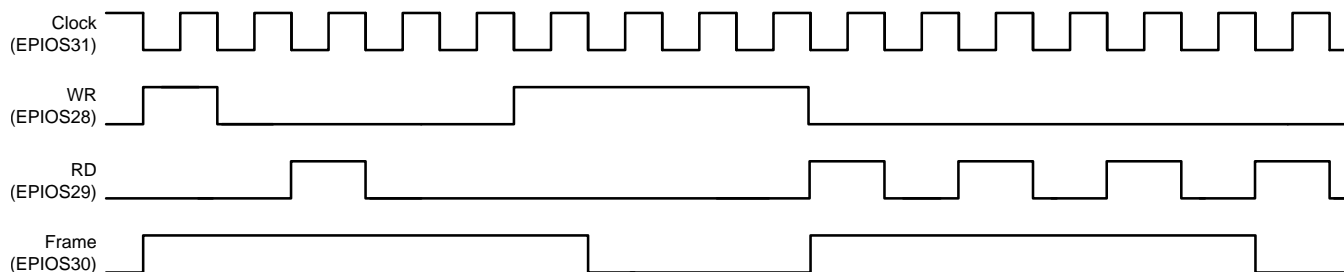


Figure 16-27. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 2

16.4.4.1.2 EPI Clock Operation

If the CLKGATE bit in the EPIGPCFG register is clear, the EPI clock always toggles when General-purpose mode is enabled. If CLKGATE is set, the clock is output only when a transaction is occurring, otherwise the clock is held high. If the WR2CYC bit is clear, the EPI clock begins toggling 1 cycle before the WR strobe goes High. If the WR2CYC bit is set, the EPI clock begins toggling when the WR strobe goes High. The clock stops toggling after the first rising edge after the WR strobe is deasserted. The RD strobe operates in the same manner as the WR strobe when the WR2CYC bit is set. See [Figure 16-28](#) and [Figure 16-29](#).

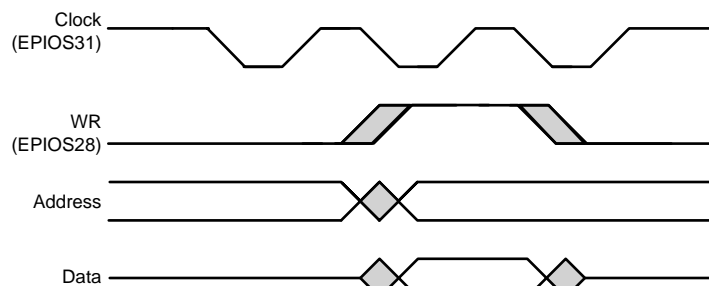


Figure 16-28. EPI Clock Operation, CLKGATE = 1, WR2CYC = 0

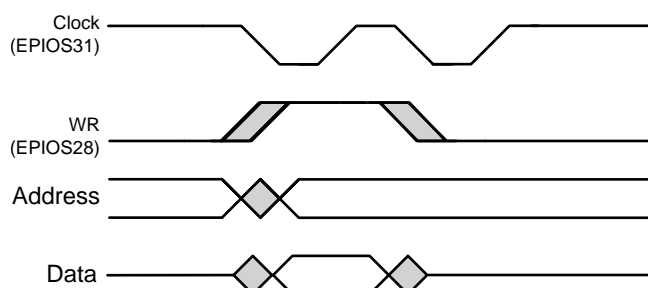


Figure 16-29. EPI Clock Operation, CLKGATE = 1, WR2CYC = 1

16.5 EPI Registers

[Table 16-12](#) lists the memory-mapped registers for the EPI. All register offset addresses not listed in [Table 16-12](#) should be considered as reserved locations and the register contents should not be modified.

The EPI controller clock must be enabled before the registers can be programmed (see [Section 4.2.89](#)). There must be a delay of 3 system clocks after the EPI module clock is enabled before any EPI module registers are accessed.

NOTE: A write immediately followed by a read of the same register may not return correct data. A delay (instruction or NOP) must be inserted between the write and the read for correct operation. Read-write does not have this issue, so use of read-write for clear of error interrupt cause is not affected.

NOTE: For all versions of EPI, only WORD read and write accesses to registers are supported.

Table 16-12. EPI Registers

Offset	Acronym	Register Name	Section
0x000	EPICFG	EPI Configuration	Section 16.5.1
0x004	EPIBAUD	EPI Main Baud Rate	Section 16.5.2
0x008	EPIBAUD2	EPI Main Baud Rate	Section 16.5.3
0x010	EPISDRAMCFG	EPI SDRAM Configuration	Section 16.5.4
0x010	EPIHB8CFG	EPI Host-Bus 8 Configuration	Section 16.5.5
0x010	EPIHB16CFG	EPI Host-Bus 16 Configuration	Section 16.5.6
0x010	EPIGPCFG	EPI General-Purpose Configuration	Section 16.5.7
0x014	EPIHB8CFG2	EPI Host-Bus 8 Configuration 2	Section 16.5.8
0x014	EPIHB16CFG2	EPI Host-Bus 16 Configuration 2	Section 16.5.9
0x01C	EPIADDRMAP	EPI Address Map	Section 16.5.10
0x020	EPIRSIZE0	EPI Read Size 0	Section 16.5.11
0x024	EPIRADDR0	EPI Read Address 0	Section 16.5.12
0x028	EPIRPSTD0	EPI Non-Blocking Read Data 0	Section 16.5.13
0x030	EPIRSIZE1	EPI Read Size 1	Section 16.5.11
0x034	EPIRADDR1	EPI Read Address 1	Section 16.5.12
0x038	EPIRPSTD1	EPI Non-Blocking Read Data 1	Section 16.5.13
0x060	EPISTAT	EPI Status	Section 16.5.14
0x06C	EPIRFIFOCNT	EPI Read FIFO Count	Section 16.5.15
0x70 to 0x8C	EPIREADFIFO0 to EPIREADFIFO7	EPI Read FIFO 0 to EPI Read FIFO 7	Section 16.5.16
0x200	EPIFIFOLVL	EPI FIFO Level Selects	Section 16.5.17
0x24	EPIWFIFOCNT	EPI Write FIFO Count	Section 16.5.18
0x28	EPIDMATXCNT	EPI DMA Transmit Count	Section 16.5.19
0x210	EPIIM	EPI Interrupt Mask	Section 16.5.20
0x214	EPIRIS	EPI Raw Interrupt Status	Section 16.5.21
0x218	EPIMIS	EPI Masked Interrupt Status	Section 16.5.22
0x21C	EPIEISC	EPI Error and Interrupt Status and Clear	Section 16.5.23
0x308	EPIHB8CFG3	EPI Host-Bus 8 Configuration 3	Section 16.5.24
0x308	EPIHB16CFG3	EPI Host-Bus 16 Configuration 3	Section 16.5.25
0x30C	EPIHB8CFG4	EPI Host-Bus 8 Configuration 4	Section 16.5.26
0x30C	EPIHB16CFG4	EPI Host-Bus 16 Configuration 4	Section 16.5.27
0x310	EPIHB8TIME	EPI Host-Bus 8 Timing Extension	Section 16.5.28
0x310	EPIHB16TIME	EPI Host-Bus 16 Timing Extension	Section 16.5.29
0x314	EPIHB8TIME2	EPI Host-Bus 8 Timing Extension	Section 16.5.30

Table 16-12. EPI Registers (continued)

Offset	Acronym	Register Name	Section
0x314	EPIHB16TIME2	EPI Host-Bus 16 Timing Extension	Section 16.5.31
0x318	EPIHB8TIME3	EPI Host-Bus 8 Timing Extension	Section 16.5.32
0x318	EPIHB16TIME3	EPI Host-Bus 16 Timing Extension	Section 16.5.33
0x31C	EPIHB8TIME4	EPI Host-Bus 8 Timing Extension	Section 16.5.34
0x31C	EPIHB16TIME4	EPI Host-Bus 16 Timing Extension	Section 16.5.35
0x360	EPIHBPSRAM	EPI Host-Bus PSRAM	Section 16.5.36

Complex bit access types are encoded to fit into small table cells. [Table 16-13](#) shows the codes that are used for access types in this section.

Table 16-13. EPI Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

16.5.1 EPICFG Register (Offset = 0x0) [reset = 0x0]

EPI Configuration (EPICFG)

NOTE: The MODE field determines which configuration register is accessed for offsets 0x010 and 0x014. Any write to the EPICFG register resets the register contents at offsets 0x010 and 0x014.

The configuration register is used to enable the block, select a mode, and select the basic pin use (based on the mode). Note that attempting to program an undefined MODE field clears the BLKEN bit and disables the EPI controller.

EPICFG is shown in [Figure 16-30](#) and described in [Table 16-14](#).

Return to [Summary Table](#).

Figure 16-30. EPICFG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							INTDIV
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			BLKEN	MODE			
R-0x0			R/W-0x0	R/W-0x0			

Table 16-14. EPICFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	INTDIV	R/W	0x0	Integer Clock Divider Enable 0x0 = EPIBAUD register values create formula clock divide. 0x1 = EPIBAUD register values create integer clock divide.
7-5	RESERVED	R	0x0	
4	BLKEN	R/W	0x0	Block Enable 0x0 = The EPI controller is disabled. 0x1 = The EPI controller is enabled.
3-0	MODE	R/W	0x0	Mode Select 0x0 = Reserved 0x1 = SDRAM. Supports SDR SDRAM. Control, address, and data pins are configured using the EPISDRAMCFG register. 0x2 = 8-Bit Host-Bus (HB8). Host-bus 8-bit interface (also known as the MCU interface). Control, address, and data pins are configured using the EPIHB8CFG and EPIHB8CFG2 registers. 0x3 = 16-Bit Host-Bus (HB16). Host-bus 16-bit interface (standard SRAM). Control, address, and data pins are configured using the EPIHB16CFG and EPIHB16CFG2 registers.

16.5.2 EPIBAUD Register (Offset = 0x4) [reset = 0x0]

EPI Main Baud Rate (EPIBAUD)

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 is the bit field used in all modes except in HB8 and HB16 modes with dual chip selects and quad chip selects when different baud rates are selected, see [Section 16.5.8](#) and [Section 16.5.9](#). If different baud rates are used, COUNT0 is associated with the address range specified by CS0n and COUNT1 is associated with the address range specified by CS1. The EPIBAUD2 register configures the baud rates for CS2n and CS3n.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,

$$\text{EPIClockFreq} = \text{SystemClockFreq} \quad (57)$$

otherwise:

$$\text{EPIClockFreq} = \frac{\text{SystemClockFreq}}{\left(\left\lfloor \frac{\text{COUNTn}}{2} \right\rfloor + 1 \right) \times 2} \quad (58)$$

where the symbol around COUNTn /2 is the floor operator, meaning the largest integer less than or equal to COUNTn /2.

So, for example, a COUNTn of 0x0001 results in a clock rate of 1/2 (system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of 1/4 (system clock).

The baud rate counter can also be configured as an integer divide by enabling INTDIV in the EPICFG register. When enabled, COUNTn of 0x0000 or 0x0001 results in a clock rate equal to system clock. COUNTn of 0x0002 results in a clock rate of 1/2 (system clock). COUNTn of 0x0003 results in a clock rate of 1/3 (system clock).

EPIBAUD is shown in [Figure 16-31](#) and described in [Table 16-15](#).

Return to [Summary Table](#).

Figure 16-31. EPIBAUD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT1																COUNT0															
R/W-0x0																R/W-0x0															

Table 16-15. EPIBAUD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	COUNT1	R/W	0x0	Baud Rate Counter 1 This bit field is only valid with multiple chip selects which are enabled when the CSCFG field is 0x2 or 0x3 or the CSCFGEXT field is set to 1, with CSCFG field as 0x1 or 0x2 and the CSBAUD bit is set in the EPIHBnCFG2 register. This bit field contains a counter used to divide the system clock by the count. A count of 0 means the system clock is used as is.
15-0	COUNT0	R/W	0x0	Baud Rate Counter 0 This bit field contains a counter used to divide the system clock by the count. A count of 0 means the system clock is used as is.

16.5.3 EPIBAUD2 Register (Offset = 0x8) [reset = 0x0]

EPI Main Baud Rate (EPIBAUD2)

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 and COUNT1 are used in quad chip select mode when different baud rates are selected, [Section 16.5.8](#) or [Section 16.5.9](#). If different baud rates are used, COUNT0 is associated with the address range specified by CS2n and COUNT1 is associated with the address range specified by CS3n.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,

$$\text{EPIClockFreq} = \text{SystemClockFreq} \quad (59)$$

otherwise:

$$\text{EPIClockFreq} = \frac{\text{SystemClockFreq}}{\left(\left\lfloor \frac{\text{COUNTn}}{2} \right\rfloor + 1 \right) \times 2} \quad (60)$$

where the symbol around COUNTn / 2 is the floor operator, meaning the largest integer less than or equal to COUNTn / 2.

So, for example, a COUNTn of 0x0001 results in a clock rate of 1/2(system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of 1/4 (system clock).

EPIBAUD2 is shown in [Figure 16-32](#) and described in [Table 16-16](#).

Return to [Summary Table](#).

Figure 16-32. EPIBAUD2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT1																COUNT0															
R/W-0x0																R/W-0x0															

Table 16-16. EPIBAUD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	COUNT1	R/W	0x0	CS3n Baud Rate Counter 1 This bit field contains a counter used to divide the system clock by the count. A count of 0 means the system clock is unchanged. This bit field is only valid when quad chip selects are enabled by setting the CSCFGEXT bit to 1 and the CSCFG field to 0x1 or 0x2. In addition, the CSBAUD bit must be set in the EPIHBnCFG2 register.
15-0	COUNT0	R/W	0x0	CS2n Baud Rate Counter 0 This bit field contains a counter used to divide the system clock by the count. A count of 0 means the system clock is unchanged. This bit field is only valid when quad chip selects are enabled by setting the CSCFGEXT to 1 and the CSCFG field to 0x1 or 0x2. In addition, the CSBAUD bit must be set in the EPIHBnCFG2 register.

16.5.4 EPISDRAMCFG Register (Offset = 0x10) [reset = 0x82EE0000]

EPI SDRAM Configuration (EPISDRAMCFG)

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPISDRAMCFG, the MODE field must be 0x1.

The SDRAM Configuration register is used to specify several parameters for the SDRAM controller. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

The SDRAM interface is designed to interface to x16 SDR SDRAMs of 64 MHz or higher, with the address and data pins overlapped (wire ORed on the board). See [Table 16-2](#) for pin assignments.

EPISDRAMCFG is shown in [Figure 16-33](#) and described in [Table 16-17](#).

Return to [Summary Table](#).

Figure 16-33. EPISDRAMCFG Register

31	30	29	28	27	26	25	24
FREQ	RESERVED				RFSH		
R/W-0x2	R-0x0				R/W-0x2EE		
23	22	21	20	19	18	17	16
RFSH							
R/W-0x2EE							
15	14	13	12	11	10	9	8
RESERVED						SLEEP	RESERVED
R-0x0						R/W-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED						SIZE	
R-0x0						R/W-0x0	

Table 16-17. EPISDRAMCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	FREQ	R/W	0x2	EPI Frequency Range. This field configures the frequency range used for delay references by internal counters. This EPI frequency is the system frequency with the divider programmed by the COUNT0 bit in the EPIBAUDn register bit. This field affects the power up, precharge, and auto refresh delays. This field does not affect the refresh counting, which is configured separately using the RFSH field (and is based on system clock rate and number of rows per bank). The ranges are: 0x0 = 0 to 15 MHz 0x1 = 15 to 30 MHz 0x2 = 30 to 50 MHz 0x3 = 50 to 100 MHz
29-27	RESERVED	R	0x0	
26-16	RFSH	R/W	0x2EE	Refresh Counter This field contains the refresh counter in EPI clocks. The reset value of 0x2EE provides a refresh period of 64 ms when using a 50 MHz EPI clock.
15-10	RESERVED	R	0x0	
9	SLEEP	R/W	0x0	Sleep Mode 0x0 = No effect. 0x1 = The SDRAM is put into low power state, but is self-refreshed.

Table 16-17. EPISDRAMCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8-2	RESERVED	R	0x0	
1-0	SIZE	R/W	0x0	Size of SDRAM The value of this field affects address pins and behavior. 0x0 = 64 megabits (8MB) 0x1 = 128 megabits (16MB) 0x2 = 256 megabits (32MB) 0x3 = 512 megabits (64MB)

16.5.5 EPIHB8CFG Register (Offset = 0x10) [reset = 0x0008FF00]

EPI Host-Bus 8 Configuration (EPIHB8CFG)

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB8CFG, the MODE field must be 0x2.

The Host Bus 8 Configuration register is activated when the HB8 mode is selected. The HB8 mode supports muxed address/data (overlay of lower 8 address and all 8 data pins), separate address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the HB8 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, CPLDs/FPGAs, and devices with an MCU/HostBus slave or 8-bit FIFO interface support.

Refer to [Table 16-7](#) for information on signal configuration controlled by this register and the EPIHB8CFG2 register.

If less address pins are required, the corresponding AFSEL bit ([Section 17.5.10](#)) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 8 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for Device 1 and 0 respectively, then address 0x68000000 accesses Device 0 (Device 1 has its CEn high), and 0x64000000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPIHB8CFG is shown in [Figure 16-34](#) and described in [Table 16-18](#).

Return to [Summary Table](#).

Figure 16-34. EPIHB8CFG Register

31	30	29	28	27	26	25	24
CLKGATE	CLKGATEI	CLKINV	RDYEN	IRDYINV	RESERVED		
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0		
23	22	21	20	19	18	17	16
XFFEN	XFEEN	WRHIGH	RDHIGH	ALEHIGH	RESERVED		
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1	R-0x0		
15	14	13	12	11	10	9	8
MAXWAIT							
R/W-0xFF							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-18. EPIHB8CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CLKGATE	R/W	0x0	<p>Clock Gated</p> <p>A software application should only set the CLKGATE bit when there are no pending transfers or no EPI register access has been issued.</p> <p>0x0 = The EPI clock is free running.</p> <p>0x1 = The EPI clock is held low.</p>
30	CLKGATEI	R/W	0x0	<p>Clock Gated when Idle</p> <p>Note that EPI0S32 is an iRDY signal if RDYEN is set. CLKGATEI is ignored if CLKPIN is 0 or if the COUNT0 field in the EPIBAUD register is cleared.</p> <p>0x0 = The EPI clock is free running.</p> <p>0x1 = The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.</p>
29	CLKINV	R/W	0x0	<p>Invert Output Clock Enable</p> <p>0x0 = No effect.</p> <p>0x1 = Invert EPI clock to ensure the rising edge is centered for outbound signal's setup and hold. Inbound signal is captured on rising edge EPI clock.</p>
28	RDYEN	R/W	0x0	<p>Input Ready Enable</p> <p>0x0 = No effect.</p> <p>0x1 = An external ready can be used to control the continuation of the current access. If this bit is set and the iRDY signal (EPI0S32) is low, the current access is stalled.</p>
27	IRDYINV	R/W	0x0	<p>Input Ready Invert</p> <p>0x0 = No effect.</p> <p>0x1 = Invert the polarity of incoming external ready (iRDY signal). If this bit is set and the iRDY signal (EPI0S32) is high the current access is stalled.</p>
26-24	RESERVED	R	0x0	
23	XFFEN	R/W	0x0	<p>External FIFO FULL Enable</p> <p>0x0 = No effect.</p> <p>0x1 = An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL full signal is high, XFIFO writes are stalled.</p>
22	XFEEN	R/W	0x0	<p>External FIFO EMPTY Enable</p> <p>0x0 = No effect.</p> <p>0x1 = An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled.</p>
21	WRHIGH	R/W	0x0	<p>WRITE Strobe Polarity</p> <p>0x0 = The WRITE strobe for CS0n is WRn (active Low).</p> <p>0x1 = The WRITE strobe for CS0n is WR (active High).</p>
20	RDHIGH	R/W	0x0	<p>READ Strobe Polarity</p> <p>0x0 = The READ strobe for CS0n is RDn (active Low).</p> <p>0x1 = The READ strobe for CS0n is RD (active High).</p>
19	ALEHIGH	R/W	0x1	<p>ALE Strobe Polarity</p> <p>0x0 = The address latch strobe for CS0n accesses is ALEn (active Low).</p> <p>0x1 = The address latch strobe for CS0n accesses is ALE (active High).</p>
18-16	RESERVED	R	0x0	

Table 16-18. EPIHB8CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-8	MAXWAIT	R/W	0xFF	<p>Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY). When the MAXWAIT value is reached the ERRRIS interrupt status bit is set in the EPIRIS register. When this field is clear, the transaction can be held off forever without a system interrupt. When the MODE field is configured to be 0x2 and the BLKEN bit is set in the EPICFG register, enabling HB8 mode, this field defaults to 0xFF.</p>
7-6	WRWS	R/W	0x0	<p>Write Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode. This field is used in conjunction with the EPIBAUD register. 0x0 = Active WRn is 2 EPI clocks. 0x1 = Active WRn is 4 EPI clocks. 0x2 = Active WRn is 6 EPI clocks. 0x3 = Active WRn is 8 EPI clocks.</p>
5-4	RDWS	R/W	0x0	<p>Read Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode. This field is used in conjunction with the EPIBAUD register 0x0 = Active RDn is 2 EPI clocks. 0x1 = Active RDn is 4 EPI clocks. 0x2 = Active RDn is 6 EPI clocks. 0x3 = Active RDn is 8 EPI clocks.</p>
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	<p>Host Bus Sub-Mode This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals. When used with multiple chip select option and the CSBAUD bit is set to 1 in the EPIHB8CFG2 register, this configuration is for CS0n. If the multiple chip select option is enabled and CSBAUD is clear, all chip-selects use the MODE encoding programmed in this register. 0x0 = ADMUX - AD[7:0]. Data and Address are muxed. 0x1 = ADNONMUX - D[7:0]. Data and address are separate. 0x2 = Continuous Read - D[7:0]. This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. 0x3 = XFIFO - D[7:0]. This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE. The XFIFO can only be used in asynchronous mode.</p>

16.5.6 EPIHB16CFG Register (Offset = 0x10) [reset = 0x0008FF00]

EPI Host-Bus 16 Configuration (EPIHB16CFG)

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB16CFG, the MODE field must be 0x3.

The Host Bus 16 sub-configuration register is activated when the HB16 mode is selected. The HB16 mode supports muxed address/data (overlay of lower 16 address and all 16 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the HB16 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, and CPLDs/FPGAs, and devices with an MCU/HostBus slave or 16-bit FIFO interface support.

Refer to [Table 16-8](#) for information on signal configuration controlled by this register and the EPIHB16CFG2 register.

If less address pins are required, the corresponding AFSEL bit ([Section 17.5.10](#)) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 16 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for Device 1 and 0 respectively, then address 0x68000000 accesses Device 0 (Device 1 has its CEn high), and 0x64000000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPIHB16CFG is shown in [Figure 16-35](#) and described in [Table 16-19](#).

Return to [Summary Table](#).

Figure 16-35. EPIHB16CFG Register

31	30	29	28	27	26	25	24
CLKGATE	CLKGATEI	CLKINV	RDYEN	IRDYINV	RESERVED		
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0		
23	22	21	20	19	18	17	16
XFFEN	XFEEN	WRHIGH	RDHIGH	ALEHIGH	WRCRE	RDCRE	BURST
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
MAXWAIT							
R/W-0xFF							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED	BSEL	MODE	
R/W-0x0		R/W-0x0		R-0x0	R/W-0x0	R/W-0x0	

Table 16-19. EPIHB16CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CLKGATE	R/W	0x0	Clock Gated A software application should only set the CLKGATE bit when there are no pending transfers or no EPI register access has been issued. 0x0 = The EPI clock is free running. 0x1 = The EPI clock is held low.
30	CLKGATEI	R/W	0x0	Clock Gated Idle Note that EPI0S32 is an iRDY signal if RDYEN is set. CLKGATEI is ignored if CLKPIN is 0 or if the COUNT0 field in the EPIBAUD register is cleared. 0x0 = The EPI clock is free running. 0x1 = The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.
29	CLKINV	R/W	0x0	Invert Output Clock Enable If operating in asynchronous mode, CLKINV must be 0. 0x0 = No effect. 0x1 = Invert EPI clock to ensure the rising edge is centered for outbound signal's setup and hold. Inbound signal is captured on rising edge EPI clock.
28	RDYEN	R/W	0x0	Input Ready Enable 0x0 = No effect. 0x1 = An external ready (iRDY) can be used to control the continuation of the current access. If this bit is set and the iRDY signal (EPIS032) is low, the current access is stalled.
27	IRDYINV	R/W	0x0	Input Ready Invert 0x0 = No effect. 0x1 = Invert polarity of incoming external ready. If this bit is set and the iRDY signal (EPIS032) is high the current access is stalled.
26-24	RESERVED	R	0x0	
23	XFFEN	R/W	0x0	External FIFO FULL Enable 0x0 = No effect. 0x1 = An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL signal is high, XFIFO writes are stalled.
22	XFEEN	R/W	0x0	External FIFO EMPTY Enable 0x0 = No effect. 0x1 = An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled.
21	WRHIGH	R/W	0x0	WRITE Strobe Polarity 0x0 = The WRITE strobe for CS0n is WRn (active Low). 0x1 = The WRITE strobe for CS0n is WR (active High).
20	RDHIGH	R/W	0x0	READ Strobe Polarity 0x0 = The READ strobe for CS0n is RDn (active Low). 0x1 = The READ strobe for CS0n is RD (active High).
19	ALEHIGH	R/W	0x1	ALE Strobe Polarity 0x0 = The address latch strobe for CS0n is ALEn (active Low). 0x1 = The address latch strobe for CS0n is ALE (active High).
18	WRCRE	R/W	0x0	PSRAM Configuration Register Write Used for PSRAM configuration registers. With WRCRE set, the next transaction by the EPI will be a write of the CR bit field in the EPIHBPSRAM register to the configuration register (CR) of the PSRAM. The WRCRE bit will self clear once the write-enabled CRE access is complete. 0x0 = No Action. 0x1 = Start CRE write transaction for CS0n.

Table 16-19. EPIHB16CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	RDCRE	R/W	0x0	<p>PSRAM Configuration Register Read Enables read of PSRAM configuration registers.</p> <p>With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR).</p> <p>This bit self clears once the read-enabled CRE access is complete.</p> <p>The address for the CRE access is located at EPIHBPSRAM [19:18].</p> <p>The read data is returned on EPIHBPSRAM [15:0].</p> <p>0x0 = No Action.</p> <p>0x1 = Start CRE read transaction for CS0n.</p>
16	BURST	R/W	0x0	<p>Burst Mode Burst mode must be used with an ALE-enabled interface.</p> <p>Burst mode must be used with ADMUX, which is configured by the MODE field in the EPIHB16CFG register.</p> <p>Burst mode is optimized for word-length accesses.</p> <p>0x0 = Burst mode is disabled.</p> <p>0x1 = Burst mode is enabled for CS0n or single chip access.</p>
15-8	MAXWAIT	R/W	0xFF	<p>Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY).</p> <p>When this field is clear, the transaction can be held off forever without a system interrupt.</p> <p>When the MODE field is configured to be 0x3 and the BLKEN bit is set in the EPICFG register, enabling HB16 mode, this field defaults to 0xFF.</p>
7-6	WRWS	R/W	0x0	<p>Write Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected).</p> <p>The effect is to delay the rising edge of WRn (or the falling edge of WR).</p> <p>Each wait state adds 2 EPI clock cycles to the access time.</p> <p>The WRWSM bit EPIHB16TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity.</p> <p>This field is not applicable in BURST mode.</p> <p>This field is used in conjunction with the EPIBAUD register.</p> <p>0x0 = Active WRn is 2 EPI clocks.</p> <p>0x1 = Active WRn is 4 EPI clocks.</p> <p>0x2 = Active WRn is 6 EPI clocks.</p> <p>0x3 = Active WRn is 8 EPI clocks.</p>
5-4	RDWS	R/W	0x0	<p>Read Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected).</p> <p>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <p>Each wait state adds 2 EPI clock cycles to the access time.</p> <p>The RDWSM bit in the EPIHB16TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity.</p> <p>This field is not applicable in BURST mode.</p> <p>This field is used in conjunction with the EPIBAUD register</p> <p>0x0 = Active RDn is 2 EPI clocks.</p> <p>0x1 = Active RDn is 4 EPI clocks.</p> <p>0x2 = Active RDn is 6 EPI clocks.</p> <p>0x3 = Active RDn is 8 EPI clocks.</p>
3	RESERVED	R	0x0	
2	BSEL	R/W	0x0	<p>Byte Select Configuration This bit enables byte select operation.</p> <p>If BSEL = 0, byte accesses cannot be executed.</p> <p>0x0 = No Byte SelectsData is read and written as 16 bits.</p> <p>0x1 = Enable Byte SelectsTwo EPI signals function as byte select signals to allow 8-bit transfers. See for details on which EPI signals are used.</p>

Table 16-19. EPIHB16CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	MODE	R/W	0x0	<p>Host Bus Sub-Mode This field determines which of three Host Bus 16 sub-modes to use.</p> <p>Submode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals.</p> <p>When used with multiple chip select option and the CSBAUD bit is set to 1 in the EPIHB16CFG2 register, this configuration is for CS0n. If the multiple chip select option is enabled and CSBAUD is clear, all chip-selects use the MODE encoding programmed in this register.</p> <p>0x0 = ADMUX - AD[15:0]Data and Address are muxed.</p> <p>0x1 = ADNONMUX - D[15:0]Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.</p> <p>0x2 = Continuous Read - D[15:0]This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.</p> <p>0x3 = XFIFO - D[15:0]This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.Note that the XFIFO can only be used in asynchronous mode.</p>

16.5.7 EPIGPCFG Register (Offset = 0x10) [reset = 0x0]

EPI General-Purpose Configuration (EPIGPCFG)

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIGPCFG, the MODE field must be 0x0.

The General-Purpose configuration register is used to configure the control, data, and address pins. This mode can be used for custom interfaces with FPGAs, CPLDs, and for digital data acquisition and actuator control. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the General-purpose mode is selected again, the register the values must be reinitialized.

This mode is designed for 3 general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs, with 3 sizes of data and optional address. Framing and clock-enable permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the baud rate in the EPIBAUD register (when used with the NBRFIFO and/or the WFIFO) or by rate of accesses from software or uDMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free running or gated), a framing signal (with frame size), a ready input (to stretch transactions), read and write strobes, address of varying sizes, and data of varying sizes. Additionally, provisions are made for splitting address and data phases on the external interface.

EPIGPCFG is shown in [Figure 16-36](#) and described in [Table 16-20](#).

Return to [Summary Table](#).

Figure 16-36. EPIGPCFG Register

31	30	29	28	27	26	25	24
CLKPIN	CLKGATE	RESERVED				FRM50	FRMCNT
R/W-0x0	R/W-0x0	R-0x0				R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
FRMCNT		RESERVED			WR2CYC	RESERVED	
R/W-0x0		R-0x0			R/W-0x0	R-0x0	
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		ASIZE			RESERVED		DSIZE
R-0x0		R/W-0x0			R-0x0		R/W-0x0

Table 16-20. EPIGPCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CLKPIN	R/W	0x0	<p>Clock Pin The EPI clock is generated from the COUNT0 field in the EPIBAUD register (as is the system clock which is divided down from it).</p> <p>0x0 = No clock output.</p> <p>0x1 = EPI0S31 functions as the EPI clock output.</p>

Table 16-20. EPIGPCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
30	CLKGATE	R/W	0x0	Clock Gated CLKGATE is ignored if CLKPIN is 0 or if the COUNT0 field in the EPIBAUD register is cleared. 0x0 = The EPI clock is free running. 0x1 = The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.
29-27	RESERVED	R	0x0	
26	FRM50	R/W	0x0	50/50 Frame 0x0 = The FRAME signal is output as a single pulse, and then held low for the count. 0x1 = The FRAME signal is output as 50/50 duty cycle using count (see FRMCNT).
25-22	FRMCNT	R/W	0x0	Frame Count This field specifies the size of the frame in EPI clocks. The frame counter is used to determine the frame size. The count is FRMCNT +1. So, a FRMCNT of 0 forms a pure transaction valid signal (held high during transactions, low otherwise). A FRMCNT of 0 with FRM50 set inverts the FRAME signal on each transaction. A FRMCNT of 1 means the FRAME signal is inverted every other transaction a value of 15 means every sixteenth transaction. If FRM50 is set, the frame is held high for FRMCNT +1 transactions, then held low for that many transactions, and so on. If FRM50 is clear, the frame is pulsed high for one EPI clock and then low for FRMCNT EPI clocks.
21-20	RESERVED	R	0x0	
19	WR2CYC	R/W	0x0	2-Cycle Writes When this bit is set, then the RW bit is forced to be set. 0x0 = Data is output on the same EPI clock cycle as the address. EPI clock begins toggling one cycle before the WR strobe goes High. 0x1 = Writes are two EPI clock cycles long, with address on one EPI clock cycle (with the WR strobe asserted) and data written on the following EPI clock cycle (with WR strobe deasserted). The next address (if any) is in the cycle following. If the WR2CYC bit is set, the EPI clock begins toggling when the WR strobe goes High.
18-6	RESERVED	R	0x0	
5-4	ASIZE	R/W	0x0	Address Bus Size This field defines the size of the address bus. The address can be up to 4-bits wide with a 24-bit data bus, up to 12-bits wide with a 16-bit data bus, and up to 20-bits wide with an 8-bit data bus. If the full address bus is not used, use the least significant address bits. Any unused address bits can be used as GPIOs by clearing the AFSEL bit for the corresponding GPIOs. 0x0 = No address 0x1 = Up to 4 bits wide. 0x2 = Up to 12 bits wide. This size cannot be used with 24-bit data. 0x3 = Up to 20 bits wide. This size cannot be used with data sizes other than 8.
3-2	RESERVED	R	0x0	
1-0	DSIZE	R/W	0x0	Size of Data Bus This field defines the size of the data bus (starting at EPI0S0). Subsets of these numbers can be created by clearing the AFSEL bit for the corresponding GPIOs. Size 32 may not be used with clock, frame, address, or other control. 0x0 = 8 bits wide (EPI0S0 to EPI0S7) 0x1 = 16 bits wide (EPI0S0 to EPI0S15) 0x2 = 24 bits wide (EPI0S0 to EPI0S23) 0x3 = 32 bits wide (EPI0S0 to EPI0S31) This size may not be used with an EPI clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.

16.5.8 EPIHB8CFG2 Register (Offset = 0x14) [reset = 0x00080000]

EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2)

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB8CFG2, the MODE field of the EPICFG register must be 0x2.

This register is used to configure operation while in Host-Bus 8 mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the Host-Bus 8 mode is selected again, the values must be reinitialized.

EPIHB8CFG2 is shown in [Figure 16-37](#) and described in [Table 16-21](#).

Return to [Summary Table](#).

Figure 16-37. EPIHB8CFG2 Register

31	30	29	28	27	26	25	24
RESERVED				CSCFGEXT	CSBAUD	CSCFG	
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	
23	22	21	20	19	18	17	16
RESERVED		WRHIGH	RDHIGH	ALEHIGH	RESERVED		
R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R-0x0		
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-21. EPIHB8CFG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0x0	
27	CSCFGEXT	R/W	0x0	<p>Chip Select Extended Configuration</p> <p>This field is used in conjunction with CSCFG, to extend the chip select options, and ALE format. The values 0x0 through 0x3 are from the CSCFG field. The CSCFGEXT bit extends the values to 0x7.</p> <p>0x0 = CSCFG bit field is used in chip select configuration.</p> <p>0x1 = The CSCFG bit field is extended with CSCFGEXT representing the MSB.</p> <p>The possible chip select configurations (CSCFGEXT + CSCFG encodings) when the CSCFGEXT bit is enabled are:</p> <p>0x0 = ALE Configuration. EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the EPIHB8CFG register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.</p> <p>0x1 = CSn Configuration. EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (MODE field in the EPIHB8CFG register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low.</p> <p>0x2 = Dual CSn Configuration. EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.</p> <p>0x3 = ALE with Dual CSn Configuration. EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.</p> <p>0x4 = ALE with Single CSn Configuration. EPI0S30 is used as address latch (ALE) and EPI0S27 is used as CSn.</p> <p>0x5 = Quad CSn Configuration. EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n.</p> <p>0x6 = ALE with Quad CSn Configuration. EPI0S30 is used as ALE, EPI0S26 is CS0n, and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n.</p> <p>0x7 = Reserved</p>

Table 16-21. EPIHB8CFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	CSBAUD	R/W	0x0	<p>Chip Select Baud Rate and Multiple Sub-Mode Configuration enable</p> <p>This bit is only valid when the CSCFGEXT + CSCFG field is programmed to 0x2 or 0x3, 0x5 or 0x6.</p> <p>This bit configures the baud rate settings for CS0n, CS1n, CS2n, and CS3n.</p> <p>This bit must also be set to allow different sub-mode configurations on chip-selects.</p> <p>If this bit is clear, all chip-select sub-modes are based on the MODE encoding defined in the EPI8HBCFG register.</p> <p>If the CSBAUD bit is set in the EPIHBnCFG2 register and dual- or quad-chip selects are enabled, then the individual chip selects can use different clock frequencies, wait states and strobe polarity.</p> <p>0x0 = Same Baud Rate and Same Sub-ModeAll CSn use the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register and the sub-mode programmed in the MODE field of the EPIHB8CFG register.</p> <p>0x1 = Different Baud RatesCS0n uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register. CS1n uses the baud rate defined by the COUNT1 field in the EPIBAUD register. CS2n uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD2 register. CS3n uses the baud rate defined by the COUNT1 field in the EPIBAUD2 register. In addition, the sub-modes for each chip select are individually programmed in their respective EPIHB8CFGn registers.</p>
25-24	CSCFG	R/W	0x0	<p>Chip Select Configuration.</p> <p>This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects.</p> <p>These bits are also used in combination with the CSCFGEXT bit for further configurations, including quad- chip select.</p> <p>0x0 = ALE Configuration. EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (HB8MODE field in the EPIHB8CFG register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.</p> <p>0x1 = CSn Configuration. EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (HB8MODE field in the EPIHB8CFG register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low.</p> <p>0x2 = Dual CSn Configuration. EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by two methods. If only external RAM or external PER is enabled in the address map, the most significant address bit for a respective external address map controls CS0n or CS1n. If both external RAM and external PER is enabled, CS0n is mapped to PER and CS1n is mapped to RAM. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.</p> <p>0x3 = ALE with Dual CSn Configuration. EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.</p>
23-22	RESERVED	R	0x0	
21	WRHIGH	R/W	0x0	<p>CS1n WRITE Strobe Polarity This field is used if the CSBAUD bit in the EPIHB8CFG2 register is enabled.</p> <p>0x0 = The WRITE strobe for CS1n accesses is WRn (active Low).</p> <p>0x1 = The WRITE strobe for CS1n accesses is WR (active High).</p>
20	RDHIGH	R/W	0x0	<p>CS1n READ Strobe Polarity This field is used if the CSBAUD bit in the EPIHB8CFG2 register is enabled.</p> <p>0x0 = The READ strobe for CS1n accesses is RDn (active Low).</p> <p>0x1 = The READ strobe for CS1n accesses is RD (active High).</p>

Table 16-21. EPIHB8CFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	ALEHIGH	R/W	0x1	CS1n ALE Strobe Polarity This field is used if the CSBAUD bit in the EPIHB8CFG2 register is enabled. 0x0 = The address latch strobe for CS1n accesses is ALEn (active Low). 0x1 = The address latch strobe for CS1n accesses is ALE (active High).
18-8	RESERVED	R	0x0	
7-6	WRWS	R/W	0x0	CS1n Write Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME2 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode. 0x0 = Active WRn is 2 EPI clocks. 0x1 = Active WRn is 4 EPI clocks 0x2 = Active WRn is 6 EPI clocks 0x3 = Active WRn is 8 EPI clocks
5-4	RDWS	R/W	0x0	CS1n Read Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state encoding adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME2 register can decrease the number of states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode. 0x0 = Active RDn is 2 EPI clocks 0x1 = Active RDn is 4 EPI clocks 0x2 = Active RDn is 6 EPI clocks 0x3 = Active RDn is 8 EPI clocks
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	CS1n Host Bus Sub-Mode This field determines which Host Bus 8 sub-mode to use for CS1n. Sub-mode use is determined by the externally connected peripheral or memory. See for information on how this bit field affects the operation of the EPI signals. The CSBAUD bit must be set to enable this CS1n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB8CFG register. 0x0 = reserved 0x1 = ADNONMUX - D[7:0]Data and address are separate.

16.5.9 EPIHB16CFG2 Register (Offset = 0x14) [reset = 0x00080000]

EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2)

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB16CFG2, the MODE field must be 0x3.

This register is used to configure operation while in Host-Bus 16 mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the Host-Bus 16 mode is selected again, the values must be reinitialized.

EPIHB16CFG2 is shown in [Figure 16-38](#) and described in [Table 16-22](#).

Return to [Summary Table](#).

Figure 16-38. EPIHB16CFG2 Register

31	30	29	28	27	26	25	24
RESERVED				CSCFGEXT	CSBAUD	CSCFG	
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	
23	22	21	20	19	18	17	16
RESERVED		WRHIGH	RDHIGH	ALEHIGH	WRCRE	RDCRE	BURST
R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-22. EPIHB16CFG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0x0	
27	CSCFGEXT	R/W	0x0	<p>Chip Select Extended Configuration.</p> <p>This field is used in conjunction with CSCFG, to extend the chip select options, and ALE format. The values 0x0 through 0x3 are from the CSCFG field. The CSCFGEXT bit extends the values to 0x7.</p> <p>0x0 = CSCFG bit field is used in chip select configuration.</p> <p>0x1 = The CSCFG bit field is extended with CSCFGEXT representing the MSB.</p> <p>The possible chip select configurations (CSCFGEXT + CSCFG encodings) when the CSCFGEXT bit is enabled are:</p> <p>0x0 = ALE Configuration. EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the EPIHB16CFG register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.</p> <p>0x1 = CSn Configuration. EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (MODE field in the EPIHB16CFG register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low.</p> <p>0x2 = Dual CSn Configuration. EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.</p> <p>0x3 = ALE with Dual CSn Configuration. EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.</p> <p>0x4 = ALE with Single CSn Configuration. EPI0S30 is used as address latch (ALE) and EPI0S27 is used as CSn.</p> <p>0x5 = Quad CSn Configuration. EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n.</p> <p>0x6 = ALE with Quad CSn Configuration. EPI0S30 is used as ALE, EPI0S26 is CS0n, and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n.</p> <p>0x7 = Reserved</p>

Table 16-22. EPIHB16CFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	CSBAUD	R/W	0x0	<p>Chip Select Baud Rate and Multiple Sub-Mode Configuration enable. This bit is only valid when the CSCFGEXT + CSCFG field is programmed to 0x2 or 0x3, 0x5 or 0x6.</p> <p>This bit configures the baud rate settings for CS0n, CS1n, CS2n, and CS3n.</p> <p>This bit must also be set to allow different sub-mode configurations on chip-selects.</p> <p>If this bit is clear, all chip-select sub-modes are based on the MODE encoding defined in the EPI8HBCFG register.</p> <p>If the CSBAUD bit is set in the EPIHBnCFG2 register and dual- or quad-chip selects are enabled, then the individual chip selects can use different clock frequencies, wait states and strobe polarity.</p> <p>0x0 = Same Baud Rate and Same Sub-Mode. All CSn use the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register and the sub-mode programmed in the MODE field of the EPIHB16CFG register.</p> <p>0x1 = Different Baud Rates. CS0n uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register. CS1n uses the baud rate defined by the COUNT1 field in the EPIBAUD register. CS2n uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD2 register. CS3n uses the baud rate defined by the COUNT1 field in the EPIBAUD2 register. In addition, the sub-modes for each chip select are individually programmed in their respective EPIHB16CFGn registers.</p>
25-24	CSCFG	R/W	0x0	<p>Chip Select Configuration This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects.</p> <p>These bits are also used in combination with the CSCFGEXT bit for further configurations, including quad- chip select.</p> <p>0x0 = ALE Configuration. EPI0S30 is used as an address latch (ALE). When using this mode, the address and data should be muxed (HB16MODE field in the EPIHB16CFG register should be configured to 0x0). If needed, the address can be latched by external logic.</p> <p>0x1 = CSn Configuration. EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (MODE field in the EPIHB16CFG register should be configured to 0x1). In this mode, the WR signal (EPI0S29) and the RD signal (EPI0S28) are used to latch the address when CSn is low.</p> <p>0x2 = Dual CSn Configuration. EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.</p> <p>0x3 = ALE with Dual CSn Configuration. EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.</p>
23-22	RESERVED	R	0x0	
21	WRHIGH	R/W	0x0	<p>CS1n WRITE Strobe Polarity. This field is used if CSBAUD bit of the EPIHB16CFG2 register is enabled.</p> <p>0x0 = The WRITE strobe for CS1n accesses is WRn (active Low).</p> <p>0x1 = The WRITE strobe for CS1n accesses is WR (active High).</p>
20	RDHIGH	R/W	0x0	<p>CS1n READ Strobe Polarity. This field is used if CSBAUD bit of the EPIHB16CFG2 register is enabled.</p> <p>0x0 = The READ strobe for CS1n accesses is RDn (active Low).</p> <p>0x1 = The READ strobe for CS1n accesses is RD (active High).</p>

Table 16-22. EPIHB16CFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	ALEHIGH	R/W	0x1	CS1n ALE Strobe Polarity. This field is used if CSBAUD bit of the EPIHB16CFG2 register is enabled. 0x0 = The address latch strobe for CS1n accesses is ALEn (active Low). 0x1 = The address latch strobe for CS1n accesses is ALE (active High).
18	WRCRE	R/W	0x0	CS1n PSRAM Configuration Register Write Used for the PSRAM configuration registers (CR). With WRCRE set, the next transaction by the EPI is a write of the CR bit field in the EPIHBPSRAM register to the configuration register (CR) of the PSRAM. The WRCRE bit self clears once the write-enabled CRE access is complete. 0x0 = No Action. 0x1 = Start CRE write transaction for CS1n.
17	RDCRE	R/W	0x0	CS1n PSRAM Configuration Register Read Used for the PSRAM configuration registers (CR). With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the CRE access is complete. The address for the CRE access is located at EPIHBPSRAM [19:18]. The read data is returned on EPIHBPSRAM [15:0]. 0x0 = No Action. 0x1 = Start CRE read transaction for CS1n.
16	BURST	R/W	0x0	CS1n Burst Mode Burst mode must be used with an ALE which is configured by programming the CSCFG and CSCFGEXT fields in the EPIHB16CFG2 register. Burst mode must be used in ADMUX, which is set by the MODE field in EPIHB16CFG2. Burst mode is optimized for word-length accesses. 0x0 = Burst mode is disabled. 0x1 = Burst mode is enabled for CS1n.
15-8	RESERVED	R	0x0	
7-6	WRWS	R/W	0x0	CS1n Write Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB16TIME2 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode. 0x0 = Active WRn is 2 EPI clocks 0x1 = Active WRn is 4 EPI clocks. 0x2 = Active WRn is 6 EPI clocks 0x3 = Active WRn is 8 EPI clocks

Table 16-22. EPIHB16CFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	RDWS	R/W	0x0	<p>CS1n Read Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <p>Each wait state encoding adds 2 EPI clock cycles to the access time.</p> <p>The RDWSM bit in the EPIHB16TIME2 register can decrease the number of states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register.</p> <p>This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode.</p> <p>0x0 = Active RDn is 2 EPI clocks 0x1 = Active RDn is 4 EPI clocks 0x2 = Active RDn is 6 EPI clocks 0x3 = Active RDn is 8 EPI clocks</p>
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	<p>CS1n Host Bus Sub-Mode This field determines which Host Bus 16 sub-mode to use for CS1n.</p> <p>Sub-mode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals.</p> <p>When used with multiple chip select option this configuration is for CS1n.</p> <p>The CSBAUD bit must be set to enable this CS1n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB16CFG register.</p> <p>0x0 = reserved 0x1 = ADNONMUX - D[15:0]Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.</p>

16.5.10 EPIADDRMAP Register (Offset = 0x1C) [reset = 0x0]

EPI Address Map (EPIADDRMAP)

This register enables address mapping. The EPI controller can directly address memory and peripherals. In addition, the EPI controller supports address mapping to allow indirect accesses in the External RAM and External Peripheral areas.

If the external device is a peripheral, including a FIFO or a directly addressable device, the EPSZ and EPADR bit fields should be configured for the address space. If the external device is SDRAM, SRAM, or NOR Flash memory, the ERADR and ERSZ bit fields should be configured for the address space.

If one of the dual chip select modes is selected (CSCFGEXT is 0x0 and CSCFG is 0x2 or 0x3 in the EPIHBnCFG2 register), both chip selects can share the peripheral or the memory space, or one chip select can use the peripheral space and the other can use the memory space. In the EPIADDRMAP register, if the EPADR field is not 0x0, the ECADR field is 0x0, and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with CS0n being asserted when the MSB of the address range is 0 and CS1n being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0, the ECADR field is 0x0, and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0 and the ECADR field is 0x0, then CS0n is asserted for either address range defined by EPADR and CS1n is asserted for either address range defined by ERADR. The two chip selects can also be shared between the code space and memory or peripheral space. If the ECADR field is 0x1, ERADR field is 0x0, and the EPADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by EPADR. If the ECADR field is 0x1, EPADR field is 0x0, and the ERADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by ERADR.

If one of the Quad-Chip-Select modes is selected (CSCFGEXT is 0x1 and CSCFG is 0x2 or 0x3 in the EPIHBnCFG2 register), both the peripheral and the memory space must be enabled. In the EPIADDRMAP register, the EPADR field is 0x3, the ERADR field is 0x3, and the ECADR field is 0x0. In this case, CS0n maps to 0x60000000; CS1n maps to 0x80000000; CS2n maps to 0xA0000000; and CS3n maps to 0xC0000000. The MODE field of the EPIHBnCFGn registers configures the interface for the individual chip selects, which support ADMUX or ADNOMUX. If the CSBAUD bit is clear, all chip selects use the mode configured in the MODE bit field of the EPIHBnCFG register. [Table 16-4](#) gives a detailed explanation of chip select address range mappings based on which combinations of peripheral and memory space are enabled.

EPIADDRMAP is shown in [Figure 16-39](#) and described in [Table 16-23](#).

Return to [Summary Table](#).

Figure 16-39. EPIADDRMAP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				ECSZ		ECADR		EPSZ		EPADR		ERSZ		ERADR	
R-0x0				R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	

Table 16-23. EPIADDRMAP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11-10	ECSZ	R/W	0x0	<p>External Code Size This field selects the size of the external code. If the size of the external code is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused).</p> <p>When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries.</p> <p>As a result, the available address space is double the amount shown below.</p> <p>0x0 = 256 bytes; lower address range: 0x00 to 0xFF 0x1 = 64 KB; lower address range: 0x0000 to 0xFFFF 0x2 = 16 MB; lower address range: 0x000000 to 0xFFFFFFF 0x3 = 256MB; lower address range: 0x00000000 to 0xFFFFFFF</p>
9-8	ECADR	R/W	0x0	<p>External Code Address This field selects address mapping for the external code area.</p> <p>0x0 = Not mapped 0x1 = At 0x10000000 0x2 = reserved 0x3 = reserved</p>
7-6	EPSZ	R/W	0x0	<p>External Peripheral Size This field selects the size of the external peripheral. If the size of the external peripheral is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused).</p> <p>When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries.</p> <p>As a result, the available address space is double the amount shown below.</p> <p>0x0 = 256 bytes; lower address range: 0x00 to 0xFF 0x1 = 64 KB; lower address range: 0x0000 to 0xFFFF 0x2 = 16 MB; lower address range: 0x000000 to 0xFFFFFFF 0x3 = 256 MB; lower address range: 0x00000000 to 0xFFFFFFF</p>
5-4	EPADR	R/W	0x0	<p>External Peripheral Address This field selects address mapping for the external peripheral area.</p> <p>0x0 = Not mapped 0x1 = At 0xA0000000 0x2 = At 0xC0000000 0x3 = Only to be used with Host Bus quad chip select. In quad chip select mode, CS2n maps to 0xA0000000 and CS3n maps to 0xC0000000.</p>
3-2	ERSZ	R/W	0x0	<p>External RAM Size This field selects the size of mapped RAM. If the size of the external memory is larger, a bus fault occurs. If the size of the external memory is smaller, it wraps (upper address bits unused):</p> <p>0x0 = 256 bytes; lower address range: 0x00 to 0xFF 0x1 = 64 KB; lower address range: 0x0000 to 0xFFFF 0x2 = 16 MB; lower address range: 0x000000 to 0xFFFFFFF 0x3 = 256 MB; lower address range: 0x00000000 to 0xFFFFFFF</p>
1-0	ERADR	R/W	0x0	<p>External RAM Address Selects address mapping for external RAM area:</p> <p>0x0 = Not mapped 0x1 = At 0x60000000 0x2 = At 0x80000000 0x3 = Only to be used with Host Bus quad chip select. In quad chip select mode, CS0n maps to 0x60000000 and CS1n maps to 0x80000000.</p>

16.5.11 EPIRSIZE0 and EPIRSIZE1 Registers [reset = 0x3]

EPI Read Size 0 (EPIRSIZE0), offset 0x020

EPI Read Size 1 (EPIRSIZE1), offset 0x030

This register selects the size of transactions when performing non-blocking reads with the EPIRPSTDn registers. This size affects how the external address is incremented.

The SIZE field must match the external data width as configured in the EPIHBnCFG or EPIGPCFG register.

SDRAM mode uses a 16-bit data interface. If SIZE is 0x1, data is returned on the least significant bits (D[7:0]), and the remaining bits D[31:8] are all zeros, therefore the data on bits D[15:8] is lost. If SIZE is 0x2, data is returned on the least significant bits (D[15:0]), and the remaining bits D[31:16] are all zeros.

Note that changing this register while a read is active has an unpredictable effect.

EPIRSIZEn is shown in [Figure 16-40](#) and described in [Table 16-24](#).

Return to [Summary Table](#).

Figure 16-40. EPIRSIZEn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														SIZE	
R-0x0														R/W-0x3	

Table 16-24. EPIRSIZEn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1-0	SIZE	R/W	0x3	Current Size 0x0 = reserved 0x1 = Byte (8 bits) 0x2 = Half-word (16 bits) 0x3 = Word (32 bits)

16.5.12 EPIRADDR0 and EPIRADDR1 Registers [reset = 0x0]

EPI Read Address 0 (EPIRADDR0), offset 0x024

EPI Read Address 1 (EPIRADDR1), offset 0x034

This register holds the current address value. When performing non-blocking reads via the EPIRPSTDn registers, this register's value forms the address (when used by the mode). That is, when an EPIRPSTDn register is written with a non-0 value, this register is used as the first address. After each read, it is incremented by the size specified by the corresponding EPIRSIZEn register. Thus at the end of a read, this register contains the next address for the next read. For example, if the last read was 0x20, and the size is word, then the register contains 0x24. When a non-blocking read is cancelled, this register contains the next address that would have been read had it not been cancelled. For example, if reading by bytes and 0x103 had been read but not 0x104, this register contains 0x104. In this manner, the system can determine the number of values in the NBRFIFO to drain.

Note that changing this register while a read is active has an unpredictable effect due to race condition.

EPIRADDRn is shown in [Figure 16-41](#) and described in [Table 16-25](#).

Return to [Summary Table](#).

Figure 16-41. EPIRADDRn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
R/W-0x0																															

Table 16-25. EPIRADDRn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0x0	Current Address Next address to read.

16.5.13 EPIRPSTD0 and EPIRPSTD1 Registers [reset = 0x0]

EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028

EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038

This register sets up a non-blocking read via the external interface. A non-blocking read is started by writing to this register with the count (other than 0). Clearing this register terminates an active non-blocking read as well as cancelling any that are pending. This register should always be cleared before writing a value other than 0; failure to do so can cause improper operation. Note that both NBR channels can be enabled at the same time, but NBR channel 0 has the highest priority and channel 1 does not start until channel 0 is finished.

The first address is based on the corresponding EPIRADDRn register. The address register is incremented by the size specified by the EPIRSIZEn register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are cleared.

Note that all three registers may be written using one STM instruction, such as with a structure copy in C/C++.

The data may be read from the EPIREADFIFO register after the read cycle is completed. The interrupt mechanism is normally used to trigger the FIFO reads via ISR or uDMA.

If the countdown has not reached 0 and the NBRFIFO is full, the external interface waits until a NBRFIFO entry becomes available to continue.

Note: if a blocking read or write is performed through the address mapped area (at 0x60000000 through 0xDFFFFFFF), any current non-blocking read is paused (at the next safe boundary), and the blocking request is inserted. After completion of any blocking reads or writes, the non-blocking reads continue from where they were paused.

The other way to read data is via the address mapped locations (see the EPIADDRMAP register), but this method is blocking (core or uDMA waits until result is returned).

To cancel a non-blocking read, clear this register. To make sure that all values read are drained from the NBRFIFO, the EPISTAT register must be consulted to be certain that bits NBRBUSY and ACTIVE are cleared. One of these registers should not be cleared until either the other EPIRPSTDn register becomes active or the external interface is not busy. At that point, the corresponding EPIRADDRn register indicates how many values were read.

EPIRPSTDn is shown in [Figure 16-42](#) and described in [Table 16-26](#).

Return to [Summary Table](#).

Figure 16-42. EPIRPSTDn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			POSTCNT												
R-0x0																			R/W-0x0												

Table 16-26. EPIRPSTDn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0x0	
12-0	POSTCNT	R/W	0x0	Post Count A write of a non-zero value starts a read operation for that count. Note that it is the software's responsibility to handle address wrap-around. Reading this register provides the current count. A write of 0 cancels a non-blocking read (whether active now or pending). Prior to writing a non-zero value, this register must first be cleared.

16.5.14 EPISTAT Register (Offset = 0x60) [reset = 0x0]

EPI Status (EPISTAT)

This register indicates which non-blocking read register is currently active; it also indicates whether the external interface is busy performing a write or non-blocking read (it cannot be performing a blocking read, as the bus would be blocked and as a result, this register could not be accessed).

This register is useful to determining which non-blocking read register is active when both are loaded with values and when implementing sequencing or sharing.

This register is also useful when canceling non-blocking reads, as it shows how many values were read by the canceled side.

EPISTAT is shown in [Figure 16-43](#) and described in [Table 16-27](#).

Return to [Summary Table](#).

Figure 16-43. EPISTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							XFFULL
R-0x0							R-0x0
7	6	5	4	3	2	1	0
XFEMPTY	INITSEQ	WBUSY	NBRBUSY	RESERVED			ACTIVE
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0			R-0x0

Table 16-27. EPISTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	XFFULL	R	0x0	External FIFO Full This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFFEN bit set in the EPIHBnCFG register. The EPI0S26 signal reflects the status of this bit. 0x0 = The external device is not gating the clock. 0x1 = The XFIFO is signaling as full (the FIFO full signal is high). Attempts to write in this case are stalled until the XFIFO full signal goes low or the counter times out as specified by the MAXWAIT field.
7	XFEMPTY	R	0x0	External FIFO Empty This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFEEN bit set in the EPIHBnCFG register. The EPI0S27 signal reflects the status of this bit. 0x0 = The external device is not gating the clock. 0x1 = The XFIFO is signaling as empty (the FIFO empty signal is high). Attempts to read in this case are stalled until the XFIFO empty signal goes low or the counter times out as specified by the MAXWAIT field.
6	INITSEQ	R	0x0	Initialization Sequence 0x0 = The SDRAM interface is not in the wakeup period. 0x1 = The SDRAM interface is running through the wakeup period (greater than 100 microseconds). If an attempt is made to read or write the SDRAM during this period, the access is held off until the wakeup period is complete.

Table 16-27. EPISTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	WBUSY	R	0x0	Write Busy 0x0 = The external interface is not performing a write. 0x1 = The external interface is performing a write.
4	NBRBUSY	R	0x0	Non-Blocking Read Busy 0x0 = The external interface is not performing a non-blocking read. 0x1 = The external interface is performing a non-blocking read, or if the non-blocking read is paused due to a write.
3-1	RESERVED	R	0x0	
0	ACTIVE	R	0x0	Register Active 0x0 = If NBRBUSY is set, the EPIRPSD0 register is active. If the NBRBUSY bit is clear, then neither EPIRPSDx register is active. 0x1 = The EPIRPSD1 register is active.

16.5.15 EPIRFIFOCNT Register (Offset = 0x6C) [reset = X]

EPI Read FIFO Count (EPIRFIFOCNT)

This register returns the number of values in the NBRFIFO (the data in the NBRFIFO can be read via the EPIREADFIFO register). A race is possible, but that only means that more values may come in after this register has been read.

EPIRFIFOCNT is shown in [Figure 16-44](#) and described in [Table 16-28](#).

Return to [Summary Table](#).

Figure 16-44. EPIRFIFOCNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNT			
R-0x0												R-X			

Table 16-28. EPIRFIFOCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	COUNT	R	X	FIFO Count. Number of filled entries in the NBRFIFO.

16.5.16 EPIREADFIFO0 to EPIREADFIFO7 Registers (Offset = 0x70 to 0x8C) [reset = X]

EPI Read FIFO (EPIREADFIFO0), offset 0x070

EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074

EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078

EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C

EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080

EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084

EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088

EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C

This register returns the contents of the NBRFIFO or 0 if the NBRFIFO is empty. Each read returns the data that is at the top of the NBRFIFO, and then empties that value from the NBRFIFO. The alias registers can be used with the LDMA instruction for more efficient operation (for up to 8 registers). See Cortex-M3/M4 Instruction Set Technical User's Manual (literature number [SPMU159](#)) for more information on the LDMA instruction.

EPIREADFIFO_n is shown in [Figure 16-45](#) and described in [Table 16-29](#).

Return to [Summary Table](#).

Figure 16-45. EPIREADFIFO_n Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R-X																															

Table 16-29. EPIREADFIFO_n Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R	X	Reads Data. This field contains the data that is at the top of the NBRFIFO. After being read, the NBRFIFO entry is removed.

16.5.17 EPIFIFOLVL Register (Offset = 0x200) [reset = 0x33]

EPI FIFO Level Selects (EPIFIFOLVL)

This register allows selection of the FIFO levels which trigger an interrupt to the interrupt controller or, more efficiently, a DMA request to the μ DMA. The NBRFIFO select triggers on fullness such that it triggers on match or above (more full) in order for the processor or the μ DMA to extract the read data. The WFIFO triggers on emptiness such that it triggers on match or below (less entries) in order for the processor or the μ DMA to insert more write data.

It should be noted that the FIFO triggers are not identical to other such FIFOs in other peripherals. In particular, empty and full triggers are provided to avoid wait states when using blocking operations.

The settings in this register are only meaningful if the μ DMA is active or the interrupt is enabled.

Additionally, this register allows protection against writes stalling and notification of performing blocking reads which stall for extra time due to preceding writes. The two functions behave in a non-orthogonal way because read and write are not orthogonal.

The write error bit configures the system such that an attempted write to an already full WFIFO abandons the write and signals an error interrupt to prevent accidental latencies due to stalling writes.

The read error bit configures the system such that after a read has been stalled due to any preceding writes in the WFIFO, the error interrupt is generated. Note that the excess stall is not prevented, but an interrupt is generated after the fact to notify that it has happened.

EPIFIFOLVL is shown in [Figure 16-46](#) and described in [Table 16-30](#).

Return to [Summary Table](#).

Figure 16-46. EPIFIFOLVL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						WFERR	RSERR
R-0x0						R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		WRFIFO		RESERVED		RDFIFO	
R-0x0		R/W-0x3		R-0x0		R/W-0x3	

Table 16-30. EPIFIFOLVL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	WFERR	R/W	0x0	<p>Write Full Error</p> <p>0x0 = The Write Full error interrupt is disabled. Writes are stalled when the WFIFO is full until a space becomes available but an error is not generated. Note that the Cortex-M4 write buffer may hide that stall if no other memory transactions are attempted during that time.</p> <p>0x1 = This bit enables the Write Full error interrupt (WTFULL in the EPIEISC register) to be generated when a write is attempted and the WFIFO is full. The write stalls until a WFIFO entry becomes available.</p>

Table 16-30. EPIFIFOLVL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
16	RSERR	R/W	0x0	Read Stall Error Note that the configuration of this bit has no effect on non-blocking reads. 0x0 = The Read Stalled error interrupt is disabled. Reads behave as normal and are stalled until any preceding writes have completed and the read has returned a result. 0x1 = This bit enables the Read Stalled error interrupt (RSTALL in the EPIEISC register) to be generated when a read is attempted and the WFIFO is not empty. The read is still stalled during the time the WFIFO drains, but this error notifies the application that this excess delay has occurred.
15-7	RESERVED	R	0x0	
6-4	WRFIFO	R/W	0x3	Write FIFO 0x0 = reserved 0x1 = reserved 0x2 = Interrupt is triggered until there are only two slots available. Thus, trigger is deasserted when there are two WRFIFO entries present. This configuration is optimized for bursts of 2. 0x3 = Interrupt is triggered until there is one WRFIFO entry available. This configuration expects only single writes. 0x4 = Trigger interrupt when WRFIFO is not full, meaning trigger will continue to assert until there are four entries in the WRFIFO.
3	RESERVED	R	0x0	
2-0	RDFIFO	R/W	0x3	Read FIFO This field configures the trigger point for the NBRFIFO. 0x0 = reserved 0x1 = Trigger when there are 1 or more entries in the NBRFIFO. 0x2 = Trigger when there are 2 or more entries in the NBRFIFO. 0x3 = Trigger when there are 4 or more entries in the NBRFIFO. 0x4 = Trigger when there are 6 or more entries in the NBRFIFO. 0x5 = Trigger when there are 7 or more entries in the NBRFIFO. 0x6 = Trigger when there are 8 entries in the NBRFIFO. 0x7 = reserved

16.5.18 EPIWFIFOCNT Register (Offset = 0x204) [reset = 0x4]

EPI Write FIFO Count (EPIWFIFOCNT)

This register contains the number of slots currently available in the WFIFO. This register may be used for polled writes to avoid stalling and for blocking reads to avoid excess stalling (due to undrained writes). An example use for writes may be:

```
for (idx = 0; idx < cnt; idx++) { while (EPIWFIFOCNT == 0) ; *ext_ram = *mydata++; }
```

The above code ensures that writes to the address mapped location do not occur unless the WFIFO has room. Although polling makes the code wait (spinning in the loop), it does not prevent interrupts being serviced due to bus stalling.

EPIWFIFOCNT is shown in [Figure 16-47](#) and described in [Table 16-31](#).

Return to [Summary Table](#).

Figure 16-47. EPIWFIFOCNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												WTAV			
R-0x0												R-0x4			

Table 16-31. EPIWFIFOCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2-0	WTAV	R	0x4	Available Write Transactions. The number of write transactions available in the WFIFO. When clear, a write is stalled waiting for a slot to become free (from a preceding write completing).

16.5.19 EPIDMATXCNT Register (Offset = 0x208) [reset = 0x0]

EPI DMA Transmit Count (EPIDMATXCNT)

This register is used to program the total number of transfers (byte, halfword or word) by the μ DMA to WRFIFO. As each transfer is processed by the EPI, the TXCNT bit field value is decreased by 1. When TXCNT = 0, the EPI's μ DMA request signal is deasserted.

EPIDMATXCNT is shown in [Figure 16-48](#) and described in [Table 16-32](#).

Return to [Summary Table](#).

Figure 16-48. EPIDMATXCNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TXCNT															
R-0x0																R/W-0x0															

Table 16-32. EPIDMATXCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	TXCNT	R/W	0x0	DMA Count. This field is used to program the total number of transfers (byte, halfword or word) from the μ DMA to the EPI WRFIFO.

16.5.20 EPIIM Register (Offset = 0x210) [reset = 0x0]

EPI Interrupt Mask (EPIIM)

This register is the interrupt mask set or clear register. For each interrupt source (read, write, and error), a mask value of 1 allows the interrupt source to trigger an interrupt to the interrupt controller; a mask value of 0 prevents the interrupt source from triggering an interrupt.

EPIIM is shown in [Figure 16-49](#) and described in [Table 16-33](#).

Return to [Summary Table](#).

Figure 16-49. EPIIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMAWRIM	DMARDIM	WRIM	RDIM	ERRIM
R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 16-33. EPIIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMAWRIM	R/W	0x0	Write μ DMA Interrupt Mask 0x0 = DMAWRRIS in the EPIRIS register is masked and does not cause an interrupt. 0x1 = DMAWRRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.
3	DMARDIM	R/W	0x0	Read μ DMA Interrupt Mask 0x0 = DMARDRIS in the EPIRIS register is masked and does not cause an interrupt. 0x1 = DMARDRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.
2	WRIM	R/W	0x0	Write FIFO Empty Interrupt Mask 0x0 = WRRIS in the EPIRIS register is masked and does not cause an interrupt. 0x1 = WRRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.
1	RDIM	R/W	0x0	Read FIFO Full Interrupt Mask 0x0 = RDRIS in the EPIRIS register is masked and does not cause an interrupt. 0x1 = RDRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.
0	ERRIM	R/W	0x0	Error Interrupt Mask 0x0 = ERRIS in the EPIRIS register is masked and does not cause an interrupt. 0x1 = ERRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.

16.5.21 EPIRIS Register (Offset = 0x214) [reset = 0x4]

EPI Raw Interrupt Status (EPIRIS)

This register is the raw interrupt status register. On a read, it gives the current state of each interrupt source. A write has no effect.

Note that raw status for read and write is set or cleared based on FIFO fullness as controlled by EPIFIFOLVL.

Raw status for error is held until the error is cleared by writing to the EPIEISC register.

EPIRIS is shown in [Figure 16-50](#) and described in [Table 16-34](#).

Return to [Summary Table](#).

Figure 16-50. EPIRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMAWRRIS	DMARDRIS	WRRIS	RDRIS	ERRRIS
R-0x0			R-0x0	R-0x0	R-0x1	R-0x0	R-0x0

Table 16-34. EPIRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMAWRRIS	R	0x0	Write μ DMA Raw Interrupt Status This bit is cleared by writing a 1 to the DMAWRIC bit in the EPIEISC register. 0x0 = The write μ DMA has not completed. 0x1 = The write μ DMA has completed.
3	DMARDRIS	R	0x0	Read μ DMA Raw Interrupt Status This bit is cleared by writing a 1 to the DMARDIC bit in the EPIEISC register. 0x0 = The read μ DMA has not completed. 0x1 = The read μ DMA has completed.
2	WRRIS	R	0x1	Write Raw Interrupt Status This bit is cleared when the level in the WFIFO is above the trigger point programmed by the WRFIFO field. 0x0 = The number of available entries in the WFIFO is above the range specified by the WRFIFO field in the EPIFIFOLVL register. 0x1 = The number of available entries in the WFIFO is within the trigger range specified by the WRFIFO field in the EPIFIFOLVL register.
1	RDRIS	R	0x0	Read Raw Interrupt Status This bit is cleared when the level in the NBRFIFO is below the trigger point programmed by the RDFIFO field. 0x0 = The number of valid entries in the NBRFIFO is below the trigger range specified by the RDFIFO field in the EPIFIFOLVL register. 0x1 = The number of valid entries in the NBRFIFO is in the trigger range specified by the RDFIFO field in the EPIFIFOLVL register.

Table 16-34. EPIRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	ERRRIS	R	0x0	<p>Error Raw Interrupt Stat Error Raw Interrupt Status</p> <p>The error interrupt occurs in the following situations:</p> <ul style="list-style-type: none"> • WFIFO Full. For a full WFIFO to generate an error interrupt, the WFERR bit in the EPIFIFOLVL register must be set. • Read Stalled. For a stalled read to generate an error interrupt, the RSERR bit in the EPIFIFOLVL register must be set. • Timeout. If the MAXWAIT field in the EPIHBnCFG register is configured to a value other than 0, a timeout error occurs when XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field. <p>0x0 = An error has not occurred. 0x1 = A WFIFO Full, a Read Stalled, or a Timeout error has occurred.</p> <p>To determine which error occurred, read the status of the EPI Error Interrupt Status and Clear (EPIEISC) register. This bit is cleared by writing a 1 to the bit in the EPIEISC register that caused the interrupt.</p>

16.5.22 EPIMIS Register (Offset = 0x218) [reset = 0x0]

EPI Masked Interrupt Status (EPIMIS)

This register is the masked interrupt status register. On read, it gives the current state of each interrupt source (read, write, and error) after being masked via the EPIIM register. A write has no effect.

The values returned are the ANDing of the EPIIM and EPIRIS registers. If a bit is set in this register, the interrupt is sent to the interrupt controller.

EPIMIS is shown in [Figure 16-51](#) and described in [Table 16-35](#).

Return to [Summary Table](#).

Figure 16-51. EPIMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMAWRMIS	DMARDMIS	WRMIS	RDMIS	ERRMIS
R-0x0			R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 16-35. EPIMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMAWRMIS	R	0x0	Write μ DMA Masked Interrupt Status. This bit is cleared by writing a 1 to the DMAWRIC bit in the EPIEISC register. 0x0 = The write μ DMA has not completed or the interrupt is masked. 0x1 = The write μ DMA has completed and the DMAWRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.
3	DMARDMIS	R	0x0	Read μ DMA Masked Interrupt Status. This bit is cleared by writing a 1 to the DMARDIC bit in the EPIEISC register. 0x0 = The read μ DMA has not completed or the interrupt is masked. 0x1 = The read μ DMA has completed and the DMAWRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.
2	WRMIS	R	0x0	Write Masked Interrupt Status 0x0 = The number of available entries in the WFIFO is above the range specified by the trigger level or the interrupt is masked. 0x1 = The number of available entries in the WFIFO is within the range specified by the trigger level (the WRFIFO field in the EPIFIFOLVL register) and the WRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.
1	RDMIS	R	0x0	Read Masked Interrupt Status 0x0 = The number of valid entries in the NBRFIFO is below the range specified by the trigger level or the interrupt is masked. 0x1 = The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the RDFIFO field in the EPIFIFOLVL register) and the RDIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.

Table 16-35. EPIMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	ERRMIS	R	0x0	<p>Error Masked Interrupt Status</p> <p>0x0 = An error has not occurred or the interrupt is masked.</p> <p>0x1 = A WFIFO Full, a Read Stalled, or a Timeout error has occurred and the ERIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.</p>

16.5.23 EPIESC Register (Offset = 0x21C) [reset = 0x0]

EPI Error and Interrupt Status and Clear (EPIESC)

This register is used to clear a pending error interrupt. Clearing any defined bit in the EPIESC has no effect; setting a bit clears the error source and the raw error returns to 0. When any of bits[2:0] of this register are read as set, it indicates that the ERRRIS bit in the EPIRIS register is set and an EPI controller error is sent to the interrupt controller if the ERIM bit in the EPIIM register is set. If any of bits [2:0] are written as 1, the register bit being written to, as well as the ERRRIS bit in the EPIRIS register and the ERIM bit in the EPIIM register are cleared. If the DMAWRIC or DMARDIC bit in this register is set, then the corresponding bit in the EPIRIS and EPIMIS register is cleared. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.

EPIESC is shown in [Figure 16-52](#) and described in [Table 16-36](#).

Return to [Summary Table](#).

Figure 16-52. EPIESC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMAWRIC	DMARDIC	WTFULL	RSTALL	TOUT
R-0x0			W1C-0x0	W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 16-36. EPIESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMAWRIC	W1C	0x0	Write μ DMA Interrupt Clear. Writing a 1 to this bit clears the DMAWRRIS bit in the EPIRIS register and the DMAWRMIS bit in the EPIMIS register.
3	DMARDIC	W1C	0x0	Read μ DMA Interrupt Clear. Writing a 1 to this bit clears the DMARDRIS bit in the EPIRIS register and the DMARDMIS bit in the EPIMIS register.
2	WTFULL	R/W1C	0x0	Write FIFO Full Error. Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits. 0x0 = The WFERR bit is not enabled or no writes are stalled. 0x1 = The WFERR bit is enabled and a write is stalled due to the WFIFO being full.
1	RSTALL	R/W1C	0x0	Read Stalled Error. Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits. 0x0 = The RSERR bit is not enabled or no pending reads are stalled. 0x1 = The RSERR bit is enabled and a pending read is stalled due to writes in the WFIFO.
0	TOUT	R/W1C	0x0	Timeout Error. This bit is the timeout error source. The timeout error occurs when the XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field (when not 0). Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits. 0x0 = No timeout error has occurred. 0x1 = A timeout error has occurred.

16.5.24 EPIHB8CFG3 Register (Offset = 0x308) [reset = 0x00080000]

EPI Host-Bus 8 Configuration 3 (EPIHB8CFG3)

NOTE: The MODE field in the EPICFG register configures whether EPI Host Bus mode is enabled.
For EPIHB8CFG3 to be valid, the MODE field must be 0x2.

EPIHB8CFG3 is shown in [Figure 16-53](#) and described in [Table 16-37](#).

Return to [Summary Table](#).

Figure 16-53. EPIHB8CFG3 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED		WRHIGH	RDHIGH	ALEHIGH	RESERVED		
R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R-0x0		
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-37. EPIHB8CFG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0x0	
21	WRHIGH	R/W	0x0	CS2n WRITE Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB8CFG2. 0x0 = The WRITE strobe for CS2n accesses is WRn (active Low). 0x1 = The WRITE strobe for CS2n accesses is WR (active High).
20	RDHIGH	R/W	0x0	CS2n READ Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB8CFG2. 0x0 = The READ strobe for CS2n accesses is RDn (active Low). 0x1 = The READ strobe for CS2n accesses is RD (active High).
19	ALEHIGH	R/W	0x1	CS2n ALE Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB8CFG2. 0x0 = The address latch strobe for CS2n accesses is ADVn (active Low). 0x1 = The address latch strobe for CS2n accesses is ALE (active High).
18-8	RESERVED	R	0x0	

Table 16-37. EPIHB8CFG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	WRWS	R/W	0x0	<p>CS2n Write Wait States. This field adds wait states to the data phase of CS2n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR).</p> <p>Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME3 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register.</p> <p>This field is not applicable in BURST mode.</p> <p>This field is used in conjunction with the EPIBAUD2 register.</p> <p>0x0 = Active WRn is 2 EPI clocks 0x1 = Active WRn is 4 EPI clocks 0x2 = Active WRn is 6 EPI clocks 0x3 = Active WRn is 8 EPI clocks</p>
5-4	RDWS	R/W	0x0	<p>CS2n Read Wait States. This field adds wait states to the data phase of CS2n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <p>Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME3 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register.</p> <p>This field is not applicable in BURST mode.</p> <p>This field is used in conjunction with the EPIBAUD2 register.</p> <p>0x0 = Active RDn is 2 EPI clocks 0x1 = Active RDn is 4 EPI clocks 0x2 = Active RDn is 6 EPI clocks 0x3 = Active RDn is 8 EPI clocks</p>
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	<p>CS2n Host Bus Sub-Mode. This field determines which Host Bus 8 sub-mode to use for CS2n in multiple chip-select mode. Sub-mode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals.</p> <p>The CSBAUD bit must be set to enable this CS2n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB8CFG register.</p> <p>0x0 = reserved 0x1 = ADNONMUX - D[7:0]Data and address are separate.</p>

16.5.25 EPIHB16CFG3 Register (Offset = 0x308) [reset = 0x00080000]

EPI Host-Bus 16 Configuration 3 (EPIHB16CFG3)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB16CFG3 to be valid, the MODE field must be 0x3.

EPIHB16CFG3 is shown in [Figure 16-54](#) and described in [Table 16-38](#).

Return to [Summary Table](#).

Figure 16-54. EPIHB16CFG3 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED		WRHIGH	RDHIGH	ALEHIGH	WRCRE	RDCRE	BURST
R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-38. EPIHB16CFG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0x0	
21	WRHIGH	R/W	0x0	CS2n WRITE Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB16CFG2. 0x0 = The WRITE strobe for CS2n accesses is WRn (active Low). 0x1 = The WRITE strobe for CS2n accesses is WR (active High).
20	RDHIGH	R/W	0x0	CS2n READ Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2. 0x0 = The READ strobe for CS2n accesses is RDn (active Low). 0x1 = The READ strobe for CS2n accesses is RD (active High).
19	ALEHIGH	R/W	0x1	CS2n ALE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2. 0x0 = The address latch strobe for CS2n accesses is ADVn (active Low). 0x1 = The address latch strobe for CS2n accesses is ALE (active High).
18	WRCRE	R/W	0x0	CS2n PSRAM Configuration Register Write Used for PSRAM configuration registers. With WRCRE set, the next transaction by the EPI is a write of the CR bit field in the EPIHBPSRAM register to the configuration register (CR) of the PSRAM. The WRCRE bit self clears once the write-enabled CRE access is complete. 0x0 = No Action. 0x1 = Start CRE write transaction for CS2n.

Table 16-38. EPIHB16CFG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	RDCRE	R/W	0x0	CS2n PSRAM Configuration Register Read. Used for PSRAM configuration registers. With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the CRE access is complete. The address for the CRE access is located at EPIHBPSRAM [19:18]. The read data is returned on EPIHBPSRAM [15:0]. 0x0 = No Action. 0x1 = Start CRE read transaction for CS2n.
16	BURST	R/W	0x0	CS2n Burst Mode. Burst mode must be used with an ALE, which is configured by programming the CSCFG and CSCFGEXT fields in the EPIHB16CFG2 register. Burst mode must be used in ADMUX, which is set by the MODE field in EPIHB16CFG3. Burst mode is optimized for word-length accesses. 0x0 = Burst mode is disabled. 0x1 = Burst mode is enabled for CS2n.
15-8	RESERVED	R	0x0	
7-6	WRWS	R/W	0x0	CS2n Write Wait States. This field adds wait states to the data phase of CS2n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB16TIME3 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the EPIHB16CFG2 register. This field is not applicable in BURST mode. This field is used in conjunction with the EPIBAUD2 register. 0x0 = Active WRn is 2 EPI clocks 0x1 = Active WRn is 4 EPI clocks 0x2 = Active WRn is 6 EPI clocks 0x3 = Active WRn is 8 EPI clocks
5-4	RDWS	R/W	0x0	CS2n Read Wait States. This field adds wait states to the data phase of CS2n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB16TIME3 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register. This field is not applicable in BURST mode. This field is used in conjunction with the EPIBAUD2 register. 0x0 = Active RDn is 2 EPI clocks 0x1 = Active RDn is 4 EPI clocks 0x2 = Active RDn is 6 EPI clocks 0x3 = Active RDn is 8 EPI clocks
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	CS2n Host Bus Sub-Mode. This field determines which Host Bus 16 sub-mode to use for CS2n in multiple chip select mode. Sub-mode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals. The CSBAUD bit must be set to enable this CS2n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB16CFG register. 0x0 = reserved 0x1 = ADNONMUX - D[15:0] Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.

16.5.26 EPIHB8CFG4 Register (Offset = 0x30C) [reset = 0x00080000]

EPI Host-Bus 8 Configuration 4 (EPIHB8CFG4)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB8CFG4 to be valid, the MODE field must be 0x2.

EPIHB8CFG4 is shown in [Figure 16-55](#) and described in [Table 16-39](#).

Return to [Summary Table](#).

Figure 16-55. EPIHB8CFG4 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED		WRHIGH	RDHIGH	ALEHIGH	RESERVED		
R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R-0x0		
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-39. EPIHB8CFG4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0x0	
21	WRHIGH	R/W	0x0	CS3n WRITE Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB8CFG2. 0x0 = The WRITE strobe for CS3n accesses is WRn (active low). 0x1 = The WRITE strobe for CS3n accesses is WR (active high).
20	RDHIGH	R/W	0x0	CS2n READ Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB8CFG2. 0x0 = The READ strobe for CS3n accesses is RDn (active low). 0x1 = The READ strobe for CS3n accesses is RD (active high).
19	ALEHIGH	R/W	0x1	CS3n ALE Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB8CFG2. 0x0 = The address latch strobe for CS3n accesses is ADVn (active low). 0x1 = The address latch strobe for CS3n accesses is ALE (active high).
18-8	RESERVED	R	0x0	

Table 16-39. EPIHB8CFG4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	WRWS	R/W	0x0	<p>CS3n Write Wait States. This field adds wait states to the data phase of CS3n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR).</p> <p>Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME4 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register.</p> <p>This field is not applicable in BURST mode.</p> <p>This field is used in conjunction with the EPIBAUD2 register.</p> <p>0x0 = Active WRn is 2 EPI clocks 0x1 = Active WRn is 4 EPI clocks 0x2 = Active WRn is 6 EPI clocks 0x3 = Active WRn is 8 EPI clocks</p>
5-4	RDWS	R/W	0x0	<p>CS3n Read Wait States. This field adds wait states to the data phase of CS3n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <p>Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME4 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used when the CSBAUD bit is set in the EPIHB8CFG2 register.</p> <p>This field is not applicable in BURST mode.</p> <p>This field is used in conjunction with the EPIBAUD2 register.</p> <p>0x0 = Active RDn is 2 EPI clocks 0x1 = Active RDn is 4 EPI clocks 0x2 = Active RDn is 6 EPI clocks 0x3 = Active RDn is 8 EPI clocks</p>
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	<p>CS3n Host Bus Sub-Mode. This field determines which Host Bus 8 sub-mode to use for CS3n in multiple chip select mode. Sub-mode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals.</p> <p>The CSBAUD bit must be set to enable this CS3n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB8CFG register.</p> <p>0x0 = reserved 0x1 = ADNONMUX - D[7:0]Data and address are separate.</p>

16.5.27 EPIHB16CFG4 Register (Offset = 0x30C) [reset = 0x00080000]

EPI Host-Bus 16 Configuration 4 (EPIHB16CFG4)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB16CFG4 to be valid, the MODE field must be 0x3.

EPIHB16CFG4 is shown in [Figure 16-56](#) and described in [Table 16-40](#).

Return to [Summary Table](#).

Figure 16-56. EPIHB16CFG4 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED		WRHIGH	RDHIGH	ALEHIGH	WRCRE	RDCRE	BURST
R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WRWS		RDWS		RESERVED		MODE	
R/W-0x0		R/W-0x0		R-0x0		R/W-0x0	

Table 16-40. EPIHB16CFG4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0x0	
21	WRHIGH	R/W	0x0	CS3n WRITE Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB16CFG2. 0x0 = The WRITE strobe for CS3n accesses is WRn (active Low). 0x1 = The WRITE strobe for CS3n accesses is WR (active High).
20	RDHIGH	R/W	0x0	CS3n READ Strobe Polarity. This field is used if the CSBAUD bit is enabled in EPIHB16CFG2. 0x0 = The READ strobe for CS3n accesses is RDn (active Low). 0x1 = The READ strobe for CS3n accesses is RD (active High).
19	ALEHIGH	R/W	0x1	CS3n ALE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2. 0x0 = The address latch strobe for CS3n accesses is ADVn (active Low). 0x1 = The address latch strobe for CS3n accesses is ALE (active High).
18	WRCRE	R/W	0x0	CS3n PSRAM Configuration Register Write. Used for PSRAM configuration registers. With WRCRE set, the next transaction by the EPI will be a write of the CR bit field in the EPIHBPSRAM register to the configuration register (CR) of the PSRAM. The WRCRE bit will self clear once the write-enabled CRE access is complete. 0x0 = No Action. 0x1 = Start CRE write transaction for CS3n.

Table 16-40. EPIHB16CFG4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	RDCRE	R/W	0x0	CS3n PSRAM Configuration Register Read. Used for PSRAM configuration registers. With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the CRE access is complete. The address for the CRE access is located at EPIHBPSRAM [19:18]. The read data is returned on EPIHBPSRAM [15:0]. 0x0 = No Action. 0x1 = Start CRE read transaction for CS3n.
16	BURST	R/W	0x0	CS3n Burst Mode. Burst mode must be used with an ALE, which is configured by programming the CSCFG and CSCFGEXT fields in the EPIHB16CFG2 register. Burst mode must be used in ADMUX, which is set by the MODE field in EPIHB16CFG4. Burst mode is optimized for word-length accesses. 0x0 = Burst mode is disabled. 0x1 = Burst mode is enabled for CS3n.
15-8	RESERVED	R	0x0	
7-6	WRWS	R/W	0x0	CS3n Write Wait States. This field adds wait states to the data phase of CS2n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB16TIME4 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is set in the EPIHB16CFG2 register. This field is not applicable in BURST mode. This field is used in conjunction with the EPIBAUD2 register. 0x0 = Active WRn is 2 EPI clocks 0x1 = Active WRn is 4 EPI clocks 0x2 = Active WRn is 6 EPI clocks 0x3 = Active WRn is 8 EPI clocks
5-4	RDWS	R/W	0x0	CS3n Read Wait States. This field adds wait states to the data phase of CS3n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB16TIME4 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used when the CSBAUD bit is set in the EPIHB16CFG2 register. This field is not applicable in BURST mode. This field is used in conjunction with the EPIBAUD2 register. 0x0 = Active RDn is 2 EPI clocks 0x1 = Active RDn is 4 EPI clocks 0x2 = Active RDn is 6 EPI clocks 0x3 = Active RDn is 8 EPI clocks
3-2	RESERVED	R	0x0	
1-0	MODE	R/W	0x0	CS3n Host Bus Sub-Mode. This field determines which Host Bus 16 sub-mode to use for CS3n in multiple chip select mode. Sub-mode use is determined by the connected external peripheral. See for information on how this bit field affects the operation of the EPI signals. The CSBAUD bit must be set to enable this CS3n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB16CFG register. 0x0 = reserved 0x1 = ADNONMUX - D[15:0]Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.

16.5.28 EPIHB8TIME Register (Offset = 0x310) [reset = 0x00022000]

EPI Host-Bus 8 Timing Extension (EPIHB8TIME)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB8TIME to be valid, the MODE field must be 0x2.

EPIHB8TIME is shown in [Figure 16-57](#) and described in [Table 16-41](#).

Return to [Summary Table](#).

Figure 16-57. EPIHB8TIME Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
reserved-1							
R-0x8							
15	14	13	12	11	10	9	8
reserved-1		CAPWIDTH			RESERVED		
R-0x8		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-41. EPIHB8TIME Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS0n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-14	RESERVED	R	0x8	
13-12	CAPWIDTH	R/W	0x2	CS0n Inter-transfer Capture Width. Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	
4	WRWSM	R/W	0x0	Write Wait State Minus One. This bit is used with the WRWS field in EPIHB8CFG. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB8CFG.
3-1	RESERVED	R	0x0	

Table 16-41. EPIHB8TIME Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RDWSM	R/W	0x0	Read Wait State Minus One. Use with RDWS field in the EPIHB8CFG register. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG. 0x1 (Write) = Wait state value is noRDWS; 1RDWS field is programmed in EPIHB8CFG.

16.5.29 EPIHB16TIME Register (Offset = 0x310) [reset = 0x00022000]

EPI Host-Bus 16 Timing Extension (EPIHB16TIME)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB16TIME to be valid, the MODE field must be 0x3.

EPIHB16TIME is shown in [Figure 16-58](#) and described in [Table 16-42](#).

[Return to Summary Table.](#)

Figure 16-58. EPIHB16TIME Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
RESERVED						PSRAMSZ	
R-0x0						R/W-0x2	
15	14	13	12	11	10	9	8
RESERVED		CAPWIDTH			RESERVED		
R-0x0		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-42. EPIHB16TIME Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS0n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-19	RESERVED	R	0x0	
18-16	PSRAMSZ	R/W	0x2	PSRAM Row Size. Defines the row size for the PSRAM controlled by CS0n 0x0 = No row size limitation 0x1 = 128 B 0x2 = 256 B 0x3 = 512 B 0x4 = 1024 B 0x5 = 2048 B 0x6 = 4096 B 0x7 = 8192 B
15-14	RESERVED	R	0x0	
13-12	CAPWIDTH	R/W	0x2	CS0n Inter-transfer Capture Width Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	

Table 16-42. EPIHB16TIME Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	WRWSM	R/W	0x0	Write Wait State Minus One. This bit is used with the WRWS field in EPIHB16CFG. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB16CFG.
3-1	RESERVED	R	0x0	
0	RDWSM	R/W	0x0	Read Wait State Minus One. Use with RDWS field in the EPIHB16CFG register. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG. 0x1 (Write) = Wait state value is noRDWS – 1RDWS field is programmed in EPIHB16CFG.

16.5.30 EPIHB8TIME2 Register (Offset = 0x314) [reset = 0xA000]

EPI Host-Bus 8 Timing Extension (EPIHB8TIME2)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB8TIME2 to be valid, the MODE field must be 0x2.

EPIHB8TIME2 is shown in [Figure 16-59](#) and described in [Table 16-43](#).

Return to [Summary Table](#).

Figure 16-59. EPIHB8TIME2 Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
reserved-1							
R-0x2							
15	14	13	12	11	10	9	8
reserved-1		CAPWIDTH			RESERVED		
R-0x2		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-43. EPIHB8TIME2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS1n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-14	RESERVED	R	0x2	
13-12	CAPWIDTH	R/W	0x2	CS1n Inter-transfer Capture Width Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	
4	WRWSM	R/W	0x0	CS1n Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG2. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG2 register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB8CFG2.
3-1	RESERVED	R	0x0	

Table 16-43. EPIHB8TIME2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RDWSM	R/W	0x0	<p>CS1n Read Wait State Minus One This field is used with RDWS field in EPIHB8CFG2.</p> <p>This bit is not applicable in BURST mode.</p> <p>0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG2.</p> <p>0x1 (Write) = Wait state value is noRDWS – 1RDWS field is programmed in EPIHB8CFG2.</p>

16.5.31 EPIHB16TIME2 Register (Offset = 0x314) [reset = 0x00022000]

EPI Host-Bus 16 Timing Extension (EPIHB16TIME2)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB16TIME2 to be valid, the MODE field must be 0x3.

EPIHB16TIME2 is shown in [Figure 16-60](#) and described in [Table 16-44](#).

Return to [Summary Table](#).

Figure 16-60. EPIHB16TIME2 Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
RESERVED						PSRAMSZ	
R-0x0						R/W-0x2	
15	14	13	12	11	10	9	8
RESERVED		CAPWIDTH			RESERVED		
R-0x0		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-44. EPIHB16TIME2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS1n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-19	RESERVED	R	0x0	
18-16	PSRAMSZ	R/W	0x2	PSRAM Row Size Defines the row size for the PSRAM controlled by CS1n 0x0 = No row size limitation 0x1 = 128 B 0x2 = 256 B 0x3 = 512 B 0x4 = 1024 B 0x5 = 2048 B 0x6 = 4096 B 0x7 = 8192 B
15-14	RESERVED	R	0x0	
13-12	CAPWIDTH	R/W	0x2	CS1n Inter-transfer Capture Width Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	

Table 16-44. EPIHB16TIME2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	WRWSM	R/W	0x0	CS1n Write Wait State Minus One This bit is used with the WRWS field in EPIHB16CFG2. This field is not applicable in BURST mode.. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG2 register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB16CFG2.
3-1	RESERVED	R	0x0	
0	RDWSM	R/W	0x0	CS1n Read Wait State Minus One This field is used with RDWS field in EPIHB16CFG2. This bit is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG2. 0x1 (Write) = Wait state value is noRDWS; 1RDWS field is programmed in EPIHB16CFG2.

16.5.32 EPIHB8TIME3 Register (Offset = 0x318) [reset = 0xA000]

EPI Host-Bus 8 Timing Extension (EPIHB8TIME3)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB8TIME 3 to be valid, the MODE field must be 0x2.

EPIHB8TIME3 is shown in [Figure 16-61](#) and described in [Table 16-45](#).

Return to [Summary Table](#).

Figure 16-61. EPIHB8TIME3 Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
reserved-1							
R-0x2							
15	14	13	12	11	10	9	8
reserved-1		CAPWIDTH			RESERVED		
R-0x2		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-45. EPIHB8TIME3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS2n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-14	RESERVED	R	0x2	
13-12	CAPWIDTH	R/W	0x2	CS2n Inter-transfer Capture Width Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	
4	WRWSM	R/W	0x0	CS2n Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG3. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG3 register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB8CFG3.
3-1	RESERVED	R	0x0	

Table 16-45. EPIHB8TIME3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RDWSM	R/W	0x0	CS2n Read Wait State Minus One This field is used with RDWS field in EPIHB8CFG3. This bit is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG3. 0x1 (Write) = Wait state value is noRDWS; 1RDWS field is programmed in EPIHB8CFG3.

16.5.33 EPIHB16TIME3 Register (Offset = 0x318) [reset = 0x00022000]

EPI Host-Bus 16 Timing Extension (EPIHB16TIME3)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB16TIME 3 to be valid, the MODE field must be 0x3.

EPIHB16TIME3 is shown in [Figure 16-62](#) and described in [Table 16-46](#).

Return to [Summary Table](#).

Figure 16-62. EPIHB16TIME3 Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
RESERVED						PSRAMSZ	
R-0x0						R/W-0x2	
15	14	13	12	11	10	9	8
RESERVED		CAPWIDTH			RESERVED		
R-0x0		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-46. EPIHB16TIME3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS2n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-19	RESERVED	R	0x0	
18-16	PSRAMSZ	R/W	0x2	PSRAM Row Size Defines the row size for the PSRAM controlled by CS2n 0x0 = No row size limitation 0x1 = 128 B 0x2 = 256 B 0x3 = 512 B 0x4 = 1024 B 0x5 = 2048 B 0x6 = 4096 B 0x7 = 8192 B
15-14	RESERVED	R	0x0	
13-12	CAPWIDTH	R/W	0x2	CS2n Inter-transfer Capture Width Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	

Table 16-46. EPIHB16TIME3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	WRWSM	R/W	0x0	CS2n Write Wait State Minus One This bit is used with the WRWS field in EPIHB16CFG3. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG3 register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB16CFG3.
3-1	RESERVED	R	0x0	
0	RDWSM	R/W	0x0	CS2n Read Wait State Minus One This field is used with RDWS field in EPIHB16CFG3. This bit is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG3. 0x1 (Write) = Wait state value is noRDWS; 1RDWS field is programmed in EPIHB16CFG3.

16.5.34 EPIHB8TIME4 Register (Offset = 0x31C) [reset = 0xA000]

EPI Host-Bus 8 Timing Extension (EPIHB8TIME4)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB8TIME4 to be valid, the MODE field must be 0x2.

EPIHB8TIME4 is shown in [Figure 16-63](#) and described in [Table 16-47](#).

Return to [Summary Table](#).

Figure 16-63. EPIHB8TIME4 Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
reserved-1							
R-0x2							
15	14	13	12	11	10	9	8
reserved-1		CAPWIDTH			RESERVED		
R-0x2		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-47. EPIHB8TIME4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS3n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-14	RESERVED	R	0x2	Bits [18:16] have the same RTL implementation as the HB16TIMEn register, even though this is not used in HB8 mode. Thus, the reset value of 0x2 is carried over from the PSRAMSZ bits of HB16TIMEn.
13-12	CAPWIDTH	R/W	0x2	CS3n Inter-transfer Capture Width Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock. 0x2 = 2 EPI clock. 0x3 = Reserved
11-5	RESERVED	R	0x0	
4	WRWSM	R/W	0x0	CS3n Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG4. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG4 register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB8CFG4.
3-1	RESERVED	R	0x0	

Table 16-47. EPIHB8TIME4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RDWSM	R/W	0x0	CS3n Read Wait State Minus One This field is used with RDWS field in EPIHB8CFG4. This bit is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG4. 0x1 (Write) = Wait state value is noRDWS; 1RDWS field is programmed in EPIHB8CFG4.

16.5.35 EPIHB16TIME4 Register (Offset = 0x31C) [reset = 0x00022000]

EPI Host-Bus 16 Timing Extension (EPIHB16TIME4)

NOTE: The MODE field in the EPICFG register determines which configuration is enabled.

For EPIHB16TIME 4 to be valid, the MODE field must be 0x3.

EPIHB16TIME4 is shown in [Figure 16-64](#) and described in [Table 16-48](#).

Return to [Summary Table](#).

Figure 16-64. EPIHB16TIME4 Register

31	30	29	28	27	26	25	24
RESERVED						IRDYDLY	
R-0x0						R/W-0x0	
23	22	21	20	19	18	17	16
RESERVED						PSRAMSZ	
R-0x0						R/W-0x2	
15	14	13	12	11	10	9	8
RESERVED		CAPWIDTH			RESERVED		
R-0x0		R/W-0x2			R-0x0		
7	6	5	4	3	2	1	0
RESERVED			WRWSM	RESERVED			RDWSM
R-0x0			R/W-0x0	R-0x0			R/W-0x0

Table 16-48. EPIHB16TIME4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0x0	
25-24	IRDYDLY	R/W	0x0	CS3n Input Ready Delay 0x0 = reserved 0x1 = Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x2 = Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. 0x3 = Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.
23-19	RESERVED	R	0x0	
18-16	PSRAMSZ	R/W	0x2	PSRAM Row Size. Defines the row size for the PSRAM controlled by CS3n 0x0 = No row size limitation 0x1 = 128 B 0x2 = 256 B 0x3 = 512 B 0x4 = 1024 B 0x5 = 2048 B 0x6 = 4096 B 0x7 = 8192 B
15-14	RESERVED	R	0x0	
13-12	CAPWIDTH	R/W	0x2	CS3n Inter-transfer Capture Width. Controls the delay between Host-Bus transfers. 0x0 = Reserved 0x1 = 1 EPI clock 0x2 = 2 EPI clock 0x3 = Reserved
11-5	RESERVED	R	0x0	

Table 16-48. EPIHB16TIME4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	WRWSM	R/W	0x0	CS3n Write Wait State Minus One. This bit is used with the WRWS field in EPIHB16CFG4. This field is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG4 register. 0x1 (Write) = Wait state value is noWRWS; 1WRWS field is programmed in EPIHB16CFG4.
3-1	RESERVED	R	0x0	
0	RDWSM	R/W	0x0	CS3n Read Wait State Minus One. This field is used with RDWS field in EPIHB16CFG4. This bit is not applicable in BURST mode. 0x0 = No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG4. 0x1 (Write) = Wait state value is noRDWS; 1RDWS field is programmed in EPIHB16CFG4.

16.5.36 EPIHBPSRAM Register (Offset = 0x360) [reset = 0x0]

EPI Host-Bus PSRAM (EPIHBPSRAM)

This register holds the PSRAM configuration register value. When the WRCRE bit in the EPIHB16CFGn register is set, all 21 bits of the EPIHBPSRAM register's CR value are written to the PSRAM's configuration register. When the RDCRE bit is set in the EPIHB16CFGn register, a read of the PSRAM's configuration register takes place and the value is written to bits[15:0] of the EPIHBPSRAM. Bits[20:16] will not contain any valid data.

EPIHBPSRAM is shown in [Figure 16-65](#) and described in [Table 16-49](#).

Return to [Summary Table](#).

Figure 16-65. EPIHBPSRAM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											CR																				
R-0x0											R/W-0x0																				

Table 16-49. EPIHBPSRAM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	RESERVED	R	0x0	
20-0	CR	R/W	0x0	PSRAM Config Register. During a configuration write, all 21 bits of the CR bit field are written to the PSRAM. During configuration reads, CR bits [15:0] of this register contain the configuration read of the PSRAM. CR[20:16] will not contain valid data.

General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H, Port J, Port K, Port L, Port M, Port N, Port P, Port Q, Port R, Port S, Port T). The GPIO module supports up to 140 programmable input/output pins, depending on the peripherals being used.

Topic	Page
17.1 Introduction	1192
17.2 Pad Capabilities	1192
17.3 Functional Description	1193
17.4 Initialization and Configuration	1198
17.5 GPIO Registers	1201

17.1 Introduction

The GPIO module has the following features:

- Up to 140 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 3.3 -V-tolerant in input configuration
- Advanced High Performance Bus accesses all ports:
 - Ports A to H and J; Ports K to N and P to T
- Fast toggle capable of a change every clock cycle for ports on AHB
- Programmable control for GPIO interrupts
 - Interrupt generation masking
 - Edge-triggered on rising, falling, or both
 - Level-sensitive on High or Low values
 - Per-pin interrupts available on Port P and Port Q
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence or a μ DMA transfer
- Pin state can be retained during Hibernation mode ; pins on port P can be programmed to wake on level in Hibernation mode
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
 - Weak pullup or pulldown resistors
 - 2-mA, 4-mA, 6-mA, 8-mA, 10-mA and 12-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
 - Slew rate control for 8-mA, 10-mA and 12-mA pad drive
 - Open drain enables
 - Digital input enables

17.2 Pad Capabilities

There are two main types of pads provided on the device:

- Fast GPIO pads: These pads provide variable, programmable drive strength and optimized voltage output levels.
- Slow GPIO pads: These pads provide 2-mA drive strength and are designed to be sensitive to voltage inputs. The following GPIOs port pins are designed with Slow GPIO Pads:
 - PJ1

See the *Specifications* chapter of the device-specific data sheet for details on the GPIO operating conditions for these two different pad types.

NOTE: Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: GPIODR2R, GPIODR4R, GPIODR8R, GPIODR12R, GPIOSLR, and GPIOODR .

NOTE: Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the GPIODR12R register, apply to these port pins.

17.3 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see [Figure 17-1](#) and [Figure 17-2](#)). The MSP432E4 microcontroller contains 18 ports and thus up to 140 of these physical GPIO blocks. Note that not all pins are implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, see the device-specific data sheet.

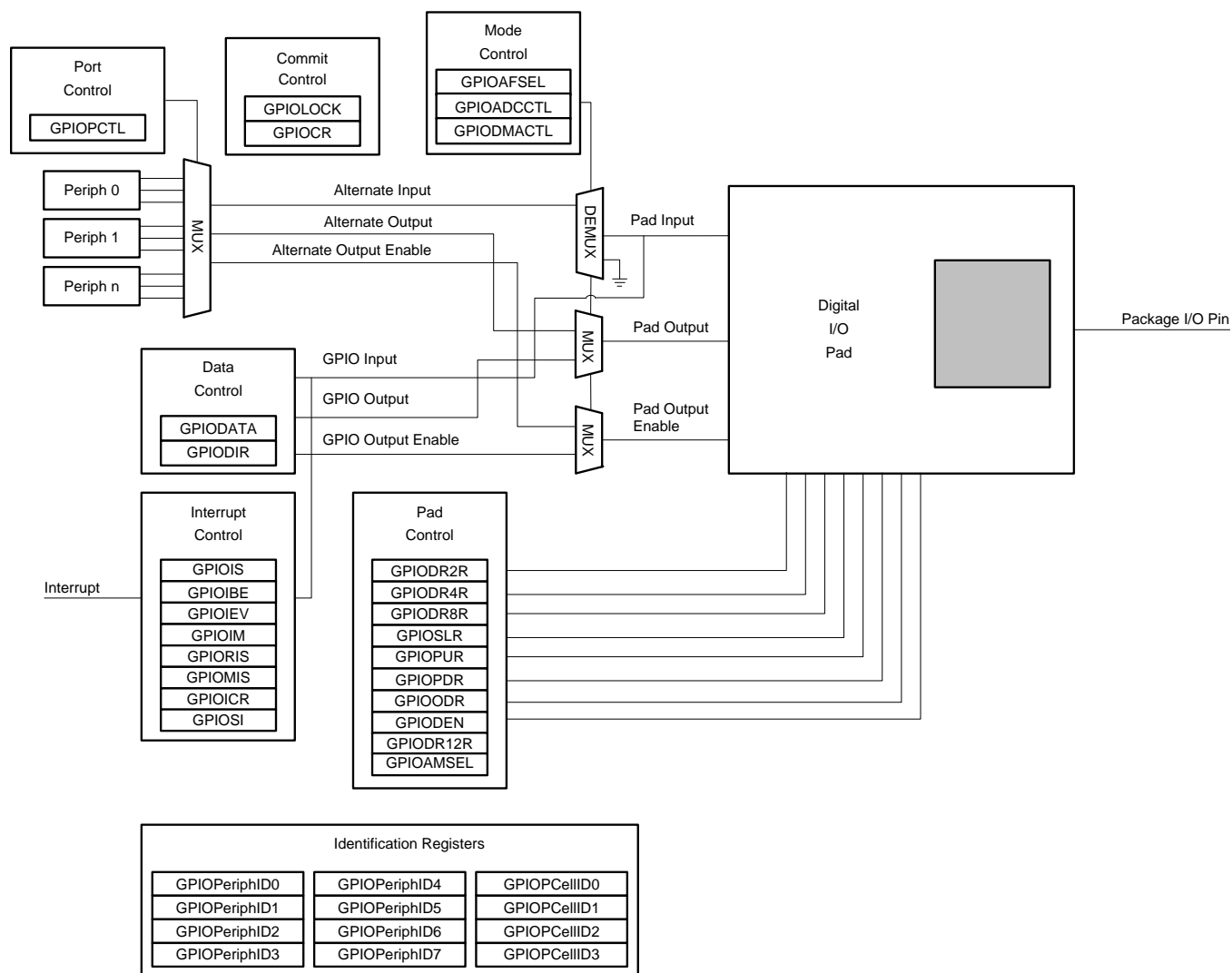


Figure 17-1. Digital I/O Pads

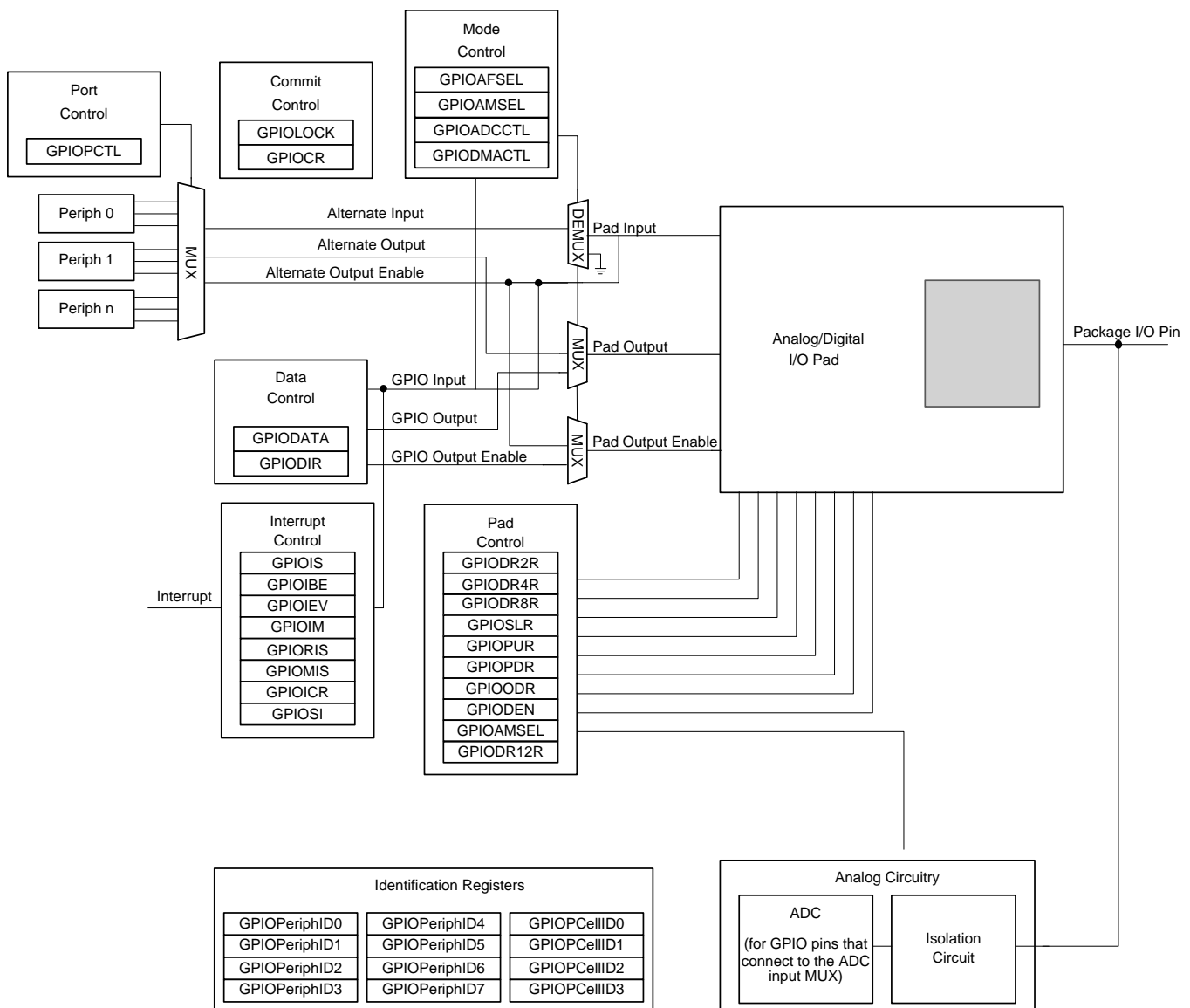


Figure 17-2. Analog and Digital I/O Pads

17.3.1 Data Control

The data control registers allow software to configure the operational modes of the GPIOs. The data direction register configures the GPIO as an input or an output while the data register either captures incoming data or drives it out to the pads.

NOTE: It is possible to create a software sequence that prevents the debugger from connecting to the MSP432E4 microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.

17.3.1.1 Data Direction Operation

The GPIO Direction (GPIODIR) register (see [Section 17.5.2](#)) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

17.3.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the GPIO Data (GPIODATA) register (see [Section 17.5.1](#)) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the GPIODATA register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set, the value of the GPIODATA register is altered. If the address bit is cleared, the data bit is left unchanged.

For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in [Figure 17-3](#), where u indicates that data is unchanged by the write. This example demonstrates how GPIODATA bits 5, 2, and 1 are written.

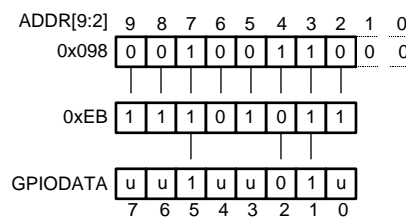


Figure 17-3. GPIODATA Write Example

During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in [Figure 17-4](#). This example shows how to read GPIODATA bits 5, 4, and 0.

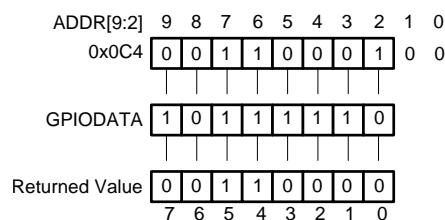


Figure 17-4. GPIODATA Read Example

17.3.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. These registers are used to select the source of the interrupt, its polarity, and the edge properties. When one or more GPIO inputs cause an interrupt, a single interrupt output is sent to the interrupt controller for the entire GPIO port. For edge-triggered interrupts, software must clear the interrupt to enable any further interrupts. For a level-sensitive interrupt, the external source must hold the level constant for the interrupt to be recognized by the controller.

Three registers define the edge or sense that causes interrupts:

- GPIO Interrupt Sense (GPIOIS) register (see [Section 17.5.3](#))
- GPIO Interrupt Both Edges (GPIOIBE) register (see [Section 17.5.4](#))

- GPIO Interrupt Event (GPIOIEV) register (see [Section 17.5.5](#))

Interrupts are enabled/disabled via the GPIO Interrupt Mask (GPIOIM) register (see [Section 17.5.6](#)).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the GPIO Raw Interrupt Status (GPIORIS) and GPIO Masked Interrupt Status (GPIOMIS) registers (see [Section 17.5.7](#) and [Section 17.5.8](#)). As the name implies, the GPIOMIS register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The GPIORIS register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

For a GPIO level-detect interrupt, the interrupt signal generating the interrupt must be held until serviced. Once the input signal deasserts from the interrupt generating logical sense, the corresponding RIS bit in the GPIORIS register clears. For a GPIO edge-detect interrupt, the RIS bit in the GPIORIS register is cleared by writing a 1 to the corresponding bit in the GPIO Interrupt Clear (GPIOICR) register (see [Section 17.5.9](#)). The corresponding GPIOMIS bit reflects the masked value of the RIS bit.

When programming the interrupt control registers (GPIOIS, GPIOIBE, or GPIOIEV), the interrupts should be masked (GPIOIM cleared). Writing any value to an interrupt control register can generate a spurious interrupt if the corresponding bits are enabled.

17.3.2.1 Interrupts Per Pin

Each pin of GPIO Port P and Port Q can trigger an interrupt. Each pin has a dedicated interrupt vector and can be handled by a separate interrupt handler. The PP0 and PQ0 interrupts serve as a master interrupt and provide a legacy aggregated interrupt version. For interrupt assignments, see .

NOTE: The OR'ed summary interrupt occurs on bit 0 of the GPIORIS register. For summary interrupt mode, software should set the GPIOIM register to 0xFF and mask the port pin interrupts 1 through 7 in the Interrupt Clear Enable (DISn) register (see [Section 2.4](#)). When servicing this interrupt, write a 1 to the corresponding bit in the UNPENDn register to clear the pending interrupt in the NVIC and clear the GPIORIS register pin interrupt bits by setting the IC field of the GPIOICR register to 0xFF.

17.3.2.2 ADC Trigger Source

Any GPIO pin can be configured to be an external trigger for the ADC using the GPIO ADC Control (GPIOADCCTL) register. If any GPIO is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set), and an interrupt for that port is generated, a trigger signal is sent to the ADC. If the ADC Event Multiplexer Select (ADCEMUX) register is configured to use the external trigger, an ADC conversion is initiated, see [Section 10.5.6](#). Note that whether the GPIO is configured to trigger on edge events or level events, a single-clock ADC trigger pulse is created in either event. Thus, when a level event is selected, the ADC sample sequence will run only one time and multiple sample sequences will not be executed if the level remains the same. It is recommended that edge events be used as ADC trigger source.

Note that if the Port B GPIOADCCTL register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode that allows code written for previous devices to operate on this microcontroller.

17.3.2.3 μ DMA Trigger Source

Any GPIO pin can be configured to be an external trigger for the μ DMA using the GPIO DMA Control (GPIODMACTL) register. If any GPIO is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set), a dma_req signal is sent to the μ DMA . If the μ DMA is configured to start a transfer based on the GPIO signal, a transfer is initiated. When transfer is complete, the dma_done signal is sent from the μ DMA to the GPIO and is reported as a DMA (done) interrupt in the GPIORIS register.

17.3.2.4 HIB Wake Source

GPIO pins K[7:4] on Port K can be configured as an external wake source for the hibernation (HIB) module. The pins can be configured in the following way:

1. Write 0x0000.0040 to the HIBCTL register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write any data to be retained during power cut to the HIBDATA register at offsets 0x030-0x06F.
3. Configure the GPIOWAKEPEN and GPIOWAKELVL registers at offsets 0x540 and 0x544 in the GPIO module. Enable the I/O wake pad configuration by writing 0x0000.0001 to the HIBIO register at offset 0x010.
4. When the IOWRC bit in the HIBIO register is read as 1, write 0x0000.0000 to the HIBO register to lock the current pad configuration so that any other writes to the GPIOWAKEPEN and GPIOWAKELVL register will be ignored.
5. The hibernation sequence may be initiated by writing 0x0000.0052 to the HIBCTL register.

The GPIOWAKESTAT register at offset 0x548 can be read to determine which port caused a wake pin assertion.

17.3.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the GPIODATA register is used to read or write the corresponding pins. When hardware control is enabled via the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the GPIO Port Control (GPIOCTL) register which selects one of several peripheral functions for each GPIO. For information on the configuration options, see .

NOTE: If any pin is to be used as an ADC input, the appropriate bit in the GPIOAMSEL register must be set to disable the analog isolation circuit.

17.3.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin. For pin numbers, see . Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), GPIO Pull Up Select (GPIOPUR) register (see [Section 17.5.15](#)), GPIO Pull-Down Select (GPIOPDR) register (see [Section 17.5.16](#)), and GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)) are not committed to storage unless the GPIO Lock (GPIOLOCK) register (see [Section 17.5.19](#)) has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register (see [Section 17.5.20](#)) have been set.

17.3.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the GPIODR2R, GPIODR4R, GPIODR8R, GPIODR12R, GPIOODR, GPIOPUR, GPIOPDR, GPIOSLR, and GPIODEN registers. These registers control drive strength, open-drain configuration, pullup and pulldown resistors, slew-rate control and digital input enable for each GPIO. If 3.3 V is applied to a GPIO configured as an open-drain output, the output voltage will depend on the strength of your pullup resistor. The GPIO pad is not electrically configured to output 3.3 V.

NOTE: Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: GPIODR2R, GPIODR4R, GPIODR8R, GPIODR12R, GPIOSLR, and GPIOODR .

NOTE: Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the GPIODR12R register, apply to these port pins.

17.3.5.1 Extended Drive Enable

The GPIO Peripheral Configuration (GPIOPC) register controls the extended drive modes of the GPIO. When the EDE bit in GPIO Peripheral Properties (GPIOPP) register is set and the EDMn bit field for a GPIO pin is non-zero in the GPIOPC register, the GPIODRnR registers do not drive their default value, but instead output an incremental drive strength, which has an additive effect. This allows for more drive strength possibilities. When the EDE bit is set and the EDMn bit field is non-zero, the 2 mA driver is always enabled. Any bits enabled in the GPIODR4R register for a pin with a non-zero EDMn value, add an additional 2 mA. Any bits set in the GPIODR8R add an extra 4 mA of drive. The GPIODR12R register is only valid when the EDMn value is 0x3. For this encoding, setting a bit in the GPIODR12R register adds 4 mA of drive to the already existing 8 mA, for a 12 mA drive strength. To attain a 10-mA drive strength, the pin's GPIODR12R and GPIODR8R register should be enabled; this would result in the addition of two, 4-mA current drivers to the already enabled 2-mA driver. The table below shows the drive capability options. If EDMn is 0x00, then the GPIODR2R, GPIODR4R, and GPIODR8R function as stated in their default register description.

NOTE: A GPIOPC register write must precede the configuration of the GPIODRnR registers for extended drive mode to take effect.

Table 17-1. GPIO Drive Strength Options

EDE (GPIOPP)	EDMn (GPIOPC)	GPIODR12R (+4 mA)	GPIODR8R (+4 mA)	GPIODR4R (+2 mA)	GPIODR2R (2 mA)	Drive (mA)
X	0x0	N/A	0	0	1	2
			0	1	0	4
			1	0	0	8
1	0x1	N/A	0	0	N/A	2
			0	1	N/A	4
			1	0	N/A	6
			1	1	N/A	8
1	0x3	0	0	0	N/A	2
		0	0	1	N/A	4
		0	1	0	N/A	6
		0	1	1	N/A	8
		1	1	0	N/A	10
		1	1	1	N/A	12
		1	0	N/A	N/A	N/A
1	0x2	N/A	N/A	N/A	N/A	N/A

17.3.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the GPIOPeriphID0 to GPIOPeriphID7 registers as well as the GPIOCellID0 to GPIOCellID3 registers.

17.4 Initialization and Configuration

To configure the GPIO pins of a particular port, follow these steps:

1. Enable the clock to the port by setting the appropriate bits in the RCGCGPIO register, see [Section 4.2.87](#). In addition, the SCGCGPIO and DCGCGPIO registers can be programmed in the same manner to enable clocking in Sleep and Deep-Sleep modes.
2. Set the direction of the GPIO port pins by programming the GPIODIR register. A write of a 1 indicates output and a write of a 0 indicates input.
3. Configure the GPIOAFSEL register to program each bit as a GPIO or alternate pin. If an alternate pin is chosen for a bit, then the PMCx field must be programmed in the GPIOPCTL register for the specific

peripheral required. There are also two registers, GPIOADCCTL and GPIODMACTL, which can be used to program a GPIO pin as a ADC or μ DMA trigger, respectively.

4. Set the EDMn field in the GPIOPC register as shown in [Table 17-1](#).
5. Set or clear the GPIODR4R register bits as shown in [Table 17-1](#).
6. Set or clear the GPIODR8R register bits as shown in [Table 17-1](#).
7. Set or clear the GPIODR12R register bits as shown in [Table 17-1](#).
8. Program each pad in the port to have either pullup, pulldown, or open drain functionality through the GPIOPUR, GPIOPDR, or GPIOODR register. Slew rate may also be programmed, if needed, through the GPIOSLR register.
9. To enable GPIO pins as digital I/Os, set the appropriate DEN bit in the GPIODEN register. To enable GPIO pins to their analog function (if available), set the GPIOAMSEL bit in the GPIOAMSEL register.
10. Program the GPIOIS, GPIOIBE, GPIOEV, and GPIOIM registers to configure the type, event, and mask of the interrupts for each port.

NOTE: To prevent false interrupts, the following steps should be taken when reconfiguring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the IME field in the GPIOIM register.
2. Configure the IS field in the GPIOIS register and the IBE field in the GPIOIBE register.
3. Clear the GPIORIS register.
4. Unmask the port by setting the IME field in the GPIOIM register.

11. Optionally, software can lock the configurations of the NMI and JTAG/SWD pins on the GPIO port pins, by setting the LOCK bits in the GPIOLOCK register.

When the internal POR signal is asserted and until otherwise configured, all GPIO pins are configured to be undriven (tristate): GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, and GPIOPUR = 0. [Table 17-2](#) shows all possible configurations of the GPIO pads and the control register settings required to achieve them. [Table 17-3](#) shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

Table 17-2. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value ⁽¹⁾										
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	DR12R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?	?
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?	?
Open Drain Input/Output (I2CSDA)	1	X	1	1	X	X	?	?	?	?	?
Digital Input/Output (I2CSCL)	1	X	0	1	X	X	?	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X	X
Digital Input (QE1)	1	X	0	1	?	?	X	X	X	X	X
Digital Output (PWM)	1	X	0	1	?	?	?	?	?	?	?
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?	?

⁽¹⁾ X = Ignored (don't care bit)

? = Can be either 0 or 1, depending on the configuration

Table 17-3. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value ⁽¹⁾							
		7	6	5	4	3	2	1	0
GPIOIS	0 = edge 1 = level	X	X	X	X	X	0	X	X
GPIOIBE	0 = single edge 1 = both edges	X	X	X	X	X	0	X	X
GPIOIEV	0 = Low level, or falling edge 1 = High level, or rising edge	X	X	X	X	X	1	X	X
GPIOIM	0 = masked 1 = not masked	0	0	0	0	0	1	0	0

⁽¹⁾ X = Ignored (don't care bit)

17.5 GPIO Registers

NOTE: The GPIO registers in this chapter are duplicated in each GPIO block; however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to unconnected bits has no effect, and reading unconnected bits returns no meaningful data. See the device-specific data sheet for the GPIOs included on any given device.

The offset is a hexadecimal increment to the register's address, relative to the base address of that GPIO port:

- GPIO Port A (AHB): 0x40058000 (ending address of 0x40058FFF)
- GPIO Port B (AHB): 0x40059000 (ending address of 0x40059FFF)
- GPIO Port C (AHB): 0x4005A000 (ending address of 0x4005AFFF)
- GPIO Port D (AHB): 0x4005B000 (ending address of 0x4005BFFF)
- GPIO Port E (AHB): 0x4005C000 (ending address of 0x4005CFFF)
- GPIO Port F (AHB): 0x4005D000 (ending address of 0x4005DFFF)
- GPIO Port G (AHB): 0x4005E000 (ending address of 0x4005EFFF)
- GPIO Port H (AHB): 0x4005F000 (ending address of 0x4005FFFF)
- GPIO Port J (AHB): 0x40060000 (ending address of 0x40060FFF)
- GPIO Port K (AHB): 0x40061000 (ending address of 0x40061FFF)
- GPIO Port L (AHB): 0x40062000 (ending address of 0x40062FFF)
- GPIO Port M (AHB): 0x40063000 (ending address of 0x40063FFF)
- GPIO Port N (AHB): 0x40064000 (ending address of 0x40064FFF)
- GPIO Port P (AHB): 0x40065000 (ending address of 0x40065FFF)
- GPIO Port Q (AHB): 0x40066000 (ending address of 0x40066FFF)
- GPIO Port R (AHB): 0x40067000 (ending address of 0x40067FFF)
- GPIO Port S (AHB): 0x40068000 (ending address of 0x40068FFF)
- GPIO Port T (AHB): 0x40069000 (ending address of 0x40069FFF)

Note that each GPIO module clock must be enabled before the registers can be programmed (see [Section 4.2.87](#)). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and high-impedance by default (GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0, and GPIOPCTL = 0). Special consideration pins may be programmed to a nonGPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset (POR) returns these GPIO to their original special consideration state.

Table 17-4. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0
PE[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0

⁽¹⁾ This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the GPIOLOCK register and uncommitting it by setting the GPIOCR register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see [Section 17.3.4](#).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

The default register type for the GPIOCR register is read-only for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins (see the device-specific data sheet for pin numbers). These six pins are the only GPIOs that are protected by the GPIOCR register. Because of this, the register type for the corresponding GPIO Ports is RW.

The default reset value for the GPIOCR register is 0x000000FF for all GPIO pins, with the exception of the NMI and JTAG/SWD pins (see the device-specific data sheet for pin numbers). To ensure that the JTAG and NMI pins are not accidentally programmed as GPIO pins, these pins default to noncommittable. Because of this, the default reset value of GPIOCR changes for the corresponding ports.

Table 17-5 lists the memory-mapped registers for the GPIO. All register offset addresses not listed in Table 17-5 should be considered as reserved locations and the register contents should not be modified.

Table 17-5. GPIO Registers

Offset	Acronym	Register Name	Section
0x0	GPIODATA	GPIO Data	Section 17.5.1
0x400	GPIODIR	GPIO Direction	Section 17.5.2
0x404	GPIOIS	GPIO Interrupt Sense	Section 17.5.3
0x408	GPIOIBE	GPIO Interrupt Both Edges	Section 17.5.4
0x40C	GPIOIEV	GPIO Interrupt Event	Section 17.5.5
0x410	GPIOIM	GPIO Interrupt Mask	Section 17.5.6
0x414	GPIORIS	GPIO Raw Interrupt Status	Section 17.5.7
0x418	GIOMIS	GPIO Masked Interrupt Status	Section 17.5.8
0x41C	GPIOICR	GPIO Interrupt Clear	Section 17.5.9
0x420	GPIOAFSEL	GPIO Alternate Function Select	Section 17.5.10
0x500	GPIODR2R	GPIO 2-mA Drive Select	Section 17.5.11
0x504	GPIODR4R	GPIO 4-mA Drive Select	Section 17.5.12
0x508	GPIODR8R	GPIO 8-mA Drive Select	Section 17.5.13
0x50C	GPIOODR	GPIO Open Drain Select	Section 17.5.14
0x510	GIOPUR	GPIO Pullup Select	Section 17.5.15
0x514	GIOPDR	GPIO Pulldown Select	Section 17.5.16
0x518	GPISLR	GPIO Slew Rate Control Select	Section 17.5.17
0x51C	GPIDEN	GPIO Digital Enable	Section 17.5.18
0x520	GPIOLOCK	GPIO Lock	Section 17.5.19
0x524	GPIOCR	GPIO Commit	Section 17.5.20
0x528	GPIOAMSEL	GPIO Analog Mode Select	Section 17.5.21
0x52C	GIOPCTL	GPIO Port Control	Section 17.5.22
0x530	GPIOADCCCTL	GPIO ADC Control	Section 17.5.23
0x534	GPIDMACTL	GPIO DMA Control	Section 17.5.24
0x538	GPISI	GPIO Select Interrupt	Section 17.5.25
0x53C	GPIDR12R	GPIO 12-mA Drive Select	Section 17.5.26
0x540	GIOWAKEPEN	GPIO Wake Pin Enable	Section 17.5.27
0x544	GIOWAKELVL	GPIO Wake Level	Section 17.5.28
0x548	GIOWAKESTAT	GPIO Wake Status	Section 17.5.29
0xFC0	GIOPP	GPIO Peripheral Property	Section 17.5.30
0xFC4	GIOPC	GPIO Peripheral Configuration	Section 17.5.31
0xFD0	GIOPeriphID4	GPIO Peripheral Identification 4	Section 17.5.32
0xFD4	GIOPeriphID5	GPIO Peripheral Identification 5	Section 17.5.33
0xFD8	GIOPeriphID6	GPIO Peripheral Identification 6	Section 17.5.34

Table 17-5. GPIO Registers (continued)

Offset	Acronym	Register Name	Section
0xFDC	GPIOPeriphID7	GPIO Peripheral Identification 7	Section 17.5.35
0xFE0	GPIOPeriphID0	GPIO Peripheral Identification 0	Section 17.5.36
0xFE4	GPIOPeriphID1	GPIO Peripheral Identification 1	Section 17.5.37
0xFE8	GPIOPeriphID2	GPIO Peripheral Identification 2	Section 17.5.38
0xFEC	GPIOPeriphID3	GPIO Peripheral Identification 3	Section 17.5.39
0xFF0	GPIOCellID0	GPIO PrimeCell Identification 0	Section 17.5.40
0xFF4	GPIOCellID1	GPIO PrimeCell Identification 1	Section 17.5.41
0xFF8	GPIOCellID2	GPIO PrimeCell Identification 2	Section 17.5.42
0xFFC	GPIOCellID3	GPIO PrimeCell Identification 3	Section 17.5.43

Complex bit access types are encoded to fit into small table cells. [Table 17-6](#) shows the codes that are used for access types in this section.

Table 17-6. GPIO Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

17.5.1 GPIODATA Register (Offset = 0x0) [reset = 0x0]

GPIO Data (GPIODATA)

The GPIODATA register is the data register. In software control mode, values written in the GPIODATA register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the GPIO Direction (GPIODIR) register (see [Section 17.5.2](#)).

In order to write to GPIODATA, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

Similarly, the values read from this register are determined for each bit by the mask bit derived from the address used to access the data register, bits [9:2]. Bits that are set in the address mask cause the corresponding bits in GPIODATA to be read, and bits that are clear in the address mask cause the corresponding bits in GPIODATA to be read as 0, regardless of their value.

A read from GPIODATA returns the last bit value written if the respective pins are configured as outputs, or it returns the value on the corresponding input pin when these are configured as inputs. All bits are cleared by a reset. See [Section 17.3.1.2](#) for examples of reads and writes.

GPIODATA is shown in [Figure 17-5](#) and described in [Table 17-7](#).

Return to [Summary Table](#).

Figure 17-5. GPIODATA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DATA							
R-0x0																								R/W-0x0							

Table 17-7. GPIODATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DATA	R/W	0x0	<p>GPIO Data.</p> <p>This register is virtually mapped to 256 locations in the address space.</p> <p>To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2].</p> <p>Reads from this register return its current state.</p> <p>Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs.</p>

17.5.2 GPIODIR Register (Offset = 0x400) [reset = 0x0]

GPIO Direction (GPIODIR)

The GPIODIR register is the data direction register. Setting a bit in the GPIODIR register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIODIR is shown in [Figure 17-6](#) and described in [Table 17-8](#).

Return to [Summary Table](#).

Figure 17-6. GPIODIR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DIR							
R-0x0																								R/W-0x0							

Table 17-8. GPIODIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DIR	R/W	0x0	GPIO Data Direction 0x0 = Corresponding pin is an input. 0x1 = Corresponding pins is an output.

17.5.3 GPIOIS Register (Offset = 0x404) [reset = 0x0]

GPIO Interrupt Sense (GPIOIS)

The GPIOIS register is the interrupt sense register. Setting a bit in the GPIOIS register configures the corresponding pin to detect levels, while clearing a bit configures the corresponding pin to detect edges. All bits are cleared by a reset.

NOTE: To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the IME field in the GPIOIM register.
2. Configure the IS field in the GPIOIS register and the IBE field in the GPIOIBE register.
3. Clear the GPIORIS register.
4. Unmask the port by setting the IME field in the GPIOIM register.

GPIOIS is shown in [Figure 17-7](#) and described in [Table 17-9](#).

Return to [Summary Table](#).

Figure 17-7. GPIOIS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																IS															
R-0x0																R/W-0x0															

Table 17-9. GPIOIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	IS	R/W	0x0	GPIO Interrupt Sense 0x0 = The edge on the corresponding pin is detected (edge-sensitive). 0x1 = The level on the corresponding pin is detected (level-sensitive).

17.5.4 GPIOIBE Register (Offset = 0x408) [reset = 0x0]

GPIO Interrupt Both Edges (GPIOIBE)

The GPIOIBE register allows both edges to cause interrupts. When the corresponding bit in the GPIO Interrupt Sense (GPIOIS) register (see [Section 17.5.3](#)) is set to detect edges, setting a bit in the GPIOIBE register configures the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the GPIO Interrupt Event (GPIOIEV) register (see [Section 17.5.5](#)). Clearing a bit configures the pin to be controlled by the GPIOIEV register. All bits are cleared by a reset.

NOTE: To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the IME field in the GPIOIM register.
2. Configure the IS field in the GPIOIS register and the IBE field in the GPIOIBE register.
3. Clear the GPIORIS register.
4. Unmask the port by setting the IME field in the GPIOIM register.

GPIOIBE is shown in [Figure 17-8](#) and described in [Table 17-10](#).

Return to [Summary Table](#).

Figure 17-8. GPIOIBE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								IBE							
R-0x0																								R/W-0x0							

Table 17-10. GPIOIBE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	IBE	R/W	0x0	GPIO Interrupt Both Edges 0x0 = Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) register (see). 0x1 = Both edges on the corresponding pin trigger an interrupt.

17.5.5 GPIOIEV Register (Offset = 0x40C) [reset = 0x0]

GPIO Interrupt Event (GPIOIEV)

The GPIOIEV register is the interrupt event register. Setting a bit in the GPIOIEV register configures the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the GPIO Interrupt Sense (GPIOIS) register (see [Section 17.5.3](#)). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in the GPIOIS register. All bits are cleared by a reset.

GPIOIEV is shown in [Figure 17-9](#) and described in [Table 17-11](#).

Return to [Summary Table](#).

Figure 17-9. GPIOIEV Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								IEV							
R-0x0																								R/W-0x0							

Table 17-11. GPIOIEV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	IEV	R/W	0x0	GPIO Interrupt Event 0x0 = A falling edge or a Low level on the corresponding pin triggers an interrupt. 0x1 = A rising edge or a High level on the corresponding pin triggers an interrupt.

17.5.6 GPIOIM Register (Offset = 0x410) [reset = 0x0]

GPIO Interrupt Mask (GPIOIM)

The GPIOIM register is the interrupt mask register. Setting a bit in the GPIOIM register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

GPIOIM is shown in [Figure 17-10](#) and described in [Table 17-12](#).

Return to [Summary Table](#).

Figure 17-10. GPIOIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							DMAIME
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
IME							
R/W-0x0							

Table 17-12. GPIOIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	DMAIME	R/W	0x0	GPIO μ DMA Done Interrupt Mask Enable 0x0 = The μ DMA done interrupt is masked and does not cause an interrupt. 0x1 = The μ DMA done interrupt is not masked and can generate an interrupt to the interrupt controller.
7-0	IME	R/W	0x0	GPIO Interrupt Mask Enable 0x0 = The interrupt from the corresponding pin is masked. 0x1 = The interrupt from the corresponding pin is sent to the interrupt controller.

17.5.7 GPIORIS Register (Offset = 0x414) [reset = 0x0]

GPIO Raw Interrupt Status (GPIORIS)

The GPIORIS register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin or if a μ DMA done interrupt occurs. If the corresponding bit in the GPIO Interrupt Mask (GPIOIM) register (see [Section 17.5.6](#)) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. For a GPIO level-detect interrupt, the interrupt signal generating the interrupt must be held until serviced. Once the input signal deasserts from the interrupt generating logical sense, the corresponding RIS bit in the GPIORIS register clears. For a GPIO edge-detect interrupt, the RIS bit in the GPIORIS register is cleared by writing a 1 to the corresponding bit in the GPIO Interrupt Clear (GPIOICR) register. The corresponding GPIOMIS bit reflects the masked value of the RIS bit.

GPIORIS is shown in [Figure 17-11](#) and described in [Table 17-13](#).

Return to [Summary Table](#).

Figure 17-11. GPIORIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							DMARIS
R-0x0							R-0x0
7	6	5	4	3	2	1	0
RIS							
R-0x0							

Table 17-13. GPIORIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	DMARIS	R	0x0	GPIO μ DMA Done Interrupt Raw Status. This bit is cleared by writing a 1 to the DMAIC bit in the GPIOICR register. 0x0 = A μ DMA done interrupt has not occurred. 0x1 = A μ DMA done interrupt has occurred and an interrupt has been triggered and is pending.
7-0	RIS	R	0x0	GPIO Interrupt Raw Status. For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the GPIOICR register. For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted. 0x0 = An interrupt condition has not occurred on the corresponding pin. 0x1 = An interrupt condition has occurred on the corresponding pin.

17.5.8 GPIOMIS Register (Offset = 0x418) [reset = 0x0]

GPIO Masked Interrupt Status (GPIOMIS)

The GPIOMIS register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

Note that if the Port B GPIOADCCTL register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

GPIOMIS is the state of the interrupt after masking.

GPIOMIS is shown in [Figure 17-12](#) and described in [Table 17-14](#).

Return to [Summary Table](#).

Figure 17-12. GPIOMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							DMAMIS
R-0x0							R-0x0
7	6	5	4	3	2	1	0
MIS							
R-0x0							

Table 17-14. GPIOMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	DMAMIS	R	0x0	GPIO μ DMA Done Masked Interrupt Status. This bit is cleared by writing a 1 to the DMAIC bit in the GPIOICR register. 0x0 = The μ DMA done interrupt is masked or has not occurred. 0x1 = An unmasked μ DMA done interrupt has occurred.
7-0	MIS	R	0x0	GPIO Masked Interrupt Status. For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the GPIOICR register. For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted. 0x0 = An interrupt condition on the corresponding pin is masked or has not occurred. 0x1 = An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.

17.5.9 GPIOICR Register (Offset = 0x41C) [reset = 0x0]

GPIO Interrupt Clear (GPIOICR)

The GPIOICR register is the interrupt clear register. Writing a 1 to the DMAIC bit in this register clears the corresponding interrupt bit in the GPIORIS and GPIOMIS registers. For edge-detect interrupts, writing a 1 to the IC bit in the GPIOICR register clears the corresponding bit in the GPIORIS and GPIOMIS registers. If the interrupt is a level-detect, the IC bit in this register has no effect. In addition, writing a 0 to any of the bits in the GPIOICR register has no effect.

GPIOICR is shown in [Figure 17-13](#) and described in [Table 17-15](#).

Return to [Summary Table](#).

Figure 17-13. GPIOICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							DMAIC
R-0x0							W1C-0x0
7	6	5	4	3	2	1	0
IC							
W1C-0x0							

Table 17-15. GPIOICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	DMAIC	W1C	0x0	GPIO μ DMA Interrupt Clear 0x0 = The μ DMA done interrupt is unaffected. 0x1 = The μ DMA done interrupt is cleared.
7-0	IC	W1C	0x0	GPIO Interrupt Clear 0x0 = The corresponding interrupt is unaffected. 0x1 = The corresponding interrupt is cleared.

17.5.10 GPIOAFSEL Register (Offset = 0x420) [reset = X]

GPIO Alternate Function Select (GPIOAFSEL)

NOTE: Tamper pins enabled in the Hibernate Tamper IO Control and Status (HIBTPIO) register override the AFSEL configuration.

The GPIOAFSEL register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The GPIO Port Control (GPIOCTL) register is used to select one of the possible functions. See the device-specific data sheet for details on which functions are muxed on each GPIO pin. The reset value for this register is 0x00000000 for GPIO ports that are not listed in [Table 17-16](#).

The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and high-impedance by default (GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0, and GPIOCTL = 0). Special consideration pins may be programmed to a nonGPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset (POR) returns these GPIO to their original special consideration state.

Table 17-16. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOCTL	GPIOCR
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0
PE[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0

⁽¹⁾ This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the GPIOLOCK register and uncommitting it by setting the GPIOCR register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see [Section 17.3.4](#).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

NOTE: It is possible to create a software sequence that prevents the debugger from connecting to the MSP432E4 microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see for pin numbers). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), GPIO Pullup Select (GPIOPUR) register (see [Section 17.5.15](#)), GPIO Pulldown Select (GPIOPDR) register (see [Section 17.5.16](#)), and GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)) are not committed to storage unless the GPIO Lock (GPIOLOCK) register (see [Section 17.5.19](#)) has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register (see [Section 17.5.20](#)) have been set.

When using the I²C module, in addition to setting the GPIOAFSEL register bits for the I²C clock and data pins, the data pins should be set to open drain using the GPIO Open Drain Select (GPIOODR) register (see examples in [Section 17.4](#)).

GPIOAFSEL is shown in [Figure 17-14](#) and described in [Table 17-17](#).

Return to [Summary Table](#).

Figure 17-14. GPIOAFSEL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																AFSEL															
R-0x0																R/W-X															

Table 17-17. GPIOAFSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	AFSEL	R/W	X	GPIO Alternate Function Select 0x0 = The associated pin functions as a GPIO and is controlled by the GPIO registers. 0x1 = The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

17.5.11 GPIODR2R Register (Offset = 0x500) [reset = 0xFF]

GPIO 2-mA Drive Select (GPIODR2R)

The GPIODR2R register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV2 bit for a GPIO signal, the corresponding DRV4 bit in the GPIODR4R register and DRV8 bit in the GPIODR8R register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

NOTE: This register has no effect on port pins PL6 and PL7.

GPIODR2R is shown in [Figure 17-15](#) and described in [Table 17-18](#).

Return to [Summary Table](#).

Figure 17-15. GPIODR2R Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DRV2							
R-0x0																								R/W-0xFF							

Table 17-18. GPIODR2R Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable. Setting a bit in either the GPIODR4 register or the GPIODR8 register clears the corresponding 2-mA enable bit. The change is effective on the next clock cycle. 0x0 = The drive for the corresponding GPIO pin is controlled by the GPIODR4R or GPIODR8R register. 0x1 = The corresponding GPIO pin has 2-mA drive.

17.5.12 GPIODR4R Register (Offset = 0x504) [reset = 0x0]

GPIO 4-mA Drive Select (GPIODR4R)

The GPIODR4R register is the 4-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV4 bit for a GPIO signal, the corresponding DRV2 bit in the GPIODR2R register and DRV8 bit in the GPIODR8R register are automatically cleared by hardware.

NOTE: This register has no effect on port pins PL6 and PL7.

GPIODR4R is shown in [Figure 17-16](#) and described in [Table 17-19](#).

Return to [Summary Table](#).

Figure 17-16. GPIODR4R Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DRV4							
R-0x0																								R/W-0x0							

Table 17-19. GPIODR4R Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DRV4	R/W	0x0	Output Pad 4-mA Drive Enable. Setting a bit in either the GPIODR2 register or the GPIODR8 register clears the corresponding 4-mA enable bit. The change is effective on the next clock cycle. 0x0 = The drive for the corresponding GPIO pin is controlled by the GPIODR2R or GPIODR8R register. 0x1 = The corresponding GPIO pin has 4-mA drive.

17.5.13 GPIODR8R Register (Offset = 0x508) [reset = 0x0]

GPIO 8-mA Drive Select (GPIODR8R)

The GPIODR8R register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV8 bit for a GPIO signal, the corresponding DRV2 bit in the GPIODR2R register and DRV4 bit in the GPIODR4R register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

NOTE: There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the V_{OH}/V_{OL} levels. See for further information.

NOTE: This register has no effect on port pins PL6 and PL7.

GPIODR8R is shown in [Figure 17-17](#) and described in [Table 17-20](#).

Return to [Summary Table](#).

Figure 17-17. GPIODR8R Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DRV8							
R-0x0																								R/W-0x0							

Table 17-20. GPIODR8R Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DRV8	R/W	0x0	Output Pad 8-mA Drive Enable. Setting a bit in either the GPIODR2 register or the GPIODR4 register clears the corresponding 8-mA enable bit. The change is effective on the next clock cycle. 0x0 = The drive for the corresponding GPIO pin is controlled by the GPIODR2R or GPIODR4R register. 0x1 = The corresponding GPIO pin has 8-mA drive.

17.5.14 GPIOODR Register (Offset = 0x50C) [reset = 0x0]

GPIO Open Drain Select (GPIOODR)

The GPIOODR register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)).

Corresponding bits in the drive strength and slew rate control registers (GPIODR2R, GPIODR4R, GPIODR8R, and GPIOSLR) can be set to achieve the desired fall times. The GPIO acts as an input if the corresponding bit in the GPIODIR register is cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the I²C module, in addition to configuring the data pin to open drain, the GPIO Alternate Function Select (GPIOAFSEL) register bits for the I²C clock and data pins should be set (see examples in [Section 17.4](#)).

NOTE: This register has no effect on port pins PL6 and PL7.

GPIOODR is shown in [Figure 17-18](#) and described in [Table 17-21](#).

Return to [Summary Table](#).

Figure 17-18. GPIOODR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ODE							
R-0x0																								R/W-0x0							

Table 17-21. GPIOODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	ODE	R/W	0x0	Output Pad Open Drain Enable 0x0 = The corresponding pin is not configured as open drain. 0x1 = The corresponding pin is configured as open drain.

17.5.15 GPIOPUR Register (Offset = 0x510) [reset = X]

GPIO Pullup Select (GPIOPUR)

The GPIOPUR register is the pullup control register. When a bit is set, a weak pullup resistor on the corresponding GPIO signal is enabled. Setting a bit in GPIOPUR automatically clears the corresponding bit in the GPIO Pulldown Select (GPIOPDR) register (see [Section 17.5.16](#)). Write access to this register is protected with the GPIOCR register. Bits in GPIOCR that are cleared prevent writes to the equivalent bit in this register.

The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and high-impedance by default (GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0, and GPIOPCTL = 0). Special consideration pins may be programmed to a nonGPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset (POR) returns these GPIO to their original special consideration state.

The reset value for this register is 0x00000000 for GPIO ports that are not listed in [Table 17-22](#).

Table 17-22. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0
PE[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0

⁽¹⁾ This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the GPIOLOCK register and uncommitting it by setting the GPIOCR register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see [Section 17.3.4](#).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

NOTE: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see for pin numbers). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), GPIO Pullup Select (GPIOPUR) register (see [Section 17.5.15](#)), GPIO Pulldown Select (GPIOPDR) register (see [Section 17.5.16](#)), and GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)) are not committed to storage unless the GPIO Lock (GPIOLOCK) register (see [Section 17.5.19](#)) has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register (see [Section 17.5.20](#)) have been set.

GPIOPUR is shown in [Figure 17-19](#) and described in [Table 17-23](#).

Return to [Summary Table](#).

Figure 17-19. GPIOPUR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PUE							
R-0x0																								R/W-X							

Table 17-23. GPIOPUR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PUE	R/W	X	Pad Weak Pullup Enable. Setting a bit in the GPIOPDR register clears the corresponding bit in the GPIOPUR register. The change is effective on the next clock cycle. 0x0 = The corresponding pin's weak pullup resistor is disabled. 0x1 = The corresponding pin's weak pullup resistor is enabled.

17.5.16 GPIOPDR Register (Offset = 0x514) [reset = 0x0]

GPIO Pulldown Select (GPIOPDR)

The GPIOPDR register is the pulldown control register. When a bit is set, a weak pulldown resistor on the corresponding GPIO signal is enabled. Setting a bit in GPIOPDR automatically clears the corresponding bit in the GPIO Pullup Select (GPIOPUR) register (see [Section 17.5.15](#)).

The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and high-impedance by default (GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0, and GPIOPCTL = 0). Special consideration pins may be programmed to a nonGPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset (POR) returns these GPIO to their original special consideration state.

Table 17-24. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0
PE[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0

⁽¹⁾ This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the GPIOLOCK register and uncommitting it by setting the GPIOCR register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see [Section 17.3.4](#).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

NOTE: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see for pin numbers). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), GPIO Pullup Select (GPIOPUR) register (see [Section 17.5.15](#)), GPIO Pulldown Select (GPIOPDR) register (see [Section 17.5.16](#)), and GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)) are not committed to storage unless the GPIO Lock (GPIOLOCK) register (see [Section 17.5.19](#)) has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register (see [Section 17.5.20](#)) have been set.

GPIOPDR is shown in [Figure 17-20](#) and described in [Table 17-25](#).

Return to [Summary Table](#).

Figure 17-20. GPIOPDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PDE							
R-0x0																								R/W-0x0							

Table 17-25. GPIOPDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PDE	R/W	0x0	Pad Weak Pulldown Enable. Setting a bit in the GPIOPUR register clears the corresponding bit in the GPIOPDR register. The change is effective on the next clock cycle. 0x0 = The corresponding pin's weak pulldown resistor is disabled. 0x1 = The corresponding pin's weak pulldown resistor is enabled.

17.5.17 GPIOSLR Register (Offset = 0x518) [reset = 0x0]

GPIO Slew Rate Control Select (GPIOSLR)

The GPIOSLR register is the slew rate control register. Slew rate control is only available when using the 8-mA, 10-mA or 12-mA drive strength option. The selection of drive strength is done through the GPIO Drive Select (GPIODRnR registers and the GPIO Peripheral Configuration (GPIOPC) register.

NOTE: This register has no effect on port pins PL6 and PL7.

GPIOSLR is shown in [Figure 17-21](#) and described in [Table 17-26](#).

Return to [Summary Table](#).

Figure 17-21. GPIOSLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
RESERVED																								SRL											
R-0x0																								R/W-0x0											

Table 17-26. GPIOSLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	SRL	R/W	0x0	Slew Rate Limit Enable (8-mA, 10-mA and 12-mA drive only) 0x0 = Slew rate control is disabled for the corresponding pin. 0x1 = Slew rate control is enabled for the corresponding pin.

17.5.18 GPIODEN Register (Offset = 0x51C) [reset = X]

GPIO Digital Enable (GPIODEN)

NOTE: Pins configured as digital inputs are Schmitt-triggered.

The GPIODEN register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and high-impedance by default (GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0, and GPIOPCTL = 0). Special consideration pins may be programmed to a nonGPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset (POR) returns these GPIO to their original special consideration state.

The reset value for this register is 0x00000000 for GPIO ports that are not listed in [Table 17-27](#).

Table 17-27. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0
PE[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0

⁽¹⁾ This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the GPIOLOCK register and uncommitting it by setting the GPIOCR register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see [Section 17.3.4](#).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

NOTE: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see for pin numbers). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)), GPIO Pullup Select (GPIOPUR) register (see [Section 17.5.15](#)), GPIO Pulldown Select (GPIOPDR) register (see [Section 17.5.16](#)), and GPIO Digital Enable (GPIODEN) register (see [Section 17.5.18](#)) are not committed to storage unless the GPIO Lock (GPIOLOCK) register (see [Section 17.5.19](#)) has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register (see [Section 17.5.20](#)) have been set.

GPIODEN is shown in [Figure 17-22](#) and described in [Table 17-28](#).

[Return to Summary Table.](#)

Figure 17-22. GPIODEN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DEN							
R-0x0																								R/W-X							

Table 17-28. GPIODEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DEN	R/W	X	Digital Enable 0x0 = The digital functions for the corresponding pin are disabled. 0x1 = The digital functions for the corresponding pin are enabled.

17.5.19 GPIOLOCK Register (Offset = 0x520) [reset = 0x1]

GPIO Lock (GPIOLOCK)

The GPIOLOCK register enables write access to the GPIOCR register (see [Section 17.5.20](#)). Writing 0x4C4F434B to the GPIOLOCK register unlocks the GPIOCR register. Writing any other value to the GPIOLOCK register re-enables the locked state. Reading the GPIOLOCK register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the GPIOLOCK register returns 0x00000001. When write accesses are enabled, or unlocked, reading the GPIOLOCK register returns 0x00000000.

GPIOLOCK is shown in [Figure 17-23](#) and described in [Table 17-29](#).

Return to [Summary Table](#).

Figure 17-23. GPIOLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK																															
R/W-0x1																															

Table 17-29. GPIOLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LOCK	R/W	0x1	<p>GPIO Lock.</p> <p>A write of the value 0x4C4F434B unlocks the GPIO Commit (GPIOCR) register for write access. A write of any other value or a write to the GPIOCR register reapplies the lock, preventing any register updates.</p> <p>A read of this register returns the following values:</p> <p>0x0 = The GPIOCR register is unlocked and may be modified.</p> <p>0x1 = The GPIOCR register is locked and may not be modified.</p>

17.5.20 GPIOCR Register (Offset = 0x524) [reset = X]

GPIO Commit (GPIOCR)

The GPIOCR register is the commit register. The value of the GPIOCR register determines which bits of the GPIOAFSEL, GPIOPUR, GPIOPDR, and GPIODEN registers are committed when a write to these registers is performed. If a bit in the GPIOCR register is cleared, the data being written to the corresponding bit in the GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN registers cannot be committed and retains its previous value. If a bit in the GPIOCR register is set, the data being written to the corresponding bit of the GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN registers is committed to the register and reflects the new value.

The contents of the GPIOCR register can only be modified if the status in the GPIOLOCK register is unlocked. Writes to the GPIOCR register are ignored if the status in the GPIOLOCK register is locked.

NOTE: This register is designed to prevent accidental programming of the registers that control connectivity to the NMI and JTAG/SWD debug hardware. By initializing the bits of the GPIOCR register to 0 for the NMI and JTAG/SWD pins (see for pin numbers), the NMI and JTAG/SWD debug port can only be converted to GPIOs through a deliberate set of writes to the GPIOLOCK, GPIOCR, and the corresponding registers.

Because this protection is currently only implemented on the NMI and JTAG/SWD pins (see for pin numbers), all of the other bits in the GPIOCR registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN register bits of these other pins.

The default register type for the GPIOCR register is RO for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins. These six pins are the only GPIOs that are protected by the GPIOCR register. Because of this, the register type for the corresponding GPIO Ports is RW.

The default reset value for the GPIOCR register is 0x000000FF for all GPIO pins, with the exception of the NMI and JTAG/SWD pins. To ensure that the JTAG and NMI pins are not accidentally programmed as GPIO pins, these pins default to noncommittable. Because of this, the default reset value of GPIOCR changes for the corresponding ports.

GPIOCR is shown in [Figure 17-24](#) and described in [Table 17-30](#).

Return to [Summary Table](#).

Figure 17-24. GPIOCR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CR							
R-0x0																								X							

Table 17-30. GPIOCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CR		X	GPIO Commit 0x0 = The corresponding GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN bits cannot be written. 0x1 = The corresponding GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN bits can be written.

17.5.21 GPIOAMSEL Register (Offset = 0x528) [reset = 0x0]

GPIO Analog Mode Select (GPIOAMSEL)

NOTE: This register is only valid for ports and pins that can be used as ADC AINx inputs.

If any pin is to be used as an ADC input, the appropriate bit in GPIOAMSEL must be set to disable the analog isolation circuit.

The GPIOAMSEL register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 3.3 -V source and affect analog operation, analog circuitry requires isolation from the pins when they are not used in their analog function.

Each bit of this register controls the isolation circuitry for the corresponding GPIO signal. For information on which GPIO pins can be used for ADC functions, refer to .

GPIOAMSEL is shown in [Figure 17-25](#) and described in [Table 17-31](#).

Return to [Summary Table](#).

Figure 17-25. GPIOAMSEL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								GPIOAMSEL							
R-0x0								R/W-0x0							

Table 17-31. GPIOAMSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	GPIOAMSEL	R/W	0x0	<p>GPIO Analog Mode Select.</p> <p>NOTE: This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.</p> <p>The reset state of this register is 0 for all signals.</p> <p>0x0 = The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.</p> <p>0x1 = The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.</p>

17.5.22 GPIOCTL Register (Offset = 0x52C) [reset = X]

GPIO Port Control (GPIOCTL)

The GPIOCTL register is used in conjunction with the GPIOAFSEL register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the GPIOAFSEL register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the GPIOAFSEL register, the corresponding GPIO signal is controlled by an associated peripheral. The GPIOCTL register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to . The reset value for this register is 0x00000000 for GPIO ports that are not listed in the table below.

NOTE: If a particular input signal to a peripheral is assigned to two different GPIO port pins, the signal is assigned to the port with the lowest letter and the assignment to the higher letter port is ignored. If a particular output signal from a peripheral is assigned to two different GPIO port pins, the signal will output to both pins. Assigning an output signal from a peripheral to two different GPIO pins is not recommended.

The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and high-impedance by default (GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0, and GPIOCTL = 0). Special consideration pins may be programmed to a nonGPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset (POR) returns these GPIO to their original special consideration state.

Table 17-32. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOCTL	GPIOCR
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0
PE[7]	GPIO ⁽¹⁾	0	0	0	0	0x0	0

⁽¹⁾ This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the GPIOLOCK register and uncommitting it by setting the GPIOCR register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see [Section 17.3.4](#).

NOTE: If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

GPIOCTL is shown in [Figure 17-26](#) and described in [Table 17-33](#).

[Return to Summary Table.](#)

Figure 17-26. GPIOCTL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMC7				PMC6				PMC5				PMC4				PMC3				PMC2				PMC1				PMC0			
R/W-X				R/W-X				R/W-X				R/W-X				R/W-X				R/W-X				R/W-X				R/W-X			

Table 17-33. GPIOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	PMC7	R/W	X	Port Mux Control 7. This field controls the configuration for GPIO pin 7.
27-24	PMC6	R/W	X	Port Mux Control 6. This field controls the configuration for GPIO pin 6.
23-20	PMC5	R/W	X	Port Mux Control 5. This field controls the configuration for GPIO pin 5.
19-16	PMC4	R/W	X	Port Mux Control 4. This field controls the configuration for GPIO pin 4.
15-12	PMC3	R/W	X	Port Mux Control 3. This field controls the configuration for GPIO pin 3.
11-8	PMC2	R/W	X	Port Mux Control 2. This field controls the configuration for GPIO pin 2.
7-4	PMC1	R/W	X	Port Mux Control 1. This field controls the configuration for GPIO pin 1.
3-0	PMC0	R/W	X	Port Mux Control 0. This field controls the configuration for GPIO pin 0.

17.5.23 GPIOADCCTL Register (Offset = 0x530) [reset = 0x0]

GPIO ADC Control (GPIOADCCTL)

This register is used to configure a GPIO pin as a source for the ADC trigger.

Note that if the Port B GPIOADCCTL register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

GPIOADCCTL is shown in [Figure 17-27](#) and described in [Table 17-34](#).

Return to [Summary Table](#).

Figure 17-27. GPIOADCCTL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																ADCEN															
R-0x0																R/W-0x0															

Table 17-34. GPIOADCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	ADCEN	R/W	0x0	ADC Trigger Enable. 0x0 = The corresponding pin is not used to trigger the ADC. 0x1 = The corresponding pin is used to trigger the ADC.

17.5.24 GPIODMACTL Register (Offset = 0x534) [reset = 0x0]

GPIO DMA Control (GPIODMACTL)

This register is used to configure a GPIO pin as a source for the μ DMA trigger.

GPIODMACTL is shown in [Figure 17-28](#) and described in [Table 17-35](#).

Return to [Summary Table](#).

Figure 17-28. GPIODMACTL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DMAEN							
R-0x0																								R/W-0x0							

Table 17-35. GPIODMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DMAEN	R/W	0x0	μ DMA Trigger Enable. 0x0 = The corresponding pin is not used to trigger the μ DMA. 0x1 = The corresponding pin is used to trigger the μ DMA.

17.5.25 GPIOSI Register (Offset = 0x538) [reset = 0x0]

GPIO Select Interrupt (GPIOSI)

This register is used to enable individual interrupts for each pin.

NOTE: This register is only available on Port P and Port Q.

GPIOSI is shown in [Figure 17-29](#) and described in [Table 17-36](#).

Return to [Summary Table](#).

Figure 17-29. GPIOSI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															SUM
R-0x0															R/W-0x0

Table 17-36. GPIOSI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	SUM	R/W	0x0	<p>Summary Interrupt.</p> <p>0x0 = All port pin interrupts are ORed together to produce a summary interrupt. The ORed summary interrupt occurs on bit 0 of the GPIORIS register. For summary interrupt mode, software should set the GPIOIM register to 0xFF and mask the port pin interrupts 1 through 7 in the Interrupt Clear Enable (DISn) register. When servicing this interrupt, write a 1 to the corresponding bit in the UNPENDn register to clear the pending interrupt in the NVIC and clear the GPIORIS register pin interrupt bits by setting the IC field of the GPIOICR register to 0xFF.</p> <p>0x1 = Each pin has its own interrupt vector.</p>

17.5.26 GPIODR12R Register (Offset = 0x53C) [reset = 0x0]

GPIO 12-mA Drive Select (GPIODR12R)

The GPIODR12R register is the 12-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. See [Table 17-1](#) for information on how to configure the drive strength. Note that changes in the GPIODR2R, GPIODR4R, or GPIODR8R registers to configure 12 mA are effective on the next clock cycle.

NOTE: This register has no effect on port pins PL6 and PL7 or PM[7:4].

GPIODR12R is shown in [Figure 17-30](#) and described in [Table 17-37](#).

Return to [Summary Table](#).

Figure 17-30. GPIODR12R Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DRV12							
R-0x0																								R/W-0x0							

Table 17-37. GPIODR12R Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DRV12	R/W	0x0	Output Pad 12-mA Drive Enable. 0x0 = The drive for the corresponding GPIO pin is controlled by the GPIODR2R, GPIODR4R, or the GPIODR8R register. 0x1 = The corresponding GPIO pin has 12-mA drive. This encoding is only valid if the GPIOPP EDE bit is set and the appropriate GPIOPC EDM bit field is programmed to 0x3.

17.5.27 GPIOWAKEPEN Register (Offset = 0x540) [reset = 0x0]

GPIO Wake Pin Enable (GPIOWAKEPEN)

This register is used to configure K[7:4] as a wake enable source for the hibernation module. The wake level must be programmed in the GPIOWAKELVL register at offset 0x544. In order for this register configuration to become implemented, the WUUNLK bit needs to be set in the HIBIO register at offset 0x02C in the hibernation module.

NOTE: This register is only available on Port K.

GPIOWAKEPEN is shown in [Figure 17-31](#) and described in [Table 17-38](#).

Return to [Summary Table](#).

Figure 17-31. GPIOWAKEPEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WAKEP7	WAKEP6	WAKEP5	WAKEP4	RESERVED			
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0			

Table 17-38. GPIOWAKEPEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	WAKEP7	R/W	0x0	K[7] Wake Enable. 0x0 = Wake-on level is not enabled. 0x1 = Wake-on level is enabled.
6	WAKEP6	R/W	0x0	K[6] Wake Enable. 0x0 = Wake-on level is not enabled. 0x1 = Wake-on level is enabled.
5	WAKEP5	R/W	0x0	K[5] Wake Enable. 0x0 = Wake-on level is not enabled. 0x1 = Wake-on level is enabled.
4	WAKEP4	R/W	0x0	K[4] Wake Enable. 0x0 = Wake-on level is not enabled. 0x1 = Wake-on level is enabled.
3-0	RESERVED	R	0x0	

17.5.28 GPIOWAKELVL Register (Offset = 0x544) [reset = 0x0]

GPIO Wake Level (GPIOWAKELVL)

This register is used to configure the wake level for K[7:4] in the hibernation module. The wake source must be enabled in the GPIOWAKEPEN register at offset 0x540. In order for this register configuration to become implemented, the WUUNLK bit needs to be set in the HIBIO register at offset 0x02C in the hibernation module.

NOTE: This register is only available on Port K.

GPIOWAKELVL is shown in [Figure 17-32](#) and described in [Table 17-39](#).

Return to [Summary Table](#).

Figure 17-32. GPIOWAKELVL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
WAKELVL7	WAKELVL6	WAKELVL5	WAKELVL4	RESERVED			
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0			

Table 17-39. GPIOWAKELVL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	WAKELVL7	R/W	0x0	K[7] Wake Level. 0x0 = Wake level low 0x1 = Wake level high
6	WAKELVL6	R/W	0x0	K[6] Wake Level. 0x0 = Wake level low 0x1 = Wake level high
5	WAKELVL5	R/W	0x0	K[5] Wake Level. 0x0 = Wake level low 0x1 = Wake level high
4	WAKELVL4	R/W	0x0	K[4] Wake Level. 0x0 = Wake level low 0x1 = Wake level high
3-0	RESERVED	R	0x0	

17.5.29 GPIOWAKESTAT Register (Offset = 0x548) [reset = 0x0]

GPIO Wake Status (GPIOWAKESTAT)

This register indicates the GPIO wake event status. If a register bit has been set for K[7:4], a wake event signal has been sent to the Hibernate module.

NOTE: This register is only available on Port K.

GPIOWAKESTAT is shown in [Figure 17-33](#) and described in [Table 17-40](#).

Return to [Summary Table](#).

Figure 17-33. GPIOWAKESTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
STAT7	STAT6	STAT5	STAT4	RESERVED			
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0			

Table 17-40. GPIOWAKESTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	STAT7	R	0x0	K[7] Wake Status. This is for future use. 0x0 = Pin is not wake-up source 0x1 = Pin wake event asserted to hibernate module
6	STAT6	R	0x0	K[6] Wake Status. This is for future use. 0x0 = Pin is not wake-up source 0x1 = Pin wake event asserted to hibernate module
5	STAT5	R	0x0	K[5] Wake Status. This is for future use. 0x0 = Pin is not wake-up source 0x1 = Pin wake event asserted to hibernate module
4	STAT4	R	0x0	K[4] Wake Status. 0x0 = Pin is not wake-up source 0x1 = Pin wake event asserted to hibernate module
3-0	RESERVED	R	0x0	

17.5.30 GPIOPP Register (Offset = 0xFC0) [reset = 0x1]

GPIO Peripheral Property (GPIOPP)

The GPIOPP register provides information regarding the GPIO properties.

GPIOPP is shown in [Figure 17-34](#) and described in [Table 17-41](#).

Return to [Summary Table](#).

Figure 17-34. GPIOPP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															EDE
R-0x0															R-0x1

Table 17-41. GPIOPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	EDE	R	0x1	<p>Extended Drive Enable.</p> <p>This bit specifies whether the extended drive capabilities are provided.</p> <p>Extended drive is configured by the EDM bits in the GPIOPC register.</p> <p>0x0 = No Extended Drive Capability provided.</p> <p>0x1 = Extended Drive Capability provided.</p>

17.5.31 GPIOPC Register (Offset = 0xFC4) [reset = 0x0]

GPIO Peripheral Configuration (GPIOPC)

This GPIOPC register controls the extended drive modes of the GPIO and must be configured before the GPIODRnR registers in order for extended drive mode to take effect. When the EDE bit in GPIOPP register is set and the EDMn bit field is nonzero, the GPIODRnR registers do not drive their default value, but instead output an incremental drive strength, which has an additive effect. This allows for more drive strength possibilities. When the EDE bit is set and the EDMn bit field is nonzero, the 2 mA driver is always enabled. Any bits enabled in the GPIODR4R register will add an additional 2 mA; any bits set in the GPIODR8R add an extra 4 mA of drive. The GPIODR12R register is only valid when the EDMn value is 0x3. For this encoding, setting a bit in the GPIODR12R register adds 4 mA of drive to the already existing 8 mA, for a 12 mA drive strength. [Table 17-1](#) shows the drive capability options. If EDMn is 0x00, then the GPIODR2R, GPIODR4R, and GPIODR8R function as stated in their default register description.

Table 17-42. GPIO Drive Strength Options

EDE (GPIOPP)	EDMn (GPIOPC)	GPIODR12R (+4 mA)	GPIODR8R (+4 mA)	GPIODR4R (+2 mA)	GPIODR2R (2 mA)	Drive (mA)
X	0x0	N/A	0	0	1	2
			0	1	0	4
			1	0	0	8
1	0x1	N/A	0	0	N/A	2
			0	1	N/A	4
			1	0	N/A	6
			1	1	N/A	8
1	0x3	0	0	0	N/A	2
		0	0	1	N/A	4
		0	1	0	N/A	6
		0	1	1	N/A	8
		1	1	0	N/A	10
		1	1	1	N/A	12
		1	0	N/A	N/A	N/A
1	0x2	N/A	N/A	N/A	N/A	N/A

GPIOPC is shown in [Figure 17-35](#) and described in [Table 17-43](#).

Return to [Summary Table](#).

Figure 17-35. GPIOPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDM7		EDM6		EDM5		EDM4		EDM3		EDM2		EDM1		EDM0	
R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	

Table 17-43. GPIOPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-14	EDM7	R/W	0x0	Extended Drive Mode Bit 7. Same encoding as EDM0, but applies to bit 7 of GPIO port.
13-12	EDM6	R/W	0x0	Extended Drive Mode Bit 6. Same encoding as EDM0, but applies to bit 6 of GPIO port.
11-10	EDM5	R/W	0x0	Extended Drive Mode Bit 5. Same encoding as EDM0, but applies to bit 5 of GPIO port.
9-8	EDM4	R/W	0x0	Extended Drive Mode Bit 4. Same encoding as EDM0, but applies to bit 4 of GPIO port.
7-6	EDM3	R/W	0x0	Extended Drive Mode Bit 3. Same encoding as EDM0, but applies to bit 3 of GPIO port.
5-4	EDM2	R/W	0x0	Extended Drive Mode Bit 2. Same encoding as EDM0, but applies to bit 2 of GPIO port.
3-2	EDM1	R/W	0x0	Extended Drive Mode Bit 1. Same encoding as EDM0, but applies to bit 1 of GPIO port.
1-0	EDM0	R/W	0x0	Extended Drive Mode Bit 0. This field controls extended drive modes of bit 0 of the GPIO port. Note that depending on the encoding used the GPIO drive strength control registers may change their decoding. Moreover, the write one, clear other register behavior may be disabled. 0x0 = Drive values of 2, 4 and 8 mA are maintained. GPIO n Drive Select (GPIODRnR) registers function as normal. 0x1 = An additional 6 mA option is provided. Write one, clear other behavior of GPIODDRnR registers is disabled. A 2 mA driver is always enabled; setting the corresponding GPIODR4R register bit adds 2 mA and setting the corresponding GPIODR8R register bit adds an additional 4 mA. 0x2 = reserved 0x3 = Additional drive strength options of 6, 10, and 12 mA are provided. The write one, clear other behavior of GPIODDRnR registers is disabled. A 2 mA driver is always enabled; setting the corresponding GPIODR4R register bit adds 2 mA and setting the corresponding GPIODR8R of GPIODR12R register bit adds an additional 4 mA.

17.5.32 GPIOPeriphID4 Register (Offset = 0xFD0) [reset = 0x0]

GPIO Peripheral Identification 4 (GPIOPeriphID4)

The GPIOPeriphID4, GPIOPeriphID5, GPIOPeriphID6, and GPIOPeriphID7 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID4 is shown in [Figure 17-36](#) and described in [Table 17-44](#).

Return to [Summary Table](#).

Figure 17-36. GPIOPeriphID4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID4															
R-0x0																R-0x0															

Table 17-44. GPIOPeriphID4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID4	R	0x0	GPIO Peripheral ID Register [7:0]

17.5.33 GPIOPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]

GPIO Peripheral Identification 5 (GPIOPeriphID5)

The GPIOPeriphID4, GPIOPeriphID5, GPIOPeriphID6, and GPIOPeriphID7 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID5 is shown in [Figure 17-37](#) and described in [Table 17-45](#).

Return to [Summary Table](#).

Figure 17-37. GPIOPeriphID5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID5															
R-0x0																R-0x0															

Table 17-45. GPIOPeriphID5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID5	R	0x0	GPIO Peripheral ID Register [15:8]

17.5.34 GPIOPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]

GPIO Peripheral Identification 6 (GPIOPeriphID6)

The GPIOPeriphID4, GPIOPeriphID5, GPIOPeriphID6, and GPIOPeriphID7 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID6 is shown in [Figure 17-38](#) and described in [Table 17-46](#).

Return to [Summary Table](#).

Figure 17-38. GPIOPeriphID6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID6															
R-0x0																R-0x0															

Table 17-46. GPIOPeriphID6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID6	R	0x0	GPIO Peripheral ID Register [23:16]

17.5.35 GPIOPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]

GPIO Peripheral Identification 7 (GPIOPeriphID7)

The GPIOPeriphID4, GPIOPeriphID5, GPIOPeriphID6, and GPIOPeriphID7 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID7 is shown in [Figure 17-39](#) and described in [Table 17-47](#).

Return to [Summary Table](#).

Figure 17-39. GPIOPeriphID7 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID7															
R-0x0																R-0x0															

Table 17-47. GPIOPeriphID7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID7	R	0x0	GPIO Peripheral ID Register [31:24]

17.5.36 GPIOPeriphID0 Register (Offset = 0xFE0) [reset = 0x61]

GPIO Peripheral Identification 0 (GPIOPeriphID0)

The GPIOPeriphID0, GPIOPeriphID1, GPIOPeriphID2, and GPIOPeriphID3 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID0 is shown in [Figure 17-40](#) and described in [Table 17-48](#).

Return to [Summary Table](#).

Figure 17-40. GPIOPeriphID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID0															
R-0x0																R-0x61															

Table 17-48. GPIOPeriphID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID0	R	0x61	GPIO Peripheral ID Register [7:0]. Can be used by software to identify the presence of this peripheral.

17.5.37 GPIOPeriphID1 Register (Offset = 0xFE4) [reset = 0x0]

GPIO Peripheral Identification 1 (GPIOPeriphID1)

The GPIOPeriphID0, GPIOPeriphID1, GPIOPeriphID2, and GPIOPeriphID3 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID1 is shown in [Figure 17-41](#) and described in [Table 17-49](#).

Return to [Summary Table](#).

Figure 17-41. GPIOPeriphID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID1															
R-0x0																R-0x0															

Table 17-49. GPIOPeriphID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID1	R	0x0	GPIO Peripheral ID Register [15:8]. Can be used by software to identify the presence of this peripheral.

17.5.38 GPIOPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]

GPIO Peripheral Identification 2 (GPIOPeriphID2)

The GPIOPeriphID0, GPIOPeriphID1, GPIOPeriphID2, and GPIOPeriphID3 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID2 is shown in [Figure 17-42](#) and described in [Table 17-50](#).

Return to [Summary Table](#).

Figure 17-42. GPIOPeriphID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID2															
R-0x0																R-0x18															

Table 17-50. GPIOPeriphID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID2	R	0x18	GPIO Peripheral ID Register [23:16]. Can be used by software to identify the presence of this peripheral.

17.5.39 GPIOPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]

GPIO Peripheral Identification 3 (GPIOPeriphID3)

The GPIOPeriphID0, GPIOPeriphID1, GPIOPeriphID2, and GPIOPeriphID3 registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIOPeriphID3 is shown in [Figure 17-43](#) and described in [Table 17-51](#).

Return to [Summary Table](#).

Figure 17-43. GPIOPeriphID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID3															
R-0x0																R-0x1															

Table 17-51. GPIOPeriphID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID3	R	0x1	GPIO Peripheral ID Register [31:24]. Can be used by software to identify the presence of this peripheral.

17.5.40 GPIOPCellID0 Register (Offset = 0xFF0) [reset = 0xD]

GPIO PrimeCell Identification 0 (GPIOPCellID0)

The GPIOPCellID0, GPIOPCellID1, GPIOPCellID2, and GPIOPCellID3 registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIOPCellID0 is shown in [Figure 17-44](#) and described in [Table 17-52](#).

Return to [Summary Table](#).

Figure 17-44. GPIOPCellID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CID0															
R-0x0																R-0xD															

Table 17-52. GPIOPCellID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID0	R	0xD	GPIO PrimeCell ID Register [7:0]. Provides software a standard cross-peripheral identification system.

17.5.41 GPIOPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]

GPIO PrimeCell Identification 1 (GPIOPCellID1)

The GPIOPCellID0, GPIOPCellID1, GPIOPCellID2, and GPIOPCellID3 registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIOPCellID1 is shown in [Figure 17-45](#) and described in [Table 17-53](#).

Return to [Summary Table](#).

Figure 17-45. GPIOPCellID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CID1															
R-0x0																R-0xF0															

Table 17-53. GPIOPCellID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID1	R	0xF0	GPIO PrimeCell ID Register [15:8]. Provides software a standard cross-peripheral identification system.

17.5.42 GPIOPCellID2 Register (Offset = 0xFF8) [reset = 0x5]

GPIO PrimeCell Identification 2 (GPIOPCellID2)

The GPIOPCellID0, GPIOPCellID1, GPIOPCellID2, and GPIOPCellID3 registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIOPCellID2 is shown in [Figure 17-46](#) and described in [Table 17-54](#).

Return to [Summary Table](#).

Figure 17-46. GPIOPCellID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CID2															
R-0x0																R-0x5															

Table 17-54. GPIOPCellID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID2	R	0x5	GPIO PrimeCell ID Register [23:16]. Provides software a standard cross-peripheral identification system.

17.5.43 GPIOPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]

GPIO PrimeCell Identification 3 (GPIOPCellID3)

The GPIOPCellID0, GPIOPCellID1, GPIOPCellID2, and GPIOPCellID3 registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIOPCellID3 is shown in [Figure 17-47](#) and described in [Table 17-55](#).

Return to [Summary Table](#).

Figure 17-47. GPIOPCellID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CID3															
R-0x0																R-0xB1															

Table 17-55. GPIOPCellID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID3	R	0xB1	GPIO PrimeCell ID Register [31:24]. Provides software a standard cross-peripheral identification system.

General-Purpose Timers

Programmable timers can be used to count or time external events that drive the timer input pins. The MSP432E4 general-purpose timer module (GPTM) contains 16- or 32-bit GPTM blocks. Each 16- or 32-bit GPTM block provides two 16-bit timers or counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or concatenated to operate as one 32-bit timer or one 32-bit real-time clock (RTC). Timers can also be used to trigger μ DMA transfers.

Topic	Page
18.1 Introduction	1254
18.2 Block Diagram	1254
18.3 Functional Description	1256
18.4 Initialization and Configuration	1268
18.5 GPTM Registers	1271

18.1 Introduction

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The GPTM is one timing resource available on the MSP432E4 microcontrollers. Other timer resources include the System Timer (SysTick) (see [Section 2.2.1](#)), and the PWM timer in the PWM module (see [Section 21.3.2](#)).

The GPTM contains eight 16- or 32-bit GPTM blocks with the following functional options:

- Operating modes:
 - 16- or 32-bit programmable one-shot timer
 - 16- or 32-bit programmable periodic timer
 - 16-bit general-purpose timer with an 8-bit prescaler
 - 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
 - 16-bit input-edge count- or time-capture modes with an 8-bit prescaler
 - 16-bit PWM mode with an 8-bit prescaler and software-programmable output inversion of the PWM signal
- Count up or down
- Sixteen 16- or 32-bit Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- Timer synchronization allows selected timers to start counting on the same clock cycle
- ADC event trigger
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Dedicated channel for each timer
 - Burst request generated on timer interrupt

18.2 Block Diagram

In [Figure 18-1](#), the specific CCP pins available depend on the MSP432E4 device. See [Table 18-1](#) for the available CCP pins and their timer assignments.

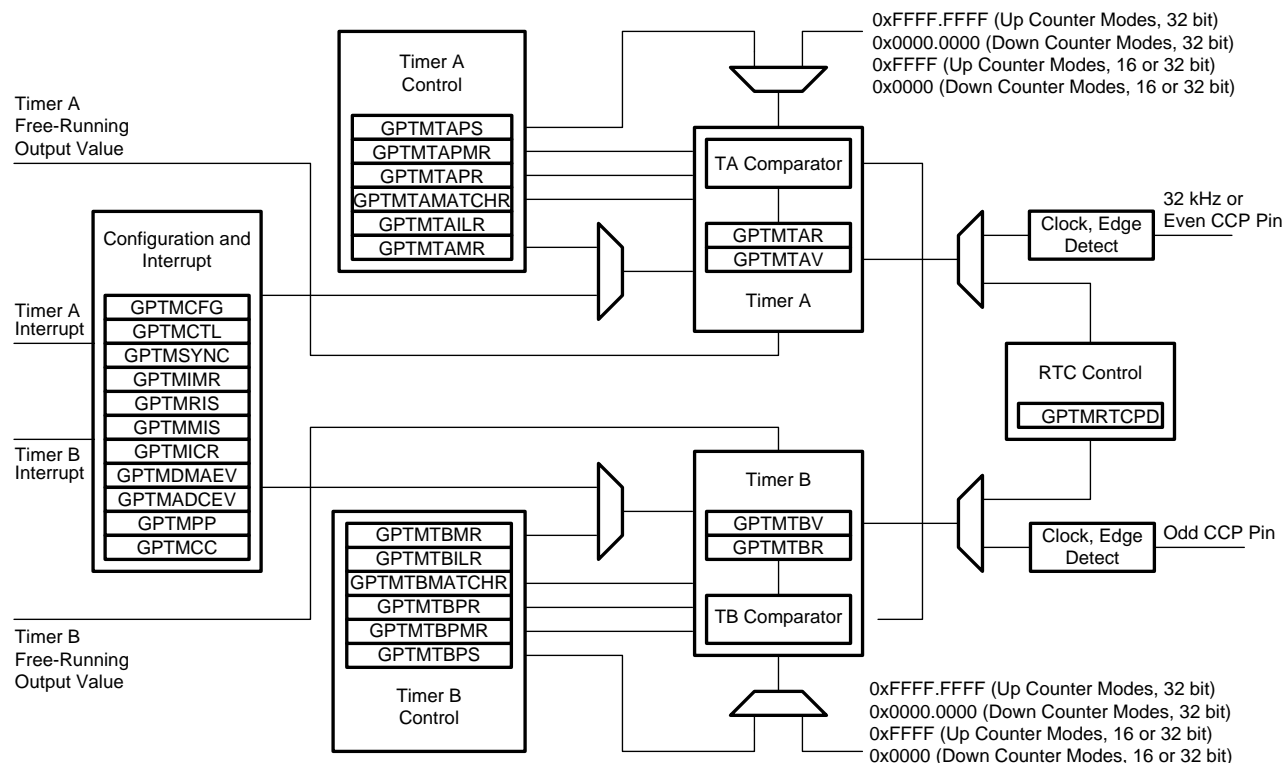


Figure 18-1. GPTM Module Block Diagram

Table 18-1. Available CCP Pins

Timer	Up or Down Counter	Even CCP Pin	Odd CCP Pin
16- or 32-Bit Timer 0	Timer A	T0CCP0	-
	Timer B	-	T0CCP1
16- or 32-Bit Timer 1	Timer A	T1CCP0	-
	Timer B	-	T1CCP1
16- or 32-Bit Timer 2	Timer A	T2CCP0	-
	Timer B	-	T2CCP1
16- or 32-Bit Timer 3	Timer A	T3CCP0	-
	Timer B	-	T3CCP1
16- or 32-Bit Timer 4	Timer A	T4CCP0	-
	Timer B	-	T4CCP1
16- or 32-Bit Timer 5	Timer A	T5CCP0	-
	Timer B	-	T5CCP1
16- or 32-Bit Timer 6	Timer A	T6CCP0	-
	Timer B	-	T6CCP1
16- or 32-Bit Timer 7	Timer A	T7CCP0	-
	Timer B	-	T7CCP1

18.3 Functional Description

The main components of each GPTM block are two, free-running, up or down counters (Timer A and Timer B), two prescaler registers, two match registers, two prescaler match registers, two shadow registers, and two load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface. Timer A and Timer B can be used individually, in which case they have a 16-bit counting range for the 16- or 32-bit GPTM blocks. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range for the 16- or 32-bit GPTM blocks. The prescaler can be used only when the timers are used individually.

Table 18-2 lists the available modes for each GPTM block. When counting down in one-shot or periodic modes, the prescaler acts as a true prescaler and contains the least-significant bits of the count. When counting up in one-shot or periodic modes, the prescaler acts as a timer extension and holds the most-significant bits of the count. In input edge count, input edge time and PWM mode, the prescaler always acts as a timer extension, regardless of the count direction.

Table 18-2. General-Purpose Timer Capabilities

Mode	Timer Use	Count Direction	Counter Size	Prescaler Size ⁽¹⁾
One-shot	Individual	Up or down	16 bit	8 bit
	Concatenated	Up or down	32 bit	-
Periodic	Individual	Up or down	16 bit	8 bit
	Concatenated	Up or down	32 bit	-
RTC	Concatenated	Up	32 bit	-
Edge count	Individual	Up or down	16 bit	8 bit
Edge time	Individual	Up or down	16 bit	8 bit
PWM	Individual	Down	16 bit	8 bit

⁽¹⁾ The prescaler is available only when the timers are used individually.

Software configures the GPTM using the GPTM Configuration (GPTMCFG) register (see [Section 18.5.1](#)), the GPTM Timer A Mode (GPTMTAMR) register (see [Section 18.5.2](#)), and the GPTM Timer B Mode (GPTMTBMR) register (see [Section 18.5.3](#)).

When configured in one of the concatenated modes, Timer A and Timer B can only operate in one mode. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

18.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to all 1s, along with their corresponding registers:

- Load registers:
 - GPTM Timer A Interval Load (GPTMTAILR) register (see [Section 18.5.10](#))
 - GPTM Timer B Interval Load (GPTMTBILR) register (see [Section 18.5.11](#))
- Shadow registers:
 - GPTM Timer A Value (GPTMTAV) register (see [Section 18.5.20](#))
 - GPTM Timer B Value (GPTMTBV) register (see [Section 18.5.21](#))

The following prescale counters are initialized to all 0s:

- GPTM Timer A Prescale (GPTMTAPR) register (see [Section 18.5.14](#))
- GPTM Timer B Prescale (GPTMTBPR) register (see [Section 18.5.15](#))
- GPTM Timer A Prescale Snapshot (GPTMTAPS) register (see [Section 18.5.23](#))
- GPTM Timer B Prescale Snapshot (GPTMTBPS) register (see [Section 18.5.24](#))

18.3.2 Timer Clock Source

The general purpose timer has the capability of being clocked by either the system clock or an alternate clock source. By setting the ALTCLK bit in the GPTM Clock Configuration (GPTMCC) register, offset 0xFC8, software can select an alternate clock source as programmed in the Alternate Clock Configuration (ALTCLKCFG) register, offset 0x138 in the System Control Module. The alternate clock source options available are PIOSC, RTCOSC, and LFIOOSC. See [Chapter 4](#) for additional information.

NOTE: When the ALTCLK bit is set in the GPTMCC register to enable using the alternate clock source, the synchronization imposes restrictions on the starting count value (down count), terminal value (up count) and the match value. This restriction applies to all modes of operation. Each event must be spaced by 4 Timer (ALTCLK) clock periods + 2 system clock periods. If some events do not meet this requirement, then it is possible that the timer block may need to be reset for correct functionality to be restored.

Example: $ALTCLK = T_{PIOSC} = 62.5 \text{ ns}$ (16 MHz trimmed)

$T_{hclk} = 1 \text{ } \mu\text{s}$ (1 MHz)

$4 \times 62.5 \text{ ns} + 2 \times 1 \text{ } \mu\text{s} = 2.25 \text{ } \mu\text{s}$ $2.25 \text{ } \mu\text{s} / 62.5 \text{ ns} = 36$ or 0x23

The minimum values for the periodic or one-shot with a match interrupt enabled are:

GPTMTAMATCHR = 0x23 and GPTMTAILR = 0x46

18.3.3 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use Timer B control and status bits. The GPTM is placed into individual/split mode by writing a value of 0x4 to the GPTM Configuration (GPTMCFG) register (see [Section 18.5.1](#)). In the following sections, the variable *n* is used in bit field and register names to imply either a Timer A function or a Timer B function. Throughout this section, the time-out event in down-count mode is 0x0 and in up-count mode is the value in the GPTM Timer *n* Interval Load (GPTMTnILR) and the optional GPTM Timer *n* Prescale (GPTMTnPR) registers, with the exception of RTC mode.

18.3.3.1 One-Shot and Periodic Timer Mode

The selection of one-shot or periodic mode is determined by the value written to the TnMR field of the GPTM Timer *n* Mode (GPTMTnMR) register (see [Section 18.5.2](#)). The timer is configured to count up or down using the TnCDIR bit in the GPTMTnMR register.

When software sets the TnEN bit in the GPTM Control (GPTMCTL) register (see [Section 18.5.4](#)), the timer begins counting up from 0x0 or down from its preloaded value. Alternatively, if the TnWOT bit is set in the GPTMTnMR register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see [Section 18.3.4](#)). [Table 18-3](#) lists the values that are loaded into the timer registers when the timer is enabled.

Table 18-3. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes

Register	Count Down Mode	Count Up Mode
GPTMTnR	GPTMTnILR	0x0
GPTMTnV	GPTMTnILR in concatenated mode; GPTMTnPR in combination with GPTMTnILR in individual mode	0x0
GPTMTnPS	GPTMTnPR in individual mode; not available in concatenated mode	0x0 in individual mode; not available in concatenated mode

When the timer is counting down and it reaches the time-out event (0x0), the timer reloads its start value from the GPTMTnILR and the GPTMTnPR registers on the next cycle. When the timer is counting up and it reaches the time-out event (the value in the GPTMTnILR and the optional GPTMTnPR registers), the timer reloads with 0x0. If configured to be a one-shot timer, the timer stops counting and clears the TnEN bit in the GPTMCTL register. If configured as a periodic timer, the timer starts counting again on the next cycle.

In periodic, snap-shot mode (TnMR field is 0x2 and the TnSNAPS bit is set in the GPTMTnMR register), the value of the timer at the time-out event is loaded into the GPTMTnR register and the value of the prescaler is loaded into the GPTMTnPS register. The free-running counter value is shown in the GPTMTnV register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry by examining the snapshot values and the current value of the free-running timer. Snapshot mode is not available when the timer is configured in one-shot mode.

In addition to reloading the count value, the GPTM can generate interrupts, CCP outputs and triggers when it reaches the time-out event. The GPTM sets the TnTORIS bit in the GPTM Raw Interrupt Status (GPTMRIS) register (see [Section 18.5.7](#)), and holds it until it is cleared by writing the GPTM Interrupt Clear (GPTMICR) register (see [Section 18.5.9](#)). If the time-out interrupt is enabled in the GPTM Interrupt Mask (GPTMIMR) register (see [Section 18.5.6](#)), the GPTM also sets the TnTOMIS bit in the GPTM Masked Interrupt Status (GPTMMIS) register (see [Section 18.5.8](#)). The time-out interrupt can be disabled entirely by setting the TnCINTD bit in the GPTM Timer n Mode (GPTMTnMR) register. In this case, the TnTORIS bit does not even set in the GPTMRIS register.

By setting the TnMIE bit in the GPTMTnMR register, an interrupt condition can also be generated when the Timer value equals the value loaded into the GPTM Timer n Match (GPTMTnMATCHR) and GPTM Timer n Prescale Match (GPTMTnPMR) registers. This interrupt has the same status, masking, and clearing functions as the time-out interrupt, but uses the match interrupt bits instead (for example, the raw interrupt status is monitored via TnMRIS bit in the GPTM Raw Interrupt Status (GPTMRIS) register). The interrupt status bits are not updated by the hardware unless the TnMIE bit in the GPTMTnMR register is set, which is different than the behavior for the time-out interrupt. The ADC trigger is enabled by setting the TnOTE bit in GPTMCTL and the event that activates the ADC is configured in the GPTM ADC Event (GPTMADCEV) register. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel as well as the type of trigger enable in the GPTM DMA Event (GPTMDMAEV) register. See [Section 8.3.4](#).

The TCACT field of the GPTM Timer n Mode (GPTMTnMR) register can be configured to clear, set or toggle an output on a time-out event.

If software updates the GPTMTnILR or the GPTMTnPR register while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting from the new value if the TnILD bit in the GPTMTnMR register is clear. If the TnILD bit is set, the counter loads the new value after the next time-out. If software updates the GPTMTnILR or the GPTMTnPR register while the counter is counting up, the time-out event is changed on the next cycle to the new value. If software updates the GPTM Timer n Value (GPTMTnV) register while the counter is counting up or down, the counter loads the new value on the next clock cycle and continues counting from the new value. If software updates the GPTMTnMATCHR or the GPTMTnPMR registers, the new values are reflected on the next clock cycle if the TnMRSU bit in the GPTMTnMR register is clear. If the TnMRSU bit is set, the new value will not take effect until the next time-out.

If the TnSTALL bit in the GPTMCTL register is set and the RTCEN bit is not set in the GPTMCTL register, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution. If the RTCEN bit is set, it prevents the TnSTALL bit from freezing the count when the processor is halted by the debugger.

[Table 18-4](#) lists a variety of configurations for a 16-bit, free-running timer while using the prescaler. All values assume a 120-MHz clock with $T_c = 8.33$ ns (clock period). The prescaler can only be used when a 16- or 32-bit timer is configured in 16-bit mode.

Table 18-4. 16-Bit Timer With Prescaler Configurations

Prescale (8-Bit Value)	No. of T_c ⁽¹⁾	Maximum Time	Unit
00000000	1	0.548258	ms
00000001	2	1.096517	ms
00000010	3	1.644775	ms
11111101	254	139.2576	ms
11111110	255	139.8059	ms
11111111	256	140.3541	ms

⁽¹⁾ T_c is the clock period.

18.3.3.1.1 Timer Compare Action Mode

The timer compare mode is an extension to the existing one-shot and periodic modes of the GPTM. This mode can be used when an application requires a pin change state at some time in the future, regardless of the processor state. The compare mode does not operate when the PWM mode is active and is mutually exclusive to the PWM mode. The compare mode is enabled when the TAMR field is set to 0x1 or 0x2 (one-shot or periodic), the TnAMS bit is 0 (capture or compare mode) and the TCACT field is nonzero in the GPTM Timer n Mode (GPTMTnMR) register. Depending on the TCACT encoding, the timer can perform a set, clear or toggle on the corresponding CCPn pin when a timer match occurs. In 16-bit mode, the corresponding CCP pin can have an action applied, but when operating in 32-bit mode, the action can only be applied to the even CCP pin.

The TCACT field can be changed while the GPTM is enabled to generate different combinations of actions. For example, during a periodic event, encodings TCACT = 0x6 or 0x7 can be used to force the initial state of the CCPn pin before the first interrupt and following that, TCACT = 0x2 and TCACT = 0x3 can be used (alternately) to change the sense of the pin for the subsequent toggle, while possible changing load value for the next period.

The time-out interrupts used for one-shot and periodic modes are used in the compare action modes. Thus, the TnTORIS bits in the GPTMRIS register are triggered if the appropriate mask bits are set in the GPTMIM register.

18.3.3.2 Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as an up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x1. All subsequent load values must be written to the GPTM Timer n Interval Load (GPTMTnILR) registers (see [Section 18.5.10](#)). If the GPTMTnILR register is loaded with a new value, the counter begins counting at that value and rolls over at the fixed value of 0xFFFFFFFF. [Table 18-5](#) lists the values that are loaded into the timer registers when the timer is enabled.

Table 18-5. Counter Values When the Timer is Enabled in RTC Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	Not available	0x1
GPTMTnV	Not available	0x1
GPTMTnPS	Not available	Not available

The input clock on a CCP 0 input must be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1-Hz rate and passed along to the input of the counter.

When software writes the TAEN bit in the GPTMCTL register, the counter starts counting up from its preloaded value of 0x1. When the current count value matches the preloaded value in the GPTMTnMATCHR registers, the GPTM asserts the RTCRIS bit in GPTMRIS and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When the timer value reaches the terminal count, the timer rolls over and continues counting up from 0x0. If the RTC interrupt is enabled in GPTMIMR, the GPTM also sets the RTCMIS bit in GPTMMIS and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in GPTMICR.

In this mode, the GPTMTnR and GPTMTnV registers always have the same value.

In addition to generating interrupts, the RTC can generate a μ DMA trigger. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel as well as the type of trigger enable in the GPTM DMA Event (GPTMDMAEV) register (see [Section 8.3.4](#)).

18.3.3.3 Input Edge-Count Mode

NOTE: For rising-edge detection, the input signal must be high for at least two clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be low for at least two clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the frequency.

In Edge-Count mode, the timer is configured as a 24-bit up- or down-counter including the optional prescaler with the upper count value stored in the GPTM Timer n Prescale (GPTMTnPR) register and the lower bits in the GPTMTnR register. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the TnCMR bit of the GPTMTnMR register must be cleared. The type of edge that the timer counts is determined by the TnEVENT fields of the GPTMCTL register. During initialization in down-count mode, the GPTMTnMATCHR and GPTMTnPMR registers are configured so that the difference between the value in the GPTMTnILR and GPTMTnPR registers and the GPTMTnMATCHR and GPTMTnPMR registers equals the number of edge events that must be counted. In up-count mode, the timer counts from 0x0 to the value in the GPTMTnMATCHR and GPTMTnPMR registers. When executing an up count, that the value of GPTMTnPR and GPTMTnILR must be greater than the value of GPTMTnPMR and GPTMTnMATCHR. [Table 18-6](#) lists the values that are loaded into the timer registers when the timer is enabled.

Table 18-6. Counter Values When the Timer is Enabled in Input Edge-Count Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	GPTMTnPR in combination with GPTMTnILR	0x0
GPTMTnV	GPTMTnPR in combination with GPTMTnILR	0x0

When software writes the TnEN bit in the GPTM Control (GPTMCTL) register, the timer is enabled for event capture. Each input event on the CCP pin decrements or increments the counter by 1 until the event count matches GPTMTnMATCHR and GPTMTnPMR. When the counts match, the GPTM asserts the CnMRIS bit in the GPTM Raw Interrupt Status (GPTMRIS) register, and holds it until it is cleared by writing the GPTM Interrupt Clear (GPTMICR) register. If the capture mode match interrupt is enabled in the GPTM Interrupt Mask (GPTMIMR) register, the GPTM also sets the CnMMIS bit in the GPTM Masked Interrupt Status (GPTMMIS) register. In up-count mode, the current count of the input events is held in both the GPTMTnR and GPTMTnV registers. In down-count mode, the current count of the input events can be obtained by subtracting the GPTMTnR or GPTMTnV from the value made up of the GPTMTnPR and GPTMTnILR register combination.

In addition to generating interrupts, an ADC and/or a μ DMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in GPTMCTL and the event that activates the ADC is configured in the GPTM ADC Event (GPTMADCEV) register. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel as well as the type of trigger enable in the GPTM DMA Event (GPTMDMAEV) register (see [Section 8.3.4](#)).

After the match value is reached in down-count mode, the counter is then reloaded using the value in GPTMTnILR and GPTMTnPR registers, and stopped because the GPTM automatically clears the TnEN bit in the GPTMCTL register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software. In up-count mode, the timer is reloaded with 0x0 and continues counting.

[Figure 18-2](#) shows how Input Edge-Count mode works. In this case, the timer start value is set to GPTMTnILR = 0x000A and the match value is set to GPTMTnMATCHR = 0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

The last two edges are not counted because the timer automatically clears the TnEN bit after the current count matches the value in the GPTMTnMATCHR register.

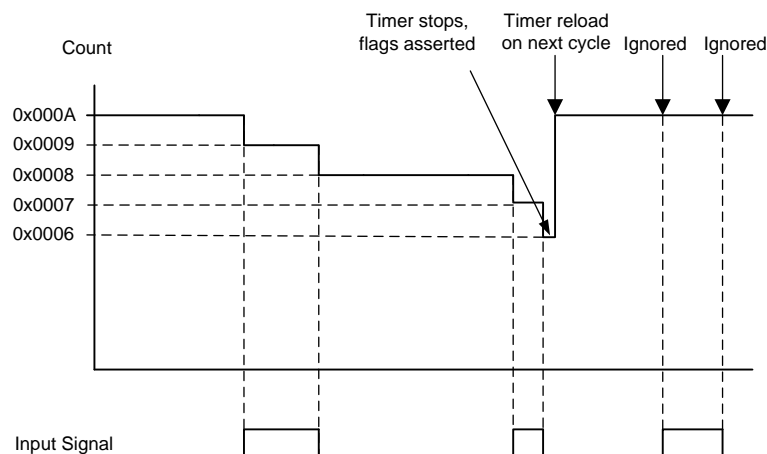


Figure 18-2. Input Edge-Count Mode Example, Counting Down

18.3.3.4 Input Edge-Time Mode

NOTE: For rising-edge detection, the input signal must be high for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Time mode, the timer is configured as a 24-bit up- or down-counter including the optional prescaler with the upper timer value stored in the GPTMTnPR register and the lower bits in the GPTMTnILR register. In this mode, the timer is initialized to the value loaded in the GPTMTnILR and GPTMTnPR registers when counting down and 0x0 when counting up. The timer is capable of capturing three types of events: rising edge, falling edge, or both. The timer is placed into Edge-Time mode by setting the TnCMR bit in the GPTMTnMR register, and the type of event that the timer captures is determined by the TnEVENT fields of the GPTMCTL register. [Table 18-7](#) lists the values that are loaded into the timer registers when the timer is enabled.

Table 18-7. Counter Values When the Timer is Enabled in Input Event-Count Mode

Register	Count Down Mode	Count Up Mode
TnR	GPTMTnILR	0x0
TnV	GPTMTnILR	0x0

When software writes the TnEN bit in the GPTMCTL register, the timer is enabled for event capture. When the selected input event is detected, the current timer counter value is captured in the GPTMTnR and GPTMTnPS register and is available to be read by the microcontroller. The GPTM then asserts the CnERIS bit in the GPTM Raw Interrupt Status (GPTMRIS) register, and holds it until it is cleared by writing the GPTM Interrupt Clear (GPTMICR) register. If the capture mode event interrupt is enabled in the GPTM Interrupt Mask (GPTMIMR) register, the GPTM also sets the CnEMIS bit in the GPTM Masked Interrupt Status (GPTMMIS) register. In this mode, the GPTMTnR and GPTMTnPS registers hold the time at which the selected input event occurred while the GPTMTnV register holds the free-running timer value. These registers can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

In addition to generating interrupts, an ADC and/or a μ DMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in GPTMCTL and the event that activates the ADC is configured in the GPTM ADC Event (GPTMADCEV) register. The μ DMA trigger is enabled by configuring the appropriate μ DMA channel as well as the type of trigger selected in the GPTM DMA Event (GPTMDMAEV) register (see [Section 8.3.4](#)).

After an event has been captured, the timer does not stop counting. It continues to count until the TnEN bit is cleared. When the timer reaches the time-out value, it is reloaded with 0x0 in up-count mode and the value from the GPTMTnILR and GPTMTnPR registers in down-count mode.

Figure 18-3 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the GPTMTnR and GPTMTnPS registers, and held there until another rising edge is detected (at which point the new count value is loaded into the GPTMTnR and GPTMTnPS registers).

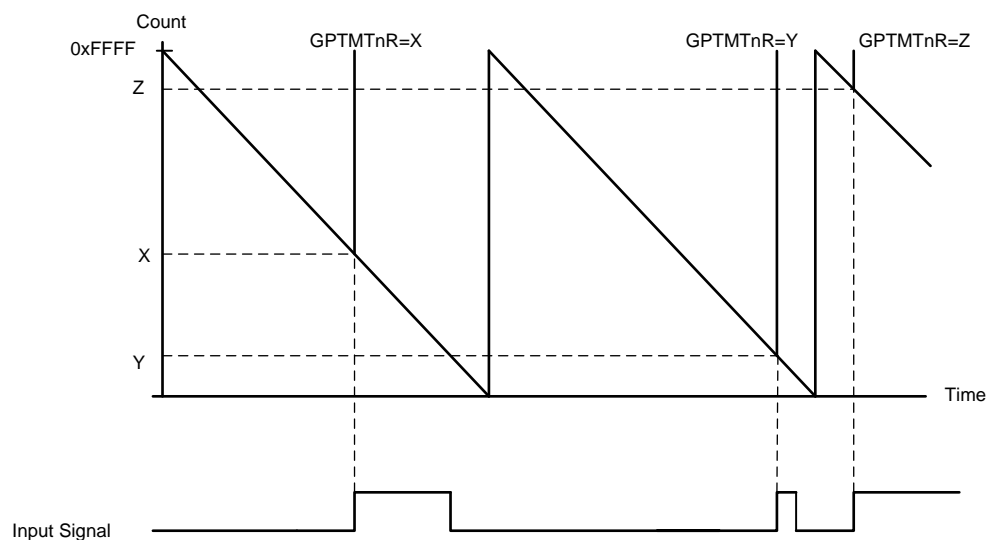


Figure 18-3. 16-Bit Input Edge-Time Mode Example

NOTE: When operating in Edge-time mode, the counter uses a modulo 2^{24} count if the prescaler is enabled, or 2^{16} if not. If there is a possibility the edge could take longer than the count, then another timer configured in periodic-timer mode can be implemented to ensure detection of the missed edge. The periodic timer should be configured in such a way that:

- The periodic timer cycles at the same rate as the edge-time timer
- The periodic timer interrupt has a higher interrupt priority than the edge-time time-out interrupt.
- If the periodic timer interrupt service routine is entered, software must check if an edge-time interrupt is pending and if it is, the value of the counter must be subtracted by 1 before being used to calculate the snapshot time of the event.

18.3.3.5 PWM Mode

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 24-bit down counter with a start value (and thus period) defined by the GPTMTnILR and GPTMTnPR registers. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch-free. PWM mode is enabled with the GPTMTnMR register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2. Table 18-8 lists the values that are loaded into the timer registers when the timer is enabled.

Table 18-8. Counter Values When the Timer is Enabled in PWM Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	GPTMTnILR	Not available
GPTMTnV	GPTMTnILR	Not available

When software writes the TnEN bit in the GPTMCTL register, the counter begins counting down until it reaches the 0x0 state. Alternatively, if the TnWOT bit is set in the GPTMTnMR register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see [Section 18.3.4](#)). On the next counter cycle in periodic mode, the counter reloads its start value from the GPTMTnILR and GPTMTnPR registers and continues counting until disabled by software clearing the TnEN bit in the GPTMCTL register. The timer is capable of generating interrupts based on three types of events: rising edge, falling edge, or both. The event is configured by the TnEVENT field of the GPTMCTL register, and the interrupt is enabled by setting the TnPWMIE bit in the GPTMTnMR register. When the event occurs, the CnERIS bit is set in the GPTM Raw Interrupt Status (GPTMRIS) register, and holds it until it is cleared by writing the GPTM Interrupt Clear (GPTMICR) register. If the capture mode event interrupt is enabled in the GPTM Interrupt Mask (GPTMIMR) register, the GPTM also sets the CnEMIS bit in the GPTM Masked Interrupt Status (GPTMMIS) register. The interrupt status bits are not updated unless the TnPWMIE bit is set.

In addition, when the TnPWMIE bit is set and a capture event occurs, the Timer automatically generates triggers to the ADC and DMA if the trigger capability is enabled by setting the TnOTE bit in the GPTMCTL register and the CnEDMAEN bit in the GPTMDMAEV register, respectively.

In this mode, the GPTMTnR and GPTMTnV registers always have the same value.

The output PWM signal asserts when the counter is at the value of the GPTMTnILR and GPTMTnPR registers (its start state), and is deasserted when the counter value equals the value in the GPTMTnMATCHR and GPTMTnPMR registers. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the GPTMCTL register.

NOTE: If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.

Figure 18-4 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and TnPWML = 0 (duty cycle would be 33% for the TnPWML = 1 configuration). For this example, the start value is GPTMTnILR = 0xC350 and the match value is GPTMTnMATCHR = 0x411A.

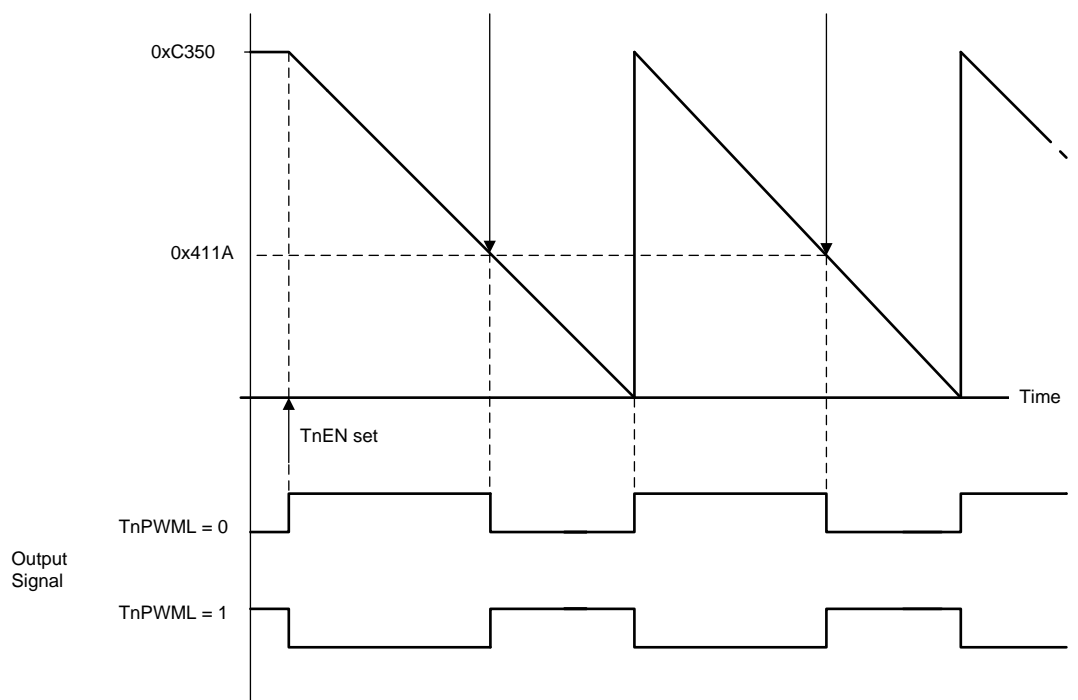


Figure 18-4. 16-Bit PWM Mode Example

When synchronizing the timers using the GPTMSYNC register, the timer must be properly configured to avoid glitches on the CCP outputs. Both the TnPLO and the TnMRSU bits must be set in the GPTMTnMR register. Figure 18-5 shows how the CCP output operates when the TnPLO and TnMRSU bits are set and the GPTMTnMATCHR value is greater than the GPTMTnILR value.

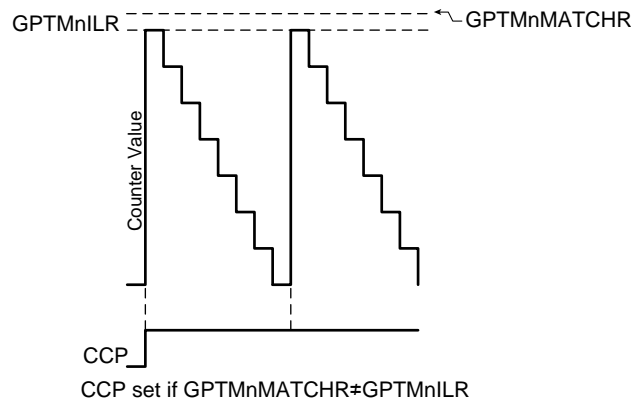


Figure 18-5. CCP Output, GPTMTnMATCHR > GPTMTnILR

Figure 18-6 shows how the CCP output operates when the PLO and MRSU bits are set and the GPTMTnMATCHR value is the same as the GPTMTnILR value. In this situation, if the PLO bit is 0, the CCP signal goes high when the GPTMTnILR value is loaded and the match would be essentially ignored.

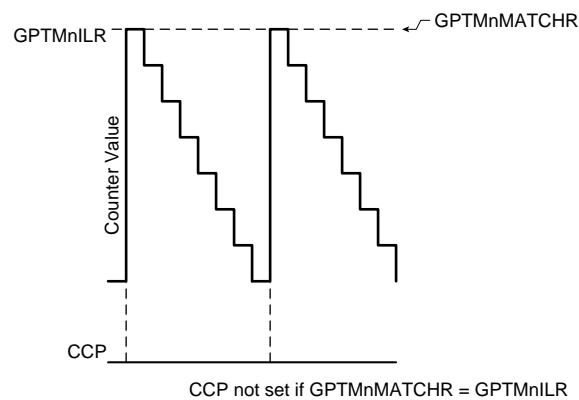


Figure 18-6. CCP Output, GPTMTnMATCHR = GPTMTnILR

Figure 18-7 shows how the CCP output operates when the PLO and MRSU bits are set and the GPTMTnILR is greater than the GPTMTnMATCHR value.

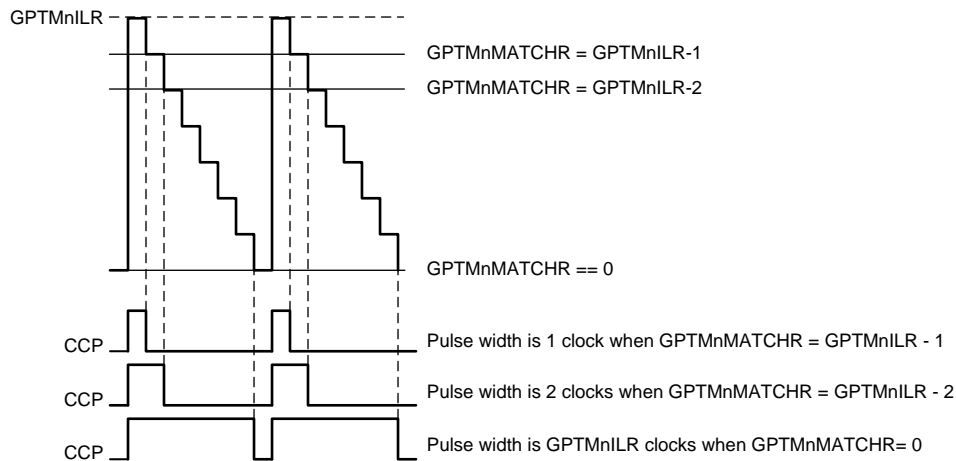


Figure 18-7. CCP Output, GPTMnILR > GPTMnMATCHR

18.3.4 Wait-for-Trigger Mode

The Wait-for-Trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate multiple timing events using the Timer triggers. Wait-for-Trigger mode is enabled by setting the TnWOT bit in the GPTMTnMR register. When the TnWOT bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so on. If Timer A is configured as a 32-bit (16- or 32-bit mode) timer (controlled by the GPTMCFG field in the GPTMCFG register), it triggers Timer A in the next module. If Timer A is configured as a 16-bit (16- or 32-bit mode) timer, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Figure 18-8 shows how the GPTMCFG bit affects the daisy chain. This function is valid for one-shot, periodic, and PWM modes.

NOTE: If the application requires cyclical daisy-chaining, the TAWOT bit in the GPTMTAMR register of Timer 0 can be set. In this case, Timer 0 waits for a trigger from the last timer module in the chain.

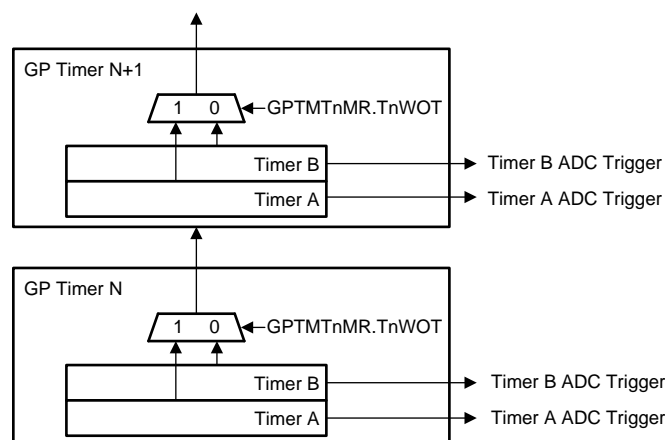


Figure 18-8. Timer Daisy Chain

18.3.5 Synchronizing GP Timer Blocks

The GPTM Synchronizer Control (GPTMSYNC) register in the GPTM0 block can be used to synchronize selected timers to begin counting at the same time. Setting a bit in the GPTMSYNC register causes the associated timer to perform the actions of a time-out event. An interrupt is not generated when the timers are synchronized. If a timer is being used in concatenated mode, only the bit for Timer A must be set in the GPTMSYNC register.

NOTE: All timers must use the same clock source for this feature to work correctly.

Table 18-9 lists the actions for the time-out event performed when the timers are synchronized in the various timer modes.

Table 18-9. Time-out Actions for GPTM Modes

Mode	Count Direction	Time-Out Action
32-bit one-shot (concatenated timers)	–	N/A
32-bit periodic (concatenated timers)	Down	Count value = ILR
	Up	Count value = 0
32-bit RTC (concatenated timers)	Up	Count value = 0
16-bit one shot (individual/split timers)	–	N/A
16-bit periodic (individual/split timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit edge count (individual/split timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit edge time (individual/split timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit PWM	Down	Count value = ILR

18.3.6 DMA Operation

The timers each have a dedicated μ DMA channel and can provide a request signal to the μ DMA controller. Pulse requests are generated by a timer via its own `dma_req` signal. A `dma_done` signal is provided from the μ DMA to each timer to indicate transfer completion and trigger a μ DMA done interrupt (DMA_nRIS) in the GPTM Raw Interrupt Status Register (GPTMRIS) register. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the μ DMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic time-out at 10 ms. Configure the μ DMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the μ DMA controller transfers 8 items, until all 256 items have been transferred. See [Chapter 8](#) for more details about programming the μ DMA controller.

A GPTM DMA Event (GPTMDMAEV) register is provided to enable the types of events that can cause a `dma_req` signal assertion by the timer module. Application software can enable a `dma_req` trigger for a match, capture or time-out event for each timer using the GPTMDMAEV register. For an individual timer, all active timer trigger events that have been enabled through the GPTMDMAEV register are ORed together to create a single `dma_req` pulse that is sent to the μ DMA. When the μ DMA transfer has completed, a `dma_done` signal is sent to the timer resulting in a DMA_nRIS bit set in the GPTMRIS register.

18.3.7 ADC Operation

The timer has the capability to trigger the ADC when the TnOTE bit is set in the GPTMCTL register at offset 0x00C. The GPTM ADC Event (GPTMADCEV) register is additionally provided so that the type of ADC trigger can be defined. For example, by setting the CBMADCEN bit in the GPTMADCEV register, a trigger pulse will be sent to the ADC whenever a Capture Match event occurs in GPTM B. Similar to the μ DMA operation, all active trigger events that have also been enabled in the GPTMADCEV register are ORed together to create an ADC trigger pulse.

18.3.8 Accessing Concatenated 16- or 32-Bit GPTM Register Values

The GPTM is placed into concatenated mode by writing 0x0 or 0x1 to the GPTMCFG bit field in the GPTM Configuration (GPTMCFG) register. In both configurations, certain 16- or 32-bit GPTM registers are concatenated to form pseudo 32-bit registers. These registers include the following:

- GPTM Timer A Interval Load (GPTMTAILR) register [15:0], see [Section 18.5.10](#)
- GPTM Timer B Interval Load (GPTMTBILR) register [15:0], see [Section 18.5.11](#)
- GPTM Timer A (GPTMTAR) register [15:0], see [Section 18.5.18](#)
- GPTM Timer B (GPTMTBR) register [15:0], see [Section 18.5.19](#)
- GPTM Timer A Value (GPTMTAV) register [15:0], see [Section 18.5.20](#)
- GPTM Timer B Value (GPTMTBV) register [15:0], see [Section 18.5.21](#)
- GPTM Timer A Match (GPTMTAMATCHR) register [15:0], see [Section 18.5.12](#)
- GPTM Timer B Match (GPTMTBMATCHR) register [15:0], see [Section 18.5.13](#)

In the 32-bit modes, the GPTM translates a 32-bit write access to GPTMTAILR into a write access to both GPTMTAILR and GPTMTBILR. The resulting word ordering for such a write operation is: GPTMTBILR[15:0]:GPTMTAILR[15:0].

Likewise, a 32-bit read access to GPTMTAR returns the value: GPTMTBR[15:0]:GPTMTAR[15:0].

A 32-bit read access to GPTMTAV returns the value: GPTMTBV[15:0]:GPTMTAV[15:0].

18.4 Initialization and Configuration

To use a GPTM, the appropriate `TIMERn` bit must be set in the `RCGCTIMER` register (see [Section 4.2.86](#)). If using any CCP pins, the clock to the appropriate GPIO module must be enabled using the `RCGCGPIO` register (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet. Configure the `PMCN` fields in the `GPIOPCTL` register to assign the CCP signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).

This section shows module initialization and configuration examples for each of the supported timer modes.

18.4.1 One-Shot and Periodic Timer Mode

The GPTM is configured for One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the `TnEN` bit in the `GPTMCTL` register is cleared) before making any changes.
2. Write the GPTM Configuration Register (`GPTMCFG`) with a value of `0x0000.0000`.
3. Configure the `TnMR` field in the GPTM Timer n Mode Register (`GPTMTnMR`) :
 1. Write a value of `0x1` for One-Shot mode.
 2. Write a value of `0x2` for Periodic mode.
4. Optionally configure the `TnSNAPS`, `TnWOT`, `TnMTE`, and `TnCDIR` bits in the `GPTMTnMR` register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down. In addition, if using CCP pins, the `TCACT` field can be programmed to configure the compare action.
 1. Load the start value into the GPTM Timer n Interval Load Register (`GPTMTnILR`).
 2. If interrupts are required, set the appropriate bits in the GPTM Interrupt Mask Register (`GPTMIMR`).
 3. Set the `TnEN` bit in the `GPTMCTL` register to enable the timer and start counting.
 4. Poll the `GPTMRIS` register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the GPTM Interrupt Clear Register (`GPTMICR`).

If the `TnMIE` bit in the `GPTMTnMR` register is set, the `RTCRIS` bit in the `GPTMRIS` register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To reen able the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

18.4.2 Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the `TAEN` bit is cleared) before making any changes.
2. If the timer has been operating in a different mode prior to this, clear any residual set bits in the GPTM Timer n Mode (`GPTMTnMR`) register before reconfiguring.
3. Write the GPTM Configuration Register (`GPTMCFG`) with a value of `0x0000.0001`.
4. Write the match value to the GPTM Timer n Match Register (`GPTMTnMATCHR`).
5. Set/clear the `RTCEN` and `TnSTALL` bit in the GPTM Control Register (`GPTMCTL`) as needed.
6. If interrupts are required, set the `RTCIM` bit in the GPTM Interrupt Mask Register (`GPTMIMR`).
7. Set the `TAEN` bit in the `GPTMCTL` register to enable the timer and start counting.

When the timer count equals the value in the `GPTMTnMATCHR` register, the GPTM asserts the `RTCRIS` bit in the `GPTMRIS` register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the `RTCCINT` bit in the `GPTMICR` register. If the `GPTMTnILR` register is loaded with a new value, the timer begins counting at this new value and continues until it reaches `0xFFFF.FFFF`, at which point it rolls over.

18.4.3 Input Edge-Count Mode

A timer is configured to Input Edge-Count mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the GPTM Configuration (GPTMCFG) register with a value of 0x0000.0004.
3. In the GPTM Timer Mode (GPTMTnMR) register, write the TnCMR field to 0x0 and the TnMR field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the TnEVENT field of the GPTM Control (GPTMCTL) register.
5. Program registers according to count direction:
 - In down-count mode, the GPTMTnMATCHR and GPTMTnPMR registers are configured so that the difference between the value in the GPTMTnILR and GPTMTnPR registers and the GPTMTnMATCHR and GPTMTnPMR registers equals the number of edge events that must be counted.
 - In up-count mode, the timer counts from 0x0 to the value in the GPTMTnMATCHR and GPTMTnPMR registers. When executing an up-count, the value of the GPTMTnPR and GPTMTnILR must be greater than the value of GPTMTnPMR and GPTMTnMATCHR.
6. If interrupts are required, set the CnMIM bit in the GPTM Interrupt Mask (GPTMIMR) register.
7. Set the TnEN bit in the GPTMCTL register to enable the timer and begin waiting for edge events.
8. Poll the CnMRIS bit in the GPTMRIS register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnMCINT bit of the GPTM Interrupt Clear (GPTMICR) register.

When counting down in Input Edge-Count Mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the TnEN bit is cleared and repeat steps 4 to 8.

18.4.4 Input Edge Time Mode

A timer is configured to Input Edge Time mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the GPTM Configuration (GPTMCFG) register with a value of 0x0000.0004.
3. In the GPTM Timer Mode (GPTMTnMR) register, write the TnCMR field to 0x1 and the TnMR field to 0x3 and select a count direction by programming the TnCDIR bit.
4. Configure the type of event that the timer captures by writing the TnEVENT field of the GPTM Control (GPTMCTL) register.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale Register (GPTMTnPR).
6. Load the timer start value into the GPTM Timer n Interval Load (GPTMTnILR) register.
7. If interrupts are required, set the CnEIM bit in the GPTM Interrupt Mask (GPTMIMR) register.
8. Set the TnEN bit in the GPTM Control (GPTMCTL) register to enable the timer and start counting.
9. Poll the CnERIS bit in the GPTMRIS register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnECINT bit of the GPTM Interrupt Clear (GPTMICR) register. The time at which the event happened can be obtained by reading the GPTM Timer n (GPTMTnR) register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the GPTMTnILR register and clearing the TnILD bit in the GPTMTnMR register. The change takes effect at the next cycle after the write.

18.4.5 PWM Mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the GPTM Configuration (GPTMCFG) register with a value of 0x0000.0004.

3. In the GPTM Timer Mode (GPTMTnMR) register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the TnPWML field of the GPTM Control (GPTMCTL) register.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale Register (GPTMTnPR).
6. If PWM interrupts are used, configure the interrupt condition in the TnEVENT field in the GPTMCTL register and enable the interrupts by setting the TnPWMIE bit in the GPTMTnMR register. Edge detect interrupt behavior is reversed when the PWM output is inverted (see [Section 18.5.4](#)).
7. Load the timer start value into the GPTM Timer n Interval Load (GPTMTnILR) register.
8. Load the GPTM Timer n Match (GPTMTnMATCHR) register with the match value.
9. Set the TnEN bit in the GPTM Control (GPTMCTL) register to enable the timer and begin generation of the output PWM signal.

In PWM Time mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the GPTMTnILR register, and the change takes effect at the next cycle after the write.

18.5 GPTM Registers

[Table 18-10](#) lists the memory-mapped registers for the GPTM. All register offset addresses not listed in [Table 18-10](#) should be considered as reserved locations and the register contents should not be modified.

The offset listed is relative to base address of each timer:

- 16/32-bit Timer 0: 0x40030000
- 16/32-bit Timer 1: 0x40031000
- 16/32-bit Timer 2: 0x40032000
- 16/32-bit Timer 3: 0x40033000
- 16/32-bit Timer 4: 0x40034000
- 16/32-bit Timer 5: 0x40035000
- 16/32-bit Timer 6: 0x400E0000
- 16/32-bit Timer 7: 0x400E1000

The GP Timer module clock must be enabled before the registers can be programmed (see [Section 4.2.86](#)). There must be a delay of 3 system clock cycles after the Timer module clock is enabled before any Timer module registers are accessed.

Table 18-10. GPTM Registers

Offset	Acronym	Register Name	Section
0x0	GPTMCFG	GPTM Configuration	Section 18.5.1
0x4	GPTMTAMR	GPTM Timer A Mode	Section 18.5.2
0x8	GPTMTBMR	GPTM Timer B Mode	Section 18.5.3
0xC	GPTMCTL	GPTM Control	Section 18.5.4
0x10	GPTMSYNC	GPTM Synchronize	Section 18.5.5
0x18	GPTMIMR	GPTM Interrupt Mask	Section 18.5.6
0x1C	GPTMRIS	GPTM Raw Interrupt Status	Section 18.5.7
0x20	GPTMMIS	GPTM Masked Interrupt Status	Section 18.5.8
0x24	GPTMICR	GPTM Interrupt Clear	Section 18.5.9
0x28	GPTMTAILR	GPTM Timer A Interval Load	Section 18.5.10
0x2C	GPTMTBILR	GPTM Timer B Interval Load	Section 18.5.11
0x30	GPTMTAMATCHR	GPTM Timer A Match	Section 18.5.12
0x34	GPTMTBMATCHR	GPTM Timer B Match	Section 18.5.13
0x38	GPTMTAPR	GPTM Timer A Prescale	Section 18.5.14
0x3C	GPTMTBPR	GPTM Timer B Prescale	Section 18.5.15
0x40	GPTMTAPMR	GPTM TimerA Prescale Match	Section 18.5.16
0x44	GPTMTBPMR	GPTM TimerB Prescale Match	Section 18.5.17
0x48	GPTMTAR	GPTM Timer A	Section 18.5.18
0x4C	GPTMTBR	GPTM Timer B	Section 18.5.19
0x50	GPTMTAV	GPTM Timer A Value	Section 18.5.20
0x54	GPTMTBV	GPTM Timer B Value	Section 18.5.21
0x58	GPTMRTCPD	GPTM RTC Predivide	Section 18.5.22
0x5C	GPTMTAPS	GPTM Timer A Prescale Snapshot	Section 18.5.23
0x60	GPTMTBPS	GPTM Timer B Prescale Snapshot	Section 18.5.24
0x6C	GPTMDMAEV	GPTM DMA Event	Section 18.5.25
0x70	GPTMADCEV	GPTM ADC Event	Section 18.5.26
0xFC0	GPTMPP	GPTM Peripheral Properties	Section 18.5.27
0xFC8	GPTMCC	GPTM Clock Configuration	Section 18.5.28

Complex bit access types are encoded to fit into small table cells. [Table 18-11](#) shows the codes that are used for access types in this section.

Table 18-11. GPTM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

18.5.1 GPTMCFG Register (Offset = 0x0) [reset = X]

GPTM Configuration (GPTMCFG)

This register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 16-bit mode.

NOTE: Bits in this register should only be changed when the TAEN and TBEN bits in the GPTMCTL register are cleared.

GPTMCFG is shown in [Figure 18-9](#) and described in [Table 18-12](#).

Return to [Summary Table](#).

Figure 18-9. GPTMCFG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													GPTMCFG		
R-X													R/W-0x0		

Table 18-12. GPTMCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	X	
2-0	GPTMCFG	R/W	0x0	GPTM Configuration. 0x0 = Reserved 0x1 = For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. 0x4 = For a 16/32-bit timer, this value selects the 16-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR. 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved

18.5.2 GPTMTAMR Register (Offset = 0x4) [reset = 0x0]

GPTM Timer A Mode (GPTMTAMR)

This register configures the GPTM based on the configuration selected in the GPTMCFG register. When in PWM mode, set the TAAMS bit, clear the TACMR bit, and configure the TAMR field to 0x1 or 0x2.

This register controls the modes for Timer A when it is used individually. When Timer A and Timer B are concatenated, this register controls the modes for both Timer A and Timer B, and the contents of GPTMTBMR are ignored.

NOTE: Except for the TCACT bit field, all other bits in this register should only be changed when the TAEN bit in the GPTMCTL register is cleared.

GPTMTAMR is shown in [Figure 18-10](#) and described in [Table 18-13](#).

Return to [Summary Table](#).

Figure 18-10. GPTMTAMR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
TCACT			TACINTD	TAPLO	TAMRSU	TAPWMIE	TAILD
R/W-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	

Table 18-13. GPTMTAMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-13	TCACT	R/W	0x0	Timer Compare Action Select. 0x0 = Disable compare operations. 0x1 = Toggle State on Time-Out 0x2 = Clear CCP on Time-Out 0x3 = Set CCP on Time-Out 0x4 = Set CCP immediately and toggle on Time-Out 0x5 = Clear CCP immediately and toggle on Time-Out 0x6 = Set CCP immediately and clear on Time-Out 0x7 = Clear CCP immediately and set on Time-Out
12	TACINTD	R/W	0x0	One-shot/Periodic Interrupt Disable. 0x0 = Time-out interrupt functions as normal. 0x1 = Time-out interrupt are disabled. Setting the TACINTD bit in the GPTMTAMR register does not have an effect on the μ DMA or ADC interrupt time-out event trigger assertions. If the TATODMAEN bit is set in the GPTMDMAEV register or the TATOADCEN bit is set in the GPTMADCEV register, a μ DMA or ADC time-out trigger is sent to the μ DMA or ADC, respectively, even if the TACINTD bit is set.
11	TAPLO	R/W	0x0	GPTM Timer A PWM Legacy Operation. This bit is only valid in PWM mode. 0x0 = Legacy operation with CCP pin driven Low when the GPTMTAILR is reloaded after the timer reaches 0. 0x1 = CCP is driven High when the GPTMTAILR is reloaded after the timer reaches 0.

Table 18-13. GPTMTAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	TAMRSU	R/W	0x0	<p>GPTM Timer A Match Register Update.</p> <p>If the timer is disabled (TAEN is clear) when this bit is set, GPTMTAMATCHR and GPTMTAPR are updated when the timer is enabled.</p> <p>If the timer is stalled (TASTALL is set), GPTMTAMATCHR and GPTMTAPR are updated according to the configuration of this bit.</p> <p>0x0 = Update the GPTMTAMATCHR register and the GPTMTAPR register, if used, on the next cycle.</p> <p>0x1 = Update the GPTMTAMATCHR register and the GPTMTAPR register, if used, on the next timeout.</p>
9	TAPWMIE	R/W	0x0	<p>GPTM Timer A PWM Interrupt Enable.</p> <p>This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output, as defined by the TAEVENT field in the GPTMCTL register.</p> <p>In addition, when this bit is set and a capture event occurs, Timer A automatically generates triggers to the ADC and DMA if the trigger capability is enabled by setting the TAOTE bit in the GPTMCTL register and the CAEDMAEN bit in the GPTMDMAEV register, respectively.</p> <p>This bit is only valid in PWM mode.</p> <p>0x0 = Capture event interrupt is disabled.</p> <p>0x1 = Capture event interrupt is enabled.</p>
8	TAILD	R/W	0x0	<p>GPTM Timer A Interval Load Write.</p> <p>Note the state of this bit has no effect when counting up.</p> <p>The bit descriptions above apply if the timer is enabled and running.</p> <p>If the timer is disabled (TAEN is clear) when this bit is set, GPTMTAR, GPTMTAV and GPTMTAPS, are updated when the timer is enabled.</p> <p>If the timer is stalled (TASTALL is set), GPTMTAR and GPTMTAPS are updated according to the configuration of this bit.</p> <p>0x0 = Update the GPTMTAR and GPTMTAV registers with the value in the GPTMTAILR register on the next cycle. Also update the GPTMTAPS register with the value in the GPTMTAPR register on the next cycle.</p> <p>0x1 = Update the GPTMTAR and GPTMTAV registers with the value in the GPTMTAILR register on the next timeout. Also update the GPTMTAPS register with the value in the GPTMTAPR register on the next timeout.</p>
7	TASNAPS	R/W	0x0	<p>GPTM Timer A Snap-Shot Mode.</p> <p>0x0 = Snap-shot mode is disabled.</p> <p>0x1 = If Timer A is configured in the periodic mode, the actual free-running, capture or snapshot value of Timer A is loaded at the time-out event/capture or snapshot event into the GPTM Timer A (GPTMTAR) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer A (GPTMTAPR).</p>
6	TAWOT	R/W	0x0	<p>GPTM Timer A Wait-on-Trigger.</p> <p>If the application requires cyclical daisy-chaining, the TAWOT bit in the GPTMTAMR register of Timer 0 can be set.</p> <p>In this case, Timer 0 waits for a trigger from the last timer module in the chain.</p> <p>0x0 = Timer A begins counting as soon as it is enabled.</p> <p>0x1 = If Timer A is enabled (TAEN is set in the GPTMCTL register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see . This function is valid for one-shot, periodic, and PWM modes.</p>

Table 18-13. GPTMTAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	TAMIE	R/W	0x0	<p>GPTM Timer A Match Interrupt Enable.</p> <p>0x0 = The match interrupt is disabled for match events. Clearing the TAMIE bit in the GPTMTAMR register prevents assertion of μDMA or ADC requests generated on a match event. Even if the TATODMAEN bit is set in the GPTMDMAEV register or the TATODADCEN bit is set in the GPTMADCEV register, a μDMA or ADC match trigger is not sent to the μDMA or ADC, respectively, when the TAMIE bit is clear.</p> <p>0x1 = An interrupt is generated when the match value in the GPTMTAMATCHR register is reached in the one-shot and periodic modes.</p>
4	TACDIR	R/W	0x0	<p>GPTM Timer A Count Direction.</p> <p>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.</p> <p>0x0 = The timer counts down.</p> <p>0x1 = The timer counts up. When counting up, the timer starts from a value of 0x0.</p>
3	TAAMS	R/W	0x0	<p>GPTM Timer A Alternate Mode Select.</p> <p>0x0 = Capture or compare mode is enabled.</p> <p>0x1 = PWM mode is enabled. To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2.</p>
2	TACMR	R/W	0x0	<p>GPTM Timer A Capture Mode.</p> <p>0x0 = Edge-Count mode</p> <p>0x1 = Edge-Time mode</p>
1-0	TAMR	R/W	0x0	<p>GPTM Timer A Mode.</p> <p>The Timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register.</p> <p>0x0 = Reserved</p> <p>0x1 = One-Shot Timer mode</p> <p>0x2 = Periodic Timer mode</p> <p>0x3 = Capture mode</p>

18.5.3 GPTMTBMR Register (Offset = 0x8) [reset = 0x0]

GPTM Timer B Mode (GPTMTBMR)

This register configures the GPTM based on the configuration selected in the GPTMCFG register. When in PWM mode, set the TBAMS bit, clear the TBCMR bit, and configure the TBMR field to 0x1 or 0x2.

This register controls the modes for Timer B when it is used individually. When Timer A and Timer B are concatenated, this register is ignored and GPTMTAMR controls the modes for both Timer A and Timer B.

NOTE: Except for the TCACT bit field, all other bits in this register should only be changed when the TBEN bit in the GPTMCTL register is cleared.

GPTMTBMR is shown in [Figure 18-11](#) and described in [Table 18-14](#).

Return to [Summary Table](#).

Figure 18-11. GPTMTBMR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
TCACT			TBCINTD	TBPLO	TBMRSU	TBPWMIE	TBILD
R/W-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	

Table 18-14. GPTMTBMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-13	TCACT	R/W	0x0	Timer Compare Action Select. 0x0 = Disable compare operations 0x1 = Toggle State on Time-Out 0x2 = Clear CCP on Time-Out 0x3 = Set CCP on Time-Out 0x4 = Set CCP immediately and toggle on Time-Out 0x5 = Clear CCP immediately and toggle on Time-Out 0x6 = Set CCP immediately and clear on Time-Out 0x7 = Clear CCP immediately and set on Time-Out
12	TBCINTD	R/W	0x0	One-Shot/Periodic Interrupt Disable. 0x0 = Time-out interrupt functions normally 0x1 = Time-out interrupt functionality is disabledSetting the TBCINTD bit in the GPTMTBMR register does not have an effect on the μ DMA or ADC interrupt time-out event trigger assertions. If the TBTODMAEN bit is set in the GPTMDMAEV register or the TBTOADCEN bit is set in the GPTMADCEV register, a μ DMA or ADC time-out trigger is sent to the μ DMA or ADC, respectively, even if the TBCINTD bit is set.
11	TBPLO	R/W	0x0	GPTM Timer B PWM Legacy Operation. This bit is only valid in PWM mode. 0x0 = Legacy operation with CCP pin driven Low when the GPTMTAILR is reloaded after the timer reaches 0. 0x1 = CCP is driven High when the GPTMTAILR is reloaded after the timer reaches 0.

Table 18-14. GPTMTBMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	TBMRSU	R/W	0x0	<p>GPTM Timer B Match Register Update.</p> <p>If the timer is disabled (TBEN is clear) when this bit is set, GPTMTBMATCHR and GPTMTBPR are updated when the timer is enabled.</p> <p>If the timer is stalled (TBSTALL is set), GPTMTBMATCHR and GPTMTBPR are updated according to the configuration of this bit.</p> <p>0x0 = Update the GPTMTBMATCHR register and the GPTMTBPR register, if used, on the next cycle.</p> <p>0x1 = Update the GPTMTBMATCHR register and the GPTMTBPR register, if used, on the next timeout.</p>
9	TBPWMIE	R/W	0x0	<p>GPTM Timer B PWM Interrupt Enable.</p> <p>This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output as defined by the TBEVENT field in the GPTMCTL register.</p> <p>In addition, when this bit is set and a capture event occurs, Timer B automatically generates triggers to the ADC and DMA if the trigger capability is enabled by setting the TBOTE bit in the GPTMCTL register and the CBEDMAEN bit in the GPTMDMAEV register, respectively.</p> <p>This bit is only valid in PWM mode.</p> <p>0x0 = Capture event interrupt is disabled.</p> <p>0x1 = Capture event is enabled.</p>
8	TBILD	R/W	0x0	<p>GPTM Timer B Interval Load Write.</p> <p>Note the state of this bit has no effect when counting up.</p> <p>The bit descriptions above apply if the timer is enabled and running.</p> <p>If the timer is disabled (TBEN is clear) when this bit is set, GPTMTBTR, GPTMTBV and are updated when the timer is enabled.</p> <p>If the timer is stalled (TBSTALL is set), GPTMTBTR and GPTMTBPS are updated according to the configuration of this bit.</p> <p>0x0 = Update the GPTMTBTR and GPTMTBV registers with the value in the GPTMTBILR register on the next cycle. Also update the GPTMTBPS register with the value in the GPTMTBPR register on the next cycle.</p> <p>0x1 = Update the GPTMTBTR and GPTMTBV registers with the value in the GPTMTBILR register on the next timeout. Also update the GPTMTBPS register with the value in the GPTMTBPR register on the next timeout.</p>
7	TBSNAPS	R/W	0x0	<p>GPTM Timer B Snap-Shot Mode.</p> <p>0x0 = Snap-shot mode is disabled.</p> <p>0x1 = If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the GPTM Timer B (GPTMTBTR) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer B (GPTMTBPR).</p>
6	TBWOT	R/W	0x0	<p>GPTM Timer B Wait-on-Trigger.</p> <p>0x0 = Timer B begins counting as soon as it is enabled.</p> <p>0x1 = If Timer B is enabled (TBEN is set in the GPTMCTL register), Timer B does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see. This function is valid for one-shot, periodic, and PWM modes.</p>
5	TBMIE	R/W	0x0	<p>GPTM Timer B Match Interrupt Enable.</p> <p>0x0 = The match interrupt is disabled for match events. Additionally, triggers to the DMA and ADC on match events are prevented.</p> <p>0x1 = An interrupt is generated when the match value in the GPTMTBMATCHR register is reached in the one-shot and periodic modes. Clearing the TBMIE bit in the GPTMTBMR register prevents assertion of μDMA or ADC requests generated on a match event. Even if the TBODMAEN bit is set in the GPTMDMAEV register or the TBODADCEN bit is set in the GPTMADCEV register, a μDMA or ADC match trigger is not sent to the μDMA or ADC, respectively, when the TBMIE bit is clear.</p>

Table 18-14. GPTMTBMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	TBCDIR	R/W	0x0	GPTM Timer B Count Direction. When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up. 0x0 = The timer counts down. 0x1 = The timer counts up. When counting up, the timer starts from a value of 0x0.
3	TBAMS	R/W	0x0	GPTM Timer B Alternate Mode Select. 0x0 = Capture or compare mode is enabled. 0x1 = PWM mode is enabled. To enable PWM mode, you must also clear the TBCMR bit and configure the TBMR field to 0x1 or 0x2.
2	TBCMR	R/W	0x0	GPTM Timer B Capture Mode 0x0 = Edge-Count mode 0x1 = Edge-Time mode
1-0	TBMR	R/W	0x0	GPTM Timer B Mode. The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register. 0x0 = Reserved 0x1 = One-Shot Timer mode 0x2 = Periodic Timer mode 0x3 = Capture mode

18.5.4 GPTMCTL Register (Offset = 0xC) [reset = 0x0]

GPTM Control (GPTMCTL)

This register is used alongside the GPTMCFG and GMTMTnMR registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

NOTE: Bits in this register should only be changed when the TnEN bit for the respective timer is cleared.

GPTMCTL is shown in [Figure 18-12](#) and described in [Table 18-15](#).

Return to [Summary Table](#).

Figure 18-12. GPTMCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED	TBPWML	TBOTE	RESERVED	TBEVENT		TBSTALL	TBEN
R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0		R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0

Table 18-15. GPTMCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0x0	
14	TBPWML	R/W	0x0	GPTM Timer B PWM Output Level 0x0 = Output is unaffected. 0x1 = Output is inverted.
13	TBOTE	R/W	0x0	GPTM Timer B Output Trigger Enable. 0x0 = The output Timer B ADC trigger is disabled. 0x1 = The output Timer B ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the ADCMUX register.
12	RESERVED	R	0x0	
11-10	TBEVENT	R/W	0x0	GPTM Timer B Event Mode. If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal. 0x0 = Positive edge 0x1 = Negative edge 0x2 = Reserved 0x3 = Both edges

Table 18-15. GPTMCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	TBSTALL	R/W	0x0	GPTM Timer B Stall Enable. If the processor is executing normally, the TBSTALL bit is ignored. 0x0 = Timer B continues counting while the processor is halted by the debugger. 0x1 = Timer B freezes counting while the processor is halted by the debugger.
8	TBEN	R/W	0x0	GPTM Timer B Enable 0x0 = Timer B is disabled. 0x1 = Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.
7	RESERVED	R	0x0	
6	TAPWML	R/W	0x0	GPTM Timer A PWM Output Level. 0x0 = Output is unaffected. 0x1 = Output is inverted.
5	TAOTE	R/W	0x0	GPTM Timer A Output Trigger Enable. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the ADCMUX register (see). 0x0 = The output Timer A ADC trigger is disabled. 0x1 = The output Timer A ADC trigger is enabled.
4	RTCEN	R/W	0x0	GPTM RTC Stall Enable If the RTCEN bit is set, it prevents the timer from stalling in all operating modes, even if TnSTALL is set. 0x0 = RTC counting freezes while the processor is halted by the debugger. 0x1 = RTC counting continues while the processor is halted by the debugger.
3-2	TAEVENT	R/W	0x0	GPTM Timer A Event Mode. If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal. 0x0 = Positive edge 0x1 = Negative edge 0x2 = Reserved 0x3 = Both edges
1	TASTALL	R/W	0x0	GPTM Timer A Stall Enable. If the processor is executing normally, the TASTALL bit is ignored. 0x0 = Timer A continues counting while the processor is halted by the debugger. 0x1 = Timer A freezes counting while the processor is halted by the debugger.
0	TAEN	R/W	0x0	GPTM Timer A Enable. 0x0 = Timer A is disabled. 0x1 = Timer A is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.

18.5.5 GPTMSYNC Register (Offset = 0x10) [reset = 0x0]

GPTM Synchronize (GPTMSYNC)

NOTE: This register is only implemented on GPTM Module 0 only.

This register allows software to synchronize a number of timers.

GPTMSYNC is shown in [Figure 18-13](#) and described in [Table 18-16](#).

Return to [Summary Table](#).

Figure 18-13. GPTMSYNC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNCT7		SYNCT6		SYNCT5		SYNCT4		SYNCT3		SYNCT2		SYNCT1		SYNCT0	
WO-0x0		WO-0x0		WO-0x0		WO-0x0		WO-0x0		WO-0x0		WO-0x0		WO-0x0	

Table 18-16. GPTMSYNC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-14	SYNCT7	W	0x0	Synchronize GPTM Timer 7 0x0 = GPT7 is not affected. 0x1 = A timeout event for Timer A of GPTM7 is triggered. 0x2 = A timeout event for Timer B of GPTM7 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM7 is triggered.
13-12	SYNCT6	W	0x0	Synchronize GPTM Timer 6 0x0 = GPTM6 is not affected. 0x1 = A timeout event for Timer A of GPTM6 is triggered. 0x2 = A timeout event for Timer B of GPTM6 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM6 is triggered.
11-10	SYNCT5	W	0x0	Synchronize GPTM Timer 5 0x0 = GPTM5 is not affected. 0x1 = A timeout event for Timer A of GPTM5 is triggered. 0x2 = A timeout event for Timer B of GPTM5 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM5 is triggered.
9-8	SYNCT4	W	0x0	Synchronize GPTM Timer 4 0x0 = GPTM4 is not affected. 0x1 = A timeout event for Timer A of GPTM4 is triggered. 0x2 = A timeout event for Timer B of GPTM4 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM4 is triggered.
7-6	SYNCT3	W	0x0	Synchronize GPTM Timer 3 0x0 = GPTM3 is not affected. 0x1 = A timeout event for Timer A of GPTM3 is triggered. 0x2 = A timeout event for Timer B of GPTM3 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM3 is triggered.

Table 18-16. GPTMSYNC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	SYNCT2	W	0x0	Synchronize GPTM Timer 2 0x0 = GPTM2 is not affected. 0x1 = A timeout event for Timer A of GPTM2 is triggered. 0x2 = A timeout event for Timer B of GPTM2 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM2 is triggered.
3-2	SYNCT1	W	0x0	Synchronize GPTM Timer 1 0x0 = GPTM1 is not affected. 0x1 = A timeout event for Timer A of GPTM1 is triggered. 0x2 = A timeout event for Timer B of GPTM1 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM1 is triggered.
1-0	SYNCT0	W	0x0	Synchronize GPTM Timer 0 0x0 = GPTM0 is not affected. 0x1 = A timeout event for Timer A of GPTM0 is triggered. 0x2 = A timeout event for Timer B of GPTM0 is triggered. 0x3 = A timeout event for both Timer A and Timer B of GPTM0 is triggered.

18.5.6 GPTMIMR Register (Offset = 0x18) [reset = 0x0]

GPTM Interrupt Mask (GPTMIMR)

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTMIMR is shown in [Figure 18-14](#) and described in [Table 18-17](#).

Return to [Summary Table](#).

Figure 18-14. GPTMIMR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		DMABIM	RESERVED	TBMIM	CBEIM	CBMIM	TBTOIM
R-0x0		R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED		DMAAIM	TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 18-17. GPTMIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	DMABIM	R/W	0x0	GPTM Timer B DMA Done Interrupt Mask. The DMABIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
12	RESERVED	R	0x0	
11	TBMIM	R/W	0x0	GPTM Timer B Match Interrupt Mask. The TBMIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
10	CBEIM	R/W	0x0	GPTM Timer B Capture Mode Event Interrupt Mask. The CBEIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
9	CBMIM	R/W	0x0	GPTM Timer B Capture Mode Match Interrupt Mask. The CBMIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
8	TBTOIM	R/W	0x0	GPTM Timer B Time-Out Interrupt Mask. The TBTOIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
7-6	RESERVED	R	0x0	
5	DMAAIM	R/W	0x0	GPTM Timer A DMA Done Interrupt Mask. The DMAAIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.

Table 18-17. GPTMIMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	TAMIM	R/W	0x0	GPTM Timer A Match Interrupt Mask. The TAMIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
3	RTCIM	R/W	0x0	GPTM RTC Interrupt Mask. The RTCIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
2	CAEIM	R/W	0x0	GPTM Timer A Capture Mode Event Interrupt Mask. The CAEIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
1	CAMIM	R/W	0x0	GPTM Timer A Capture Mode Match Interrupt Mask. The CAMIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.
0	TATOIM	R/W	0x0	GPTM Timer A Time-Out Interrupt Mask. The TATOIM values are defined as follows: 0x0 = Interrupt is disabled. 0x1 = Interrupt is enabled.

18.5.7 GPTMRIS Register (Offset = 0x1C) [reset = 0x0]

GPTM Raw Interrupt Status (GPTMRIS)

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the GPTMIMR register. Each bit can be cleared by writing a 1 to its corresponding bit in GPTMICR.

NOTE: The state of the GPTMRIS register is not affected by disabling and then re-enabling the timer using the TnEN bits in the GPTM Control (GPTMCTL) register. If an application requires that all or certain status bits should not carry over after re-enabling the timer, then the appropriate bits in the GPTMRIS register should be cleared using the GPTMICR register prior to re-enabling the timer. If this is not done, any status bits set in the GPTMRIS register and unmasked in the GPTMIMR register generate an interrupt once the timer is re-enabled.

GPTMRIS is shown in [Figure 18-15](#) and described in [Table 18-18](#).

Return to [Summary Table](#).

Figure 18-15. GPTMRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		DMABRIS	RESERVED	TBMRIS	CBERIS	CBMRIS	TBTORIS
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
RESERVED		DMAARIS	TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 18-18. GPTMRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	DMABRIS	R	0x0	GPTM Timer B DMA Done Raw Interrupt Status. 0x0 = The Timer B DMA transfer has not completed. 0x1 = The Timer B DMA transfer has completed.
12	RESERVED	R	0x0	
11	TBMRIS	R	0x0	GPTM Timer B Match Raw Interrupt. This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR register. 0x0 = The match value has not been reached. 0x1 = The TBMIE bit is set in the GPTMTBMR register, and the match values in the GPTMTBMATCHR and (optionally) GPTMTBPMR registers have been reached when configured in one-shot or periodic mode.
10	CBERIS	R	0x0	GPTM Timer B Capture Mode Event Raw Interrupt. This bit is cleared by writing a 1 to the CBECINT bit in the GPTMICR register. 0x0 = The capture mode event for Timer B has not occurred. 0x1 = A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode.

Table 18-18. GPTMRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	CBMRIS	R	0x0	GPTM Timer B Capture Mode Match Raw Interrupt. This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR register. 0x0 = The capture mode match for Timer B has not occurred. 0x1 = The capture mode match has occurred for Timer B. This interrupt asserts when the values in the GPTMTBR and GPTMTBPR match the values in the GPTMTBMATCHR and GPTMTBPMR when configured in Input Edge-Time mode.
8	TBTORIS	R	0x0	GPTM Timer B Time-Out Raw Interrupt. This bit is cleared by writing a 1 to the TBTOCINT bit in the GPTMICR register. 0x0 = Timer B has not timed out. 0x1 = Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into GPTMTBILR, depending on the count direction).
7-6	RESERVED	R	0x0	
5	DMAARIS	R	0x0	GPTM Timer A DMA Done Raw Interrupt Status. 0x0 = The Timer A DMA transfer has not completed. 0x1 = The Timer A DMA transfer has completed.
4	TAMRIS	R	0x0	GPTM Timer A Match Raw Interrupt. This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR register. 0x0 = The match value has not been reached. 0x1 = The TAMIE bit is set in the GPTMTAMR register, and the match value in the GPTMTAMATCHR and (optionally) GPTMTAPMR registers have been reached when configured in one-shot or periodic mode.
3	RTCRIS	R	0x0	GPTM RTC Raw Interrupt. This bit is cleared by writing a 1 to the RTCCINT bit in the GPTMICR register. 0x0 = The RTC event has not occurred. 0x1 = The RTC event has occurred.
2	CAERIS	R	0x0	GPTM Timer A Capture Mode Event Raw Interrupt This bit is cleared by writing a 1 to the CAECINT bit in the GPTMICR register. 0x0 = The capture mode event for Timer A has not occurred. 0x1 = A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode.
1	CAMRIS	R	0x0	GPTM Timer A Capture Mode Match Raw Interrupt. This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR register. 0x0 = The capture mode match for Timer A has not occurred. 0x1 = A capture mode match has occurred for Timer A. This interrupt asserts when the values in the GPTMTAR and GPTMTAPR match the values in the GPTMTAMATCHR and GPTMTAPMR when configured in Input Edge-Time mode.
0	TATORIS	R	0x0	GPTM Timer A Time-Out Raw Interrupt. This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register. 0x0 = Timer A has not timed out. 0x1 = Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into GPTMTAILR, depending on the count direction).

18.5.8 GPTMMIS Register (Offset = 0x20) [reset = X]

GPTM Masked Interrupt Status (GPTMMIS)

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in GPTMIMR, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in GPTMICR.

GPTMMIS is shown in [Figure 18-16](#) and described in [Table 18-19](#).

Return to [Summary Table](#).

Figure 18-16. GPTMMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-X							
23	22	21	20	19	18	17	16
RESERVED							
R-X							
15	14	13	12	11	10	9	8
RESERVED		DMABMIS	RESERVED	TBMMIS	CBEMIS	CBMMIS	TBTOMIS
R-X		R-0x0	R-0x0	R-0x0	0x0	R-0x0	R-X
7	6	5	4	3	2	1	0
RESERVED		DMAAMIS	TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 18-19. GPTMMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	X	
13	DMABMIS	R	0x0	GPTM Timer B DMA Done Masked Interrupt. This bit is cleared by writing a 1 to the DMABINT bit in the GPTMICR register. 0x0 = A Timer B DMA done interrupt has not occurred or is masked. 0x1 = An unmasked Timer B DMA done interrupt has occurred.
12	RESERVED	R	0x0	
11	TBMMIS	R	0x0	GPTM Timer B Match Masked Interrupt. This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR register. 0x0 = A Timer B Mode Match interrupt has not occurred or is masked. 0x1 = An unmasked Timer B Mode Match interrupt has occurred.
10	CBEMIS		0x0	GPTM Timer B Capture Mode Event Masked Interrupt. This bit is cleared by writing a 1 to the CBECINT bit in the GPTMICR register. 0x0 = A Capture B event interrupt has not occurred or is masked. 0x1 = An unmasked Capture B event interrupt has occurred.
9	CBMMIS	R	0x0	GPTM Timer B Capture Mode Match Masked Interrupt. This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR register. 0x0 = A Capture B Mode Match interrupt has not occurred or is masked. 0x1 = An unmasked Capture B Match interrupt has occurred.
8	TBTOMIS	R	X	GPTM Timer B Time-Out Masked Interrupt. This bit is cleared by writing a 1 to the TBTOCINT bit in the GPTMICR register. 0x0 = A Timer B Time-Out interrupt has not occurred or is masked. 0x1 = An unmasked Timer B Time-Out interrupt has occurred.
7-6	RESERVED	R	0x0	

Table 18-19. GPTMMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	DMAAMIS	R	0x0	GPTM Timer A DMA Done Masked Interrupt. This bit is cleared by writing a 1 to the DMAAINT bit in the GPTMICR register. 0x0 = A Timer A DMA done interrupt has not occurred or is masked. 0x1 = An unmasked Timer A DMA done interrupt has occurred.
4	TAMMIS	R	0x0	GPTM Timer A Match Masked Interrupt This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR register. 0x0 = A Timer A Mode Match interrupt has not occurred or is masked. 0x1 = An unmasked Timer A Mode Match interrupt has occurred.
3	RTCMIS	R	0x0	GPTM RTC Masked Interrupt. This bit is cleared by writing a 1 to the RTCCINT bit in the GPTMICR register. 0x0 = An RTC event interrupt has not occurred or is masked. 0x1 = An unmasked RTC event interrupt has occurred.
2	CAEMIS	R	0x0	GPTM Timer A Capture Mode Event Masked Interrupt. This bit is cleared by writing a 1 to the CAECINT bit in the GPTMICR register. 0x0 = A Capture A event interrupt has not occurred or is masked. 0x1 = An unmasked Capture A event interrupt has occurred.
1	CAMMIS	R	0x0	GPTM Timer A Capture Mode Match Masked Interrupt. This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR register. 0x0 = A Capture A Mode Match interrupt has not occurred or is masked. 0x1 = An unmasked Capture A Match interrupt has occurred.
0	TATOMIS	R	0x0	GPTM Timer A Time-Out Masked Interrupt. This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register. 0x0 = A Timer A Time-Out interrupt has not occurred or is masked. 0x1 = An unmasked Timer A Time-Out interrupt has occurred.

18.5.9 GPTMICR Register (Offset = 0x24) [reset = X]

GPTM Interrupt Clear (GPTMICR)

This register is used to clear the status bits in the GPTMRIS and GPTMMIS registers. Writing a 1 to a bit clears the corresponding bit in the GPTMRIS and GPTMMIS registers.

GPTMICR is shown in [Figure 18-17](#) and described in [Table 18-20](#).

Return to [Summary Table](#).

Figure 18-17. GPTMICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		DMABINT	RESERVED	TBMCINT	CBECINT	CBMCINT	TBTOCINT
R-X		W1C-0x0	R-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0
7	6	5	4	3	2	1	0
RESERVED		DMAAINT	TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
R-0x0		W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0

Table 18-20. GPTMICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	DMABINT	W1C	0x0	GPTM Timer B DMA Done Interrupt Clear. Writing a 1 to this bit clears the DMABRIS bit in the GPTMRIS register and the DMABMIS bit in the GPTMMIS register.
12	RESERVED	R	0x0	
11	TBMCINT	W1C	0x0	GPTM Timer B Match Interrupt Clear. Writing a 1 to this bit clears the TBMRIS bit in the GPTMRIS register and the TBMMIS bit in the GPTMMIS register.
10	CBECINT	W1C	0x0	GPTM Timer B Capture Mode Event Interrupt Clear. Writing a 1 to this bit clears the CBERIS bit in the GPTMRIS register and the CBEMIS bit in the GPTMMIS register.
9	CBMCINT	W1C	0x0	GPTM Timer B Capture Mode Match Interrupt Clear. Writing a 1 to this bit clears the CBMRIS bit in the GPTMRIS register and the CBMMIS bit in the GPTMMIS register.
8	TBTOCINT	W1C	0x0	GPTM Timer B Time-Out Interrupt Clear. Writing a 1 to this bit clears the TBTORIS bit in the GPTMRIS register and the TBTOMIS bit in the GPTMMIS register.
7-6	RESERVED	R	0x0	
5	DMAAINT	W1C	0x0	GPTM Timer A DMA Done Interrupt Clear. Writing a 1 to this bit clears the DMAARIS bit in the GPTMRIS register and the DMAAMIS bit in the GPTMMIS register.
4	TAMCINT	W1C	0x0	GPTM Timer A Match Interrupt Clear. Writing a 1 to this bit clears the TAMRIS bit in the GPTMRIS register and the TAMMIS bit in the GPTMMIS register.
3	RTCCINT	W1C	0x0	GPTM RTC Interrupt Clear. Writing a 1 to this bit clears the RTCRIS bit in the GPTMRIS register and the RTCMIS bit in the GPTMMIS register.
2	CAECINT	W1C	0x0	GPTM Timer A Capture Mode Event Interrupt Clear. Writing a 1 to this bit clears the CAERIS bit in the GPTMRIS register and the CAEMIS bit in the GPTMMIS register.

Table 18-20. GPTMICR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	CAMCINT	W1C	0x0	GPTM Timer A Capture Mode Match Interrupt Clear. Writing a 1 to this bit clears the CAMRIS bit in the GPTMRIS register and the CAMMIS bit in the GPTMMIS register.
0	TATOCINT	W1C	0x0	GPTM Timer A Time-Out Raw Interrupt. Writing a 1 to this bit clears the TATORIS bit in the GPTMRIS register and the TATOMIS bit in the GPTMMIS register.

18.5.10 GPTMTAILR Register (Offset = 0x28) [reset = 0xFFFFFFFF]

GPTM Timer A Interval Load (GPTMTAILR)

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, GPTMTAILR appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Interval Load (GPTMTBILR) register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of GPTMTBILR.

GPTMTAILR is shown in [Figure 18-18](#) and described in [Table 18-21](#).

Return to [Summary Table](#).

Figure 18-18. GPTMTAILR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAILR																															
R/W-0xFFFFFFFF																															

Table 18-21. GPTMTAILR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAILR	R/W	0xFFFFFFFF F	GPTM Timer A Interval Load Register. Writing this field loads the counter for Timer A. A read returns the current value of GPTMTAILR.

18.5.11 GPTMTBILR Register (Offset = 0x2C) [reset = 0xFFFF]

GPTM Timer B Interval Load (GPTMTBILR)

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAILR register. Reads from this register return the current value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the load value. Bits 31:16 are RESERVED in both cases.

GPTMTBILR is shown in [Figure 18-19](#) and described in [Table 18-22](#).

Return to [Summary Table](#).

Figure 18-19. GPTMTBILR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBILR																															
R/W-0xFFFF																															

Table 18-22. GPTMTBILR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBILR	R/W	0xFFFF	GPTM Timer B Interval Load Register. Writing this field loads the counter for Timer B. A read returns the current value of GPTMTBILR. When a 16/32-bit GPTM is in 32-bit mode, writes are ignored, and reads return the current value of GPTMTBILR.

18.5.12 GPTMTAMATCHR Register (Offset = 0x30) [reset = 0xFFFFFFFF]

GPTM Timer A Match (GPTMTAMATCHR)

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with GPTMTAILR, determines how many edge events are counted. The total number of edge events counted is equal to the value in GPTMTAILR minus this value. Note that in edge-count mode, when executing an up-count, the value of GPTMTnPR and GPTMTnILR must be greater than the value of GPTMTnPMR and GPTMTnMATCHR.

In PWM mode, this value along with GPTMTAILR, determines the duty cycle of the output PWM signal.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, GPTMTAMATCHR appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Match (GPTMTBMATCHR) register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of GPTMTBMATCHR.

GPTMTAMATCHR is shown in [Figure 18-20](#) and described in [Table 18-23](#).

Return to [Summary Table](#).

Figure 18-20. GPTMTAMATCHR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMR																															
R/W-0xFFFFFFFF																															

Table 18-23. GPTMTAMATCHR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAMR	R/W	0xFFFFFFFF F	GPTM Timer A Match Register. This value is compared to the GPTMTAR register to determine match events.

18.5.13 GPTMTBMATCHR Register (Offset = 0x34) [reset = 0xFFFF]

GPTM Timer B Match (GPTMTBMATCHR)

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with GPTMTBILR determines how many edge events are counted. The total number of edge events counted is equal to the value in GPTMTBILR minus this value. Note that in edge-count mode, when executing an up-count, the value of GPTMTnPR and GPTMTnILR must be greater than the value of GPTMTnPMR and GPTMTnMATCHR.

In PWM mode, this value along with GPTMTBILR, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAMATCHR register. Reads from this register return the current match value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the match value. Bits 31:16 are RESERVED in both cases.

GPTMTBMATCHR is shown in [Figure 18-21](#) and described in [Table 18-24](#).

Return to [Summary Table](#).

Figure 18-21. GPTMTBMATCHR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBMR																															
R/W-0xFFFF																															

Table 18-24. GPTMTBMATCHR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBMR	R/W	0xFFFF	GPTM Timer B Match Register. This value is compared to the GPTMTBR register to determine match events.

18.5.14 GPTMTAPR Register (Offset = 0x38) [reset = 0x0]

GPTM Timer A Prescale (GPTMTAPR)

This register allows software to extend the range of the timers when they are used individually. When in one-shot or periodic down count modes, this register acts as a true prescaler for the timer counter. When acting as a true prescaler, the prescaler counts down to 0 before the value in the GPTMTAR and GPTMTAV registers are incremented. In all other individual/split modes, this register is a linear extension of the upper range of the timer counter, holding bits 23:16 in the 16-bit modes of the 16/32-bit GPTM. See [Table 18-4](#) for more details and an example.

GPTMTAPR is shown in [Figure 18-22](#) and described in [Table 18-25](#).

Return to [Summary Table](#).

Figure 18-22. GPTMTAPR Register

7	6	5	4	3	2	1	0
TAPSR							
R/W-0x0							

Table 18-25. GPTMTAPR Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	TAPSR	R/W	0x0	GPTM Timer A Prescale. The register loads this value on a write. A read returns the current value of the register. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler.

18.5.15 GPTMTBPR Register (Offset = 0x3C) [reset = 0x0]

GPTM Timer B Prescale (GPTMTBPR)

This register allows software to extend the range of the timers when they are used individually. When in one-shot or periodic down count modes, this register acts as a true prescaler for the timer counter. When acting as a true prescaler, the prescaler counts down to 0 before the value in the GPTMTBR and GPTMTBV registers are incremented. In all other individual/split modes, this register is a linear extension of the upper range of the timer counter, holding bits 23:16 in the 16-bit modes of the 16/32-bit GPTM. See [Table 18-4](#) for more details and an example.

GPTMTBPR is shown in [Figure 18-23](#) and described in [Table 18-26](#).

Return to [Summary Table](#).

Figure 18-23. GPTMTBPR Register

7	6	5	4	3	2	1	0
TBPSR							
R/W-0x0							

Table 18-26. GPTMTBPR Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	TBPSR	R/W	0x0	GPTM Timer B Prescale. The register loads this value on a write. A read returns the current value of this register. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler.

18.5.16 GPTMTAPMR Register (Offset = 0x40) [reset = 0x0]

GPTM TimerA Prescale Match (GPTMTAPMR)

This register allows software to extend the range of the GPTMTAMATCHR when the timers are used individually. This register holds bits 23:16 in the 16-bit modes of the 16/32-bit GPTM.

GPTMTAPMR is shown in [Figure 18-24](#) and described in [Table 18-27](#).

Return to [Summary Table](#).

Figure 18-24. GPTMTAPMR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TAPSMR															
R-0x0																R/W-0x0															

Table 18-27. GPTMTAPMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	TAPSMR	R/W	0x0	GPTM TimerA Prescale Match. This value is used alongside GPTMTAMATCHR to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler match value.

18.5.17 GPTMTBPMR Register (Offset = 0x44) [reset = 0x0]

GPTM TimerB Prescale Match (GPTMTBPMR)

This register allows software to extend the range of the GPTMTBMATCHR when the timers are used individually. This register holds bits 23:16 in the 16-bit modes of the 16/32-bit GPTM.

GPTMTBPMR is shown in [Figure 18-25](#) and described in [Table 18-28](#).

Return to [Summary Table](#).

Figure 18-25. GPTMTBPMR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								TBPSMR							
R-0x0																								R/W-0x0							

Table 18-28. GPTMTBPMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	TBPSMR	R/W	0x0	GPTM TimerB Prescale Match. This value is used alongside GPTMTBMATCHR to detect timer match events while using a prescaler.

18.5.18 GPTMTAR Register (Offset = 0x48) [reset = 0xFFFFFFFF]

GPTM Timer A (GPTMTAR)

This register shows the current value of the Timer A counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

NOTE: When an alternate clock source is enabled, a read of this register returns the current count -1.

When a GPTM is configured to one of the 32-bit modes, GPTMTAR appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B (GPTMTBR) register). In the 16-bit Input Edge Count, Input Edge Time, and PWM modes, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the GPTMTAV register. To read the value of the prescaler in periodic snapshot mode, read the Timer A Prescale Snapshot (GPTMTAPS) register.

GPTMTAR is shown in [Figure 18-26](#) and described in [Table 18-29](#).

Return to [Summary Table](#).

Figure 18-26. GPTMTAR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR																															
R-0xFFFFFFFF																															

Table 18-29. GPTMTAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAR	R	0xFFFFFFFF F	GPTM Timer A Register. A read returns the current value of the GPTM Timer A Count Register, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

18.5.19 GPTMTBR Register (Offset = 0x4C) [reset = 0xFFFF]

GPTM Timer B (GPTMTBR)

This register shows the current value of the Timer B counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

NOTE: When an alternate clock source is enabled, a read of this register returns the current count -1.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAR register. Reads from this register return the current value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler in Input Edge Count, Input Edge Time, and PWM modes, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the GPTMTBV register. To read the value of the prescaler in periodic snapshot mode, read the Timer B Prescale Snapshot (GPTMTBPS) register.

GPTMTBR is shown in [Figure 18-27](#) and described in [Table 18-30](#).

Return to [Summary Table](#).

Figure 18-27. GPTMTBR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBR																															
R-0xFFFF																															

Table 18-30. GPTMTBR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBR	R	0xFFFF	GPTM Timer B Register. A read returns the current value of the GPTM Timer B Count Register, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

18.5.20 GPTMTAV Register (Offset = 0x50) [reset = 0xFFFFFFFF]

GPTM Timer A Value (GPTMTAV)

When read, this register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry when using the snapshot feature with the periodic operating mode. When written, the value written into this register is loaded into the GPTMTAR register on the next clock cycle.

NOTE: When an alternate clock source is enabled, a read of this register returns the current count -1.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, GPTMTAV appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Value (GPTMTBV) register). In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

GPTMTAV is shown in [Figure 18-28](#) and described in [Table 18-31](#).

Return to [Summary Table](#).

Figure 18-28. GPTMTAV Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAV																															
R/W-0xFFFFFFFF																															

Table 18-31. GPTMTAV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAV	R/W	0xFFFFFFFF F	GPTM Timer A Value. A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the GPTMTAR register on the next clock cycle. In 16-bit mode, only the lower 16-bits of the GPTMTAV register can be written with a new value. Writes to the prescaler bits have no effect.

18.5.21 GPTMTBV Register (Offset = 0x54) [reset = 0xFFFF]

GPTM Timer B Value (GPTMTBV)

When read, this register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the GPTMTBR register on the next clock cycle.

NOTE: When an alternate clock source is enabled, a read of this register returns the current count -1.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAV register. Reads from this register return the current free-running value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

GPTMTBV is shown in [Figure 18-29](#) and described in [Table 18-32](#).

Return to [Summary Table](#).

Figure 18-29. GPTMTBV Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBV																															
R/W-0xFFFF																															

Table 18-32. GPTMTBV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBV	R/W	0xFFFF	GPTM Timer B Value. A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the GPTMTAR register on the next clock cycle. In 16-bit mode, only the lower 16-bits of the GPTMTBV register can be written with a new value. Writes to the prescaler bits have no effect.

18.5.22 GPTMRTCPD Register (Offset = 0x58) [reset = 0x7FFF]

GPTM RTC Predivide (GPTMRTCPD)

This register provides the current RTC predivider value when the timer is operating in RTC mode. Software must perform an atomic access with consecutive reads of the GPTMTAR, GPTMTBR, and GPTMRTCPD registers.

NOTE: When an alternate clock source is enabled, a read of this register returns the current count -1.

GPTMRTCPD is shown in [Figure 18-30](#) and described in [Table 18-33](#).

Return to [Summary Table](#).

Figure 18-30. GPTMRTCPD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RTCPD															
R-0x0																R-0x7FFF															

Table 18-33. GPTMRTCPD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	RTCPD	R	0x7FFF	RTC Predivide Counter Value. The current RTC predivider value when the timer is operating in RTC mode. This field has no meaning in other timer modes.

18.5.23 GPTMTAPS Register (Offset = 0x5C) [reset = 0x0]

GPTM Timer A Prescale Snapshot (GPTMTAPS)

For 16-/32-bit wide GPTM, this register shows the current value of the Timer A prescaler for periodic snapshot mode.

GPTMTAPS is shown in [Figure 18-31](#) and described in [Table 18-34](#).

Return to [Summary Table](#).

Figure 18-31. GPTMTAPS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PSS															
R-0x0																R-0x0															

Table 18-34. GPTMTAPS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	PSS	R	0x0	GPTM Timer A Prescaler Snapshot. A read returns the current value of the GPTM Timer A Prescaler.

18.5.24 GPTMTBPS Register (Offset = 0x60) [reset = 0x0]

GPTM Timer B Prescale Snapshot (GPTMTBPS)

For 16-/32-bit wide GPTM, this register shows the current value of the Timer B prescaler for periodic snapshot mode.

GPTMTBPS is shown in [Figure 18-32](#) and described in [Table 18-35](#).

[Return to Summary Table.](#)

Figure 18-32. GPTMTBPS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PSS															
R-0x0																R-0x0															

Table 18-35. GPTMTBPS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	PSS	R	0x0	GPTM Timer A Prescaler Value. A read returns the current value of the GPTM Timer A Prescaler.

18.5.25 GPTMDMAEV Register (Offset = 0x6C) [reset = 0x0]

GPTM DMA Event (GPTMDMAEV)

This register allows software to enable/disable GPTM DMA trigger events. Setting a bit enables the corresponding DMA trigger, while clearing a bit disables it.

GPTMDMAEV is shown in [Figure 18-33](#) and described in [Table 18-36](#).

Return to [Summary Table](#).

Figure 18-33. GPTMDMAEV Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				TBMDMAEN	CBEDMAEN	CBMDMAEN	TBTODMAEN
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			TAMDMAEN	RTCDMAEN	CAEDMAEN	CAMDMAEN	TATODMAEN
R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 18-36. GPTMDMAEV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	TBMDMAEN	R/W	0x0	GPTM B Mode Match Event DMA Trigger Enable. When this bit is enabled, a Timer B dma_req signal is sent to the μ DMA when a mode match has occurred. 0x0 = Timer B Mode Match DMA trigger is disabled. 0x1 = Timer B DMA Mode Match trigger is enabled.
10	CBEDMAEN	R/W	0x0	GPTM B Capture Event DMA Trigger Enable. When this bit is enabled, a Timer B dma_req signal is sent to the μ DMA when a capture event has occurred. 0x0 = Timer B Capture Event DMA trigger is disabled. 0x1 = Timer B Capture Event DMA trigger is enabled.
9	CBMDMAEN	R/W	0x0	GPTM B Capture Match Event DMA Trigger Enable. When this bit is enabled, a Timer B dma_req signal is sent to the μ DMA when a capture match event has occurred. 0x0 = Timer B Capture Match DMA trigger is disabled. 0x1 = Timer B Capture Match DMA trigger is enabled.
8	TBTODMAEN	R/W	0x0	GPTM B Time-Out Event DMA Trigger Enable. When this bit is enabled, a Timer B dma_req signal is sent to the μ DMA on a time-out event. 0x0 = Timer B Time-Out DMA trigger is disabled. 0x1 = Timer B Time-Out DMA trigger is enabled.
7-5	RESERVED	R	0x0	
4	TAMDMAEN	R/W	0x0	GPTM A Mode Match Event DMA Trigger Enable. When this bit is enabled, a Timer A dma_req signal is sent to the μ DMA when a mode match has occurred. 0x0 = Timer A Mode Match DMA trigger is disabled. 0x1 = Timer A DMA Mode Match trigger is enabled.
3	RTCDMAEN	R/W	0x0	GPTM A RTC Match Event DMA Trigger Enable. When this bit is enabled, a Timer A dma_req signal is sent to the μ DMA when a RTC match has occurred. 0x0 = Timer A RTC Match DMA trigger is disabled. 0x1 = Timer A RTC Match DMA trigger is enabled.

Table 18-36. GPTMDMAEV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	CAEDMAEN	R/W	0x0	GPTM A Capture Event DMA Trigger Enable. When this bit is enabled, a Timer A dma_req signal is sent to the μ DMA when a capture event has occurred. 0x0 = Timer A Capture Event DMA trigger is disabled. 0x1 = Timer A Capture Event DMA trigger is enabled.
1	CAMDMAEN	R/W	0x0	GPTM A Capture Match Event DMA Trigger Enable. When this bit is enabled, a Timer A dma_req signal is sent to the μ DMA when a capture match event has occurred. 0x0 = Timer A Capture Match DMA trigger is disabled. 0x1 = Timer A Capture Match DMA trigger is enabled.
0	TATODMAEN	R/W	0x0	GPTM A Time-Out Event DMA Trigger Enable. When this bit is enabled, a Timer A dma_req signal is sent to the μ DMA on a time-out event. 0x0 = Timer A Time-Out DMA trigger is disabled. 0x1 = Timer A Time-Out DMA trigger is enabled.

18.5.26 GPTMADCEV Register (Offset = 0x70) [reset = 0x0]

GPTM ADC Event (GPTMADCEV)

This register allows software to enable/disable GPTM ADC trigger events. Setting a bit enables the corresponding ADC trigger, while clearing a bit disables it.

GPTMADCEV is shown in [Figure 18-34](#) and described in [Table 18-37](#).

Return to [Summary Table](#).

Figure 18-34. GPTMADCEV Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				TBMADCEN	CBEADCEN	CBMADCEN	TBTOADCEN
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED			TAMADCEN	RTCADCEV	CAEADCEN	CAMADCEN	TATOADCEN
R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 18-37. GPTMADCEV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	TBMADCEN	R/W	0x0	GPTM B Mode Match Event ADC Trigger Enable. When this bit is enabled, a trigger pulse is sent to the ADC when a mode match has occurred. 0x0 = Timer B Mode Match ADC trigger is disabled. 0x1 = Timer B Mode Match ADC trigger is enabled.
10	CBEADCEN	R/W	0x0	GPTM B Capture Event ADC Trigger Enable. When this bit is enabled, a trigger pulse is sent to the ADC when a capture event has occurred. 0x0 = Timer B Capture Event ADC trigger is disabled. 0x1 = Timer B Capture Event ADC trigger is enabled.
9	CBMADCEN	R/W	0x0	GPTM B Capture Match Event ADC Trigger Enable. When this bit is enabled, a trigger signal is sent to the ADC when a capture match event has occurred. 0x0 = Timer B Capture Match ADC trigger is disabled. 0x1 = Timer B Capture Match ADC trigger is enabled.
8	TBTOADCEN	R/W	0x0	GPTM B Time-Out Event ADC Trigger Enable. When this bit is enabled, a trigger signal is sent to the ADC on a time-out event. 0x0 = Timer B Time-Out ADC trigger is disabled. 0x1 = Timer B Time-Out ADC trigger is enabled.
7-5	RESERVED	R	0x0	
4	TAMADCEN	R/W	0x0	GPTM A Mode Match Event ADC Trigger Enable. When this bit is enabled, a trigger pulse is sent to the ADC when a mode match has occurred. 0x0 = Timer A Mode Match ADC trigger is disabled. 0x1 = Timer A Mode Match ADC trigger is enabled.
3	RTCADCEV	R/W	0x0	GPTM RTC Match Event ADC Trigger Enable. When this bit is enabled, a trigger signal is sent to the ADC when a RTC match has occurred. 0x0 = Timer A RTC Match ADC trigger is disabled. 0x1 = Timer A RTC Match ADC trigger is enabled.

Table 18-37. GPTMADCEV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	CAEADCEN	R/W	0x0	GPTM A Capture Event ADC Trigger Enable. When this bit is enabled, a trigger pulse is sent to the ADC when a capture event has occurred. 0x0 = Timer A Capture Event ADC trigger is disabled. 0x1 = Timer A Capture Event ADC trigger is enabled.
1	CAMADCEN	R/W	0x0	GPTM A Capture Match Event ADC Trigger Enable. When this bit is enabled, a trigger signal is sent to the ADC when a capture match event has occurred. 0x0 = Timer A Capture Match ADC trigger is disabled. 0x1 = Timer A Capture Match ADC trigger is enabled.
0	TATOADCEN	R/W	0x0	GPTM A Time-Out Event ADC Trigger Enable. When this bit is enabled, a trigger signal is sent to the ADC on a time-out event. 0x0 = Timer A Time-Out Event ADC trigger is disabled. 0x1 = Timer A Time-Out Event ADC trigger is enabled.

18.5.27 GPTMPP Register (Offset = 0xFC0) [reset = 0x70]

GPTM Peripheral Properties (GPTMPP)

The GPTMPP register provides information regarding the properties of the General-Purpose Timer module.

GPTMPP is shown in [Figure 18-35](#) and described in [Table 18-38](#).

Return to [Summary Table](#).

Figure 18-35. GPTMPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	ALTCLK	SYNCCNT	CHAIN	SIZE			
R-0x0	R-0x1	R-0x1	R-0x1	R-0x0			

Table 18-38. GPTMPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6	ALTCLK	R	0x1	Alternate Clock Source 0x0 = The alternate clock source (ALTCLK) is not available to the Timer module. 0x1 = The alternate clock source (ALTCLK) is available to the Timer module.
5	SYNCCNT	R	0x1	Synchronize Start 0x0 = Timer is not capable of synchronizing the counter value with other GPTimers. 0x1 = Timer is capable of synchronizing the counter value with other Timers.
4	CHAIN	R	0x1	Chain with Other Timers. Note that although this bit is set for Timer 0A, this timer cannot chain because there is not a previously numbered Timer. 0x0 = Timer is not capable of chaining with the previously numbered Timer. 0x1 = Timer is capable of chaining with the previously numbered Timer.
3-0	SIZE	R	0x0	Count Size 0x0 = Timer A and Timer B counters are 16 bits each with an 8-bit prescale counter. 0x1 = Timer A and Timer B counters are 32 bits each with a 16-bit prescale counter.

18.5.28 GPTMCC Register (Offset = 0xFC8) [reset = 0x0]

GPTM Clock Configuration (GPTMCC)

The GPTMCC register controls the clock source for the General-Purpose Timer module.

NOTE: When the ALTCLK bit is set in the GPTMCC register to enable using the alternate clock source, the synchronization imposes restrictions on the starting count value (down-count), terminal value (up-count) and the match value. This restriction applies to all modes of operation. Each event must be spaced by 4 Timer (ALTCLK) clock periods + 2 system clock periods. If some events do not meet this requirement, then it is possible that the timer block may need to be reset for correct functionality to be restored.

Example: ALTCLK= $T_{PIOSC} = 62.5 \text{ ns}$ (16 MHz trimmed)

$T_{hclk} = 1 \mu\text{s}$ (1 MHz)

$4 \times 62.5 \text{ ns} + 2 \times 1 \mu\text{s} = 2.25 \mu\text{s}$ $2.25 \mu\text{s} / 62.5 \text{ ns} = 36$ (or 0x23)

The minimum values for the periodic or one-shot with a match interrupt enabled are:

GPTMTAMATCHR = 0x23 GPTMTAILR = 0x46

GPTMCC is shown in [Figure 18-36](#) and described in [Table 18-39](#).

Return to [Summary Table](#).

Figure 18-36. GPTMCC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							ALTCLK
R-0x0							R/W-0x0

Table 18-39. GPTMCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	ALTCLK	R/W	0x0	Alternate Clock Source 0x0 = System clock (based on clock source and divisor factor programmed in RSCLKCFG register in the System Control Module) 0x1 = Alternate clock source as defined by ALTCLKCFG register in System Control Module.

Inter-Integrated Circuit (I²C) Interface

The Inter-Integrated Circuit (I²C) bus provides bidirectional data transfer through a two-wire design (a serial data line [SDA] and a serial clock line [SCL]), and interfaces to external I²C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I²C bus may also be used for system testing and diagnostic purposes in product development and manufacturing.

Topic	Page
19.1 Introduction	1314
19.2 Block Diagram	1315
19.3 Functional Description	1315
19.4 Initialization and Configuration	1333
19.5 I2C Registers	1335

19.1 Introduction

The I²C modules support the following features:

- Devices on the I²C bus can be designated as either a master or a slave.
 - Supports both transmitting and receiving data as either a master or a slave
 - Supports simultaneous master and slave operation
- Four I²C modes:
 - Master transmit
 - Master receive
 - Slave transmit
 - Slave receive
- Two 8-entry FIFOs for receive and transmit data
 - FIFOs can be independently assigned to master or slave
- Four transmission speeds:
 - Standard (100 kbps)
 - Fast mode (400 kbps)
 - Fast-mode plus (1 Mbps)
 - High-speed mode (3.33 Mbps)
- Glitch suppression
- SMBus support through software
 - Clock low time-out interrupt
 - Dual slave address capability
 - Quick command capability
- Master and slave interrupt generation
 - Master generates interrupts when a transmit or receive operation completes (or aborts because of an error)
 - Slave generates interrupts when data has been transferred or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multiple-master support, and 7-bit addressing mode
- Efficient transfers using a Micro Direct Memory Access Controller (μDMA)
 - Separate channels for transmit and receive
 - Ability to execute single data transfers or burst data transfers using the RX and TX FIFOs in the I²C

19.2 Block Diagram

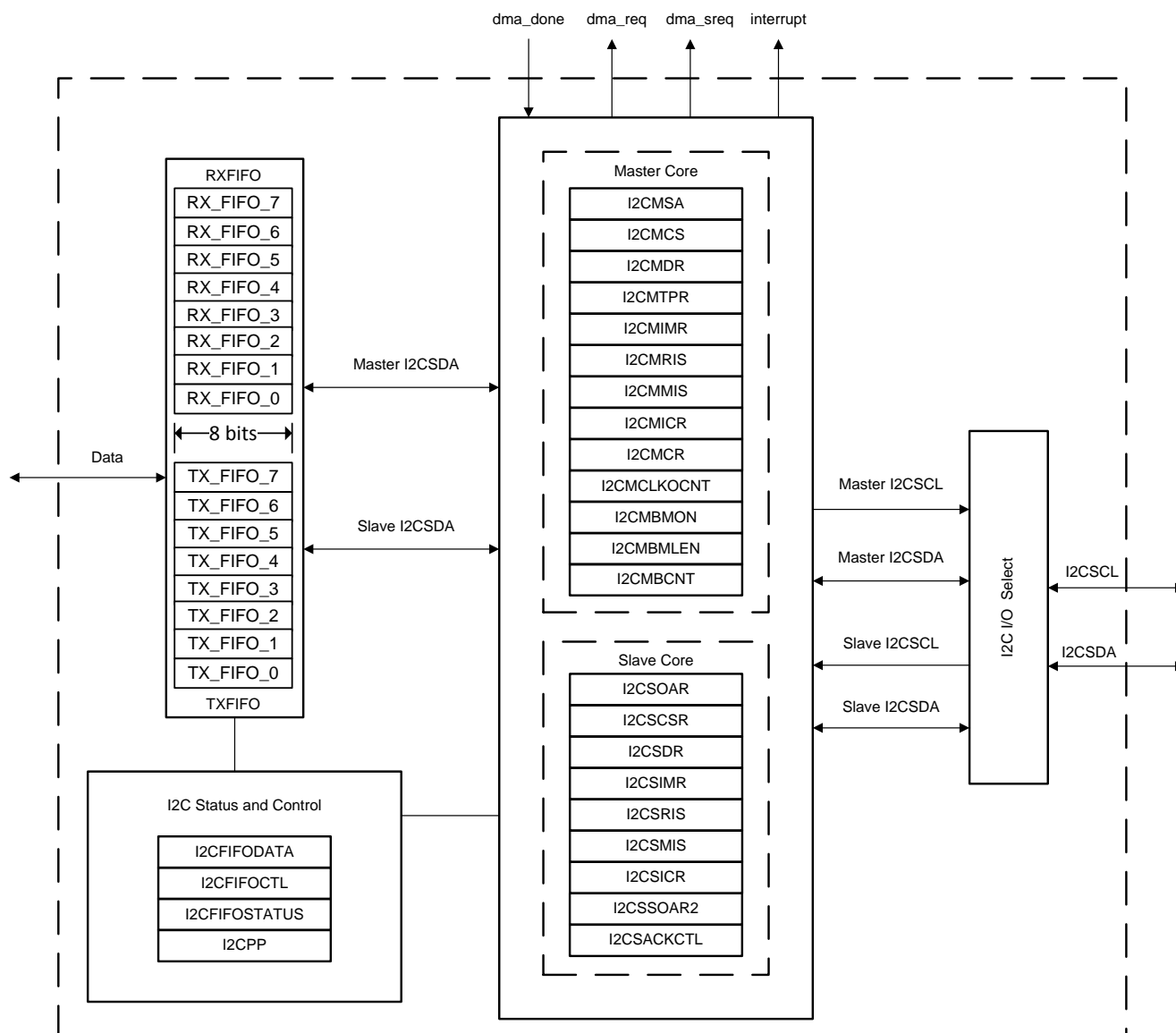


Figure 19-1. I²C Block Diagram

19.3 Functional Description

Each I²C module is comprised of both master and slave functions and is identified by a unique address. A master-initiated communication generates the clock signal, SCL. For proper operation, the SDA pin must be configured as an open-drain signal. Due to the internal circuitry that supports high-speed operation, the SCL pin must not be configured as an open-drain signal, although the internal circuitry causes it to act as if it were an open-drain signal. Both SDA and SCL signals must be connected to a positive supply voltage using a pullup resistor. Figure 19-2 shows a typical I²C bus configuration. See the I²C bus specification and user manual to determine the size of the pullups required for proper operation.

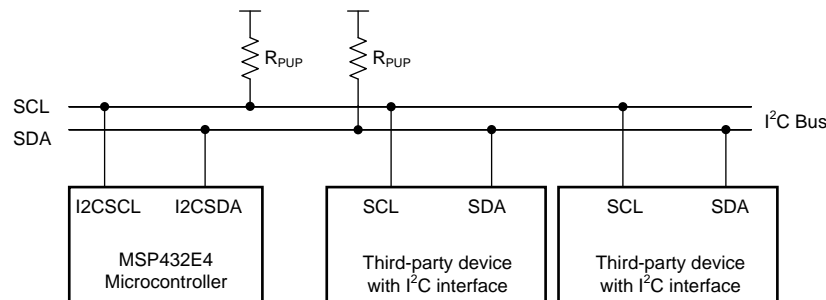


Figure 19-2. I²C Bus Configuration

19.3.1 I²C Bus Functional Overview

The I²C bus uses only two signals: SDA and SCL (named I2CSDA and I2CSCL on MSP432E4 microcontrollers). SDA is the bidirectional SDA and SCL is the bidirectional serial clock line. The bus is considered idle when both lines are high.

Every transaction on the I²C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in [Section 19.3.1.1](#)) is unrestricted, but each data byte must be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

19.3.1.1 START and STOP Conditions

The protocol of the I²C bus defines two states to begin and end a transaction: START and STOP. A high-to-low transition on the SDA line while the SCL is high is defined as a START condition, and a low-to-high transition on the SDA line while SCL is high is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See [Figure 19-3](#).

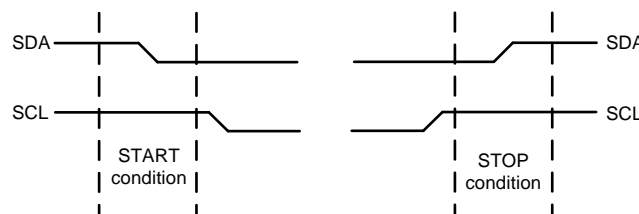


Figure 19-3. START and STOP Conditions

The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the I²C Master Slave Address (I2CMSA) register is written with the desired address, the R/S bit is cleared, and the Control register is written with ACK = X (0 or 1), STOP = 1, START = 1, and RUN = 1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the I²C Master Data (I2CMDR) register. When the I²C module operates in master receiver mode, the ACK bit is normally set causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

When operating in slave mode, the STARTRIS and STOPRIS bits in the I²C Slave Raw Interrupt Status (I2CSRIS) register indicate detection of start and stop conditions on the bus and the I²C Slave Masked Interrupt Status (I2CSMIS) register can be configured to allow STARTRIS and STOPRIS to be promoted to controller interrupts (when interrupts are enabled).

19.3.1.2 Data Format With 7-Bit Address

Data transfers follow the format in [Figure 19-4](#). After the START condition, a slave address is transmitted. This address is 7 bits long followed by an eighth bit, which is a data direction bit (R/S bit in the I2CMSA register). If the R/S bit is clear, it indicates a transmit operation (send), and if it is set, it indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive and transmit formats are then possible within a single transfer.

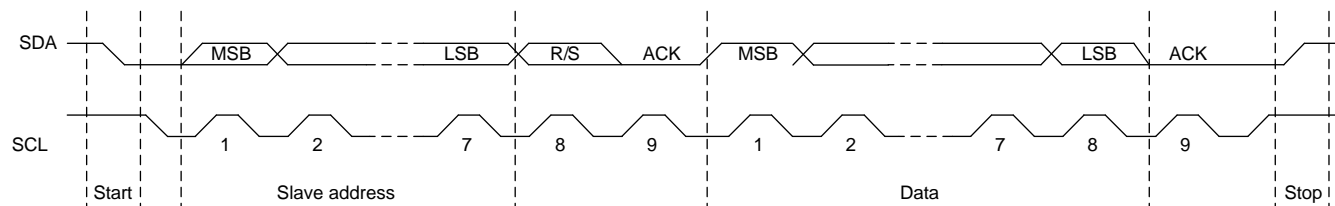


Figure 19-4. Complete Data Transfer With a 7-Bit Address

The first seven bits of the first byte make up the slave address (see [Figure 19-5](#)). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

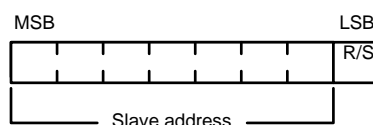


Figure 19-5. R/S Bit in First Byte

19.3.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can change only when SCL is low (see [Figure 19-6](#)).

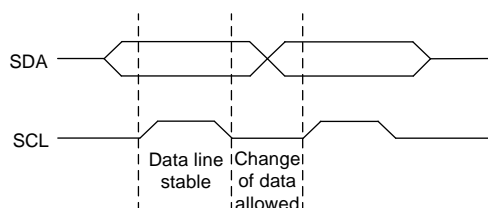


Figure 19-6. Data Validity During Bit Transfer on the I²C Bus

19.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted out by the receiver during the acknowledge cycle must comply with the data validity requirements described in [Section 19.3.1.3](#).

When a slave receiver does not acknowledge the slave address, SDA must be left high by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

If the slave is required to provide a manual ACK or NACK, the I²C Slave ACK Control (I2CSACKCTL) register allows the slave to NACK for invalid data or command or ACK for valid data or command. When this operation is enabled, the MCU slave module I²C clock is pulled low after the last data bit until this register is written with the indicated response.

19.3.1.5 Repeated START

The I²C master module can execute a repeated START (transmit or receive) after an initial transfer has occurred.

NOTE: When reading the I2CMCS register to check the BUSY bit, also read the ADRACK and DATAACK bits, because these are cleared on register read, and status may be lost if they are not checked on every read of the register.

Alternatively, the NACKRIS bit of the I2CMRIS register can be used to monitor NACK status.

For more information on repeated START, see [Figure 19-12](#) and [Figure 19-13](#).

19.3.1.5.1 Repeated Start For Master Transmit

A repeated start sequence for a master transmit is as follows:

1. When the device is in the idle state, the master writes the slave address to the I2CMSA register and configures the R/S bit for the desired transfer type.
2. Data is written to the I2CMDR register.
3. When the BUSY bit in the I2CMCS register is 0, the master writes 0x3 to the I2CMCS register to initiate a transfer.
4. The master does not generate a STOP condition but instead writes another slave address to the I2CMSA register and then writes 0x3 to initiate the repeated START.

19.3.1.5.2 Repeated Start For Master Receive

A repeated start sequence for a master receive is similar:

1. When the device is in idle, the master writes the slave address to the I2CMSA register and configures the R/S bit for the desired transfer type.
2. The master reads data from the I2CMDR register.
3. When the BUSY bit in the I2CMCS register is 0, the master writes 0x3 to the I2CMCS register to initiate a transfer.
4. The master does not generate a STOP condition but instead writes another slave address to the I2CMSA register and then writes 0x3 to initiate the repeated START.

19.3.1.6 Clock Low Time-out (CLTO)

The I²C slave can extend the transaction by pulling the clock low periodically to create a slow bit transfer rate. The I²C module has a 12-bit programmable counter that is used to track how long the clock has been held low. The upper 8 bits of the count value are software programmable through the I²C Master Clock Low Time-out Count (I2CMCLKOCNT) register. The lower four bits are not user visible and are 0x0. The CNTL value programmed in the I2CMCLKOCNT register has to be greater than 0x01. The application can program the eight most significant bits of the counter to reflect the acceptable cumulative low period in transaction. The count is loaded at the START condition and counts down on each falling edge of the internal bus clock of the master. Note that the internal bus clock generated for this counter keeps running at the programmed I²C speed even if SCL is held low on the bus. Upon reaching terminal count, the master state machine forces ABORT on the bus by issuing a STOP condition at the instance of SCL and SDA release.

As an example, if an I²C module was operating at 100-kHz speed, programming the I2CMCLKOCNT register to 0xDA would translate to the value 0xDA0 since the lower four bits are set to 0x0. This would translate to a decimal value of 3488 clocks or a cumulative clock low period of 34.88 ms at 100 kHz.

The CLKRIS bit in the I²C Master Raw Interrupt Status (I2CMRIS) register is set when the clock time-out period is reached, allowing the master to start corrective action to resolve the remote slave state. In addition, the CLKTO bit in the I²C Master Control/Status (I2CMCS) register is set; this bit is cleared when a STOP condition is sent or during the I²C master reset. The status of the raw SDA and SCL signals are readable by software through the SDA and SCL bits in the I²C Master Bus Monitor (I2CMBMON) register to help determine the state of the remote slave.

In the event of a CLTO condition, application software must choose how it intends to attempt bus recovery. Most applications may attempt to manually toggle the I²C pins to force the slave to let go of the clock signal (a common solution is to attempt to force a STOP on the bus). If a CLTO is detected before the end of a burst transfer, and the bus is successfully recovered by the master, the master hardware attempts to finish the pending burst operation. The behavior of the bus varies depending on the state of the slave after bus recovery. If the slave resumes in a state where it can acknowledge the master (essentially, where it was before the bus hang), it continues where it left off. However, if the slave resumes in a reset state (or if a forced STOP by the master causes the slave to enter the idle state), it may ignore the master's attempt to complete the burst operation and NAK the first data byte that the master sends or requests.

Since the behavior of slaves cannot always be predicted, it is suggested that the application software always write the STOP bit in the I²C Master Configuration (I2CMCR) register during the CLTO interrupt service routine. This limits the amount of data the master attempts to send or receive upon bus recovery to a single byte, and after the single byte is on the wire, the master issues a STOP. An alternative solution is to have the application software reset the I²C peripheral before attempting to manually recover the bus. This solution allows the I²C master hardware to be returned to a known good (and idle) state before attempting to recover a stuck bus and prevents any unwanted data from appearing on the wire.

NOTE: The Master Clock Low Time-out counter counts for the entire time SCL is held low continuously. If SCL is deasserted at any point, the Master Clock Low Time-out Counter is reloaded with the value in the I2CMCLKOCNT register and begins counting down from this value.

19.3.1.7 Dual Address

The I²C interface supports dual address capability for the slave. The additional programmable address is provided and can be matched if enabled. In legacy mode with dual address disabled, the I²C slave provides an ACK on the bus if the address matches the OAR field in the I2CSOAR register. In dual address mode, the I²C slave provides an ACK on the bus if either the OAR field in the I2CSOAR register or the OAR2 field in the I2CSOAR2 register is matched. The enable for dual address is programmable through the OAR2EN bit in the I2CSOAR2 register and there is no disable on the legacy address.

The OAR2SEL bit in the I2CSCSR register indicates if the address that was ACKed is the alternate address or not. When this bit is clear, it indicates either legacy operation or no address match.

19.3.1.8 Arbitration

A master may start a transfer only if the bus is idle. It is possible for two or more masters to generate a START condition within the minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place a 1 (high) on SDA, while another master transmits a 0 (low), switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

If arbitration is lost when the I²C master is initiating a BURST with the TX FIFO enabled, the application should execute the following steps to correctly handle the arbitration loss:

1. Flush and disable the TX FIFO
2. Clear and mask the TXFE interrupt by clearing the TXFEIM bit in the I2CMIMR register.

Once the bus is IDLE, the TXFIFO can be filled and enabled, the TXFE bit can be unmasked and a new BURST transaction can be initiated.

19.3.1.9 Glitch Suppression in Multi-Master Configuration

When a multi-master configuration is being used, the PULSESEL bit in the I2CMTPR register can be programmed to provide glitch suppression on the SCL and SDA lines and assure proper signal values. The glitch suppression value is in terms of buffered system clocks. Note that all signals will be delayed internally when glitch suppression is nonzero. For example, if PULSESEL is set to 0x7, 31 clocks should be added onto the calculation for the expected transaction time.

19.3.1.10 SMBus Operation

The SMBus interface is based on the I²C protocol; however, some differences exist between the two. These differences must be handled through software in order to make sure the SMBus protocol, including timing specifications, is met. Note that the SMBus 2.0 specification limits the maximum frequency of the interface to 100 KHz; as a result, I²C standard speed operation is used for SMBus.

The SMBus and I²C slave can extend the transaction if it is not ready by pulling the clock low. The SMBus specification allows the maximum time-out for such elongated transaction to be 25 to 35 ms. The I²C specification does not have this requirement. The I²C module supports a programmable count to support clock-low time-out for the master to error out and take action as required; this feature is explained in [Section 19.3.1.6](#). Note that if transactions are extended, a time-out period should be programmed in the I2CMCLKCNT register, and the CLKRIS bit in the I2CMRIS register should not be masked.

Unlike the I²C slave, the SMBus slave must respond with an ACK response to its address regardless of whether it is ready or not. As a result, the I²C slave sends an ACK response to its address and a NACK response on the data byte if it is not ready. The ARBLST bit in the I2CMCS register is set if there were any issues with the transfer. In addition, the slave can send a NACK at any time to force the master to stop sending additional bytes.

The I²C interface supports μ DMA for efficient data handling. The μ DMA operation needs FIFOs to be enabled for appropriate transfer type to perform I²C master for burst transfers and all types of slave transfers. The I²C interface is supported by two channels: one for Rx (I²C-to-Memory) and one for Tx (Memory-to-I²C) transfers. See [Section 19.3.5](#) for more information.

19.3.1.10.1 Quick Command

Quick command is a simple, compact SMBus protocol that sends an address and one bit of data in the R/S bit of the I²C header byte to communicate a command to the slave, typically a turnoff or turnon. The I²C master peripheral can send a quick command by writing the target address and R/S value into the I2CMSA register followed by a write to I2CMCS with a value of 0x27. SMBus requires the slave to be able to accept and process commands and the master to generate the quick command transactions. The master also has the capability to stop the transaction after acknowledgement from a slave.

The I²C slave peripheral requires special handling when a quick command is sent. In the case where a master sends a quick command with the R/S (data) bit cleared, the QCMDST bit in I2CSCSR is set, and the QCMDRW bit shows the data value (which, in this case, is 0) when the STOPRIS bit is set in I2CSRIS and the STOP interrupt is asserted. In this scenario, a DATARIS interrupt bit is not set. When the master sends a quick command with the R/S (data) bit set, the DATARIS bit is set to notify the slave to write a data byte to I2CSDR in which bit 7 is set. A dummy writeM of 0xFF to the I2CSDR register is recommended. After the write to I2CSDR, the STOP interrupt is asserted and the QCMDST and QCMDRW bits are set in the I2CSCSR register to indicate that a quick command read occurred and the last transaction was a quick command. Therefore, when the slave must receive a quick command, it expects such a command because it must write the I2CSDR with a specific value when R/S is set.

19.3.2 Available Speed Modes

The I²C bus can run in standard mode (100 kbps), fast mode (400 kbps), fast mode plus (1 Mbps) or high-speed mode (3.4 Mbps, if the correct system clock frequency is set and there is appropriate pull strength on SCL and SDA). The selected mode should match the speed of the other I²C devices on the bus.

19.3.2.1 Standard, Fast, and Fast Plus Modes

Standard, fast, and fast plus modes are selected using a value in the I²C Master Timer Period (I2CMTPR) register that results in an SCL frequency of 100 kbps for standard mode, 400 kbps for fast mode, or 1 Mbps for fast mode plus.

The I²C clock rate is determined by the parameters CLK_PRD, TIMER_PRD, SCL_LP, and SCL_HP where:

- CLK_PRD is the system clock period
- SCL_LP is the low phase of SCL (fixed at 6)
- SCL_HP is the high phase of SCL (fixed at 4)

TIMER_PRD is the programmed value in the I2CMTPR register (see [Section 19.5.4](#)). This value is determined by replacing the known variables in [Equation 61](#) and solving for TIMER_PRD. The I²C clock period is calculated as follows:

$$\text{SCL_PERIOD} = 2 \times (1 + \text{TIMER_PRD}) \times (\text{SCL_LP} + \text{SCL_HP}) \times \text{CLK_PRD} \quad (61)$$

For example:

- CLK_PRD = 50 ns
- TIMER_PRD = 2
- SCL_LP = 6
- SCL_HP = 4

Yields an SCL frequency of: $1 / \text{SCL_PERIOD} = 338 \text{ KHz}$

[Table 19-1](#) lists examples of the timer periods that should be used to generate standard, fast mode, and fast mode plus SCL frequencies based on various system clock frequencies.

Table 19-1. Examples of I²C Master Timer Period Versus Speed Mode

System Clock	Standard Mode		Fast Mode		Fast Mode Plus	
	Timer Period	Data Rate	Timer Period	Data Rate	Timer Period	Data Rate
4 MHz	0x01	100 kbps	—	—	—	—
6 MHz	0x02	100 kbps	—	—	—	—
12.5 MHz	0x06	89 kbps	0x01	312 kbps	—	—
16.7 MHz	0x08	93 kbps	0x02	278 kbps	—	—
20 MHz	0x09	100 kbps	0x02	333 kbps	—	—
25 MHz	0x0C	96.2 kbps	0x03	312 kbps	—	—
33 MHz	0x10	97.1 kbps	0x04	330 kbps	—	—
40 MHz	0x13	100 kbps	0x04	400 kbps	0x01	1000 kbps
50 MHz	0x18	100 kbps	0x06	357 kbps	0x02	833 kbps
80 MHz	0x27	100 kbps	0x09	400 kbps	0x03	1000 kbps
100 MHz	0x31	100 kbps	0x0C	385 kbps	0x04	1000 kbps
120 MHz	0x3B	100 kbps	0xE	400 kbps	0x5	1000 kbps

19.3.2.2 High-Speed Mode

The I²C peripheral has support for high-speed operation as both a master and slave. High-speed mode is configured by setting the HS bit in the I²C Master Control/Status (I2CMCS) register. High-speed mode transmits data at a high bit rate with a 66.6%/33.3% duty cycle, but communication and arbitration are done at standard, fast mode, or fast-mode plus speed, depending on which is selected by the user. When the HS bit in the I2CMCS register is set, current mode pullups are enabled.

The clock period can be selected using Equation 62, but in this case, SCL_LP = 2 and SCL_HP = 1.

$$\text{SCL_PERIOD} = 2 \times (1 + \text{TIMER_PRD}) \times (\text{SCL_LP} + \text{SCL_HP}) \times \text{CLK_PRD} \quad (62)$$

For example:

- CLK_PRD = 25 ns
- TIMER_PRD = 1
- SCL_LP = 2
- SCL_HP = 1

Yields a SCL frequency of: $1 / T = 3.33 \text{ MHz}$

Table 19-2 gives examples of timer period and system clock in high-speed mode. Note that the HS bit in the I2CMTPR register must be set for the TPR value to be used in high-speed mode.

Table 19-2. Examples of I²C Master Timer Period in High-Speed Mode

System Clock	Timer Period	Transmission Mode
40 MHz	0x01	3.33 Mbps
50 MHz	0x02	2.77 Mbps
80 MHz	0x03	3.33 Mbps

When operating as a master, the protocol is shown in Figure 19-7. The master is responsible for sending a master code byte in either standard (100 kbps) or fast mode (400 kbps) before it begins transferring in high-speed mode. The master code byte must contain data in the form of 0000.1XXX and is used to tell the slave devices to prepare for a high-speed transfer. The master code byte should never be acknowledged by a slave since it is only used to indicate that the upcoming data is going to be transferred at a higher data rate. To send the master code byte for a standard high-speed transfer, software should place the value of the master code byte into the I2CMSA register and write the I2CMCS register with a value of 0x13. If a high-speed burst transfer is required, then to send the master code byte, software should place the value of the master code byte into the I2CMSA register and write the I2CMCS register with 0x50. Either configuration places the I²C master peripheral in high-speed mode, and all subsequent transfers (until STOP) are carried out at high-speed data rate using the normal I2CMCS command bits, without setting the HS bit in the I2CMCS register. Again, setting the HS bit in the I2CMCS register is only necessary during the master code byte.

When operating as a high-speed slave, no additional software is required.

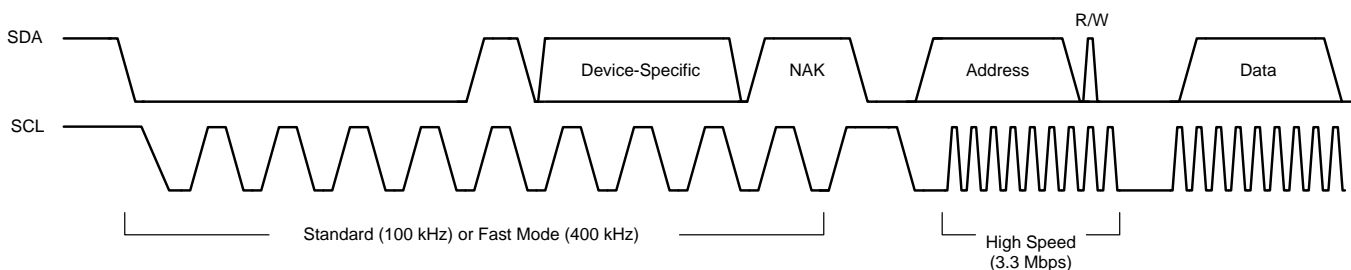


Figure 19-7. High-Speed Data Format

NOTE: High-Speed mode is 3.4 Mbps if the correct system clock frequency is set and there is appropriate pull strength on SCL and SDA lines.

19.3.3 Interrupts

The I²C can generate interrupts when the following conditions are observed in the master module:

- Master transaction completed (RIS bit)
- Master arbitration lost (ARBLOSTRIS bit)
- Master Address/Data NACK (NACKRIS bit)
- Master bus time-out (CLKRIS bit)
- Next byte request (RIS bit)
- Stop condition on bus detected (STOPRIS bit)
- Start condition on bus detected (STARTRIS bit)
- RX DMA interrupt pending (DMARXRIS bit)
- TX DMA interrupt pending (DMATXRIS bit)
- Trigger value for FIFO has been reached and a TX FIFO request interrupt is pending (TXRIS bit)
- Trigger value for FIFO has been reached and a RX FIFO request interrupt is pending (RXRIS bit)
- Transmit FIFO is empty (TXFERIS bit)
- Receive FIFO is full (RXFFRIS bit)

Interrupts are generated when the following conditions are observed in the slave module:

- Slave transaction received (DATARIS bit)
- Slave transaction requested (DATARIS bit)
- Slave next byte transfer request (DATARIS bit)
- Stop condition on bus detected (STOPRIS bit)
- Start condition on bus detected (STARTRIS bit)
- RX DMA interrupt pending (DMARXRIS bit)
- TX DMA interrupt pending (DMATXRIS bit)
- Programmable trigger value for FIFO has been reached and a TX FIFO request interrupt is pending (TXRIS bit)
- Programmable trigger value for FIFO has been reached and a RX FIFO request interrupt is pending (RXRIS bit)
- Transmit FIFO is empty (TXFERIS bit)
- Receive FIFO is full (RXFFRIS bit)

The I²C master and slave modules have separate interrupt registers. Interrupts can be masked by clearing the appropriate bit in the I2CMIMR or I2CSIMR register. Note that the RIS bit in the Master Raw Interrupt Status (I2CMRIS) register and the DATARIS bit in the Slave Raw Interrupt Status (I2CSRIS) register have multiple interrupt causes, including a next byte transfer request interrupt. This interrupt is generated when the master and slave are requesting a receive or transmit transaction.

19.3.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the I²C Master Configuration (I2CMCR) register. In loopback mode, the SDA and SCL signals from the master and are tied to the SDA and SCL signals of the slave module to allow internal testing of the device without having to go through I/O.

19.3.5 FIFO and μ DMA Operation

Both the master and the slave module have the capability to access two 8-byte FIFOs that can be used in conjunction with the μ DMA for fast transfer of data. The transmit (TX) FIFO and receive (RX) FIFO can be independently assigned to either the I²C master or I²C slave. Thus, the following FIFO assignments are allowed:

- The transmit and receive FIFOs can be assigned to the master
- The transmit and receive FIFOs can be assigned to the slave
- The transmit FIFO can be assigned to the master, while the receive FIFO is assigned to the slave and vice versa

In most cases, both FIFOs will be assigned to either the master or the slave. The FIFO assignment is configured by programming the TXASGNMT and RXASGNMT bit in the I²C FIFO Control (I2CFIFOCTL) register.

Each FIFO has a programmable threshold point which indicates when the FIFO service interrupt should be generated. Additionally, a FIFO receive full and transmit empty interrupt can be enabled in the Interrupt Mask (I2CxIMR) registers of both the master and slave. Note that if we clear the TXFERIS interrupt (by setting the TXFEIC bit) when the TX FIFO is empty, the TXFERIS interrupt does not reassert even though the TX FIFO remains empty in this situation.

When a FIFO is not assigned to a master or a slave module, the FIFO interrupt and status signals to the module are forced to a state that indicates the FIFO is empty. For example, if the TX FIFO is assigned to the master module, the status signals to the slave transmit interface indicates that the FIFO is empty.

NOTE: The FIFOs must be empty when reassigning the FIFOs for proper functionality

19.3.5.1 Master Module Burst Mode

A BURST command is provided for the master module which allows a sequence of data transfers using the μ DMA (or software, if desired) to handle the data in the FIFO. The BURST command is enabled by setting the BURST bit in the Master Control/Status (I2CMCS) register. The number of bytes transferred by a BURST request is programmed in the I²C Master Burst Length (I2CMBLEN) register and a copy of this value is automatically written to the I²C Master Burst Count (I2CMBCNT) register to be used as a down-counter during the BURST transfer. The bytes written to the I²C FIFO Data (I2CFIFODATA) register are transferred to the RX FIFO or TX FIFO depending on whether a transmit or receive is being executed. If data is NACKed during a BURST and the STOP bit is set in the I2CMCS register, the transfer terminates. If the STOP bit is not set, the software application must issue a repeated STOP or START when a NACK interrupt is asserted. In the case of a NACK, the I2CMBCNT register can be used to determine the amount of data that was transferred prior to the BURST termination. If the Address is NACKed during a transfer, then a STOP is issued.

19.3.5.1.1 Master Module μ DMA Functionality

When the Master Control/Status (I2CMCS) register is set to enable BURST and the master I²C μ DMA channel is enabled in the DMA Channel Map Select n (DMACHMAPn) registers in the μ DMA, the master control module will assert either the internal single μ DMA request signal (dma_sreq) or multiple μ DMA request signal (dma_req) to the μ DMA. Note that there are separate dma_req and dma_sreq signals for transmit and receive. A single μ DMA request (dma_sreq) will be asserted by the master module when the Rx FIFO has at least one data byte present in the FIFO and/or when the Tx FIFO has at least one space available to fill. The dma_req (or Burst) signal will be asserted when Rx FIFO fill level is higher than trigger level and/or the Tx FIFO burst length remaining is less than 4 bytes and the FIFO fill level is less than trigger level. If a single transfer or BURST operation has completed, the μ DMA sends a dma_done signal to the master module represented by the DMATX and DMARX interrupts in the I2CMIMR, I2CMRIS, I2CMMIS, and I2CMICR registers.

If the μ DMA I²C channel is disabled and software is used to handle the BURST command, software can read the FIFO Status (I2CFIFOSTAT) Register and the Master Burst Count (I2CMBC) register to determine whether the FIFO needs servicing during the BURST transaction. A trigger value can be programmed in the I2CFIFOCTL register to allow for interrupts at various fill levels of the FIFOs.

The NACK and ARBLOST bits in the interrupt status registers can be enabled to indicate no acknowledgement of data transfer or an arbitration loss on the bus.

When the master module is transmitting FIFO data, software can fill the Tx FIFO in advance of setting the BURST bit in the I2CMCS register. If the FIFO is empty when the μ DMA is enabled for BURST mode, the dma_req and dma_sreq both assert (assuming the I2CMBLEN register is programmed to at least four bytes and the Tx FIFO fill level is less than the trigger set). If the I2CMBLEN register value is less than four and the Tx FIFO is not full but more than trigger level, only dma_sreq asserts. Single requests will be generated as required to keep the FIFO full until the number of bytes specified in the I2CMBLEN register has been transferred to the FIFO (and the I2CMBCOUNT register reaches 0x0). At this point, no further requests are generated until the next BURST command is issued. If the μ DMA is disabled, FIFOs will be serviced based on the interrupts active in the master interrupt status registers, the FIFO trigger values shown in the I2CFIFOSTATUS register and completion of a BURST transfer.

When the master module is receiving FIFO data, the Rx FIFO is initially empty and no requests are asserted. If data is read from the slave and placed into the Rx FIFO, the dma_sreq signal to the μ DMA is asserted to indicate there is data to be transferred. If the Rx FIFO contains at least 4 bytes, the dma_req signal is also asserted. The μ DMA will continue to transfer data out of the Rx FIFO until it has reached the amount of bytes programmed in the I2CMBLEN register.

NOTE: The TXFEIM interrupt mask bit in the I2CMIMR register should be clear (masking the TXFE interrupt) when the master is performing an RX Burst from the RXFIFO and should be unmasked before starting a TX FIFO transfers.

19.3.5.2 Slave Module

The slave module also has the capability to use the μ DMA in Rx and Tx FIFO data transfers. If the Tx FIFO is assigned to the slave module and the TXFIFO bit is set in the I2CSCSR register, the slave module will generate a single μ DMA request, dma_sreq, if the master module requests the next byte transfer. If the FIFO fill level is less than the trigger level, a μ DMA multiple transfer request, dma_req, will be asserted to continue data transfers from the μ DMA.

If the Rx FIFO is assigned to the slave module and the RXFIFO bit is set in the I2CSCSR register, then the slave module will generate a signal μ DMA request, dma_sreq, if there is any data to be transferred. The dma_req signal will be asserted when the Rx FIFO has more data than the trigger level programmed by the RXTRIG bit in the I2CFIFOCTL register.

NOTE: Best practice recommends that an application should not switch between the I2CSDR register and TX FIFO or vice versa for successive transactions.

19.3.6 Command Sequence Flow Charts

This section details the steps required to perform the various I²C transfer types in both master and slave mode. See [Table 19-8](#) for further sequence information.

19.3.6.1 I²C Master Command Sequences

The following figures show the command sequences available for the I²C master.

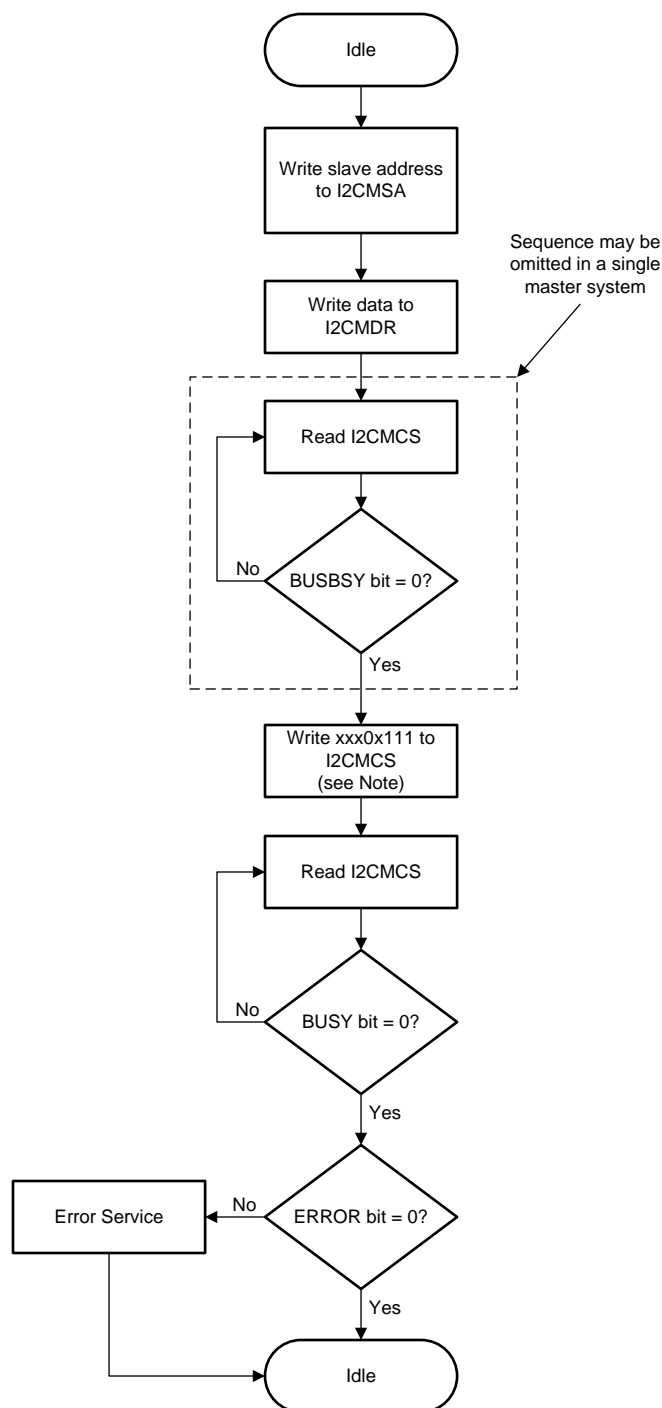
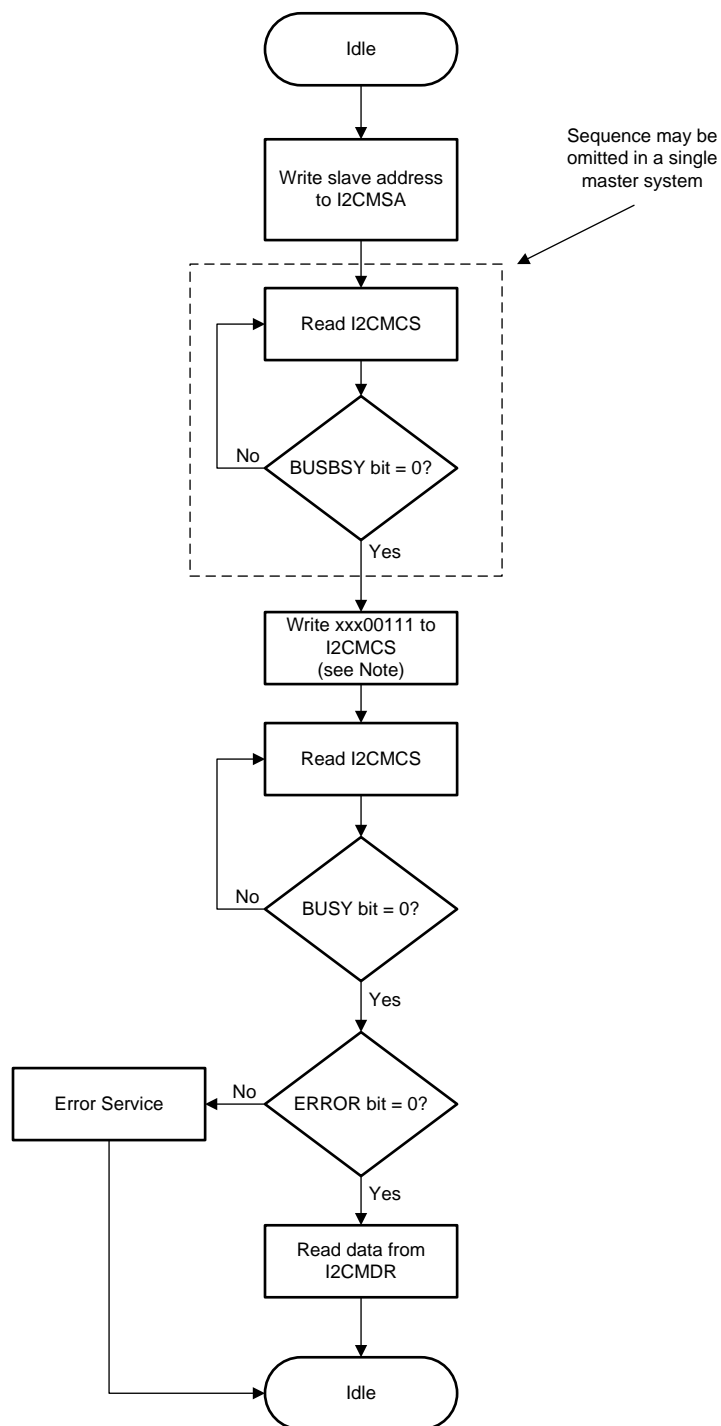
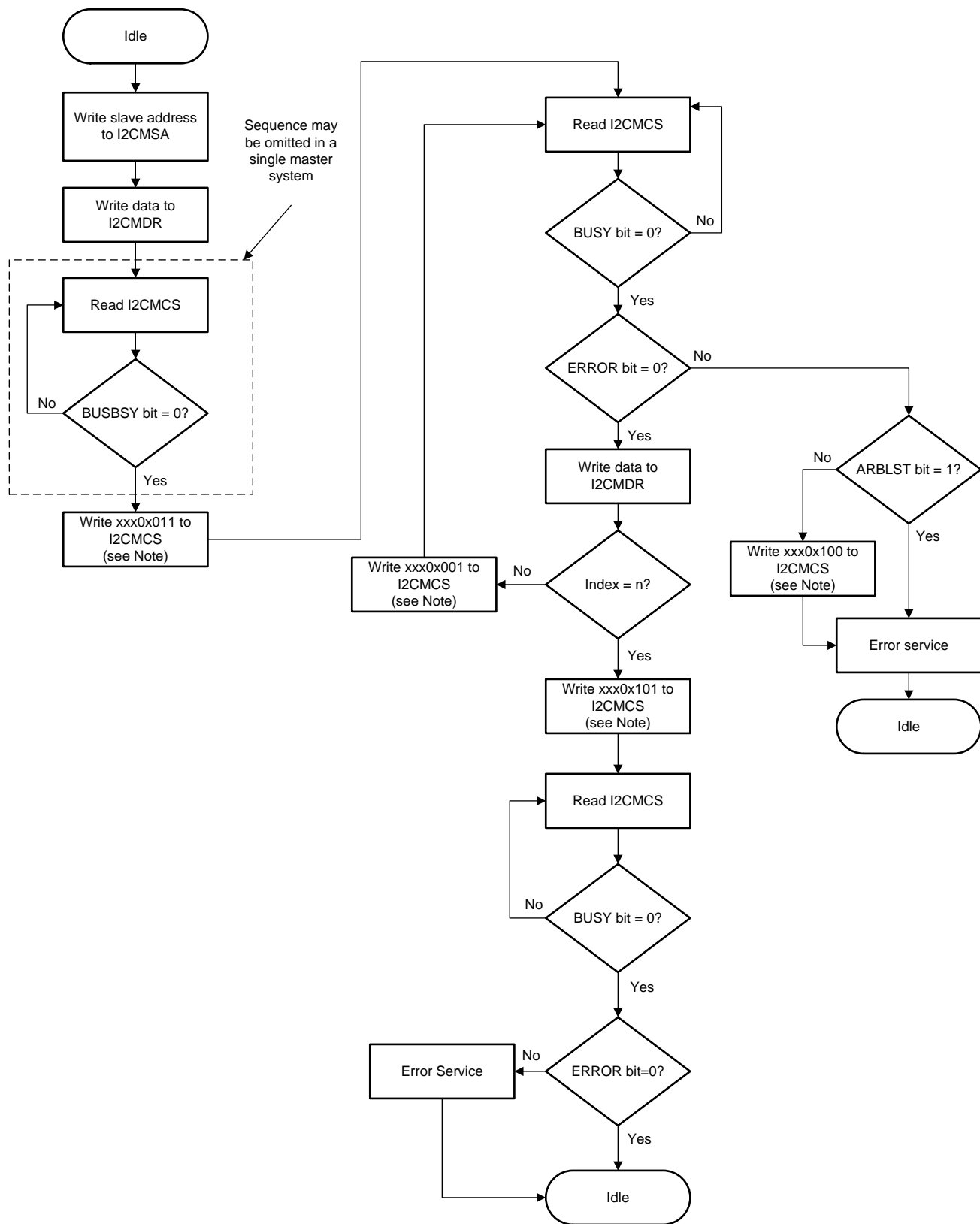


Figure 19-8. Master Single Transmit



NOTE: x = application-specific bit

Figure 19-9. Master Single Receive



NOTE: x = application-specific bit

Figure 19-10. Master Transmit of Multiple Data Bytes

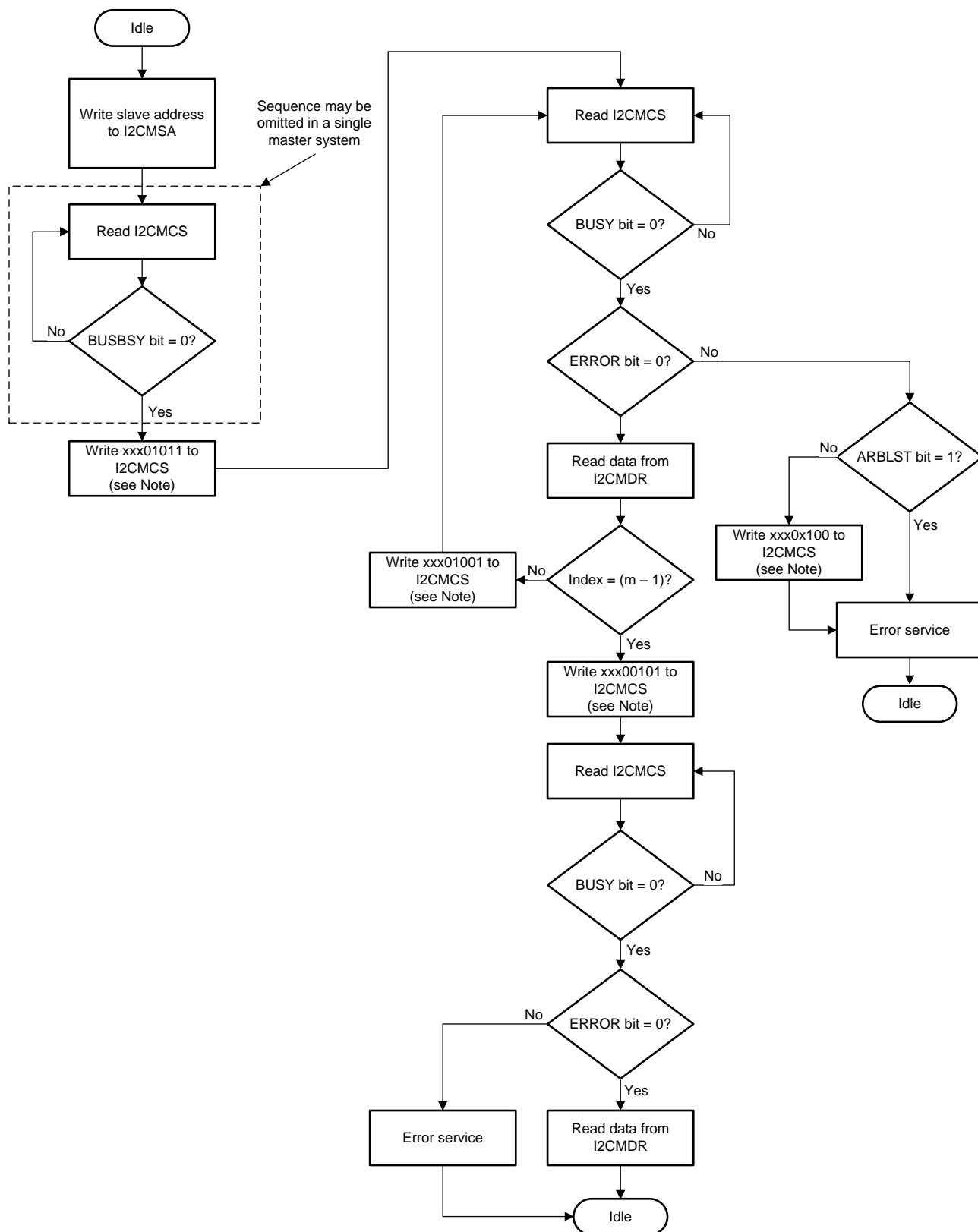
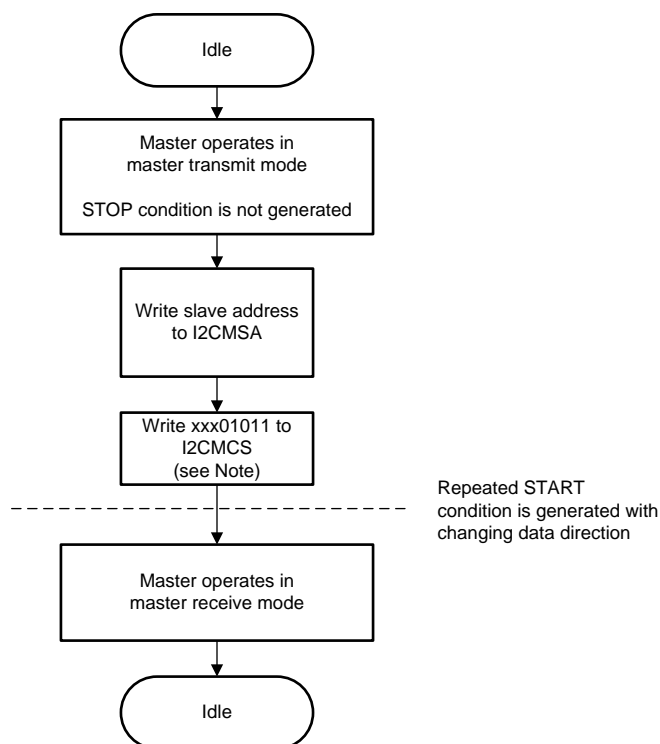
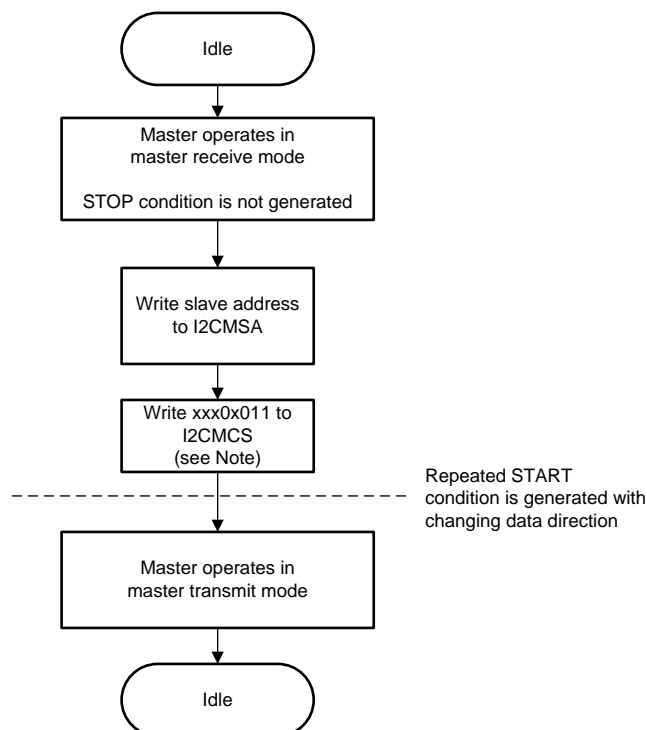


Figure 19-11. Master Receive of Multiple Data Bytes



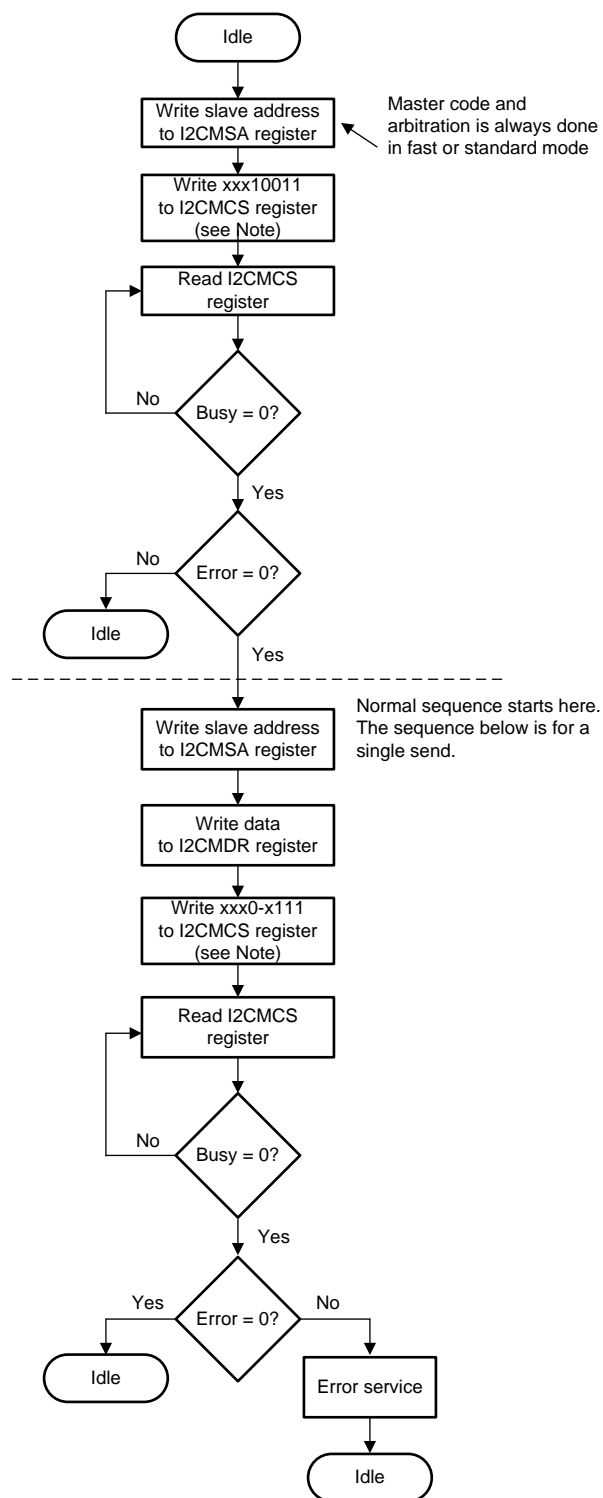
NOTE: x = application-specific bit

Figure 19-12. Master Receive With Repeated START After Master Transmit



NOTE: x = application-specific bit

Figure 19-13. Master Transmit With Repeated START After Master Receive

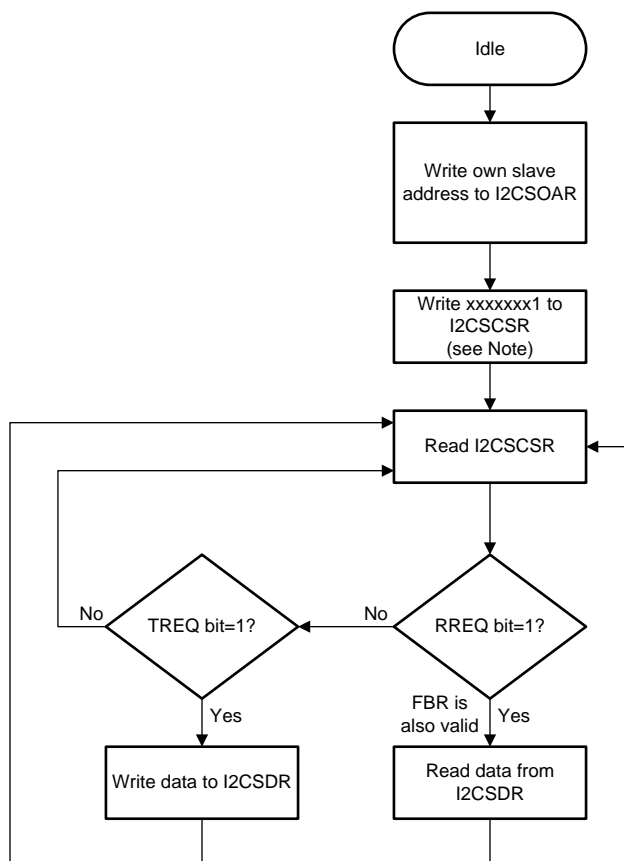


NOTE: x = application-specific bit

Figure 19-14. Standard High-Speed Mode Master Transmit

19.3.6.2 I²C Slave Command Sequences

Figure 19-15 shows the command sequence available for the I²C slave.



NOTE: x = application-specific bit

Figure 19-15. Slave Command Sequence

19.4 Initialization and Configuration

19.4.1 Configure the I²C Module to Transmit a Single Byte as a Master

The following example shows how to configure the I²C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I²C clock using the RCGCI2C register in the System Control module (see [Section 4.2.93](#)).
2. Enable the clock to the appropriate GPIO module using the RCGCGPIO register in the System Control module (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet.
3. In the GPIO module, enable the appropriate pins for their alternate function using the GPIOAFSEL register (see [Section 17.5.10](#)). To determine which GPIOs to configure, see the device-specific data sheet.
4. Enable the I2CSDA pin for open-drain operation. See [Section 17.5.14](#).
5. Configure the PMCN fields in the GPIOCTL register to assign the I²C signals to the appropriate pins. See [Section 17.5.22](#) and the device-specific data sheet.
6. Initialize the I²C master by writing the I2CMCR register with a value of 0x0000.0010.
7. Set the desired SCL clock speed of 100 kbps by writing the I2CMTPR register with the correct value. The value written to the I2CMTPR register represents the number of system clock periods in one SCL clock period. The TPR value is calculated by using [Equation 63](#):

$$\text{TPR} = (\text{System Clock} / (2 \times (\text{SCL_LP} + \text{SCL_HP}) \times \text{SCL_CLK})) - 1$$

$$\text{TPR} = (20 \text{ MHz} / (2 \times (6 + 4) \times 100000)) - 1$$

$$\text{TPR} = 9$$

(63)

Write the I2CMTPR register with the value of 0x0000.0009.

8. Specify the slave address of the master and that the next operation is a Transmit by writing the I2CMSA register with a value of 0x0000.0076. This sets the slave address to 0x3B.
9. Place data (byte) to be transmitted in the data register by writing the I2CMDR register with the desired data.
10. Initiate a single byte transmit of the data from master to slave by writing the I2CMCS register with a value of 0x0000.0007 (STOP, START, RUN).
11. Wait until the transmission completes by polling the BUSBSY bit in the I2CMCS register until it has been cleared.
12. Check the ERROR bit in the I2CMCS register to confirm the transmit was acknowledged.

19.4.2 Configure the I²C Master to High Speed Mode

To configure the I²C master to high-speed mode:

1. Enable the I²C clock using the RCGCI2C register in the System Control module (see [Section 4.2.93](#)).
2. Enable the clock to the appropriate GPIO module via the RCGCGPIO register in the System Control module (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet.
3. In the GPIO module, enable the appropriate pins for their alternate function using the GPIOAFSEL register (see [Section 17.5.10](#)). To determine which GPIOs to configure, see the device-specific data sheet.
4. Enable the I2CSDA pin for open-drain operation. See [Section 17.5.14](#).
5. Configure the PMCN fields in the GPIOCTL register to assign the I²C signals to the appropriate pins. See [Section 17.5.22](#) and the device-specific data sheet.
6. Initialize the I²C master by writing the I2CMCR register with a value of 0x0000.0010.
7. Set the desired SCL clock speed of 3.33 Mbps by writing the I2CMTPR register with the correct value. The value written to the I2CMTPR register represents the number of system clock periods in one SCL clock period. The TPR value is determined by [Equation 64](#):

$$\text{TPR} = (\text{System Clock} / (2 \times (\text{SCL_LP} + \text{SCL_HP}) \times \text{SCL_CLK})) - 1$$

$$\text{TPR} = (80 \text{ MHz} / (2 \times (2 + 1) \times 3330000)) - 1$$

$$\text{TPR} = 3$$

(64)

Write the I2CMTPR register with the value of 0x0000.0003.

8. To send the master code byte, software should place the value of the master code byte into the I2CMSA register and write the I2CMCS register with the following value depending on the required operation:
 - For standard high-speed mode, write 0x13 to the I2CMCS register.
 - For burst high-speed mode, write 0x50 to the I2CMCS register.
9. This places the I²C master peripheral in high-speed mode, and all subsequent transfers (until STOP) are carried out at high-speed data rate using the normal I2CMCS command bits, without setting the HS bit in the I2CMCS register.
10. The transaction is ended by setting the STOP bit in the I2CMCS register.
11. Wait until the transmission completes by polling the BUSBSY bit in the I2CMCS register until it has been cleared.
12. Check the ERROR bit in the I2CMCS register to confirm the transmit was acknowledged.

19.5 I2C Registers

Table 19-3 lists the memory-mapped registers for the I2C. All register offset addresses not listed in Table 19-3 should be considered as reserved locations and the register contents should not be modified.

Table 19-3. I2C Registers

Offset	Acronym	Register Name	Section
0x0	I2CMSA	I2C Master Slave Address	Section 19.5.1
0x4	I2CMCS	I2C Master Control/Status	Section 19.5.2
0x8	I2CMDR	I2C Master Data	Section 19.5.3
0xC	I2CMTPR	I2C Master Timer Period	Section 19.5.4
0x10	I2CMIMR	I2C Master Interrupt Mask	Section 19.5.5
0x14	I2CMRIS	I2C Master Raw Interrupt Status	Section 19.5.6
0x18	I2CMMIS	I2C Master Masked Interrupt Status	Section 19.5.7
0x1C	I2CMICR	I2C Master Interrupt Clear	Section 19.5.8
0x20	I2CMCR	I2C Master Configuration	Section 19.5.9
0x24	I2CMCLKOCNT	I2C Master Clock Low Time-out Count	Section 19.5.10
0x2C	I2CMBMON	I2C Master Bus Monitor	Section 19.5.11
0x30	I2CMBLEN	I2C Master Burst Length	Section 19.5.12
0x34	I2CMBCNT	I2C Master Burst Count	Section 19.5.13
0x800	I2CSOAR	I2C Slave Own Address	Section 19.5.14
0x804	I2CSCSR	I2C Slave Control/Status	Section 19.5.15
0x808	I2CSDR	I2C Slave Data	Section 19.5.16
0x80C	I2CSIMR	I2C Slave Interrupt Mask	Section 19.5.17
0x810	I2CSRIS	I2C Slave Raw Interrupt Status	Section 19.5.18
0x814	I2CSMIS	I2C Slave Masked Interrupt Status	Section 19.5.19
0x818	I2CSICR	I2C Slave Interrupt Clear	Section 19.5.20
0x81C	I2CSOAR2	I2C Slave Own Address 2	Section 19.5.21
0x820	I2CSACKCTL	I2C Slave ACK Control	Section 19.5.22
0xF00	I2CFIFODATA	I2C FIFO Data	Section 19.5.23
0xF04	I2CFIFOCTL	I2C FIFO Control	Section 19.5.24
0xF08	I2CFIFOSTATUS	I2C FIFO Status	Section 19.5.25
0xFC0	I2CPP	I2C Peripheral Properties	Section 19.5.26
0xFC4	I2CPC	I2C Peripheral Configuration	Section 19.5.27

Complex bit access types are encoded to fit into small table cells. Table 19-4 shows the codes that are used for access types in this section.

Table 19-4. I2C Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
RC	C R	to Clear Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

19.5.1 I2CMSA Register (Offset = 0x0) [reset = 0x0]

I2C Master Slave Address (I2CMSA)

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Transmit (Low).

I2CMSA is shown in [Figure 19-16](#) and described in [Table 19-5](#).

Return to [Summary Table](#).

Figure 19-16. I2CMSA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SA							R/S
R-0x0								R/W-0x0							R/W-0x0

Table 19-5. I2CMSA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-1	SA	R/W	0x0	I2C Slave Address. This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0x0	Receive/Send. The R/S bit specifies if the next master operation is a Receive (High) or Transmit (Low). 0x0 = Transmit 0x1 = Receive

19.5.2 I2CMCS Register (Offset = 0x4) [reset = 0x20]

I2C Master Control/Status (I2CMCS)

This register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I²C bus controller. When written, the control register configures the I²C controller operation.

The START bit generates the START or repeated START condition. The STOP bit determines if the cycle stops at the end of the data cycle or continues to the next transfer cycle, which could be a repeated START. To generate a single transmit cycle, the I2CMSA register is written with the desired address, the R/S bit is cleared, and this register is written with ACK = X (0 or 1), STOP = 1, START = 1, and RUN = 1 to perform the operation and stop. When the operation is completed (or aborted due an error), an interrupt becomes active and the data may be read from the I2CMDR register. When the I²C module operates in master receiver mode, the ACK bit is normally set, causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

NOTE: After the CPU starts a transaction, up to 60% of the I²C clock period is required before the BUSY bit is set. Therefore, a delay is required before reading this bit.

NOTE: When reading the I2CMCS register to check the BUSY bit, also read the ADRACK and DATAACK bits, because these are cleared on register read, and status may be lost if they are not checked on every read of the register.

Alternatively, the NACKRIS bit of the I2CMRIS register can be used to monitor NACK status.

I2CMCS as a read-only status register is shown in [Figure 19-17](#) and described in [Table 19-6](#).

I2CMCS as a write-only control register is shown in [Figure 19-18](#) and described in [Table 19-7](#).

Return to [Summary Table](#).

Figure 19-17. I2CMCS Register — Read-Only Status Register

31	30	29	28	27	26	25	24
ACTDMARX	ACTDMATX	RESERVED					
R-0x0	R-0x0	R-0x0					
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
CLKTO	BUSBSY	IDLE	ARBLST	DATAACK	ADRACK	ERROR	BUSY
R-0x0	R-0x0	R-0x1	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 19-6. I2CMCS Register Field Descriptions — Read-Only Status Register

Bit	Field	Type	Reset	Description
31	ACTDMARX	R	0x0	DMA RX Active Status. 0x0 = DMA RX is not active 0x1 = DMA RX is active.
30	ACTDMATX	R	0x0	DMA TX Active Status. 0x0 = DMA TX is not active 0x1 = DMA TX is active.
29-8	RESERVED	R	0x0	

Table 19-6. I2CMCS Register Field Descriptions — Read-Only Status Register (continued)

Bit	Field	Type	Reset	Description
7	CLKTO	R	0x0	Clock Time-out Error. This bit is cleared when the master sends a STOP condition or if the I2C master is reset. 0x0 = No clock timeout error. 0x1 = The clock timeout error has occurred.
6	BUSBSY	R	0x0	Bus Busy. The bit changes based on the START and STOP conditions. 0x0 = The I ² C bus is idle. 0x1 = The I ² C bus is busy.
5	IDLE	R	0x1	I2C Idle. 0x0 = The I ² C controller is not idle. 0x1 = The I ² C controller is idle.
4	ARBLST	R	0x0	Arbitration Lost. 0x0 = The I ² C controller won arbitration. 0x1 = The I ² C controller lost arbitration.
3	DATAACK	R	0x0	Acknowledge Data. 0x0 = The transmitted data was acknowledged 0x1 = The transmitted data was not acknowledged.
2	ADRACK	R	0x0	Acknowledge Address. 0x0 = The transmitted address was acknowledged 0x1 = The transmitted address was not acknowledged.
1	ERROR	R	0x0	Error. The error can be from the slave address not being acknowledged or the transmit data not being acknowledged. 0x0 = No error was detected on the last operation. 0x1 = An error occurred on the last operation.
0	BUSY	R	0x0	I2C Busy. When the BUSY bit is set, the other status bits are not valid. Note: After the CPU starts a transaction, up to 60% of the I ² C clock period is required before the BUSY bit is set. 0x0 = The controller is idle. 0x1 = The controller is busy.

Figure 19-18. I2CMCS Register — Write-Only Control Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	BURST	QCMD	HS	ACK	STOP	START	RUN
R-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0

Table 19-7. I2CMCS Register Field Descriptions — Write-Only Control Register

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6	BURST	W	0x0	Burst Enable. The BURST and RUN bits are mutually exclusive. 0 = Burst operation is disabled. 1 = The master is enabled to burst using the receive and transmit FIFOs (see Table 19-8).
5	QCMD	W	0x0	Quick Command. 0 = Bus transaction is not a quick command. 1 = The bus transaction is a quick command. To execute a quick command, the START, STOP and RUN bits also need to be set. After the quick command is issued, the master generates a STOP.
4	HS	W	0x0	High-Speed Enable. 0 = The master operates in Standard, Fast mode, or Fast mode plus as selected by using a value in the I2CMTPR register that results in an SCL frequency of 100 kbps for Standard mode, 400 kbps for Fast mode, or 1 Mbps for Fast mode plus. 1 = The master operates in High-Speed mode with transmission speeds up to 3.33 Mbps.
3	ACK	W	0x0	Data Acknowledge Enable. 0 = The received data byte is not acknowledged automatically by the master. 1 = The received data byte is acknowledged automatically by the master (see Table 19-8).
2	STOP	W	0x0	Generate STOP. 0 = The controller does not generate the STOP condition. 1 = The controller generates the STOP condition (see Table 19-8).
1	START	W	0x0	Generate START. 0 = The controller does not generate the START condition. 1 = The controller generates the START or repeated START condition (see Table 19-8).
0	RUN	W	0x0	I2C Master Enable. The BURST and RUN bits are mutually exclusive. 0 = In standard and high speed mode, this encoding means the master is unable to transmit or receive data. In Burst mode, this bit is not used and must be set to 0. 1 = The master is able to transmit or receive data. Note that this bit cannot be set in Burst mode (see Table 19-8).

[Table 19-8](#) can be read from left to right to determine the next state after programming bits in the I2CMSA and I2CMCS registers.

Table 19-8. Write Field Decoding for I2CMCS[6:0]

Current State	I2CM SA[0]	I2CMCS[6:0]							Next State Description
	R/S	BUR ST	QCC MD	HS	ACK	STOP	START	RUN	
Idle	0	0	0	0	X ⁽¹⁾	0	1	1	START condition followed by TRANSMIT (master enters to the Master Transmit status).
	0	0	0	0	X	1	1	1	START condition followed by a TRANSMIT and STOP condition (master remains in Idle status).
	0	1	0	0	X	0	1	0	START condition followed by N FIFO-serviced TRANSMITs (master goes to the Master Transmit status).
	0	1	0	0	X	1	1	0	START condition followed by N FIFO-serviced TRANSMITs and STOP condition (master remains in Idle status).
	1	0	0	0	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive status).
	0	0	1	0	0	1	1	1	Quick Command (Send). After Quick Command is executed, the master returns to Idle status.
	1	0	1	0	0	1	1	1	Quick Command (Receive). After Quick Command is executed, the master returns to Idle status.
	1	0	0	0	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle status).
	1	0	0	0	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive status).
	1	1	0	0	0	0	1	0	START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE operation (master goes to the Master Receive status).
	1	1	0	0	0	1	1	0	START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE and STOP condition (master remains in Idle status).
	1	1	0	0	1	0	1	0	START condition followed by N FIFO-serviced RECEIVE operations (master goes to the Master Receive status).
	0	0	0	1	0	0	1	1	START/RUN condition where master byte is sent with no ACK; followed by High Speed transmit Operation. All subsequent transfers are carried out using normal transmit commands.
	0	1	0	1	0	0	0	0	RUN/BURST condition where master byte is sent with no ACK; followed by High Speed Burst transmit Operation.
	1	0	0	0	1	1	1	1	Illegal
	1	0	0	0	1	1	1	0	Illegal
All other combinations not listed are nonoperations.									NOP

⁽¹⁾ An X in a table cell indicates the bit can be 0 or 1.

Table 19-8. Write Field Decoding for I2CMCS[6:0] (continued)

Current State	I2C SA[0]	I2CMCS[6:0]							Next State Description
	R/S	BUR ST	QCC MD	HS	ACK	STO P	STA RT	RUN	
Master Transmit	X	0	0	0	X	0	0	1	TRANSMIT operation (master remains in Master Transmit status).
	X	0	0	0	X	1	0	0	STOP condition (master goes to Idle status).
	X	0	0	0	X	1	0	1	TRANSMIT followed by STOP condition (master goes to Idle status).
	X	1	0	0	X	0	0	0	N FIFO-serviced TRANSMIT operations (master remains in Master Transmit status).
	X	1	0	0	X	1	0	0	N FIFO-serviced TRANSMIT operations followed by STOP condition (master goes to Idle status).
	0	0	0	0	X	0	1	1	Repeated START condition followed by a TRANSMIT (master remains in Master Transmit status).
	0	0	0	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle status).
	0	1	0	0	X	0	1	0	Repeated START condition followed by N FIFO-serviced TRANSMIT operations (master remains in Master Transmit status).
	0	1	0	0	X	1	1	0	Repeated START condition followed by N FIFO-serviced TRANSMIT operations and STOP condition (master goes to Idle status).
	1	0	0	0	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive status).
	1	0	0	0	0	1	1	1	Repeated START condition followed by a RECEIVE and STOP condition (master goes to Idle status).
	1	0	0	0	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive status).
	1	1	0	0	0	0	1	0	Repeated START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE operation (master goes to Master Receive status).
	1	1	0	0	0	1	1	0	Repeated START condition followed by N FIFO-serviced RECEIVE operations and STOP condition (master goes to Idle status).
	1	1	0	0	1	0	1	0	Repeated START condition followed by N FIFO-serviced RECEIVE operations (master goes to Master Receive status).
	1	0	0	0	1	1	1	1	Illegal
	1	1	0	0	1	1	1	0	Illegal
	All other combinations not listed are nonoperations.								NOP

Table 19-8. Write Field Decoding for I2CMCS[6:0] (continued)

Current State	I2CM SA[0]	I2CMCS[6:0]							Next State Description
	R/S	BUR ST	QCC MD	HS	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive status).
	X	0	0	0	X	1	0	0	STOP condition (master goes to Idlestatus). ⁽²⁾
	X	0	0	0	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idlestatus).
	X	0	0	0	1	0	0	1	RECEIVE operation (master remains in Master Receivestatus).
	X	1	0	0	0	0	0	0	N FIFO-serviced RECEIVE operations with negative ACK on the last RECEIVE (master remains in Master Receivestatus).
	X	1	0	0	0	1	0	0	N FIFO-serviced RECEIVE operations followed by STOP condition (master goes to Idlestatus).
	X	1	0	0	1	0	0	0	N FIFO-serviced RECEIVE operations (master remains in Master Receivestatus).
	X	0	0	0	1	1	0	1	Illegal
	X	1	0	0	1	1	0	0	Illegal
	1	0	0	0	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receivestatus).
	1	0	0	0	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idlestatus).
	1	0	0	0	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receivestatus).
	1	1	0	0	0	0	1	0	Repeated START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE (master remains in Master Receivestatus).
	1	1	0	0	0	1	1	0	Repeated START condition followed by N FIFO-serviced RECEIVE operations and STOP condition (master goes to Idlestatus).
	1	1	0	0	1	0	1	0	Repeated START condition followed by N FIFO-serviced RECEIVE operations (master remains in Master Receivestatus).
	0	0	0	0	X	0	1	1	Repeated START condition followed by TRANSMIT (master goes to Master Transmitstatus).
	0	0	0	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idlestatus).
	0	1	0	0	X	0	1	0	Repeated START condition followed by N FIFO-serviced TRANSMIT operations (master goes to Master Transmitstatus).
	0	1	0	0	X	1	1	0	Repeated START condition followed by N FIFO-serviced TRANSMIT operations and STOP condition (master goes to Idlestatus).
	All other combinations not listed are nonoperations.								NOP

⁽²⁾ In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

19.5.3 I2CMDBR Register (Offset = 0x8) [reset = 0x0]

I2C Master Data (I2CMDBR)

This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state. If the BURST bit is enabled in the I2CMCS register, then the I2CFIFODATA register is used for the current data transmit or receive value and this register is ignored.

I2CMDBR is shown in [Figure 19-19](#) and described in [Table 19-9](#).

Return to [Summary Table](#).

Figure 19-19. I2CMDBR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DATA							
R-0x0																								R/W-0x0							

Table 19-9. I2CMDBR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DATA	R/W	0x0	This byte contains the data transferred during a transaction.

19.5.4 I2CMTPR Register (Offset = 0xC) [reset = 0x1]

I2C Master Timer Period (I2CMTPR)

This register is programmed to set the timer period for the SCL clock and assign the SCL clock to either standard or high-speed mode.

I2CMTPR is shown in [Figure 19-20](#) and described in [Table 19-10](#).

Return to [Summary Table](#).

Figure 19-20. I2CMTPR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED													PULSESEL		
R-0x0													R/W-0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HS	TPR						
R-0x0								W-0x0		R/W-0x1					

Table 19-10. I2CMTPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0x0	
18-16	PULSESEL	R/W	0x0	Glitch Suppression Pulse Width. This field controls the pulse width select for glitch suppression on the SCL and SDA lines. The following values are the glitch suppression values in terms of system clocks. 0x0 = Bypass 0x1 = 1 clock 0x2 = 2 clocks 0x3 = 3 clocks 0x4 = 4 clocks 0x5 = 8 clocks 0x6 = 16 clocks 0x7 = 31 clocks
15-8	RESERVED	R	0x0	
7	HS	W	0x0	High-Speed Enable. 0x0 = The SCL Clock Period set by TPR applies to Standard mode (100 Kbps), Fast-mode (400 Kbps), or Fast-mode plus (1 Mbps). 0x1 = The SCL Clock Period set by TPR applies to High-speed mode (3.33 Mbps).
6-0	TPR	R/W	0x1	Timer Period. This field is used in the equation to configure SCL_PERIOD: $SCL_PERIOD = 2 * (1 + TPR) * (SCL_LP + SCL_HP) * CLK_PRD$, where, SCL_PRD is the SCL line period (I2C clock), TPR is the Timer Period register value (range of 1 to 127), SCL_LP is the SCL Low period (fixed at 6), SCL_HP is the SCL High period (fixed at 4), and CLK_PRD is the system clock period in ns.

19.5.5 I2CMIMR Register (Offset = 0x10) [reset = 0x0]

I2C Master Interrupt Mask (I2CMIMR)

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2CMIMR is shown in [Figure 19-21](#) and described in [Table 19-11](#).

Return to [Summary Table](#).

Figure 19-21. I2CMIMR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				RXFFIM	TXFEIM	RXIM	TXIM
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
ARBLOSTIM	STOPIM	STARTIM	NACKIM	DMATXIM	DMARXIM	CLKIM	IM
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 19-11. I2CMIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	RXFFIM	R/W	0x0	Receive FIFO Full Interrupt Mask 0x0 = The RXFFRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The Receive FIFO Full interrupt is sent to the interrupt controller when the RXFFRIS bit in the I2CMRIS register is set.
10	TXFEIM	R/W	0x0	Transmit FIFO Empty Interrupt Mask. The TXFEIM interrupt mask bit in the I2CMIMR register should be clear (masking the TXFE interrupt) when the master is performing an RX Burst from the RXFIFO and should be unmasked before starting a TX FIFO transfers. 0x0 = The TXFERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The Transmit FIFO Empty interrupt is sent to the interrupt controller when the TXFERIS bit in the I2CMRIS register is set.
9	RXIM	R/W	0x0	Receive FIFO Request Interrupt Mask 0x0 = The RXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The RX FIFO Request interrupt is sent to the interrupt controller when the RXRIS bit in the I2CMRIS register is set.
8	TXIM	R/W	0x0	Transmit FIFO Request Interrupt Mask 0x0 = The TXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The TX FIFO Request interrupt is sent to the interrupt controller when the TXRIS bit in the I2CMRIS register is set.
7	ARBLOSTIM	R/W	0x0	Arbitration Lost Interrupt Mask 0x0 = The ARBLOSTRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The Arbitration Lost interrupt is sent to the interrupt controller when the ARBLOSTRIS bit in the I2CMRIS register is set.

Table 19-11. I2CMIMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	STOPIM	R/W	0x0	STOP Detection Interrupt Mask 0x0 = The STOPRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The STOP detection interrupt is sent to the interrupt controller when the STOPRIS bit in the I2CMRIS register is set.
5	STARTIM	R/W	0x0	START Detection Interrupt Mask 0x0 = The STARTRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The START detection interrupt is sent to the interrupt controller when the STARTRIS bit in the I2CMRIS register is set.
4	NACKIM	R/W	0x0	Address/Data NACK Interrupt Mask 0x0 = The NACKRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The address/data NACK interrupt is sent to the interrupt controller when the NACKRIS bit in the I2CMRIS register is set.
3	DMATXIM	R/W	0x0	Transmit DMA Interrupt Mask 0x0 = The DMATXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The transmit DMA complete interrupt is sent to the interrupt controller when the DMATXRIS bit in the I2CMRIS register is set.
2	DMARXIM	R/W	0x0	Receive DMA Interrupt Mask 0x0 = The DMARXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The receive DMA complete interrupt is sent to the interrupt controller when the DMARXRIS bit in the I2CMRIS register is set.
1	CLKIM	R/W	0x0	Clock Time-out Interrupt Mask 0x0 = The CLKRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The clock timeout interrupt is sent to the interrupt controller when the CLKRIS bit in the I2CMRIS register is set.
0	IM	R/W	0x0	Master Interrupt Mask 0x0 = The RIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The master interrupt is sent to the interrupt controller when the RIS bit in the I2CMRIS register is set.

19.5.6 I2CMRIS Register (Offset = 0x14) [reset = 0x0]

I2C Master Raw Interrupt Status (I2CMRIS)

This register specifies whether an interrupt is pending.

I2CMRIS is shown in [Figure 19-22](#) and described in [Table 19-12](#).

Return to [Summary Table](#).

Figure 19-22. I2CMRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				RXFFRIS	TXFERIS	RXRIS	TXRIS
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
ARBLOSTRIS	STOPRIS	STARTRIS	NACKRIS	DMATXRIS	DMARXRIS	CLKRIS	RIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 19-12. I2CMRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	RXFFRIS	R	0x0	Receive FIFO Full Raw Interrupt Status. This bit is cleared by writing a 1 to the RXFFIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The Receive FIFO Full interrupt is pending.
10	TXFERIS	R	0x0	Transmit FIFO Empty Raw Interrupt Status. This bit is cleared by writing a 1 to the TXFEIC bit in the I2CMICR register. Note that if we clear the TXFERIS interrupt (by setting the TXFEIC bit) when the TX FIFO is empty, the TXFERIS interrupt does not reassert even though the TX FIFO remains empty in this situation. 0x0 = No interrupt 0x1 = The Transmit FIFO Empty interrupt is pending.
9	RXRIS	R	0x0	Receive FIFO Request Raw Interrupt Status. This bit is cleared by writing a 1 to the RXIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The trigger level for the RX FIFO has been reached or there is data in the FIFO and the burst count is zero. Thus, a RX FIFO request interrupt is pending.
8	TXRIS	R	0x0	Transmit Request Raw Interrupt Status. This bit is cleared by writing a 1 to the TXIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The trigger level for the TX FIFO has been reached and more data is needed to complete the burst. Thus, a TX FIFO request interrupt is pending.
7	ARBLOSTRIS	R	0x0	Arbitration Lost Raw Interrupt Status. This bit is cleared by writing a 1 to the ARBLOSTIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The Arbitration Lost interrupt is pending.
6	STOPRIS	R	0x0	STOP Detection Raw Interrupt Status This bit is cleared by writing a 1 to the STOPIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The STOP Detection interrupt is pending.

Table 19-12. I2CMRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	STARTRIS	R	0x0	START Detection Raw Interrupt Status. This bit is cleared by writing a 1 to the STARTIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The START Detection interrupt is pending.
4	NACKRIS	R	0x0	Address/Data NACK Raw Interrupt Status. This bit is cleared by writing a 1 to the NACKIC bit in the I2CMICR register. 0x0 = No interrupt 0x1 = The address/data NACK interrupt is pending.
3	DMATXRIS	R	0x0	Transmit DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMATXIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = The transmit DMA complete interrupt is pending.
2	DMARXRIS	R	0x0	Receive DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMARXIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = The receive DMA complete interrupt is pending.
1	CLKRIS	R	0x0	Clock Time-out Raw Interrupt Status. This bit is cleared by writing a 1 to the CLKIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = The clock timeout interrupt is pending.
0	RIS	R	0x0	Master Raw Interrupt Status. This interrupt includes Master transaction completed and Next byte transfer request. This bit is cleared by writing a 1 to the IC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = A master interrupt is pending.

19.5.7 I2CMMIS Register (Offset = 0x18) [reset = 0x0]

I2C Master Masked Interrupt Status (I2CMMIS)

This register specifies whether an interrupt was signaled.

I2CMMIS is shown in [Figure 19-23](#) and described in [Table 19-13](#).

Return to [Summary Table](#).

Figure 19-23. I2CMMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				RXFFMIS	TXFEMIS	RXMIS	TXMIS
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
ARBLOSTMIS	STOPMIS	STARTMIS	NACKMIS	DMATXMIS	DMARXMIS	CLKMIS	MIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 19-13. I2CMMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	RXFFMIS	R	0x0	Receive FIFO Full Interrupt Mask. This bit is cleared by writing a 1 to the RXFFIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked Receive FIFO Full interrupt was signaled and is pending.
10	TXFEMIS	R	0x0	Transmit FIFO Empty Interrupt Mask. This bit is cleared by writing a 1 to the TXFEIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked Transmit FIFO Empty interrupt was signaled and is pending.
9	RXMIS	R	0x0	Receive FIFO Request Interrupt Mask. This bit is cleared by writing a 1 to the RXIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked Receive FIFO Request interrupt was signaled and is pending.
8	TXMIS	R	0x0	Transmit Request Interrupt Mask. This bit is cleared by writing a 1 to the TXIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked Transmit FIFO Request interrupt was signaled and is pending.
7	ARBLOSTMIS	R	0x0	Arbitration Lost Interrupt Mask. This bit is cleared by writing a 1 to the ARBLOSTIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked Arbitration Lost interrupt was signaled and is pending.
6	STOPMIS	R	0x0	STOP Detection Interrupt Mask. This bit is cleared by writing a 1 to the STOPIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked STOP Detection interrupt was signaled and is pending.

Table 19-13. I2CMMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	STARTMIS	R	0x0	START Detection Interrupt Mask. This bit is cleared by writing a 1 to the STARTIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked START Detection interrupt was signaled and is pending.
4	NACKMIS	R	0x0	Address/Data NACK Interrupt Mask. This bit is cleared by writing a 1 to the NACKIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked Address/Data NACK interrupt was signaled and is pending.
3	DMATXMIS	R	0x0	Transmit DMA Interrupt Status. This bit is cleared by writing a 1 to the DMATXIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked transmit DMA complete interrupt was signaled and is pending.
2	DMARXMIS	R	0x0	Receive DMA Interrupt Status. This bit is cleared by writing a 1 to the DMARXIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked receive DMA complete interrupt was signaled and is pending.
1	CLKMIS	R	0x0	Clock Time-out Masked Interrupt Status. This bit is cleared by writing a 1 to the CLKIC bit in the I2CMICR register. 0x0 = No interrupt. 0x1 = An unmasked clock timeout interrupt was signaled and is pending.
0	MIS	R	0x0	Masked Interrupt Status. This bit is cleared by writing a 1 to the IC bit in the I2CMICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked master interrupt was signaled and is pending.

19.5.8 I2CMICR Register (Offset = 0x1C) [reset = 0x0]

I2C Master Interrupt Clear (I2CMICR)

This register clears the raw and masked interrupts.

I2CMICR is shown in [Figure 19-24](#) and described in [Table 19-14](#).

Return to [Summary Table](#).

Figure 19-24. I2CMICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				RXFFIC	TXFEIC	RXIC	TXIC
R-0x0				W-0x0	W-0x0	W-0x0	W-0x0
7	6	5	4	3	2	1	0
ARBLOSTIC	STOPIC	STARTIC	NACKIC	DMATXIC	DMARXIC	CLKIC	IC
W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0

Table 19-14. I2CMICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	RXFFIC	W	0x0	Receive FIFO Full Interrupt Clear. Writing a 1 to this bit clears the RXFFIS bit in the I2CMRIS register and the RXFFMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
10	TXFEIC	W	0x0	Transmit FIFO Empty Interrupt Clear. Writing a 1 to this bit clears the TXFERIS bit in the I2CMRIS register and the TXFEMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
9	RXIC	W	0x0	Receive FIFO Request Interrupt Clear. Writing a 1 to this bit clears the RXRIS bit in the I2CMRIS register and the RXMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
8	TXIC	W	0x0	Transmit FIFO Request Interrupt Clear. Writing a 1 to this bit clears the TXRIS bit in the I2CMRIS register and the TXMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
7	ARBLOSTIC	W	0x0	Arbitration Lost Interrupt Clear. Writing a 1 to this bit clears the ARBLOSTRIS bit in the I2CMRIS register and the ARBLOSTMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
6	STOPIC	W	0x0	STOP Detection Interrupt Clear. Writing a 1 to this bit clears the STOPRIS bit in the I2CMRIS register and the STOPMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
5	STARTIC	W	0x0	START Detection Interrupt Clear. Writing a 1 to this bit clears the STARTRIS bit in the I2CMRIS register and the STARTMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
4	NACKIC	W	0x0	Address/Data NACK Interrupt Clear. Writing a 1 to this bit clears the NACKRIS bit in the I2CMRIS register and the NACKMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
3	DMATXIC	W	0x0	Transmit DMA Interrupt Clear. Writing a 1 to this bit clears the DMATXRIS bit in the I2CMRIS register and the DMATXMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.

Table 19-14. I2CMICR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	DMARXIC	W	0x0	Receive DMA Interrupt Clear. Writing a 1 to this bit clears the DMARXRIS bit in the I2CMRIS register and the DMARXMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
1	CLKIC	W	0x0	Clock Time-out Interrupt Clear. Writing a 1 to this bit clears the CLKRIS bit in the I2CMRIS register and the CLKMIS bit in the I2CMMIS register. A read of this register returns no meaningful data.
0	IC	W	0x0	Master Interrupt Clear. Writing a 1 to this bit clears the RIS bit in the I2CMRIS register and the MIS bit in the I2CMMIS register. A read of this register returns no meaningful data.

19.5.9 I2CMCR Register (Offset = 0x20) [reset = 0x0]

I2C Master Configuration (I2CMCR)

This register configures the mode (Master or Slave), and sets the interface for test mode loopback.

I2CMCR is shown in [Figure 19-25](#) and described in [Table 19-15](#).

Return to [Summary Table](#).

Figure 19-25. I2CMCR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		SFE	MFE	RESERVED		LPBK	
R-0x0		R/W-0x0	R/W-0x0	R-0x0		R/W-0x0	

Table 19-15. I2CMCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5	SFE	R/W	0x0	I2C Slave Function Enable. 0x0 = Slave mode is disabled. 0x1 = Slave mode is enabled.
4	MFE	R/W	0x0	I2C Master Function Enable. 0x0 = Master mode is disabled. 0x1 = Master mode is enabled.
3-1	RESERVED	R	0x0	
0	LPBK	R/W	0x0	I2C Loopback. 0x0 = Normal operation. 0x1 = The controller in a test mode loopback configuration.

19.5.10 I2CMCLKOCNT Register (Offset = 0x24) [reset = 0x0]

I2C Master Clock Low Time-out Count (I2CMCLKOCNT)

This register contains the upper 8 bits of a 12-bit counter that can be used to keep the timeout limit for clock stretching by a remote slave. The lower four bits of the counter are not user visible and are always 0x0.

NOTE: The Master Clock Low Time-out counter counts for the entire time SCL is held Low continuously. If SCL is deasserted at any point, the Master Clock Low Time-out Counter is reloaded with the value in the I2CMCLKOCNT register and begins counting down from this value.

I2CMCLKOCNT is shown in [Figure 19-26](#) and described in [Table 19-16](#).

Return to [Summary Table](#).

Figure 19-26. I2CMCLKOCNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CNTL							
R-0x0																								R/W-0x0							

Table 19-16. I2CMCLKOCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CNTL	R/W	0x0	I2C Master Count. This field contains the upper 8 bits of a 12-bit counter for the clock low timeout count. The value of CNTL must be greater than 0x1.

19.5.11 I2CMBMON Register (Offset = 0x2C) [reset = 0x3]

I2C Master Bus Monitor (I2CMBMON)

This register is used to determine the SCL and SDA signal status.

I2CMBMON is shown in [Figure 19-27](#) and described in [Table 19-17](#).

Return to [Summary Table](#).

Figure 19-27. I2CMBMON Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														SDA	SCL
R-0x0														R-0x1	R-0x1

Table 19-17. I2CMBMON Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	SDA	R	0x1	I2C SDA Status. 0x0 = The I2CSDA signal is low. 0x1 = The I2CSDA signal is high.
0	SCL	R	0x1	I2C SCL Status. 0x0 = The I2CSCL signal is low. 0x1 = The I2CSCL signal is high.

19.5.12 I2CMBLEN Register (Offset = 0x30) [reset = 0x0]

I2C Master Burst Length (I2CMBLEN)

This register contains the programmed length of bytes that are transferred during a Burst request.

I2CMBLEN is shown in [Figure 19-28](#) and described in [Table 19-18](#).

Return to [Summary Table](#).

Figure 19-28. I2CMBLEN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CNTL							
R-0x0																								R/W-0x0							

Table 19-18. I2CMBLEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CNTL	R/W	0x0	I2C Burst Length. This field contains the programmed length of bytes of the Burst Transaction. If BURST is enabled this register must be set to a nonzero value otherwise an error will occur.

19.5.13 I2CMBCNT Register (Offset = 0x34) [reset = 0x0]

I2C Master Burst Count (I2CMBCNT)

When BURST is active, the value in the I2CMLEN register is copied into this register and decremented during the BURST transaction. This register can be used to determine the number of transfers that occurred when a BURST terminates early (as a result of a data NACK). When a BURST completes successfully, this register will contain 0.

I2CMBCNT is shown in [Figure 19-29](#) and described in [Table 19-19](#).

Return to [Summary Table](#).

Figure 19-29. I2CMBCNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CNTL							
R-0x0																								R-0x0							

Table 19-19. I2CMBCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CNTL	R	0x0	I2C Master Burst Count. This field contains the current count-down value of the BURST transaction.

19.5.14 I2CSOAR Register (Offset = 0x800) [reset = 0x0]

I2C Slave Own Address (I2CSOAR)

This register consists of seven address bits that identify this I2C device on the I2C bus.

I2CSOAR is shown in [Figure 19-30](#) and described in [Table 19-20](#).

Return to [Summary Table](#).

Figure 19-30. I2CSOAR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								OAR							
R-0x0																								R/W-0x0							

Table 19-20. I2CSOAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6-0	OAR	R/W	0x0	I2C Slave Own Address. This field specifies bits A6 through A0 of the slave address.

19.5.15 I2CSCSR Register (Offset = 0x804) [reset = 0x0]

I2C Slave Control/Status (I2CSCSR)

This register functions as a control register when written, and a status register when read.

I2CSCSR as a read-only status register is shown in [Figure 19-31](#) and described in [Table 19-21](#).

I2CSCSR as a write-only control register is shown in [Figure 19-32](#) and described in [Table 19-22](#).

Return to [Summary Table](#).

Figure 19-31. I2CSCSR Register — Read-Only Status Register

31	30	29	28	27	26	25	24
ACTDMARX	ACTDMATX	RESERVED					
R-0x0	R-0x0	R-0x0					
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		QCMDRW	QCMDST	OAR2SEL	FBR	TREQ	RREQ
R-0x0		RC-0x0	RC-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 19-21. I2CSCSR Register Field Descriptions — Read-Only Status Register

Bit	Field	Type	Reset	Description
31-3	ACTDMARX	R	0x0	DMA RX Active Status. 0x0 = DMA RX is not active 0x1 = DMA RX is active.
30	ACTDMATX	R	0x0	DMA TX Active Status. 0x0 = DMA TX is not active 0x1 = DMA TX is active.
5	QCMDRW	RC	0x0	Quick Command Read / Write This bit only has meaning when the QCMDST bit is set. 0x0 = Quick command was a write 0x1 = Quick command was a read
4	QCMDST	RC	0x0	Quick Command Status. 0x0 = The last transaction was a normal transaction or a transaction has not occurred. 0x1 = The last transaction was a Quick Command transaction.
3	OAR2SEL	R	0x0	OAR2 Address Matched. This bit gets reevaluated after every address comparison. 0x0 = Either the address is not matched or the match is in legacy mode. 0x1 = OAR2 address matched and ACKed by the slave.
2	FBR	R	0x0	First Byte Received. This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the I2CSDR register. This bit is not used for slave transmit operations. 0x0 = The first byte has not been received. 0x1 = The first byte following the slave's own address has been received.

Table 19-21. I2CSCSR Register Field Descriptions — Read-Only Status Register (continued)

Bit	Field	Type	Reset	Description
1	TREQ	R	0x0	Transmit Request. 0x0 = No outstanding transmit request. 0x1 = The I2C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the I2CSDR register.
0	RREQ	R	0x0	Receive Request. 0x0 = No outstanding receive data. 0x1 = The I2C controller has outstanding receive data from the I2C master and is using clock stretching to delay the master until the data has been read from the I2CSDR register.

Figure 19-32. I2CSCSR Register — Write-Only Control Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					RXFIFO	TXFIFO	DA
R-0x0					W-0x0	W-0x0	W-0x0

Table 19-22. I2CSCSR Register Field Descriptions — Write-Only Control Register

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	RXFIFO	W	0x0	RX FIFO Enable. 0 = Disables RX FIFO 1 = Enables RX FIFO
1	TXFIFO	W	0x0	TX FIFO Enable. 0 = Disables TX FIFO 1 = Enables TX FIFO
0	DA	W	0x0	Device Active. 0 = Disables the I2C slave operation. 1 = Enables the I2C slave operation. After this bit has been set, it should not be set again unless it has been cleared by writing a 0 or by a reset, otherwise transfer failures may occur.

19.5.16 I2CSDR Register (Offset = 0x808) [reset = 0x0]

I2C Slave Data (I2CSDR)

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state. If the RXFIFO bit or TXFIFO bit are enabled in the I2CSCSR register, then this register is ignored and the data value being transferred from the FIFO is contained in the I2CFIFODATA register.

NOTE: Best practice recommends that an application should not switch between the I2CSDR register and TX FIFO or vice versa for successive transactions.

I2CSDR is shown in [Figure 19-33](#) and described in [Table 19-23](#).

Return to [Summary Table](#).

Figure 19-33. I2CSDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DATA							
R-0x0																								R/W-0x0							

Table 19-23. I2CSDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DATA	R/W	0x0	Data for Transfer. This field contains the data for transfer during a slave receive or transmit operation.

19.5.17 I2CSIMR Register (Offset = 0x80C) [reset = 0x0]

I2C Slave Interrupt Mask (I2CSIMR)

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2CSIMR is shown in [Figure 19-34](#) and described in [Table 19-24](#).

Return to [Summary Table](#).

Figure 19-34. I2CSIMR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							RXFFIM
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
TXFEIM	RXIM	TXIM	DMATXIM	DMARXIM	STOPIM	STARTIM	DATAIM
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 19-24. I2CSIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	RXFFIM	R/W	0x0	Receive FIFO Full Interrupt Mask. 0x0 = The RXFFRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The Receive FIFO Full interrupt is sent to the interrupt controller when the RXFFRIS bit in the I2CSRIS register is set.
7	TXFEIM	R/W	0x0	Transmit FIFO Empty Interrupt Mask. 0x0 = The TXFERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The Transmit FIFO Empty interrupt is sent to the interrupt controller when the TXFERIS bit in the I2CSRIS register is set.
6	RXIM	R/W	0x0	Receive FIFO Request Interrupt Mask. 0x0 = The RXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The RX FIFO Request interrupt is sent to the interrupt controller when the RXRIS bit in the I2CSRIS register is set.
5	TXIM	R/W	0x0	Transmit FIFO Request Interrupt Mask. 0x0 = The TXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The TX FIFO Request interrupt is sent to the interrupt controller when the TXRIS bit in the I2CSRIS register is set.
4	DMATXIM	R/W	0x0	Transmit DMA Interrupt Mask. 0x0 = The DMATXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The transmit DMA complete interrupt is sent to the interrupt controller when the DMATXRIS bit in the I2CSRIS register is set.
3	DMARXIM	R/W	0x0	Receive DMA Interrupt Mask. 0x0 = The DMARXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The receive DMA complete interrupt is sent to the interrupt controller when the DMARXRIS bit in the I2CSRIS register is set.

Table 19-24. I2CSIMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	STOPIM	R/W	0x0	Stop Condition Interrupt Mask. 0x0 = The STOPRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The STOP condition interrupt is sent to the interrupt controller when the STOPRIS bit in the I2CSRIS register is set.
1	STARTIM	R/W	0x0	Start Condition Interrupt Mask. 0x0 = The STARTRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = The START condition interrupt is sent to the interrupt controller when the STARTRIS bit in the I2CSRIS register is set.
0	DATAIM	R/W	0x0	Data Interrupt Mask. 0x0 = The DATARIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = Data interrupt sent to interrupt controller when DATARIS bit in the I2CSRIS register is set.

19.5.18 I2CSRIS Register (Offset = 0x810) [reset = 0x0]

I2C Slave Raw Interrupt Status (I2CSRIS)

This register specifies whether an interrupt is pending.

I2CSRIS is shown in [Figure 19-35](#) and described in [Table 19-25](#).

Return to [Summary Table](#).

Figure 19-35. I2CSRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							RXFFRIS
R-0x0							R-0x0
7	6	5	4	3	2	1	0
TXFERIS	RXRIS	TXRIS	DMATXRIS	DMARXRIS	STOPRIS	STARTRIS	DATARIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 19-25. I2CSRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	RXFFRIS	R	0x0	Receive FIFO Full Raw Interrupt Status. This bit is cleared by writing a 1 to the RXFFIC bit in the I2CSICR register. 0x0 = No interrupt 0x1 = The Receive FIFO Full interrupt is pending.
7	TXFERIS	R	0x0	Transmit FIFO Empty Raw Interrupt Status. This bit is cleared by writing a 1 to the TXFEIC bit in the I2CSICR register. Note that if the TXFERIS interrupt is cleared (by setting the TXFEIC bit) when the TX FIFO is empty, the TXFERIS interrupt does not reassert even though the TX FIFO remains empty in this situation. 0x0 = No interrupt 0x1 = The Transmit FIFO Empty interrupt is pending.
6	RXRIS	R	0x0	Receive FIFO Request Raw Interrupt Status. This bit is cleared by writing a 1 to the RXIC bit in the I2CSICR register. 0x0 = No interrupt 0x1 = The trigger value for the FIFO has been reached and a RX FIFO Request interrupt is pending.
5	TXRIS	R	0x0	Transmit Request Raw Interrupt Status. This bit is cleared by writing a 1 to the TXIC bit in the I2CSICR register. 0x0 = No interrupt 0x1 = The trigger value for the FIFO has been reached and a TX FIFO Request interrupt is pending.
4	DMATXRIS	R	0x0	Transmit DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMATXIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = A transmit DMA complete interrupt is pending.
3	DMARXRIS	R	0x0	Receive DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMARXIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = A receive DMA complete interrupt is pending.

Table 19-25. I2CSRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	STOPRIS	R	0x0	Stop Condition Raw Interrupt Status. This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = A STOP condition interrupt is pending.
1	STARTRIS	R	0x0	Start Condition Raw Interrupt Status. This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = A START condition interrupt is pending.
0	DATARIS	R	0x0	Data Raw Interrupt Status. This interrupt encompasses: (1) Slave transaction received, (2) Slave transaction requested, and (3) Next byte transfer request This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = Slave Interrupt is pending.

19.5.19 I2CSMIS Register (Offset = 0x814) [reset = 0x0]

I2C Slave Masked Interrupt Status (I2CSMIS)

This register specifies whether an interrupt was signaled.

I2CSMIS is shown in [Figure 19-36](#) and described in [Table 19-26](#).

Return to [Summary Table](#).

Figure 19-36. I2CSMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							RXFFMIS
R-0x0							R-0x0
7	6	5	4	3	2	1	0
TXFEMIS	RXMIS	TXMIS	DMATXMIS	DMARXMIS	STOPMIS	STARTMIS	DATAMIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 19-26. I2CSMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	RXFFMIS	R	0x0	Receive FIFO Full Interrupt Mask. This bit is cleared by writing a 1 to the RXFFIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = An unmasked Receive FIFO Full interrupt was signaled and is pending.
7	TXFEMIS	R	0x0	Transmit FIFO Empty Interrupt Mask. This bit is cleared by writing a 1 to the TXFEIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = An unmasked Transmit FIFO Empty interrupt was signaled and is pending.
6	RXMIS	R	0x0	Receive FIFO Request Interrupt Mask. This bit is cleared by writing a 1 to the RXIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = An unmasked Receive FIFO Request interrupt was signaled and is pending.
5	TXMIS	R	0x0	Transmit FIFO Request Interrupt Mask. This bit is cleared by writing a 1 to the TXIC bit in the I2CSICR register. 0x0 = No interrupt. 0x1 = An unmasked Transmit FIFO Request interrupt was signaled and is pending.
4	DMATXMIS	R	0x0	Transmit DMA Masked Interrupt Status. This bit is cleared by writing a 1 to the DMATXIC bit in the I2CSICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked transmit DMA complete interrupt was signaled is pending.
3	DMARXMIS	R	0x0	Receive DMA Masked Interrupt Status. This bit is cleared by writing a 1 to the DMARXIC bit in the I2CSICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked receive DMA complete interrupt was signaled is pending.

Table 19-26. I2CSMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	STOPMIS	R	0x0	Stop Condition Masked Interrupt Status. This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked STOP condition interrupt was signaled is pending.
1	STARTMIS	R	0x0	Start Condition Masked Interrupt Status. This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked START condition interrupt was signaled is pending.
0	DATAMIS	R	0x0	Data Masked Interrupt Status. This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked slave data interrupt was signaled is pending.

19.5.20 I2CSICR Register (Offset = 0x818) [reset = 0x0]

I2C Slave Interrupt Clear (I2CSICR)

This register clears the raw interrupt. A read of this register returns no meaningful data.

I2CSICR is shown in [Figure 19-37](#) and described in [Table 19-27](#).

Return to [Summary Table](#).

Figure 19-37. I2CSICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							RXFFIC
R-0x0							W-0x0
7	6	5	4	3	2	1	0
TXFEIC	RXIC	TXIC	DMATXIC	DMARXIC	STOPIC	STARTIC	DATAIC
W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0	W-0x0

Table 19-27. I2CSICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	RXFFIC	W	0x0	Receive FIFO Full Interrupt Mask. Writing a 1 to this bit clears the RXFFIS bit in the I2CSRIS register and the RXFFMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
7	TXFEIC	W	0x0	Transmit FIFO Empty Interrupt Mask. Writing a 1 to this bit clears the TXFERIS bit in the I2CSRIS register and the TXFEMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
6	RXIC	W	0x0	Receive Request Interrupt Mask. Writing a 1 to this bit clears the RXRIS bit in the I2CSRIS register and the RXMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
5	TXIC	W	0x0	Transmit Request Interrupt Mask. Writing a 1 to this bit clears the TXRIS bit in the I2CSRIS register and the TXMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
4	DMATXIC	W	0x0	Transmit DMA Interrupt Clear. Writing a 1 to this bit clears the DMATXRIS bit in the I2CSRIS register and the DMATXMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
3	DMARXIC	W	0x0	Receive DMA Interrupt Clear. Writing a 1 to this bit clears the DMARXRIS bit in the I2CSRIS register and the DMARXMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
2	STOPIC	W	0x0	Stop Condition Interrupt Clear. Writing a 1 to this bit clears the STOPRIS bit in the I2CSRIS register and the STOPMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
1	STARTIC	W	0x0	Start Condition Interrupt Clear. Writing a 1 to this bit clears the STARTRIS bit in the I2CSRIS register and the STARTMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
0	DATAIC	W	0x0	Data Interrupt Clear. Writing a 1 to this bit clears the DATARIS bit in the I2CSRIS register and the DATMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.

19.5.21 I2CSOAR2 Register (Offset = 0x81C) [reset = 0x0]

I2C Slave Own Address 2 (I2CSOAR2)

This register consists of seven address bits that identify the alternate address for the I2C device on the I2C bus.

I2CSOAR2 is shown in [Figure 19-38](#) and described in [Table 19-28](#).

Return to [Summary Table](#).

Figure 19-38. I2CSOAR2 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
OAR2EN	OAR2						
R/W-0x0	R/W-0x0						

Table 19-28. I2CSOAR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	OAR2EN	R/W	0x0	I2C Slave Own Address 2 Enable. 0x0 = The alternate address is disabled. 0x1 = Enables the use of the alternate address in the OAR2 field.
6-0	OAR2	R/W	0x0	I2C Slave Own Address 2. This field specifies the alternate OAR2 address.

19.5.22 I2CSACKCTL Register (Offset = 0x820) [reset = 0x0]

I2C Slave ACK Control (I2CSACKCTL)

This register enables the I2C slave to NACK for invalid data or command or ACK for valid data or command. The I2C clock is pulled low after the last data bit until this register is written.

I2CSACKCTL is shown in [Figure 19-39](#) and described in [Table 19-29](#).

Return to [Summary Table](#).

Figure 19-39. I2CSACKCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						ACKOVAL	ACKOEN
R-0x0						R/W-0x0	R/W-0x0

Table 19-29. I2CSACKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	ACKOVAL	R/W	0x0	I2C Slave ACK Override Value. 0x0 = An ACK is sent indicating valid data or command. 0x1 = A NACK is sent indicating invalid data or command.
0	ACKOEN	R/W	0x0	I2C Slave ACK Override Enable. 0x0 = A response is not provided. 0x1 = An ACK or NACK is sent according to the value written to the ACKOVAL bit.

19.5.23 I2CFIFODATA Register (Offset = 0xF00) [reset = 0x0]

I2C FIFO Data (I2CFIFODATA)

The I2C FIFO Data (I2CFIFODATA) register contains the current value of the top of the RX or TX FIFO stack being used in the a transfer.

I2CFIFODATA is shown in [Figure 19-40](#) and described in [Table 19-30](#).

Return to [Summary Table](#).

Figure 19-40. I2CFIFODATA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DATA							
R-0x0																								R-0x0							

Table 19-30. I2CFIFODATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	DATA	R	0x0	I2C RX FIFO Read Data Byte. This field contains the current byte being read in the RX FIFO stack.

19.5.24 I2CFIFOCTL Register (Offset = 0xF04) [reset = 0x00040004]

I2C FIFO Control (I2CFIFOCTL)

The FIFO Control Register can be programmed to control various aspects of the FIFO transaction, such as RX and TX FIFO assignment, byte count value for FIFO triggers and flushing of the FIFOs.

I2CFIFOCTL is shown in [Figure 19-41](#) and described in [Table 19-31](#).

Return to [Summary Table](#).

Figure 19-41. I2CFIFOCTL Register

31	30	29	28	27	26	25	24
RXASGNMT	RXFLUSH	DMARXENA	RESERVED				
R/W-0x0	R/W-0x0	R/W-0x0	R-0x0				
23	22	21	20	19	18	17	16
RESERVED					RXTRIG		
R-0x0					R/W-0x4		
15	14	13	12	11	10	9	8
TXASGNMT	TXFLUSH	DMATXENA	RESERVED				
R/W-0x0	R/W-0x0	R/W-0x0	R-0x0				
7	6	5	4	3	2	1	0
RESERVED					TXTRIG		
R-0x0					R/W-0x4		

Table 19-31. I2CFIFOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RXASGNMT	R/W	0x0	RX Control Assignment. 0x0 = RX FIFO is assigned to Master 0x1 = RX FIFO is assigned to Slave
30	RXFLUSH	R/W	0x0	RX FIFO Flush. Setting this bit will Flush the RX FIFO. This bit will self-clear when the flush has completed.
29	DMARXENA	R/W	0x0	DMA RX Channel Enable. 0x0 = DMA RX channel disabled 0x1 = DMA RX channel enabled
28-19	RESERVED	R	0x0	
18-16	RXTRIG	R/W	0x4	RX FIFO Trigger. Indicates at what fill level the RX FIFO will generate a trigger. Programming RXTRIG to 0x0 has no effect since no data is present to transfer out of RX FIFO. 0x0 = Trigger when RX FIFO contains no bytes 0x1 = Trigger when Rx FIFO contains 1 or more bytes 0x2 = Trigger when Rx FIFO contains 2 or more bytes 0x3 = Trigger when Rx FIFO contains 3 or more bytes 0x4 = Trigger when Rx FIFO contains 4 or more bytes 0x5 = Trigger when Rx FIFO contains 5 or more bytes 0x6 = Trigger when Rx FIFO contains 6 or more bytes 0x7 = Trigger when Rx FIFO contains 7 or more bytes.
15	TXASGNMT	R/W	0x0	TX Control Assignment. 0x0 = TX FIFO is assigned to Master 0x1 = TX FIFO is assigned to Slave
14	TXFLUSH	R/W	0x0	TX FIFO Flush. Setting this bit will Flush the TX FIFO. This bit will self-clear when the flush has completed.
13	DMATXENA	R/W	0x0	DMA TX Channel Enable. 0x0 = DMA TX channel disabled 0x1 = DMA TX channel enabled

Table 19-31. I2CFIFOCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12-3	RESERVED	R	0x0	
2-0	TXTRIG	R/W	0x4	<p>TX FIFO Trigger. Indicates at what fill level in the TX FIFO a trigger will be generated.</p> <p>0x0 = Trigger when the TX FIFO is empty.</p> <p>0x1 = Trigger when TX FIFO contains <= 1 byte</p> <p>0x2 = Trigger when TX FIFO contains <= 2 bytes</p> <p>0x3 = Trigger when TX FIFO <= 3 bytes</p> <p>0x4 = Trigger when FIFO <= 4 bytes</p> <p>0x5 = Trigger when FIFO <= 5 bytes</p> <p>0x6 = Trigger when FIFO <= 6 bytes</p> <p>0x7 = Trigger when FIFO <= 7 bytes</p>

19.5.25 I2CFIFOSTATUS Register (Offset = 0xF08) [reset = 0x00010005]

I2C FIFO Status (I2CFIFOSTATUS)

This register contains the real-time status of the RX and TX FIFOs.

I2CFIFOSTATUS is shown in [Figure 19-42](#) and described in [Table 19-32](#).

Return to [Summary Table](#).

Figure 19-42. I2CFIFOSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED					RXABVTRIG	RXFF	RXFE
R-0x0					R-0x0	R-0x0	R-0x1
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					TXBLWTRIG	TXFF	TXFE
R-0x0					R-0x1	R-0x0	R-0x1

Table 19-32. I2CFIFOSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0x0	
18	RXABVTRIG	R	0x0	RX FIFO Above Trigger Level. 0x0 = The number of bytes in RX FIFO is below the trigger level programmed by the RXTRIG bit in the I2CFIFOCTL register 0x1 = The number of bytes in the RX FIFO is above the trigger level programmed by the RXTRIG bit in the I2CFIFOCTL register
17	RXFF	R	0x0	RX FIFO Full. 0x0 = The RX FIFO is not full. 0x1 = The RX FIFO is full.
16	RXFE	R	0x1	RX FIFO Empty. 0x0 = The RX FIFO is not empty. 0x1 = The RX FIFO is empty.
15-3	RESERVED	R	0x0	
2	TXBLWTRIG	R	0x1	TX FIFO Below Trigger Level. 0x0 = The number of bytes in TX FIFO is above the trigger level programmed by the TXTRIG bit in the I2CFIFOCTL register 0x1 = The number of bytes in the TX FIFO is below the trigger level programmed by the TXTRIG bit in the I2CFIFOCTL register
1	TXFF	R	0x0	TX FIFO Full. 0x0 = The TX FIFO is not full. 0x1 = The TX FIFO is full.
0	TXFE	R	0x1	TX FIFO Empty. 0x0 = The TX FIFO is not empty. 0x1 = The TX FIFO is empty.

19.5.26 I2CPP Register (Offset = 0xFC0) [reset = 0x1]

I2C Peripheral Properties (I2CPP)

The I2CPP register provides information regarding the properties of the I2C module.

I2CPP is shown in [Figure 19-43](#) and described in [Table 19-33](#).

Return to [Summary Table](#).

Figure 19-43. I2CPP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															HS
R-0x0															R-0x1

Table 19-33. I2CPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	HS	R	0x1	High-Speed Capable 0x0 = The interface is capable of Standard, Fast, or Fast mode plus operation. 0x1 = The interface is capable of High-Speed operation.

19.5.27 I2CPC Register (Offset = 0xFC4) [reset = 0x1]

I2C Peripheral Configuration (I2CPC)

The I2CPC register allows software to enable features present in the I2C module.

I2CPC is shown in [Figure 19-44](#) and described in [Table 19-34](#).

Return to [Summary Table](#).

Figure 19-44. I2CPC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															
R-0x1															

Table 19-34. I2CPC Register Field Descriptions

Bit	Field	Type	Reset	Description
0	HS	R/W	0x1	High-Speed Capable 0x0 = The interface is set to Standard, Fast or Fast mode plus operation. 0x1 = The interface is set to High-Speed operation. Note that this encoding may only be used if the HS bit in the I2CPP register is set. Otherwise, this encoding is not available.
31-0	RESERVED	R	0x1	

LCD Controller

The Liquid Crystal Display (LCD) controller provides support for a variety of LCD and OLED panels.

Topic	Page
20.1 Introduction	1378
20.2 Block Diagram	1378
20.3 Functional Description	1379
20.4 Interrupts	1394
20.5 Bus Transaction Modes	1395
20.6 Initialization and Configuration	1395
20.7 LCD Registers	1397

20.1 Introduction

The LCD controller includes the following features:

- Character-based panels
 - Support for two character panels (CS0 and CS1) with independent and programmable bus timing parameters when in asynchronous Hitachi, Motorola, and Intel modes
 - Support for one character panel (CS0) with programmable bus timing parameters when in synchronous Motorola and Intel modes
 - Can be used as a generic 16-bit address and data interleaved MPU bus master with no external stall
- Passive matrix LCD panels
 - Panel types including STN, DSTN, and C-DSTN
 - AC bias control
- Active matrix LCD panels
 - Panel types including TN TFT
 - 1, 2, 4, or 8 bits per pixel with palette RAM and 16 or 24 bits per pixel without palette RAM
- OLED Panels
 - Passive matrix (PM OLED) with frame buffer and controller IC inside the panel
 - Active matrix (AM OLED)
- Bus mastering capability from either system SRAM or EPI memory.

20.2 Block Diagram

The LCD controller consists of two independent controllers, the Raster Controller and the LCD Interface Display Driver (LIDD) controller. Each controller operates independently from the other and only one of them is active at any given time.

- The Raster Controller provides a synchronous LCD interface. It provides timing and data for constant graphics refresh to a passive display. It supports a wide variety of monochrome and full-color display types and sizes by use of programmable timing controls, a built-in palette, and a gray-scale serializer. Graphics data is processed and stored in frame buffers. A frame buffer is a contiguous memory block in the system. A built-in DMA engine supplies the graphics data to the Raster engine which, in turn, outputs to the external LCD device.
- The LIDD controller provides an asynchronous or synchronous interface depending on the operating mode. The LIDD controller provides full timing programmability of control signals (chip selects, read/write strobes, enable, direction), and output data.

Figure 20-1 shows the LCD controller details. The raster and LIDD controllers are responsible for generating the correct external timing. The LCD DMA engine provides a constant flow of data from the frame buffers to the external LCD panel through the Raster and LIDD controllers. In addition, CPU access is provided to read and write registers.

Figure 20-1 shows the LCD block diagram. For more information on the Raster controller data path, see Figure 20-3.

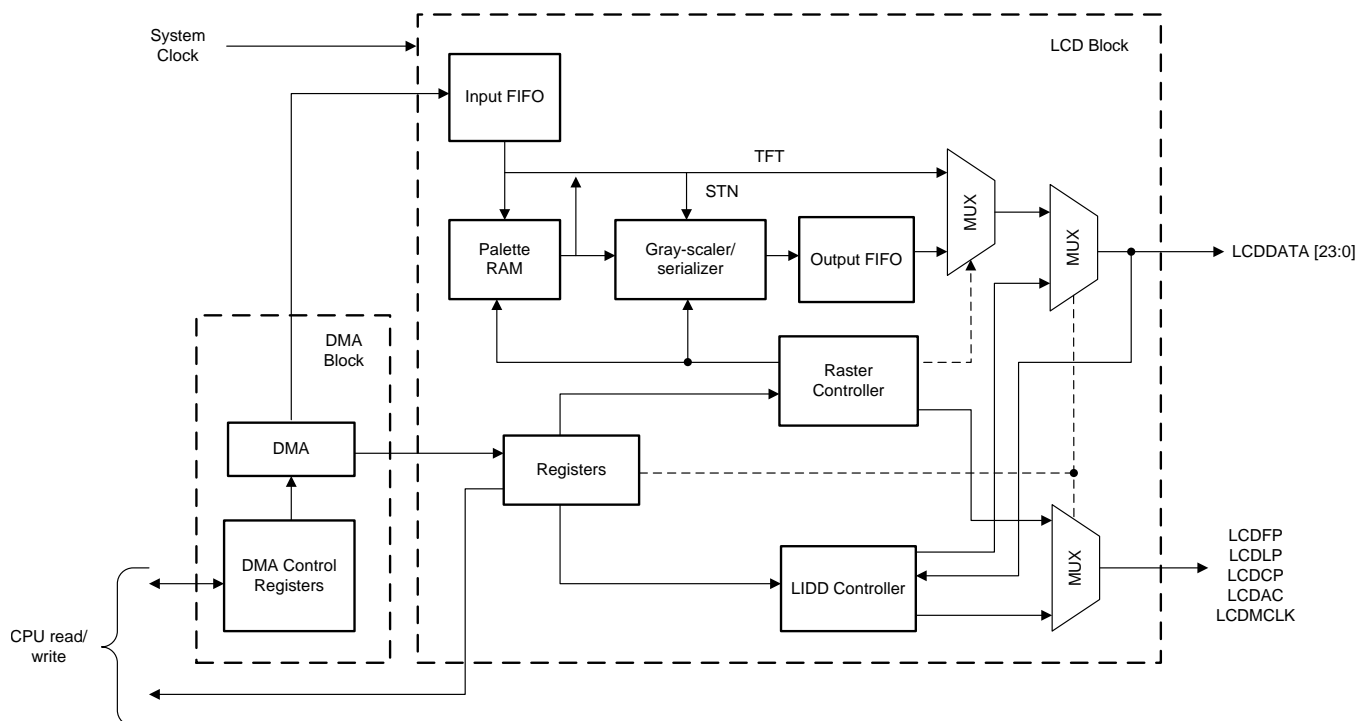


Figure 20-1. LCD Block Diagram

20.3 Functional Description

The following sections describe the functional capability of the LCD controller.

20.3.1 Clocking

This section details the various clocks and signals. Figure 20-2 shows the input and output LCD controller clocks.

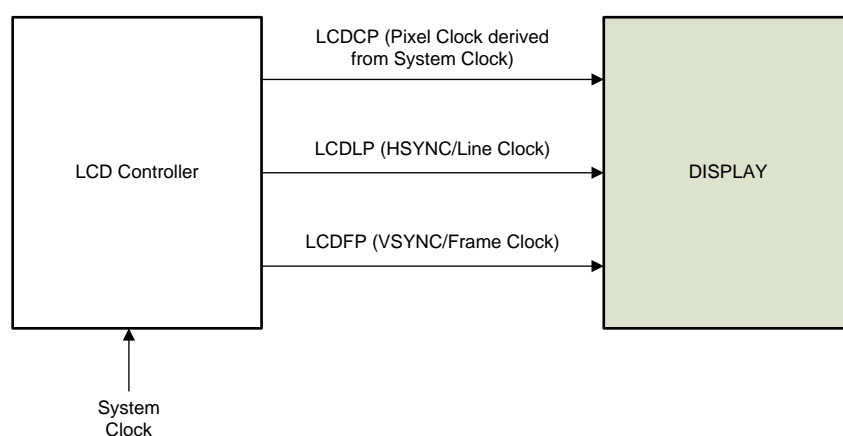


Figure 20-2. Input and Output Clocks

20.3.1.1 Pixel Clock (LCDCP Signal)

The pixel clock (LCDCP) frequency is derived from the system clock, which is the internal reference clock (MCLK) to the LCD module. The pixel clock is used by the LCD display to clock the pixel data into the line shift register. The formula used for the pixel clock is:

$LDCDP = \text{System Clock} / CLKDIV$

where CLKDIV is a field in the LCD Control (LCDCTL) register. The value of CLKDIV should not be 0 or 1 in Raster Mode. The pixel clock operates differently depending on the mode of the LCD controller:

- Passive (STN) mode: LCDCP only transitions when valid data is available for output. It does not transition when the horizontal clock (HSYNC) is asserted or during wait state insertion.
- Active (TFT) mode: LCDCP continuously toggles as long as the Raster Controller is enabled.

20.3.1.2 HSYNC Horizontal Clock (LCDLP Signal)

HSYNC (LCDLP signal) toggles after all pixels in a horizontal line have been transmitted to the LCD and a programmable number of pixel clock wait states have elapsed both at the beginning and end of each line. The LCD Raster Timing 0 (LCDRASTRIM0) register, offset 0x02C, fully defines the behavior of this signal.

HSYNC can be programmed to be synchronized with the rising or falling edge of the pixel clock (LCDCP). The PXLCLKCTL and PSYNCRF bits in the LCD Raster Timing 2 (LCDRASTRIM2) register, offset 0x034, are used to configure the synchronization.

- Active (TFT) mode: The horizontal clock or the line clock is also used by TFT displays as the horizontal synchronization signal (HSYNC).

The timings of the horizontal clock (line clock) pins are programmable to support:

- Delay insertion both at the beginning and end of each line
- Line clock polarity
- Line clock pulse width, driven on rising or falling edge of pixel clock

20.3.1.3 VSYNC Vertical Clock (LCDFP Signal)

VSYNC (LCDFP signal) toggles after all lines in a frame have been transmitted to the LCD and a programmable number of line clock cycles has elapsed both at the beginning and end of each frame. The LCD Raster Timing 1 (LCDRASTRIM1) register, offset 0x030, fully defines the behavior of this signal.

VSYNC can be programmed to be synchronized with the rising or falling edge of LCDCP pixel clock. The PXLCLKCTL and PSYNCRF bits in the LCD Raster Timing 2 (LCDRASTRIM2) register, offset 0x034, are used to configure the synchronization.

- Passive (STN) mode: The vertical, or frame, clock toggles during the first line of the screen.
- Active (TFT) mode: The vertical, or frame, clock is also used by TFT displays as the vertical synchronization signal (VSYNC). The timings of the vertical clock pins are programmable to support:
 - Delay insertion both at the beginning and end of each frame
 - Frame clock polarity

20.3.1.4 LCD AC Bias Enable (LCDAC)

- Passive (STN) mode: To prevent a DC charge within the screen pixels, the power and ground supplies of the display are periodically switched. The Raster Controller signals the LCD to switch the polarity by toggling this pin.
- Active (TFT) mode: This signal acts as an output enable (OE) signal. It is used to signal the external LCD that the data is valid on the data bus.

20.3.2 LCD DMA Engine

The LCD DMA engine can output graphics data to constantly refresh the external LCD display, without burdening the CPU, through interrupts or a firmware timer. The DMA operates on one or two frame buffers, which are set up during initialization. Using two frame buffers (ping-pong buffers) enables the simultaneous operation of outputting the current video frame to the external display and updating the next video frame. The ping-pong buffering approach is preferred in most applications.

When the Raster controller is used, the LCD DMA engine reads data from a frame buffer and writes it to the input FIFO. The Raster controller requests data from the FIFO for frame refresh, so the DMA is used to keep the FIFO filled.

When the LIDD controller is used, the LCD DMA engine accesses the address or data registers of the LIDD controller. The following steps are needed to configure the DMA engine:

1. Configure the data format in the LCD DMA Control (LCDDMACTL) register, offset 0x040.
2. Configure frame buffer 0 by programming the LCD DMA Frame Buffer 0 Base Address (LCDDMABAFB0) register, offset 0x044, and the LCD DMA Frame Buffer 0 Ceiling Address (LCDDMACAFB0) register, offset 0x048.
3. Configure frame buffer 1 by programming the LCD DMA Frame Buffer 1 Base Address (LCDDMABAFB1) register, offset 0x04C, and the LCD DMA Frame Buffer 1 Ceiling Address (LCDDMACAFB1) register, offset 0x050.

In addition, depending on whether the application is operating in LIDD mode or Raster mode, the LCDLIDDCTL or the LCDRASTRCTL register should be configured appropriately, along with all of the timing registers. To enable DMA transfers, write 1 to the DMAEN bit in the LCDLIDDCTL or the LCDEN bit in the LCDRASTRCTL register.

NOTE: When the LCD DMA is enabled, do not read or write the LCD registers for the base and ceiling addresses (LCDDMABAFB0, LCDDMACAFB0, LCDDMABAFB1, and LCDDMACAFB1) with the CPU.

To change any of these registers, disable the DMA (clear the DMAEN bit in the LCDLIDDCTL register or the LCDEN bit in the LCDRASTRCTL register), update the registers, and enable the LCD DMA again.

20.3.2.1 Interrupts

Interrupts in this LCD module are related to DMA engine operation. Five registers are used to control and monitor the interrupts:

- LCDLIDDCTL and LCDRASTRCTL registers enable and disable individual features that can generate interrupts.
- The LCD Interrupt Raw Status and Set Register (LCDRISSET) register, offset 0x058, collects all of the interrupt status information.
- The LCD Interrupt Status and Clear (LCDMISCLR) register, offset 0x05C, is used to read and clear the status of the masked interrupt triggers.
- The LCD Interrupt Mask (LCDIM) register, offset 0x060, is used to control whether an interrupt source is masked or not.

20.3.2.1.1 LIDD Mode

When operating in LIDD mode, the DMA engine generates one interrupt signal every time the specified frame buffer has been transferred completely. The EOF1, EOF0, and DONE bits of the LCDRISSET register reflect the interrupt signal, regardless of whether or not the interrupt is enabled.

20.3.2.1.2 Raster Mode

When operating in Raster mode, the DMA engine generates the interrupts in the following scenarios:

- Output FIFO under-run: This interrupt occurs when the DMA engine cannot keep up with the data rate consumed by the LCD (which is determined by the LCDCP.) This is likely due to a system memory throughput issue or an incorrect LCDCP setting. The FIFOU bit in LCD Interrupt Raw Status and Set Register (LCDRISSET) register is set when this error occurs. This bit is cleared by disabling the Raster Controller, that is clearing the LCDEN bit in LCD Raster Control (LCDRASTRCTL) register.
- Frame synchronization lost: This error happens when the DMA engine attempts to read what it believes to be the first word of the video buffer but the data cannot be recognized as such. This situation can be caused by an invalid frame buffer address or an invalid BPP value. The SYNC bit in the IRQSTATUS_RAW register is set when such an error is detected. This field is cleared by disabling

the Raster Controller (clearing the LCDEN bit in LCD Raster Control (LCDRASTRCTL) register).

- **Palette loaded:** This interrupt can be generated when the palette is loaded into the memory by the DMA engine. At the same time, the PALLOAD bit in the LCD Interrupt Raw Status and Set Register (LCDRISSET) register is set. In data-only (PALMODE = 0x2) and palette-plus-data (PALMODE = 0x0) modes, writing 0 to this bit clears the interrupt. In the palette-only (PALMODE = 0x1) mode, this bit is cleared by disabling the Raster Controller (clearing the LCDEN bit in the LCD Raster Control (LCDRASTRCTL) register).
- **AC bias transition:** If the ACBI field in the LCD Raster Timing 2 (LCDRASTRTIM2) register is programmed with a nonzero value, an internal counter is loaded with this value and starts to decrement each time LCDAC (AC-bias signal) switches its state. When the counter reaches zero, the ACBS bit in the LCD Interrupt Raw Status and Set Register (LCDRISSET) register is set, which delivers an interrupt signal to the system interrupt controller (if the interrupt is enabled.) The counter reloads the value in ACBI field, but does not start to decrement until the ABC bit is cleared by writing 0 to this bit.
- **Frame transfer completed:** When one frame of data is transferred completely, the DONE bit in the LCD Interrupt Raw Status and Set Register (LCDRISSET) register is set. This bit is cleared by disabling the Raster Controller (clearing the LCDEN bit in LCD Raster Control (LCDRASTRCTL) register). The EOF0 and EOF1 bits in LCD Interrupt Raw Status and Set Register (LCDRISSET) register are set accordingly.

The function enable bits are in the in LCD Raster Control (LCDRASTRCTL) register and must be set to generate an interrupt to the CPU.

20.3.3 LIDD Bus Operation

The integrated LIDD controller has programmable timing parameters that support a wide variety of character-based LCD panels. Alternatively, the DMA module can mimic the CPU and perform a sequence of write-only data bus transactions to the character-based LCD panel.

LIDD mode is enabled by clearing the LCDMODE bit in the LCD Control (LCDCTL) register.

LIDD Controller operation is summarized as follows:

- During initialization, the LCD LIDD CS0 Configuration (LIDDCS0CFG) and LCD LIDD CS1 Configuration (LIDDCS1CFG) registers are configured to match the requirements of the LCD panel being used.
- During normal operation, the CPU writes display data to the LIDD CS0 Data Read/Write Initiation (LIDDCS0DATA) and LIDD CS1 Data Read/Write Initiation (LIDDCS1DATA) registers. The LIDD interface converts the CPU write into the proper signal transition sequence for the display, as programmed earlier. Note that the first CPU write should send the beginning address of the update to the LCD panel and the subsequent writes update data at display locations starting from the first address and continuing sequentially. Note that DMA may be used instead of the CPU.
- The LIDD controller can also back status or data from the LCD panel, if the latter has this capability. This is set up and activated in a similar manner to the write function.

Table 20-1 describes how the signals are used to interface to external LCD modules, which are configured by the LCDLIDDCTL register.

Table 20-1. LIDD I/O Name Map

Display Type	Interface Type	Data Bits	LCDLIDDCTL [2:0]	Signal Name	LCD Display I/O Name	LCD Display I/O Description
Character Display	HD44780 Type	4	0x4	LCDDATA[7:4]	DATA[7:4]	LCD data bus (length defined by instruction)
				LCDAC	E (or E0)	Enable strobe (first display)
				LCDLP	R/W	Read/Write
				LCDFP	RS	Register select (data/not instruction)
				LCDMCLK	E1	Enable strobe (second display optional)

Table 20-1. LIDD I/O Name Map (continued)

Display Type	Interface Type	Data Bits	LCDLDDCTL [2:0]	Signal Name	LCD Display I/O Name	LCD Display I/O Description
Character Display	HD44780 Type	8	0x4	LCDDATA[7:0]	DATA[7:0]	LCD data bus (length defined by instruction)
				LCDAC	E (or E0)	Enable strobe (first display)
				LCDLP	R/W	Read/Write
				LCDFP	RS	Register select (data/not instruction)
				LCDMCLK	E1	Enable strobe (second display optional)
Micro Interface Graphic Display	6800 Family	Up to 16	0x1	LCDDATA[15:0]	DATA[15:0]	LCD data bus (16 bits always available)
				LCDCP	E	Enable clock
				LCDLP	R/W	Read/Write
				LCDFP	A0	Address and data select
				LCDAC	CS (or CS0)	Chip select (first display)
				LCDMCLK	CS1	Chip select (second display optional)
			0x0	LCDMCLK	None	Synchronous clock (optional)
Micro Interface Graphic Display	8080 Family	Up to 16	0x3	LCDDATA[15:0]	DATA[15:0]	LCD data bus (16 bits always available)
				LCDCP	RD	Read strobe
				LCDLP	WR	Write strobe
				LCDFP	A0	Address and data select
				LCDAC	CS (or CS0)	Chip select (first display)
				LCDMCLK	CS1	Chip select (second display optional)
			0x2	LCDMCLK	None	Synchronous clock (optional)

20.3.4 Raster Control

The Raster mode is enabled by setting the MODESEL bit in the LCD Control (LCDCTL) register. The following table shows the active external signals when this mode is active:

Table 20-2. Operation Modes Supported by Raster Controller

Interface	Data Bus Width	RASTERCTRL [9, 7, 1]	Signal Name	Description
Passive (STN) Mono 4-bit	4	001	LCDDATA[3:0]	Data Bus
			LCDCP	Pixel Clock
			LCDLP	Horizontal clock (line clock)
			LCDFP	Vertical clock (frame clock)
			LCDAC	AC Bias
			LCDMCLK	Not used
Passive (STN) Mono 8-bit	8	101	LCDDATA[7:0]	Data Bus
			LCDCP	Pixel Clock
			LCDLP	Horizontal clock (line clock)
			LCDFP	Vertical clock (frame clock)
			LCDAC	AC Bias
			LCDMCLK	Not used

Table 20-2. Operation Modes Supported by Raster Controller (continued)

Interface	Data Bus Width	RASTERCTRL [9, 7, 1]	Signal Name	Description
Passive (STN) Color	8	100	LCDDATA[7:0]	Data Bus
			LCDCP	Pixel Clock
			LCDLP	Horizontal clock (line clock)
			LCDFP	Vertical clock (frame clock)
			LCDAC	AC Bias
			LCDMCLK	Not used
Active (TFT) Color	16	x10	LCDDATA[15:0]	Data Bus
			LCDCP	Pixel Clock
			LCDLP	Horizontal clock (line clock)
			LCDFP	Vertical clock (frame clock)
			LCDAC	AC Bias
			LCDMCLK	Not used

20.3.4.1 Logical Data Path

The block diagram of the Raster controller is shown in [Figure 20-1](#). [Figure 20-3](#) illustrates its logical data path for various operation modes (passive [STN] versus active [TFT] and various BPP size). [Figure 20-3](#) shows that:

- The grayscale and serializer and output FIFO blocks are bypassed in active (TFT) modes.
- The palette is bypassed in both 12- and 16-BPP modes.

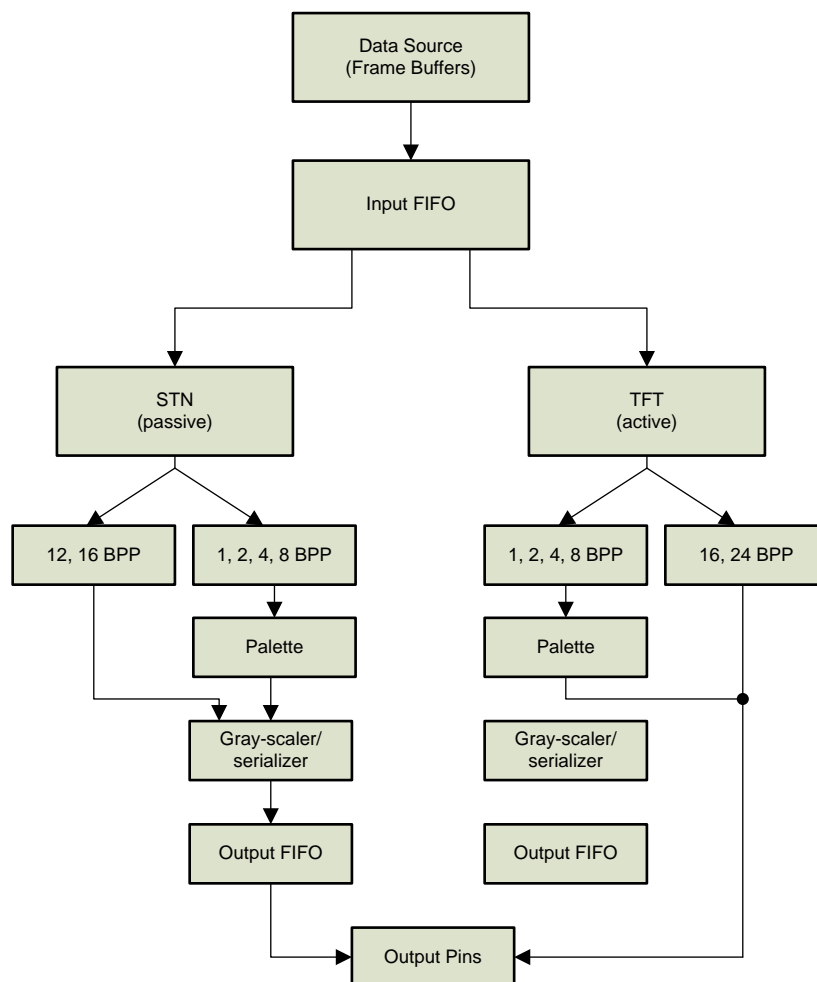


Figure 20-3. LCD Raster Data Path

In summary:

- The display image is stored in frame buffers.
- The built-in DMA engine constantly transfers the data stored in the frame buffers to the input FIFO.
- The Raster Controller relays data to the external pins according to the specified format.

20.3.5 LCD Frame Buffer

The LCD controller has the option to use the internal SRAM or external memory through the EPI interface to hold the frame buffer. The frame buffer is a contiguous memory block, storing enough data to fill a full LCD screen. The frame buffer contains the palette look-up table (palette RAM) and the frame data for a given source image. When moving data into the LCD, the PALMODE bit in the LCD Raster Control (LCDRASTRCTL) register can be configured for palette and data loading, palette loading only or data loading only.

NOTE: When using the LCD with EPI to interface to external memory, the external code address space 0x1000.0000 must be selected by programming the ECADR field to 0x1 in the EPI Address Map (EPIADDRMAP) register at EPI offset 0x01C.

20.3.6 Palette RAM

The palette RAM is a programmable cross reference table that maps a small set of input codes into a larger set of output codes. For active matrix displays, the palette RAM programmed in the LCD is used only for color source images. For passive matrix displays, the palette RAM is used for both grayscale and color source images. For color components, the passive matrix mode only supports, 4 bits per color component. One, two, three, and eight bit per pixel (bpp) source images always use the palette. Twelve, 16, and 24 bits per pixel do not use the palette RAM, since raw data transfers occur.

The LCD palette RAM uses a 12-bit output code such that there is a possibility of 4096 color options to choose from. Depending on the bpp frame storage of palette RAM, two, four, 16, or 256 palette colors can be stored in the palette RAM.

20.3.6.1 Palette RAM Structure for 1, 2, and 4 Bits Per Pixel

When palette RAM is enabled for 1-, 2-, and 4-bpp frame storage, the palette buffer is 16 word entries, where the upper half of the word is unused. The first entry of the palette RAM contains the encoding for the palette type (1, 2, 3, 6, or 12, 16, 24) of the stored picture on bits [14:12], as well as the first input code for one palette entry (bits [11:0]). Subsequent entries only contain the input codes for next palette entries. [Figure 20-4](#) shows the frame buffer structure.

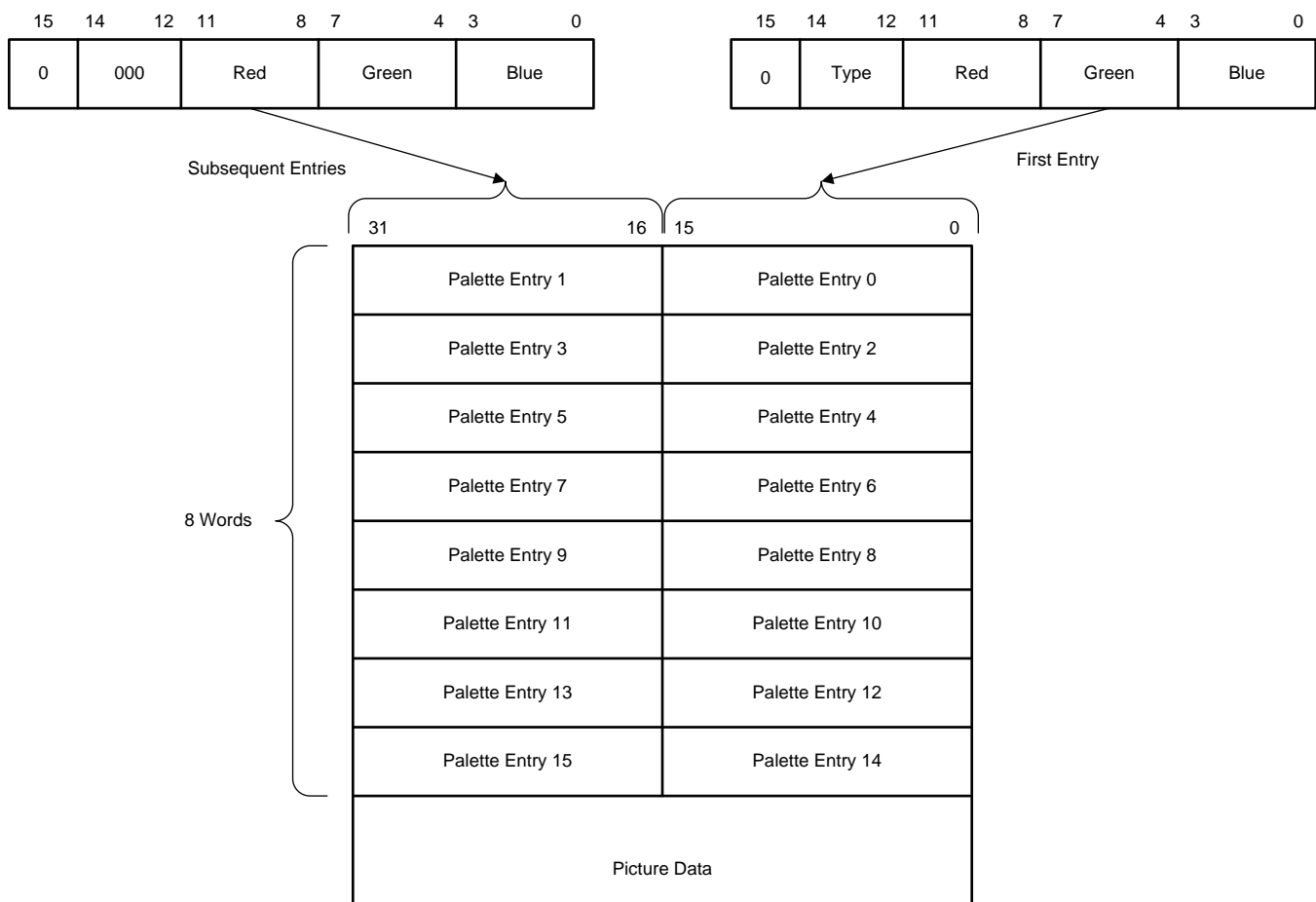


Figure 20-4. Palette RAM Structure for 1, 2, and 4 Bits Per Pixel

For 1-bpp source pictures, the first two palette entries are valid; for 2-bpp sources, the first four palette entry sources are valid; for 4-bpp, all 16 palette entries are valid. Even if the palette entries are not valid for a particular type, there is a fixed number of 8 words reserved for the palette RAM.

The first palette entry represents the RGB value for the source pixel encoded as 0x0. The second palette entry represents the RGB value for the source pixel encoded as 0x1, and so on, such that the sixteenth palette entry is used to represent the RGB value for a pixel encoded as 0xF.

Source data encoded as 12, 16, or 24 bpp do not use the palette RAM since the encoded data already represents the actual RGB values. However, 16 palette RAM entries still precede the picture data in this case and the first palette entry must still define the bpp in bits [14:12]. The type field encoding is defined below in [Table 20-3](#)

Table 20-3. Type Encoding in Palette Entry 0 Buffer

First Palette RAM Entry [14:12]	Source Image Type
0x0	1 bpp
0x1	2 bpp
0x2	4 bpp
0x3	8 bpp
0x4	12, 16, or 24 bpp

20.3.6.2 Palette RAM Structure for 8 bpp

For image sources that are 8 bpp, 256 palette entries (2^8) are required. This structure of the palette RAM is similar to the one described for 1, 2, 4, or 4 bpp except that the palette requires 256 entries (128 words), instead of 16. The first palette entry again contains the "type" definition of the picture along with the RGB values in bits [11:0].

20.3.6.2.1 LCD RAM Configuration

The LCD RAM can be configured to hold palette only, picture data only or both palette and picture data. Depending on the mode, the RAM is used differently:

- **Palette only:** The PALMODE field in the LCD Raster Control (LCDRASTRCTL) register is programmed to 0x01. If the type field in the first palette entry is for 1, 2, 4, or 12, 16, and 24 bpp, an 8-word palette buffer is read from DDR by the DMA. If the bpp field in the first entry is 8 bpp, a 128-word palette is loaded by the DMA. Note that the DMA Frame Buffer n Base Address (LCDDMABAn) and LCD DMA Frame Buffer n Ceiling Address (LCDMACAn) registers must encompass the palette information size. Note that although an 8-word buffer is created for 12, 16, and 24 bpp, it is ignored during data transfers.
- **Data only:** When PALMODE is programmed to 0x10, the bpp field from the last palette read is used. Thus, a single palette in DDR can be associated with multiple stored pictures in DDR, enabling the palette loading to occur once (PALMODE = 0x01) and subsequent data-only frame buffers to be transferred by the DMA into the RAM (PALMODE = 0x10). For 12, 16, and 24-bpp frames the bpp field in the previously read palette RAM buffer is irrelevant.
- **Palette and data:** When the PALMODE encoding is 0x00, the frame buffer is comprised of palette data followed immediately by pixel information. For 12, 14, and 24 bit (packed or unpacked) raw formats, an 8 word palette is prepended to the pixel data. Thus, palette information will be read from DDR even though it is not applicable data and the configuration will look similar to [Figure 20-4](#).

20.3.6.2.2 Raw Data and Color Component Formats

Depending on the value programmed in the raster control register you can have different color formats. Raw data formats are used for active matrix panels. The LCD controller supports 24-bpp, 16-bpp, and 12-bpp raw formats.

NOTE: The TYPE code in the first palette entry must be set to 1XX for 12, 16, and 24-bpp raw data formats.

20.3.6.2.2.1 Data Format for 24 bpp

For raw formats used in active matrix panels, the palette RAM provides a 12-bpp output, or a color depth of 12 bits (4096 colors per pixel). Data formats of 24 bpp can be stored in packed or unpacked format. In packed format, four 24 bit pixels are stored in 3 contiguous 32-bit words (also called a quad-pixel triplet) in DDR, as show in [Figure 20-5](#).

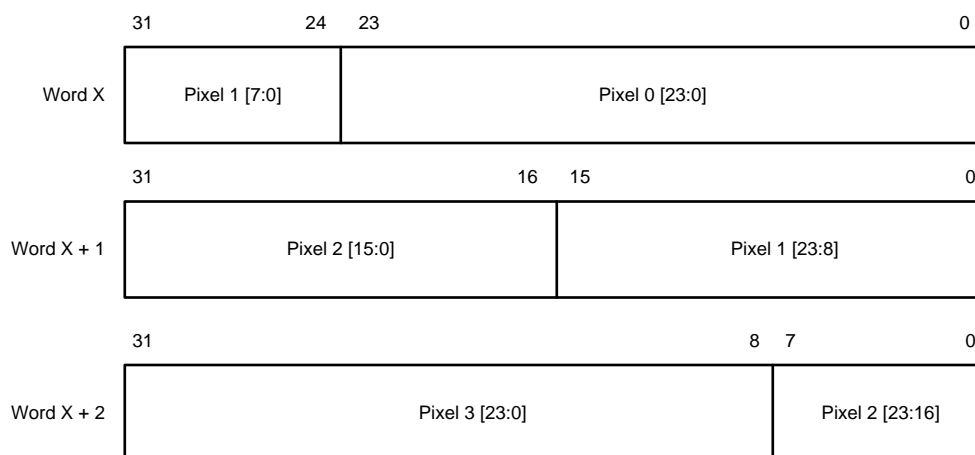


Figure 20-5. 24-bpp Packed Data Format

When using a 24-bit packed format, four consecutive pixels provide three word-aligned accesses for the DMA. If the MSBPPL and PPL bits in the LCDRASTRIM0 register are programmed to a multiple of 16, then the MSBLPP and LPP bits of the LCDRASTRIM1 and LCDRASTRIM2 register must follow a multiple of 16 so the DMA can access the data using word-aligned accesses.

When LCDTFT = 0x1, TFT24UPCK = 0x1 and TFT24 = 0x1 in the LCDRASTRCTL register, a 24bit unpacked format is used. In this mode, each 24-bpp data is stored in a 32-bit word, with the upper 8 bits unused. [Figure 20-6](#) shows the unpacked implementation.

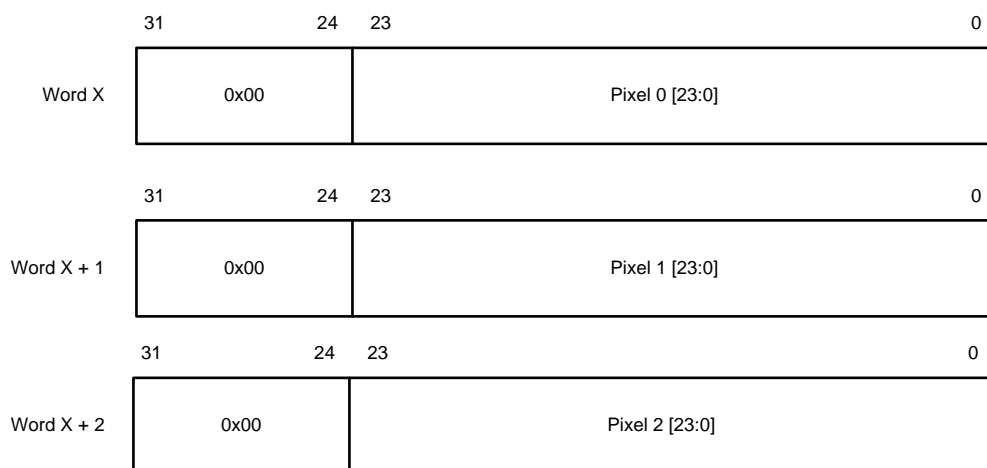


Figure 20-6. 24-bpp Unpacked Data Format

For 24-bpp color components, data is stored in the DDR in B-G-R order with each color representing 8 bits. When the data is transferred to the LCD pixel data bus (LCDDATA[23:0]), the output is transposed such that a portion of the color components are now displayed on LCDDATA [15:0] in RGB565 format. The three least significant bits of the RGB 24-bit data are prepended to RGB565 data on LCDDATA[23:15]. [Figure 20-7](#) shows the transposition of data.

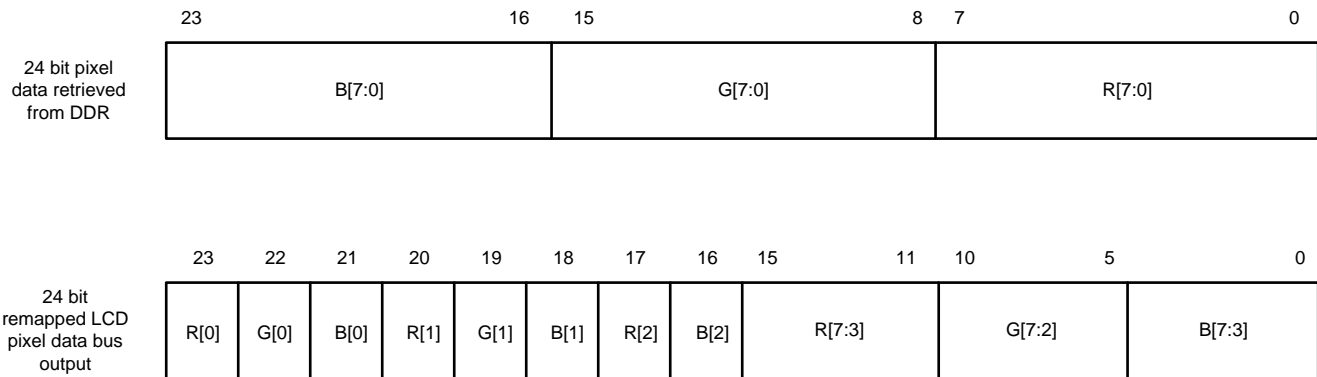


Figure 20-7. 24-bpp Color RGB Remapping on LCDDATA[23:0]

20.3.6.2.2.2 Data Format for 16 bpp

The LCD controller is configured to support 16-bpp raw data format when TFT24 = 0x0, LCDTFT = 0x1, TFTMAP = 0x0 in the LCDRASTRCTL register. Two 16-bpp raw data pixels can be stored in one word in DDR (see [Figure 20-8](#)). For color component ordering, pixel data is in a RGB565 configuration and is output on the LCD pixel data bus LCDDATA [15:0] in the same order. LCDDATA bits [23:16] are undefined in 16-bpp mode. [Figure 20-9](#) shows the configuration of the pixel data retrieved from memory and LCD bus output.

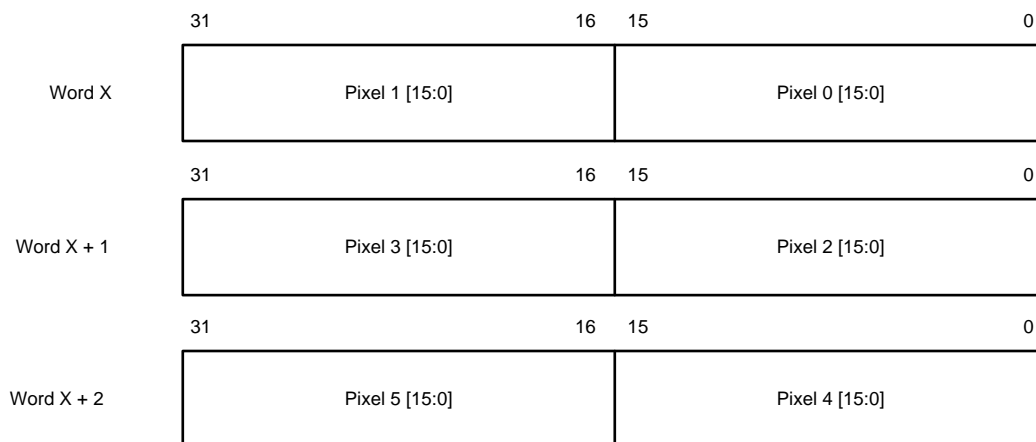


Figure 20-8. 16-bpp Data Format

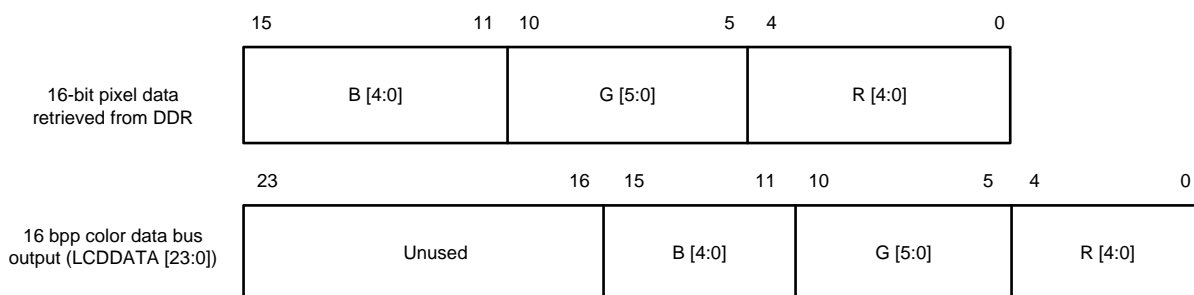


Figure 20-9. 16-bpp Color Component Ordering

20.3.6.2.2.3 Data Format for 12 bpp

Two pixel data are stored in each 32-bit word in memory. The upper four bits in each 16-bit halfword are unused. Figure 20-10 shows the 12-bpp data format.

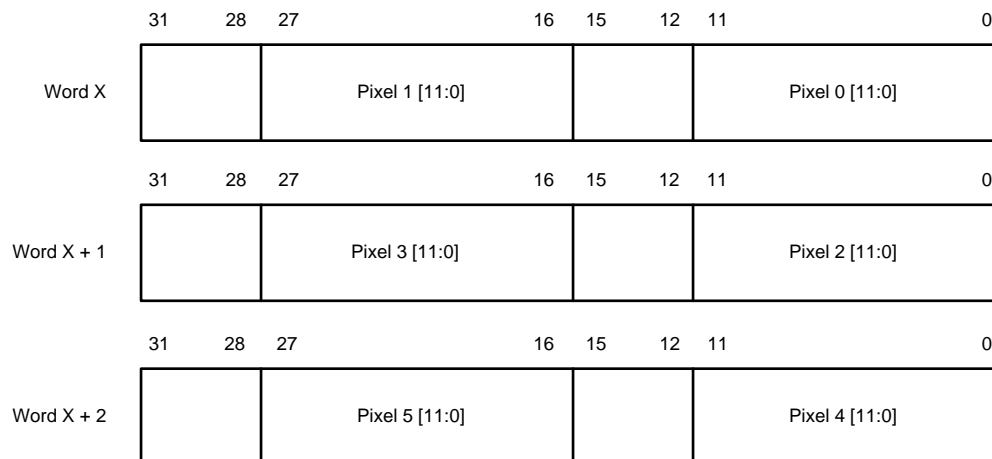


Figure 20-10. 12-bpp Data Format

12-bpp color format has 4 pixels per red, green and blue color component. The BGR configuration is output on LCDDATA [11:0], as shown in Figure 20-11. LCDDATA [23:12] are unused.

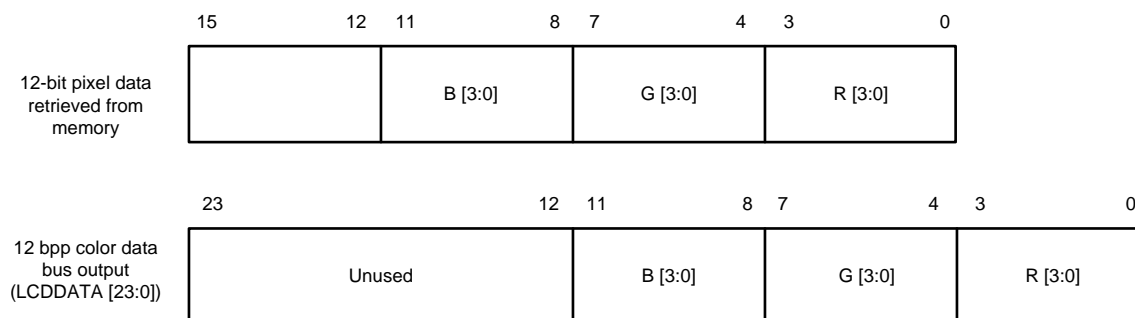


Figure 20-11. 12-bpp Color Component Ordering

20.3.6.2.2.4 1-, 2-, 4-, and 8-bpp Data Format

The palette frame buffer for 1, 2, 4, and 8-bpp data supports passive and active matrix displays. Active matrix is enabled by setting the TFT bit in the LCD Raster Control (LCDRASTRCTL) register. When the TFTMAP is set to 0, the output from the palette look-up is a 12-bpp BGR pixel as shown in Figure 20-4, where each color is a 4-bit value. This output is sent to LCDDATA [11:0]. By setting TFTMAP to 1, the pixel data output is converted to a 16-bit BGR565 format. The extra least significant are dithered by replication of the component values' most significant bits. Figure 20-12 shows this conversion.

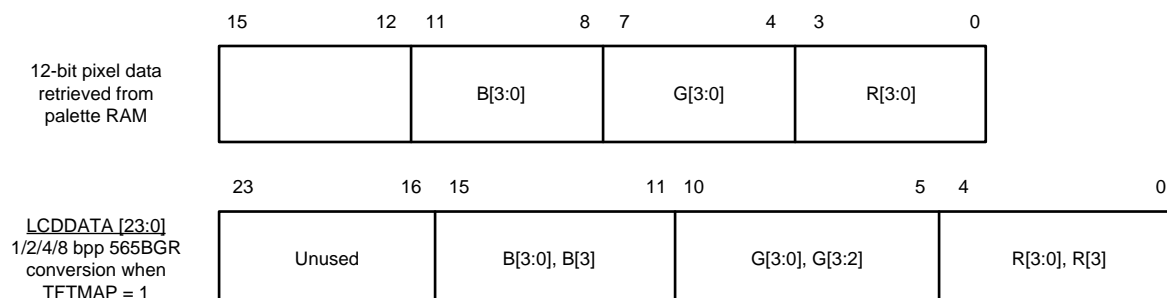


Figure 20-12. 1-, 2-, 4-, and 8-bpp Dither Output

20.3.7 Palette

As explained in [Section 20.3.6](#), the pixel data is an index of palette entry (when palette is used). The number of colors supported is given by $2^{\text{number of BPP}}$. However, due to a limitation of the grayscale and serializer block, fewer grayscales or colors may be supported.

The PALMODE field in the LCD Raster Control (LCDRASTRCTL) register affects the palette loading:

- If PALMODE is 0x0 (palette-plus-data mode) or 0x1 (palette-only mode), the palette is loaded by the DMA engine at the very beginning, which is followed by the loading of pixel data.
- If PALMODE is 0x2 (data-only mode), the palette is not loaded. Instead, the DMA engine immediately loads the pixel data.

20.3.8 Grayscale and Serializer – Passive (STN) Mode

Once a palette entry is selected from the look-up palette by the pixel data, its content is sent to the grayscale/ serializer. If it is monochrome data, it is encoded as 4 bits. If it is color data, it is encoded as 4 bits (Red), 4 bits (Green), and 4 bits (Blue). These 4-bit values are used to select one of the 16 intensity levels, as shown in [Table 20-4](#). A patented algorithm is used during this processing to provide an optimized intensity value that matches the eye's visual perception of color/gray gradations.

20.3.9 Grayscale and Serializer – Active (TFT) Mode

The grayscale and serializer is bypassed in this mode.

20.3.10 Color and Grayscale Intensities and Modulation Rates

[Table 20-4](#) lists the intensities and modulation rates for varying dither values.

Table 20-4. Intensities and Modulation Rates

Dither Value (4-Bit Value from Palette)	Intensity (0% is White)	Modulation Rate (Ratio of ON to ON + OFF Pixels)
0x0	0.0%	0
0x1	14.3%	1 / 7
0x2	20.0%	1 / 5
0x3	25%	1 / 4
0x4	33.3%	3 / 9
0x5	40.0%	2 / 5
0x6	44.4%	4 / 9
0x7	50.0%	1 / 2
0x8	55.6%	5 / 9
0x9	60.0%	3 / 5
0xA	66.6%	6 / 9
0xB	75%	3 / 4
0xC	80.0%	4 / 5
0xD	85.7%	6 / 7
0xE	93.3%	14 / 15
0xF	100.0%	1

20.3.11 Summary of Color Depth

[Table 20-5](#) summarizes the color depths depending on the number of BPP.

Table 20-5. Number of Colors and Shades of Gray Available on Screen

Passive Mode (LCDTFT = 0)		Active Mode (LDCTFT = 1)	Color Only (LCDBW = 0)
Bits	Monochrome (LCDBW = 1)	Color (LDCBW = 0)	
1	2 palette entries to select within 15 grayscales	2 palette entries to select within 3375 possible colors	2 palette entries to select within 4096 possible colors
2	4 palette entries to select within 15 grayscales	4 palette entries to select within 3375 possible colors	4 palette entries to select within 4096 possible colors
4	16 palette entries to select within 15 grayscales	16 palette entries to select within 3375 possible colors	16 palette entries to select within 4096 possible colors
8	Not relevant because it would consist in 256 palette entries to select within 15 grayscales, but exists anyways	256 palette entries to select within 3375 possible colors	256 palette entries to select within 4096 possible colors
12	–	3375 possible colors	4096 possible colors
16	–	3375 possible colors (STN_565 = 1)	Up to 65536 possible colors
24	–	–	Up to 16.7 million colors

20.3.12 Output Format

The following section describes the output format for both passive and active LCD modes.

20.3.12.1 Passive (STN) Mode

As shown in [Figure 20-13](#), the pixel data stored in frame buffers go through palette (if applicable) and grayscale and serializer before reaching the output FIFO. As a result, it is likely that the data fed to the output FIFO is numerically different from the data in the frame buffers. (However, they represent the same color or grayscale.)

The output FIFO formats the received data according to display modes. [Figure 20-13](#), shows the actual data output on the external pins.

20.3.12.2 Active (TFT) Mode

As shown in [Figure 20-13](#), the grayscale and serializer and output FIFO are bypassed in active (TFT) mode. Namely, at each pixel clock, one pixel data (16 bits) is output to the external LCD.

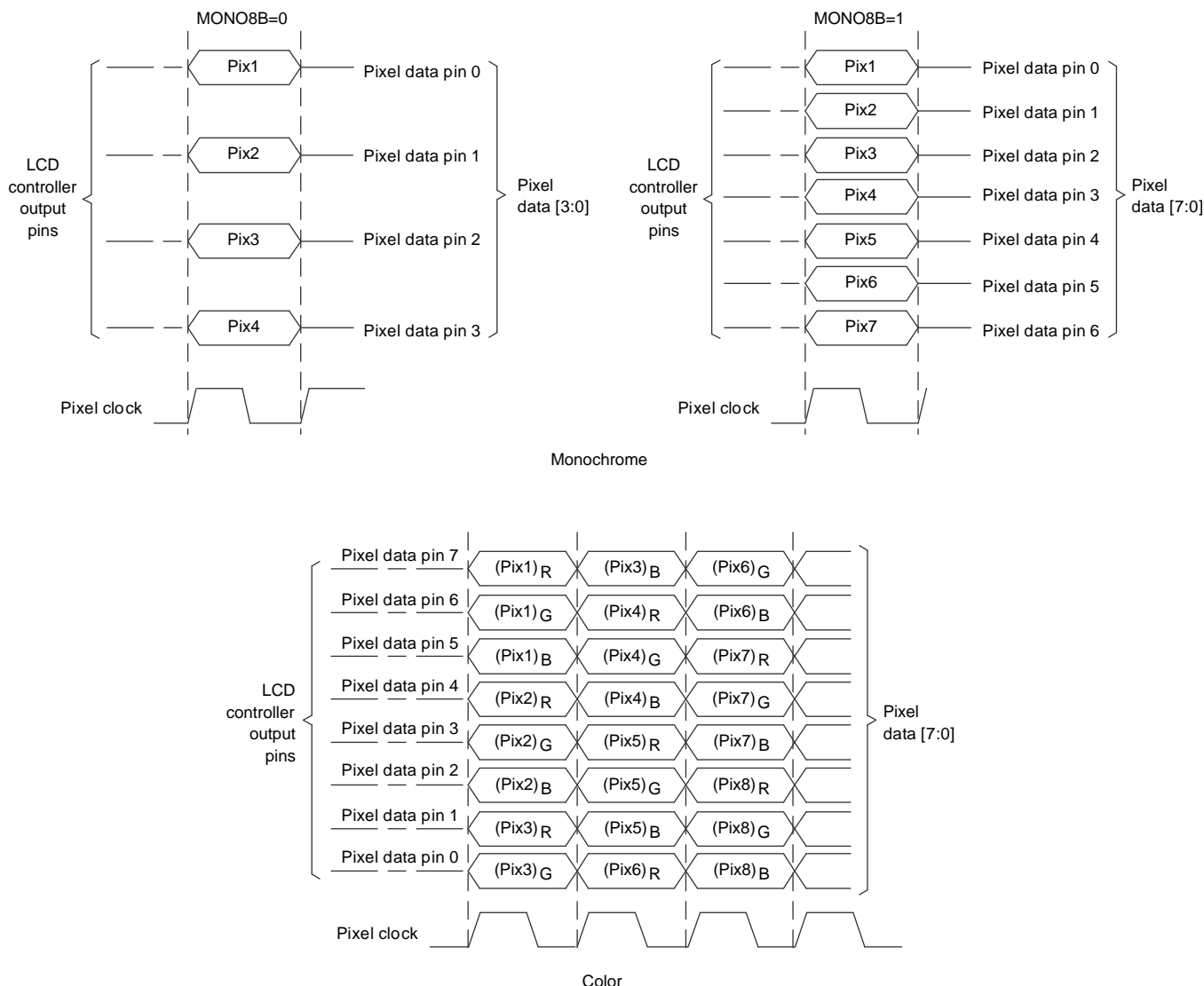


Figure 20-13. Monochrome and Color Output

20.3.13 Subpicture Feature

A feature exists in the LCD to cover either the top or lower portion of the display with a default color. This feature is called a subpicture and is illustrated in [Figure 20-14](#). Subpictures are only allowed for Active Matrix Mode (when LCDTFT = 0x1 in the LCD Raster Control (LCDRASTRCTL) register).

Subpicture reduce the bandwidth since lines containing default pixel data are not read from memory. For example, suppose the panel has 100 lines of which 50 are default pixel data lines. Then, only 50 lines of data are written by the DMA from DDR for this subpicture setup. That is, the LCD DMA Frame Buffer n Base Address (LCDDMABAFBn) and LCD DMA Frame Buffer n Base Address (LCDDMABAFBn) registers only encompass 50 lines of data instead of 100.



Figure 20-14. Example of Subpicture

The subpicture feature is enabled when the SPEN control bit in the LCD Raster Subpanel Display n (LCDRASTRSUBPn) register is set to 1. The HOLS bit, when programmed to 0, puts the Default Pixel Data lines at the top of the screen and the active video lines at the bottom of the screen. When the HOLS bit is set to 1, Active video lines are at the top of the screen and Default Pixel Data lines are at the bottom of the screen. The HOLS bit behavior is shown in [Figure 20-15](#).

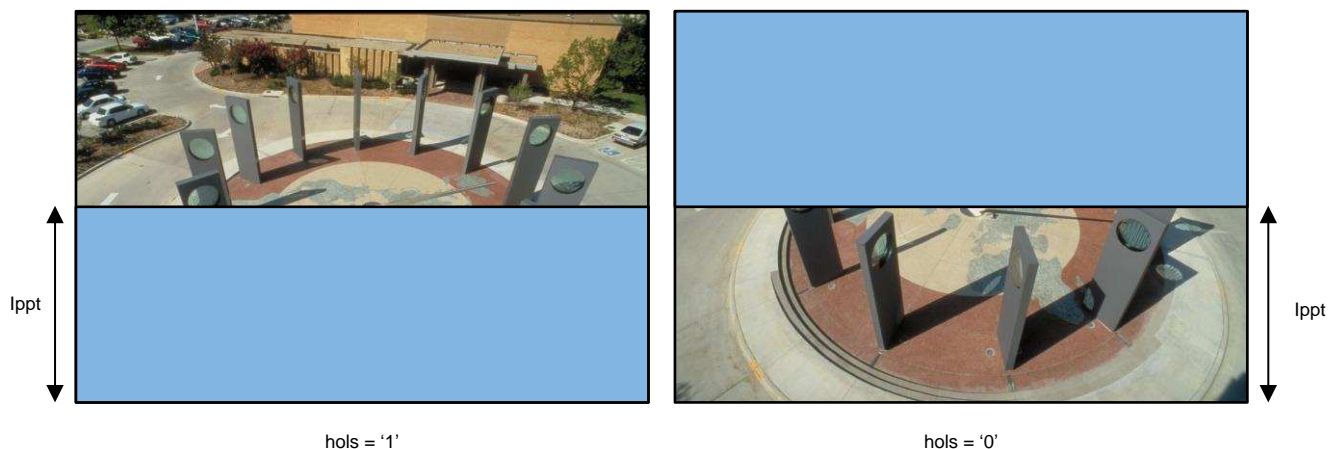


Figure 20-15. Subpicture Output With HOLS Bit Variations

The LPPT (lines per panel threshold) bit field defines the number of lines at the bottom of the picture for both HOLS = 1 or 0. LPPT is an encoded value in the range [0:2047] used to represent the number of lines in the range [1:2048].

20.4 Interrupts

The application can program the LCD controller to trigger an interrupt for the following events:

- DMA end-of-frame 1: The DMA end-of-frame 1 (EOF1) interrupt is triggered when the DMA module

has completed transferring the contents of a bounded frame buffer 1.

- **DMA end-of-frame 0:** The DMA end-of-frame 0 (EOF0) interrupt is triggered when the DMA module has completed transferring the contents of a bounded frame buffer 0.
- **DMA palette loaded:** When PALMODE is set to Palette-only or Palette+data, the palette loaded interrupt is triggered when the palette portion of the DMA transfer has been stored in the Palette RAM.
- **DMA FIFO Underflow:** The FIFO Underflow (FIFOU) interrupt is triggered when the real-time output needs to send a value for pixel data but one cannot be found in the FIFO.
- **AC Bias Count Decrement to Zero:** For Passive Matrix displays, a count can be kept of the number of times the AC Bias line toggles. Once the specified number of transitions has been seen, the AC Bias Count (ACBS) interrupt is triggered. The module does not post any further interrupts or keep counting AC Bias transitions until the interrupt has been cleared.
- **Frame synchronization lost:** When the DMA module reads a frame buffer and stores it in the FIFO, it sets a start frame and an end frame indicator embedded with the data. On retrieving the data from the FIFO, the Sync Lost (SYNCS) interrupt is triggered if the start indicator is not found at the first pixel of a new frame.
- **Recurrent Raster mode frame done:** In raster mode, the Recurrent Frame Done (RRASTRDONE) interrupt is triggered each time a complete frame has been sent to the interface pins.
- **Raster or LIDD frame done (shared interrupt):** In LIDD DMA mode, a frame buffer of data is sent. When the frame buffer has completed, the LIDD Frame Done (DONE) interrupt is triggered. In order to do another LIDD DMA, the DMA engine must be disabled and then re-enabled.

In Raster mode, the interrupt is triggered after LCDEN bit is programmed to 0 in the LCD Raster Control (LCDRASTRCTL) register and after the last frame is sent to the pins. After the Raster mode DMA is running, the interrupt occurs only once after the module is disabled.

The interrupts are enabled in the LCD Interrupt Mask register (LCDIM) register. All pending interrupts in the LCD module must be serviced by the Host's Interrupt Service Routine before it exits.

20.5 Bus Transaction Modes

The LCD Controller supports character-based LCD panels by ensuring the bus cycles for reading/writing commands and data are met. The LIDD (LCD Interface Display Driver) provides this support. The LIDD interfaces with an Arm CPU in the SoC using a VBUS slave port. Abundant programmable timing parameters can be configured such that the off-chip interface can support a wide variety of character based LCD panels. Alternatively, the DMA module can be used to mimic the CPU and perform a sequence of write-only data bus transactions to the character based LCD panel. The Motorola and Intel interfaces support asynchronous and synchronous modes. The only difference between the two is that, for synchronous mode, the internal bus clock is output on a pin that replaces CS1. In other words, only one character panel can be supported under synchronous mode while two panels can be supported for asynchronous mode.

20.6 Initialization and Configuration

Basic configuration of the LCD controller is as follows:

1. Enable the LCD Controller in the System Control module using the RCGCLCD register. See [Section 4.2.103](#).
2. Enable the DMA, core and LIDD controller (if required) in the LCDCLKEN register.
3. Select the LCD mode (LIDD or Raster) by programming the LCDMODE bit in the LCDCTL register. In addition, configure the LCDCP or LCDMCLK frequency through the CLKDIV field.

When operating in LIDD mode:

1. Program the LCDLIDDCTL register along with the chip select configuration register, LIDDCSnCFG.
 2. Select the DMA parameters in the LCDDMACTL register. Program the frame buffer boundaries in the LCDDMABAFB0 and LCDDMACAFB0 registers.
 3. Enable any required interrupts in the LCDIM register.
 4. Initiate LIDD mode transactions by setting the DMAEN bit in the LCDLIDDCTL register.
4. When operating in Raster mode:

1. Program the LCDRASTRCTL register along with the raster timings in the LCDRASTRTIMn registers.
2. Configure the subpanel display in the LCDRASTRSUBPn registers.
3. Select the DMA parameters such as burst size, and memory layout in the LCDDMACTL register. Program the frame buffer boundaries in the LCDDMABAFBn and LCDDMACAFBn registers.
4. Enable any required interrupts in the LCDIM register.
5. Initiate Raster mode transactions by setting the LCDEN bit in the LCDRASTRCTL register.
5. The LCD DMA may interface to internal SRAM or the external memory through EPI for transfer of data. To configure the EPI for use with LCD, see [Chapter 16](#).

NOTE: When using the LCD with EPI to interface to external memory, the external code address space 0x1000.0000 must be selected by programing the ECADR field to 0x1 in the EPI Address Map (EPIADDRMAP) register at EPI offset 0x01C.

20.7 LCD Registers

Table 20-6 lists the memory-mapped registers for the LCD. All register offset addresses not listed in Table 20-6 should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the LCD module base address of 0x44050000.

Table 20-6. LCD Registers

Offset	Acronym	Register Name	Section
0x0	LCDPID	LCD PID Register Format	Section 20.7.1
0x4	LCDCTL	LCD Control	Section 20.7.2
0xC	LCDLIDDCCTL	LCD LIDD Control	Section 20.7.3
0x10	LIDDCS0CFG	LCD LIDD CS0 Configuration	Section 20.7.4
0x14	LIDDCS0ADDR	LIDD CS0 Read/Write Address	Section 20.7.5
0x18	LIDDCS0DATA	LIDD CS0 Data Read/Write Initiation	Section 20.7.6
0x1C	LIDDCS1CFG	LIDD CS1 Configuration	Section 20.7.7
0x20	LIDDCS1ADDR	LIDD CS1 Address Read/Write Initiation	Section 20.7.8
0x24	LIDDCS1DATA	LIDD CS1 Data Read/Write Initiation	Section 20.7.9
0x28	LCDRASTRCTL	LCD Raster Control	Section 20.7.10
0x2C	LCDRASTRTIM0	LCD Raster Timing 0	Section 20.7.11
0x30	LCDRASTRTIM1	LCD Raster Timing 1	Section 20.7.12
0x34	LCDRASTRTIM2	LCD Raster Timing 2	Section 20.7.13
0x38	LCDRASTRSUBP1	LCD Raster Subpanel Display 1	Section 20.7.14
0x3C	LCDRASTRSUBP2	LCD Raster Subpanel Display 2	Section 20.7.15
0x40	LCDDMACTL	LCD DMA Control	Section 20.7.16
0x44	LCDDMABAFB0	LCD DMA Frame Buffer 0 Base Address	Section 20.7.17
0x48	LCDDMACAFB0	LCD DMA Frame Buffer 0 Ceiling Address	Section 20.7.18
0x4C	LCDDMABAFB1	LCD DMA Frame Buffer 1 Base Address	Section 20.7.19
0x50	LCDDMACAFB1	LCD DMA Frame Buffer 1 Ceiling Address	Section 20.7.20
0x54	LCDSYSCFG	LCD System Configuration Register	Section 20.7.21
0x58	LCDRISSET	LCD Interrupt Raw Status and Set Register	Section 20.7.22
0x5C	LCDMISCLR	LCD Interrupt Status and Clear	Section 20.7.23
0x60	LCDIM	LCD Interrupt Mask	Section 20.7.24
0x64	LCDIENC	LCD Interrupt Enable Clear	Section 20.7.25
0x6C	LCDCLKEN	LCD Clock Enable	Section 20.7.26
0x70	LCDCLKRESET	LCD Clock Resets	Section 20.7.27

Complex bit access types are encoded to fit into small table cells. Table 20-7 shows the codes that are used for access types in this section.

Table 20-7. LCD Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.7.1 LCDPID Register (Offset = 0x0) [reset = X]

LCD PID Register Format (LCDPID)

This register contains information regarding the Peripheral ID (PID) of the LCD module.

LCDPID is shown in [Figure 20-16](#) and described in [Table 20-8](#).

Return to [Summary Table](#).

Figure 20-16. LCDPID Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED					MAJOR		
R-0x0					R-X		
7	6	5	4	3	2	1	0
RESERVED		MINOR					
R-0x0		R-X					

Table 20-8. LCDPID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0x0	
10-8	MAJOR	R	X	Major release number. This field contains the major release number for the IP.
7-6	RESERVED	R	0x0	
5-0	MINOR	R	X	Minor release number. This field contains the minor release number for the IP.

20.7.2 LCDCTL Register (Offset = 0x4) [reset = 0x0]

LCD Control (LCDCTL)

The LCD Control (LCDCTL) register configures the mode, clock frequencies, and restart behavior of the LCD Controller.

LCDCTL is shown in [Figure 20-17](#) and described in [Table 20-9](#).

Return to [Summary Table](#).

Figure 20-17. LCDCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
CLKDIV							
R/W-0x0							
7	6	5	4	3	2	1	0
RESERVED							LCDMODE
R-0x0							R/W-0x0

Table 20-9. LCDCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-8	CLKDIV	R/W	0x0	<p>Clock Divisor.</p> <p>This field contains a value (from 0 to 255) used to specify the frequency of the pixel clock, LCDCP (in Raster Mode) or MCLK (in LIDD mode).</p> <p>The equation for the two clock outputs is $\text{SYSCLK}/\text{CLKDIV}$, where: LCDCP can range from $\text{SYSCLK}/2$ to $\text{SYSCLK}/255$. $\text{CLKDIV} = 0x0$ or $\text{CLKDIV} = 0x1$ are not allowed.</p> <p>MCLK can vary from SYSCLK to $\text{SYSCLK}/255$ (using $\text{CLKDIV} = 0x0$ or $\text{CLKDIV} = 0x1$ sets $\text{MCLK} = \text{SYSCLK}$).</p> <p>See Table 20-10 for f_{SYSCLK} to f_{LCDCP} frequency conversions based on CLKDIV value.</p>
7-1	RESERVED	R	0x0	
0	LCDMODE	R/W	0x0	<p>LCD Mode Select.</p> <p>0x0 = LCD Controller is operating in LIDD Mode</p> <p>0x1 = LCD Controller is operating in Raster Mode</p>

**Table 20-10. SYSCLK to Pixel Clock (LCDP)
Frequency Conversion Table**

f_{SYSCLK} (MHz)	CLKDIV Value	f_{LCDP} (MHz)
120	2	60
	3	40
	4	30
	5	24
	6	20
	7	17.1428
	8	15

	N	f_{SYSCLK}/N
80	2	40
	3	26.666
	4	20
	5	16
	6	13.333
	7	11.428

	N	f_{SYSCLK}/N
60	2	30
	3	20
	4	15
	5	12
	6	10
	N	f_{SYSCLK}/N

20.7.3 LCDLIDDCTL Register (Offset = 0xC) [reset = 0x0]

LCD LIDD Control (LCDLIDDCTL)

This register configures the functionality of the LCD controller interface when it is programmed to function in LIDD mode.

LCDLIDDCTL is shown in [Figure 20-18](#) and described in [Table 20-11](#).

Return to [Summary Table](#).

Figure 20-18. LCDLIDDCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						DMACS	DMAEN
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
CS1E1	CS0E0	WRDIRINV	RDEN	ALE		MODE	
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0	

Table 20-11. LCDLIDDCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	DMACS	R/W	0x0	CS0/CS1 Select for LIDD DMA writes. In Motorola 6800 or Intel 8080 synchronous modes, CS1 is not used and thus, DMACS should not be set to 0x1. 0x0 = DMA writes to LIDD CS0 (LCDAC). 0x1 = DMA writes to LIDD CS1 (LCDMCLK).
8	DMAEN	R/W	0x0	LIDD DMA enable. 0x0 = Deactivate DMA control of LIDD interface. DMA control is released upon completion of transfer of the current frame of data (LIDD Frame Done) when this bit is clear. The microcontroller has direct read and write access to the panel in this mode. 0x1 = Activate DMA to drive LIDD interface to support streaming data to smart panels. The microcontroller cannot access the panel directly in this mode. Note: When the DMA is enabled, do not read or write the LCD registers for the base and ceiling addresses (LCDDMABAFB0, LCDDMACAFB0, LCDDMABAFB1, and LCDDMACAFB1) with the CPU.
7	CS1E1	R/W	0x0	Chip Select 1 (CS1) / Enable 1 (E1) Polarity Control. CS1 is active low by default. EN1 is active high by default. 0x0 = CS1/E1 (LCDMCLK) are not inverted. CS1 remains active low and E1 remains active high. 0x1 = Invert CS1/E1 (LCDMCLK). CS1 is active high and E1 is active low.
6	CS0E0	R/W	0x0	Chip Select 0 (CS0) / Enable 0 (E0) Polarity Control. CS0 is active low by default. E0 is active high by default. 0x0 = CS0/E0 (LCDAC) are not inverted. CS0 remains active low and E0 remains active high. 0x1 = Invert CS0/E0. CS0 is active high and E0 is active low.

Table 20-11. LCDLIDDCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	WRDIRINV	R/W	0x0	Write Strobe (WR) / Direction (DIR) Polarity Control. WR active low by default. DIR is write low/read high by default. 0x0 = WR/DIR (LCDLP) are not inverted. WR remains active low and DIR is write low/read high. 0x1 = Invert WR/DIR. WR is active high and DIR is write high/read low.
4	RDEN	R/W	0x0	Read Strobe (RD) / Direct Enable (EN) Polarity Control. RD is active low by default. EN is active high by default. 0x0 = RD/EN (LCDCP) are not inverted. RD is active low and EN is active high. 0x1 = Invert RD/EN (LCDCP). RD is active high and EN is active low.
3	ALE	R/W	0x0	Address Latch Enable (ALE) Polarity Control. ALE is active low by default. 0x0 = ALE (LCDFP) is not inverted. 0x1 = Invert ALE (LCDFP). ALE is active high.
2-0	MODE	R/W	0x0	LIDD Mode Select. Selects type of LCD display interface for the LIDD to drive. See for external pin assignments based on MODE encoding. 0x0 = Synchronous Motorola 6800 Mode 0x1 = Asynchronous Motorola 6800 Mode 0x2 = Synchronous Intel 8080 mode 0x3 = Asynchronous Intel 8080 mode 0x4 = Asynchronous Hitachi mode 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved

The MODE bit field described above defines the function of the LCD external pins as follows:

Table 20-12. LCD Alternate Signal Functions in LIDD Mode

Pin	MODE = 0x0	MODE = 0x1	MODE = 0x2	MODE = 0x3	MODE = 0x4
LCDCP	RS/WS	RS/WS	RS	RS	N/A
LCDLP	DIR	DIR	WS	WS	DIR
LCDFP	ALE	ALE	ALE	ALE	ALE
LCDAC	CS0	CS0	CS0	CS0	E0
LCDMCLK	MCLK	CS1	MCLK	CS1	E1

20.7.4 LIDDCS0CFG Register (Offset = 0x10) [reset = 0x00440044]

LCD LIDD CS0 Configuration (LIDDCS0CFG)

The LIDD CS0 Configuration (LIDDCS0CFG) register defines the timings for the read and write strobes with respect to CS0.

LIDDCS0CFG is shown in [Figure 20-19](#) and described in [Table 20-13](#).

Return to [Summary Table](#).

Figure 20-19. LIDDCS0CFG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRSU					WRDUR					WRHOLD					RDSU
R/W-0x0					R/W-0x2					R/W-0x2					R/W-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDSU					RDDUR					RDHOLD					GAP
R/W-0x0					R/W-0x1					R/W-0x1					R/W-0x0

Table 20-13. LIDDCS0CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	WRSU	R/W	0x0	Write strobe (WR) setup cycles. When performing a write access, this field defines the number of MCLK cycles that the LCDDATA bus, output enable, ADE, DIR and CS0 signals have to be ready before the write strobe. The minimum value is 0x0.
26-21	WRDUR	R/W	0x2	Write strobe (WR) duration cycles. This field value defines the number of MCLK cycles for which the write strobe is held active when performing a write access. The minimum value is 0x1.
20-17	WRHOLD	R/W	0x2	Write strobe (WR) hold cycles. This field value defines the number of MCLK cycles for which the LCDDATA bus, output enable, ALE, DIR and CS0 signals are held after the write strobe is deasserted when performing a write access. The minimum value is 0x1.
16-12	RDSU	R/W	0x0	Read strobe (RD) setup cycles. When performing a read access, this field defines the number of MCLK cycles that the LCDDATA bus, output enable, ALE, DIR and CS0 signals have to be ready before the read strobe is asserted.
11-6	RDDUR	R/W	0x1	Read strobe (RD) duration cycles. This field defines the number of MCLK cycles for which the read strobe is held active when performing a read access. The minimum value is 0x1.
5-2	RDHOLD	R/W	0x1	Read strobe (RD) hold cycles. This field defines of MCLK cycles for which the LCDDATA bus, output enable, ALE, DIR and CS0 signals are held after the read strobe is deasserted when performing a read access. The minimum value is 0x1.
1-0	GAP	R/W	0x0	Field value defines the number of LCDMCLK cycles (GAP +1) between the end of one CS0 (LCDAC) device access and the start of another CS0 (LCDAC) device access unless the two accesses are both reads. In this case, this delay is not incurred. The minimum value is 0x0.

20.7.5 LIDDCS0ADDR Register (Offset = 0x14) [reset = 0x0]

LIDD CS0 Read/Write Address (LIDDCS0ADDR)

This register contains the read and write address of the current access enabled by CS0 (LCDAC).

LIDDCS0ADDR is shown in [Figure 20-20](#) and described in [Table 20-14](#).

Return to [Summary Table](#).

Figure 20-20. LIDDCS0ADDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CS0ADDR															
R-0x0																R/W-0x0															

Table 20-14. LIDDCS0ADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	CS0ADDR	R/W	0x0	<p>LCD address.</p> <p>The LCD Controller supports a shared address and data output bus. A write to this register initiates a bus write transaction. A read from this register initiates a bus read transaction.</p> <p>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the LIDDCTRL register). If the LIDD is being used as a generic bus interface, writing to this register can store CS0ADDR to an external transparent latch holding a 16-bit address.</p> <p>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the command instruction area of the panel.</p>

20.7.6 LIDDCS0DATA Register (Offset = 0x18) [reset = 0x0]

LIDD CS0 Data Read/Write Initiation (LIDDCS0DATA)

This register contains the read and write data of the current access enabled by CS0 (LCDAC).

LIDDCS0DATA is shown in [Figure 20-21](#) and described in [Table 20-15](#).

Return to [Summary Table](#).

Figure 20-21. LIDDCS0DATA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CS0DATA															
R-0x0																R/W-0x0															

Table 20-15. LIDDCS0DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	CS0DATA	R/W	0x0	<p>LCD data read/write.</p> <p>The LCD Controller supports a shared Address/Data output bus. A write to this register initiates a bus write transaction. A read from this register initiates a bus read transaction.</p> <p>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the LIDDCTRL register).</p> <p>If the LIDD is being used as a generic bus interface, writing to this register can store CS0ADDR to an external transparent latch holding a 16-bit address.</p> <p>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the data area of the panel.</p>

20.7.7 LIDDCS1CFG Register (Offset = 0x1C) [reset = 0x00440044]

LIDD CS1 Configuration (LIDDCS1CFG)

The LIDD CS1 Configuration (LIDDCS1CFG) register defines the timings for the read and write strobes with respect to CS1.

LIDDCS1CFG is shown in [Figure 20-22](#) and described in [Table 20-16](#).

Return to [Summary Table](#).

Figure 20-22. LIDDCS1CFG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRSU					WRDUR					WRHOLD					RDSU
R/W-0x0					R/W-0x2					R/W-0x2					R/W-0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDSU					RDDUR					RDHOLD					GAP
R/W-0x0					R/W-0x1					R/W-0x1					R/W-0x0

Table 20-16. LIDDCS1CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	WRSU	R/W	0x0	Write strobe (WR) setup cycles. When performing a write access, this field defines the number of LCDMCLK cycles that Data Bus/Pad Output Enable, ALE, DIR, and CS1 have to be ready before WR (LCDLP) is asserted. The minimum value is 0x0.
26-21	WRDUR	R/W	0x2	Write strobe (WR) duration cycles. Field value defines the number of LCDMCLK cycles for which WR (LCDLP) is held active when performing a write access. The minimum value is 0x1.
20-17	WRHOLD	R/W	0x2	Write strobe (WR) hold cycles. Field value defines the number of LCDMCLK cycles for which Data Bus/Pad Output Enable, ALE, the DIR, and CS1 signals are held after WR (LCDLP) is deasserted when performing a write access. The minimum value is 0x1.
16-12	RDSU	R/W	0x0	Read strobe (RD) setup cycles. When performing a read access, this field defines the number of LCDMCLK cycles that Data Bus/Pad Output Enable, ALE, the DIR, and CS1 signals have to be ready before RD (LCDCP) is asserted.
11-6	RDDUR	R/W	0x1	Read strobe (RD) duration cycles. Field value defines the number of LCDMCLK cycles for which RD (LCDCP) is held active when performing a read access. The minimum value is 0x1.
5-2	RDHOLD	R/W	0x1	Read strobe (RD) hold cycles. Field value defines the number of LCDMCLK cycles for which Data Bus/Pad Output Enable, ALE, DIR, and CS1 signals are held after RD (LCDCP) is deasserted when performing a read access. The minimum value is 0x1.
1-0	GAP	R/W	0x0	Field value defines the number of LCDMCLK cycles (GAP + 1) between the end of one CS1 (LCDAC) device access and the start of another CS0 (LCDAC) device access unless the two accesses are both reads. In this case, this delay is not incurred. The minimum value is 0x0.

20.7.8 LIDDCS1ADDR Register (Offset = 0x20) [reset = 0x0]

LIDD CS1 Address Read/Write Initiation (LIDDCS1ADDR)

This register contains the read and write address of the current access enabled by CS1 (LCDAC).

LIDDCS1ADDR is shown in [Figure 20-23](#) and described in [Table 20-17](#).

Return to [Summary Table](#).

Figure 20-23. LIDDCS1ADDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CS1ADDR															
R-0x0																R/W-0x0															

Table 20-17. LIDDCS1ADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	CS1ADDR	R/W	0x0	<p>LCD Address Bus.</p> <p>The LCD Controller supports a shared Address/Data output bus. A write to this register would initiate a bus write transaction. A read from this register would initiate a bus read transaction.</p> <p>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the LIDDCTRL register). If the LIDD is being used as a generic bus interface, writing to this register can store CS1ADDR to an external transparent latch holding a 16-bit address.</p> <p>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the command instruction area of the panel.</p>

20.7.9 LIDDCS1DATA Register (Offset = 0x24) [reset = 0x0]

LIDD CS1 Data Read/Write Initiation (LIDDCS1DATA)

What is the difference between this register and the above register

LIDDCS1DATA is shown in [Figure 20-24](#) and described in [Table 20-18](#).

Return to [Summary Table](#).

Figure 20-24. LIDDCS1DATA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CS0DATA															
R-0x0																R/W-0x0															

Table 20-18. LIDDCS1DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	CS0DATA	R/W	0x0	<p>LCD Data Read/Write Initiation.</p> <p>The LCD Controller supports a shared Address/Data output bus. A write to this register would initiate a bus write transaction. A read from this register would initiate a bus read transaction.</p> <p>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the LIDDCTRL register). If the LIDD is being used as a generic bus interface, writing to this register can store CS1ADDR to an external transparent latch holding a 16-bit address.</p> <p>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the data area of the panel.</p>

20.7.10 LCDRASTRCTL Register (Offset = 0x28) [reset = 0x0]

LCD Raster Control (LCDRASTRCTL)

The LCD Raster Control (LCDRASTRCTL) register is used to configure the features of Raster mode.

LCDRASTRCTL is shown in [Figure 20-25](#) and described in [Table 20-19](#).

Return to [Summary Table](#).

Figure 20-25. LCDRASTRCTL Register

31	30	29	28	27	26	25	24
RESERVED					TFT24UPCK	TFT24	FRMBUFSZ
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
TFTMAP	NIBMODE	PALMODE		REQDLY			
R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0			
15	14	13	12	11	10	9	8
REQDLY				RESERVED		MONO8B	RDORDER
R/W-0x0				R-0x0		R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
LCDTFT	RESERVED					LCDBW	LCDEN
R/W-0x0	R-0x0					R/W-0x0	R/W-0x0

Table 20-19. LCDRASTRCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0x0	
26	TFT24UPCK	R/W	0x0	<p>24-bit TFT mode packing.</p> <p>This bit is only used when TFT24 and LCDTFT are both set to 1.</p> <p>If TFT24UPCK is clear, 24-bit pixels are packed in 32-bit boundaries which means four pixels are saved in every three words, as shown below.</p> <p>Word0: pix1[7:0], pix0[23:0] Word1: pix2[15:0], pix1[23:8] Word2: pix3[23:0], pix2[23:16]</p> <p>See Figure 20-5 for information on how the pixels are packed.</p> <p>If this bit is set to 1, then 24-bit pixels are stored unpacked in DDR with the uppermost byte unused, as shown below:</p> <p>Word0: Unused[7:0], pix0[23:0] Word1: Unused[7:0], pix1[23:0] Word2: Unused[7:0], pix2[23:0] Word3: Unused[7:0], pix3[23:0]</p> <p>0x0 = 24-bit pixels are packed into 32 bit boundaries, which means 4 pixels are saved in every three words 0x1 = 24-bit pixels are stored unpacked in DDR with the uppermost byte unused</p>
25	TFT24	R/W	0x0	<p>24-bit TFT mode.</p> <p>0x0 = 24-Bit TFT Mode disabled. Palette RAM lookup is used for output pixel data.</p> <p>0x1 = 24-Bit TFT Mode enabled. 24-bit data in TFT Active mode. The format of the framebuffer data depends on TFT24UPCK.</p>

Table 20-19. LCDRASTRCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24	FRMBUFSZ	R/W	0x0	<p>Frame buffer select.</p> <p>This mode is valid when LCDTFT is 0 and there are 16 bpp raw data frame buffers (bpp= 00)</p> <p>Only for this case, this bit selects whether the frame buffer format is 16 bpp 565 or 12bpp.</p> <p>The Grayscale can only take 12 bits per pixel. The frame buffer data is 16 bits per pixel 565 when FRMBUFSZ is set to 1 and only the 4 most significant bits of each color component are sent to the Grayscale input.</p> <p>0x0 = Framebuffer is 12 bpp packed in bits [11:0]</p> <p>0x1 = Framebuffer is 16 bpp 565</p>
23	TFTMAP	R/W	0x0	<p>TFT mode alternate signal mapping for palettized framebuffer.</p> <p>This bit must be 0 for all 12-, 16-, or 24-bpp raw data formats. This bit must be 1 for 1-, 2-, 4-, or 8-bpp palette lookup data. Valid only in active matrix mode when LCDTFT = 1.</p> <p>0x0 = 4 bits per component output data for 1-, 2-, 4-, and 8-bpp modes are right aligned on LCDDATA[11:0]</p> <p>0x1 = 4 bits per component output data for 1-, 2-, 4-, and 8-bpp modes are converted to 5,6,5, format and use LCDDATA[15:0] = {R3 R2 R1 R0 R3 G3 G2 G1 G0 G3 G2 B3 B2 B1 B0 B3}</p>
22	NIBMODE	R/W	0x0	<p>Nibble mode. This bit is used to determine palette indexing and is used in conjunction with RDORDER.</p> <p>0x0 = Nibble mode is disabled</p> <p>0x1 = Nibble mode is enabled</p>
21-20	PALMODE	R/W	0x0	<p>Palette loading mode. For raw data (12, 16, or 24 bpp) frame buffers, no palette lookup is employed. Thus, these frame buffers use the data-only loading mode.</p> <p>0x0 = Palette and data loading, reset value</p> <p>0x1 = Palette loading only</p> <p>0x2 = Data loading only</p>
19-12	REQDLY	R/W	0x0	<p>Palette loading delay.</p> <p>This 8-bit parameter pauses reading of the Palette data from the asynchronous FIFO between each burst of 16 words. The delay is in terms of system clock (SYSCLK) cycles. The value (0-255) used to specify the number of system clock cycles that should be paused between bursts of 16 word reads from the asynchronous FIFO while loading the Palette SRAM. Programming REQDLY = 0x00 disables this pause when loading the Palette table.</p> <p>When loading the Palette from system memory, palette data is burst into the internal Palette SRAM from the Asynchronous FIFO. 1, 2, and 4-bit per pixel frame buffer encodings use a fixed 16-word entry palette residing above the video data.</p> <p>The 8-bit per pixel frame buffer encoding uses a 256-word entry palette residing above the video data. Likewise, 12, 16, and 24-bit per pixel frame buffer encodings also define a 256-word entry palette even though these encodings will not do a full bit-depth palette lookup. However, the 256-word palette entry must still be read from DDR as a frame buffer is fetched.</p> <p>Bursting in 256 words in sequential SYSCLK cycles may cause the asynchronous FIFO to underflow depending on the DDR burst bandwidth.</p>
11-10	RESERVED	R	0x0	
9	MONO8B	R/W	0x0	<p>Mono 8-bit.</p> <p>This bit is ignored in all other modes.</p> <p>0x0 = lcd_pixel_o[3:0] is used to output four pixel values to the panel each pixel clock transition</p> <p>0x1 = lcd_pixel_o[7:0] is used to output eight pixel values to the panel each pixel clock transition.</p>

Table 20-19. LCDRASTRCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	RDORDER	R/W	0x0	<p>Raster data order select.</p> <p>For 1, 2, 4, and 8 BPP framebuffers. This bit has no effect on raw data framebuffers (12/16/24 bpp). This bit is used to determine palette indexing and is used in conjunction with NIBMODE.</p> <p>0x0 = The framebuffer parsing for Palette Data lookup is from Bit 0 to bit 31 of the input word from the DMA output.</p> <p>0x1 = The framebuffer parsing for Palette Data lookup is from Bit 31 to Bit 0 of the input word from the DMA output.</p>
7	LCDTFT	R/W	0x0	<p>LCD TFT</p> <p>0x0 = Passive or STN display operation enabled; dither logic is enabled.</p> <p>0x1 = Active or TFT display operation enabled, external palette and DAC required, dither logic bypassed, pin timing changes to support continuous pixel clock, output enable, vsync, and hsync.</p>
6-2	RESERVED	R	0x0	
1	LCDBW	R/W	0x0	<p>LCD monochrome. Only applies for passive matrix panels.</p> <p>0x0 = Color operation enabled</p> <p>0x1 = Monochrome operation enabled</p>
0	LCDEN	R/W	0x0	<p>LCD controller enable for raster operations.</p> <p>This bit does not affect LIDD mode behavior.</p> <p>0x0 = LCD controller disabled</p> <p>0x1 = LCD controller enabled</p>

20.7.11 LCDRASTRTIM0 Register (Offset = 0x2C) [reset = 0x0]

LCD Raster Timing 0 (LCDRASTRTIM0)

LCDRASTRTIM0 is shown in [Figure 20-26](#) and described in [Table 20-20](#).

Return to [Summary Table](#).

Figure 20-26. LCDRASTRTIM0 Register

31	30	29	28	27	26	25	24
HBP							
R/W-0x0							
23	22	21	20	19	18	17	16
HFP							
R/W-0x0							
15	14	13	12	11	10	9	8
HSW						PPL	
R/W-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
PPL				MSBPPL	RESERVED		
R/W-0x0				R/W-0x0	R-0x0		

Table 20-20. LCDRASTRTIM0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	HBP	R/W	0x0	Horizontal Back Porch Lowbits. Bits 7:0 of the horizontal back porch field. Encoded value (from 1 to 1024) used to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display (programmed value plus 1). Note that pixel clock is held in its inactive state during the beginning of the line wait period in passive display mode, and is permitted to transition in active display mode.
23-16	HFP	R/W	0x0	Horizontal Front Porch Lowbits. Encoded value (from 1 to 1024) used to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted (programmed value plus 1). Note that pixel clock is held in its inactive state during the end of line wait period in passive display mode, and is permitted to transition in active display mode.
15-10	HSW	R/W	0x0	Horizontal Sync Pulse Width Lowbits. Bits 5:0 of the horizontal sync pulse width field. Encoded value (from 1 to 1024) used to specify the number of pixel clock periods to pulse the line clock at the end of each line (programmed value plus 1). Note that pixel clock is held in its inactive state during the generation of line clock in passive display mode, and is permitted to transition in active display mode.
9-4	PPL	R/W	0x0	Pixels-per-line LSB[9:4]. The PPL field is set to the (number of horizontal pixels minus 1) divided by 16.
3	MSBPPL	R/W	0x0	Pixels-per-line MSB[10]. Needed to support up to 2048 ppl.
2-0	RESERVED	R	0x0	

20.7.12 LCDRASTRTIM1 Register (Offset = 0x30) [reset = 0x0]

LCD Raster Timing 1 (LCDRASTRTIM1)

LCDRASTRTIM1 is shown in [Figure 20-27](#) and described in [Table 20-21](#).

Return to [Summary Table](#).

Figure 20-27. LCDRASTRTIM1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBP								VFP								VSW								LPP							
R/W-0x0								R/W-0x0								R/W-0x0								R/W-0x0							

Table 20-21. LCDRASTRTIM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	VBP	R/W	0x0	Vertical Back Porch. Value (from 0 to 255) use to specify the number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display. Note that line clock transitions during the insertion of the extra line clock periods.
23-16	VFP	R/W	0x0	Vertical Front Porch. Value (from 0 to 255) used to specify the number of line clock periods to add to the end of each frame. Note that the line clock transitions during the insertion of the extra line clock periods.
15-10	VSW	R/W	0x0	Vertical Sync Width Pulse. In active mode (LCDTFT = 1), encoded value (from 1-64) used to specify the number of line clock periods to set the lcd_fp pin active at the end of each frame after the (vfp) period elapses. The number of clock cycles is programmed value plus one. The frame clock is used as the VSYNC signal in active mode. In passive mode (LCDTFT = 0), encoded value (from 1-64) used to specify the number of extra line clock periods to insert after the vertical front porch (vfp) period has elapsed. Note that the width of lcd_fp is not affected by vsw in passive mode and line clock transitions during the insertion of the extra line clock periods (programmed value plus one).
9-0	LPP	R/W	0x0	Lines Per Panel. Encoded value (programmed value range of 0 to 2047 represents an actual range of 1 to 2048) used to specify the number of lines per panel. It represents the total number of lines on the LCD (programmed value plus one). Bit 10 of this field is in LCDRASTRTIM2.

20.7.13 LCDRASTRIM2 Register (Offset = 0x34) [reset = 0x0]

LCD Raster Timing 2 (LCDRASTRIM2)

LCDRASTRIM2 is shown in [Figure 20-28](#) and described in [Table 20-22](#).

Return to [Summary Table](#).

Figure 20-28. LCDRASTRIM2 Register

31	30	29	28	27	26	25	24
RESERVED	HSW				MSBLPP	PXLCLKCTL	PSYNCRF
R-0x0	R/W-0x0				R/W-0x0	R/W-0x0	R/W-0x0
23	22	21	20	19	18	17	16
INVOE	INVPXLCLK	IHS	IVS	ACBI			
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0			
15	14	13	12	11	10	9	8
ACBF							
R/W-0x0							
7	6	5	4	3	2	1	0
RESERVED	MSBHBP			RESERVED	MSBHFP		
R-0x0	R/W-0x0			R-0x0	R/W-0x0		

Table 20-22. LCDRASTRIM2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0x0	
30-27	HSW	R/W	0x0	Bits 9:6 of the horizontal sync width field.
26	MSBLPP	R/W	0x0	MSB of lines per panel. Bit 10 of the LPP field in LCDRASTRIM1.
25	PXLCLKCTL	R/W	0x0	HSYNC/VSYSN pixel clock control on/off. This bit MUST be programmed to 0 for passive matrix displays. The edge timing is fixed. 0x0 = LCDLP and LCDFP are driven on opposite edges of pixel clock than the LCD pixel output. 0x1 = LCDLP and LCDFP are driven according to bit 24, PSYNCRF
24	PSYNCRF	R/W	0x0	Program HSYNC/VSYSN rise or fall. 0x0 = LCDLP and LCDFP are driven on the falling edge of pixel clock (PXLCLKCTL must be set to 1). 0x1 = LCDLP and LCDFP are driven on the rising edge of pixel clock (PXLCLKCTL must be set to 1).
23	INVOE	R/W	0x0	Invert output enable. Active display mode: data driven out of the LCD data lines on programmed pixel clock edge where AC-bias is active. Passive display mode: INVOE is ignored. 0x0 = LCDAC pin is active high in active display mode 0x1 = LCDAC pin is active low in active display mode

Table 20-22. LCDRASTRIM2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
22	INVPXLCLK	R/W	0x0	<p>Invert pixel clock.</p> <p>For active matrix output (LCDTFT = 1), the output pixel clock is a free running clock in that it transitions in horizontal blanking (including horizontal front porch, horizontal back porch) areas and all vertical blanking times.</p> <p>For Passive Matrix output (LCDTFT = 0), the output pixel clock on occurs when an output data value is written. It is in a return-to-zero state when INVPXLCLK = 0 and a return-to-one state when INVPXLCLK = 1.</p> <p>0x0 = Data is driven on the LCD data lines on the rising edge of LCDCP.</p> <p>0x1 = Data is driven on the LCD data lines in the falling edge of LCDCP.</p>
21	IHS	R/W	0x0	<p>Invert hsync.</p> <p>Active and passive mode: horizontal sync pulse/line clock active between lines, after the end of line wait period.</p> <p>0x0 = LCDLP pin is active high and inactive low</p> <p>0x1 = LCDLP pin is active low and inactive high</p>
20	IVS	R/W	0x0	<p>Invert vsync.</p> <p>Active mode: vertical sync pulse active between frames, after end of frame wait period.</p> <p>Passive mode: frame clock active during first line of each frame.</p> <p>0x0 = LCDFP pin is active high and inactive low</p> <p>0x1 = LCDFP pin is active low and inactive high</p>
19-16	ACBI	R/W	0x0	<p>AC bias pins transitions per interrupt.</p> <p>Value (from 0x0 to 0xF) used to specify the number of AC Bias pin transitions to count before setting the line count status (ACBS) bit, signaling an interrupt request. Counter frozen when ACBS is set, and is restarted when ACBS is cleared by software. This function is disabled when ACBI = 0x0000.</p>
15-8	ACBF	R/W	0x0	<p>AC bias pin frequency.</p> <p>Value (from 0x0 to 0xFF) used to specify the number of line clocks to count before transitioning the AC Bias pin. This pin is used to periodically invert the polarity of the power supply to prevent DC charge build-up within the display. ACBF = Number of line clocks/toggle of the LCDAC pin.</p>
7-6	RESERVED	R	0x0	
5-4	MSBHBP	R/W	0x0	Bits 9:8 of the horizontal back porch field.
3-2	RESERVED	R	0x0	
1-0	MSBHFP	R/W	0x0	Bits 9:8 of the horizontal front porch field.

20.7.14 LCDRASTRSUBP1 Register (Offset = 0x38) [reset = 0x0]

LCD Raster Subpanel Display 1 (LCDRASTRSUBP1)

Note that subpictures are only allowed for Active Matrix mode (LCDTFT = 1) in LCDRASTRCTL

LCDRASTRSUBP1 is shown in [Figure 20-29](#) and described in [Table 20-23](#).

Return to [Summary Table](#).

Figure 20-29. LCDRASTRSUBP1 Register

31	30	29	28	27	26	25	24
SPEN	RESERVED	HOLS	RESERVED			LPPT	
R/W-0x0	R-0x0	R/W-0x0	R-0x0			R/W-0x0	
23	22	21	20	19	18	17	16
LPPT							
R/W-0x0							
15	14	13	12	11	10	9	8
DPDLSB							
R/W-0x0							
7	6	5	4	3	2	1	0
DPDLSB							
R/W-0x0							

Table 20-23. LCDRASTRSUBP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SPEN	R/W	0x0	Subpanel enable. 0x0 = Function disabled 0x1 = Sub-panel function mode enabled
30	RESERVED	R	0x0	
29	HOLS	R/W	0x0	High or low signal. This field indicates the position of the sub-panel based on the LPPT value. 0x0 = Default Pixel Data lines are at the top of the screen and the active video lines are at the bottom of the screen. 0x1 = Active video lines are at the top of the screen and Default Pixel Data lines are at the bottom of the screen.
28-26	RESERVED	R	0x0	
25-16	LPPT	R/W	0x0	Line per panel threshold. Encoded value (programmed value range of {0:2047} represents an actual range of {1:2048}) used to specify the number of lines on the bottom part of the panel. Bit 10 of this field is in LCDRASTRSUBP2. HOLS determines whether Default Pixel Data is on the top (HOLS = 0) or on the bottom (HOLS = 1). LPPT defines the number of lines on the bottom part of the output.
15-0	DPDLSB	R/W	0x0	Default pixel data LSB[15:0]. DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reach or after passing LPPT. DPDMSB is defined in bit field [7:0] in LCDRASTRSUBP2.

20.7.15 LCDRASTRSUBP2 Register (Offset = 0x3C) [reset = 0x0]

LCD Raster Subpanel Display 2 (LCDRASTRSUBP2)

Note that subpictures are only allowed for Active Matrix mode (LCDTFT= 1) in LCDRASTRCTL

LCDRASTRSUBP2 is shown in [Figure 20-30](#) and described in [Table 20-24](#).

Return to [Summary Table](#).

Figure 20-30. LCDRASTRSUBP2 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							LPPTMSB
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
DPDMSB							
R/W-0x0							

Table 20-24. LCDRASTRSUBP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	LPPTMSB	R/W	0x0	Lines per panel threshold bit 10. This register is bit 10 of the LPPT field in LCDRASTRSUBP1.
7-0	DPDMSB	R/W	0x0	Default pixel data MSB [23:16]. DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reached or after passing the LPPT.

20.7.16 LCDDMACTL Register (Offset = 0x40) [reset = 0x0]

LCD DMA Control (LCDDMACTL)

LCDDMACTL is shown in [Figure 20-31](#) and described in [Table 20-25](#).

Return to [Summary Table](#).

Figure 20-31. LCDDMACTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED					FIFORDY		
R-0x0					R/W-0x0		
7	6	5	4	3	2	1	0
RESERVED	BURSTSZ			BYTESWAP	RESERVED	BIGDEND	FMODE
R-0x0	R/W-0x0			R/W-0x0	R-0x0	R/W-0x0	R/W-0x0

Table 20-25. LCDDMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0x0	
10-8	FIFORDY	R/W	0x0	DMA FIFO threshold. The DMA FIFO becomes ready when the number of words specified by this register from the frame buffer have been loaded. 0x0 = 8 words 0x1 = 16 words 0x2 = 32 words 0x3 = 64 words 0x4 = 128 words 0x5 = 256 words 0x6 = 512 words 0x7 = Reserved
7	RESERVED	R	0x0	
6-4	BURSTSZ	R/W	0x0	Burst size setting for DMA transfers (all DMA transfers are 32 bits wide). For Raster mode, configuring BURSTSZ should be done only after an LCD peripheral reset using the SRLCD register in System Control. The BURSTSZ field should not be changed once the LCD DMA is enabled. 0x0 = Reserved 0x1 = Reserved 0x2 = Burst size of 4 0x3 = Burst size of 8 0x4 = Burst size of 16 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved
3	BYTESWAP	R/W	0x0	This bit controls the byte lane ordering of the data on the output of the DMA module. It works in conjunction with the big-endian bit. See the big-endian description for configuration guidelines.
2	RESERVED	R	0x0	

Table 20-25. LCDDMACTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	BIGDEND	R/W	0x0	<p>Big endian enable.</p> <p>Use this bit when the processor is operating in big endian mode and writes to the frame buffers are less than 32-bits wide. Only this needs to change the byte alignment for data coming into the FIFO from the frame buffer. The BIGEND and BYTESWAP bits control the byte lane ordering of the data on the output of the DMA module.</p> <p>0x0 = Big endian reordering disabled.</p> <p>0x1 = Big endian reordering enabled.</p>
0	FMODE	R/W	0x0	<p>Frame mode.</p> <p>0x0 = One frame buffer (FB0 only) used</p> <p>0x1 = Two frame buffers used. DMA ping-pongs between FB0 and FB1 in this mode.</p>

20.7.17 LCDDMABAFB0 Register (Offset = 0x44) [reset = 0x0]

LCD DMA Frame Buffer 0 Base Address (LCDDMABAFB0)

NOTE: When the LCD DMA is enabled, do not read or write the LCD registers for the base and ceiling addresses (LCDDMABAFB0, LCDDMACAFB0, LCDDMABAFB1, and LCDDMACAFB1) with the CPU. To change any of these registers, disable the DMA (clear the DMAEN bit in the LCDLIDDCTL register or the LCDEN bit in the LCDRASTRCTL register), update the registers, and enable the LCD DMA again.

LCDDMABAFB0 is shown in [Figure 20-32](#) and described in [Table 20-26](#).

[Return to Summary Table.](#)

Figure 20-32. LCDDMABAFB0 Register

31	30	29	28	27	26	25	24
FB0BA							
R/W-0x0							
23	22	21	20	19	18	17	16
FB0BA							
R/W-0x0							
15	14	13	12	11	10	9	8
FB0BA							
R/W-0x0							
7	6	5	4	3	2	1	0
FB0BA						RESERVED	
R/W-0x0						R-0x0	

Table 20-26. LCDDMABAFB0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	FB0BA	R/W	0x0	Frame buffer 0 base address pointer
1-0	RESERVED	R	0x0	

20.7.18 LCDDMACAFB0 Register (Offset = 0x48) [reset = 0x0]

LCD DMA Frame Buffer 0 Ceiling Address (LCDDMACAFB0)

NOTE: When the LCD DMA is enabled, do not read or write the LCD registers for the base and ceiling addresses (LCDDMABAFB0, LCDDMACAFB0, LCDDMABAFB1, and LCDDMACAFB1) with the CPU. To change any of these registers, disable the DMA (clear the DMAEN bit in the LCDLIDCTL register or the LCDEN bit in the LCDRASTRCTL register), update the registers, and enable the LCD DMA again.

LCDDMACAFB0 is shown in [Figure 20-33](#) and described in [Table 20-27](#).

Return to [Summary Table](#).

Figure 20-33. LCDDMACAFB0 Register

31	30	29	28	27	26	25	24
FB0CA							
R/W-0x0							
23	22	21	20	19	18	17	16
FB0CA							
R/W-0x0							
15	14	13	12	11	10	9	8
FB0CA							
R/W-0x0							
7	6	5	4	3	2	1	0
FB0CA						RESERVED	
R/W-0x0						R-0x0	

Table 20-27. LCDDMACAFB0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	FB0CA	R/W	0x0	Frame buffer 0 ceiling address pointer. For raster mode (MODESEL = 1), this register cannot be the same value as FB0BA.
1-0	RESERVED	R	0x0	

20.7.19 LCDDMABAFB1 Register (Offset = 0x4C) [reset = 0x0]

LCD DMA Frame Buffer 1 Base Address (LCDDMABAFB1)

NOTE: When the LCD DMA is enabled, do not read or write the LCD registers for the base and ceiling addresses (LCDDMABAFB0, LCDDMACAFB0, LCDDMABAFB1, and LCDDMACAFB1) with the CPU. To change any of these registers, disable the DMA (clear the DMAEN bit in the LCDLIDDCTL register or the LCDEN bit in the LCDRASTRCTL register), update the registers, and enable the LCD DMA again.

LCDDMABAFB1 is shown in [Figure 20-34](#) and described in [Table 20-28](#).

Return to [Summary Table](#).

Figure 20-34. LCDDMABAFB1 Register

31	30	29	28	27	26	25	24
FB1BA							
R/W-0x0							
23	22	21	20	19	18	17	16
FB1BA							
R/W-0x0							
15	14	13	12	11	10	9	8
FB1BA							
R/W-0x0							
7	6	5	4	3	2	1	0
FB1BA						RESERVED	
R/W-0x0						R-0x0	

Table 20-28. LCDDMABAFB1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	FB1BA	R/W	0x0	Frame buffer 1 base address pointer.
1-0	RESERVED	R	0x0	

20.7.20 LCDDMACAFB1 Register (Offset = 0x50) [reset = 0x0]

LCD DMA Frame Buffer 1 Ceiling Address (LCDDMACAFB1)

NOTE: When the LCD DMA is enabled, do not read or write the LCD registers for the base and ceiling addresses (LCDDMABAFB0, LCDDMACAFB0, LCDDMABAFB1, and LCDDMACAFB1) with the CPU. To change any of these registers, disable the DMA (clear the DMAEN bit in the LCDLIDCTL register or the LCDEN bit in the LCDRASTRCTL register), update the registers, and enable the LCD DMA again.

LCDDMACAFB1 is shown in [Figure 20-35](#) and described in [Table 20-29](#).

Return to [Summary Table](#).

Figure 20-35. LCDDMACAFB1 Register

31	30	29	28	27	26	25	24
FB1CA							
R/W-0x0							
23	22	21	20	19	18	17	16
FB1CA							
R/W-0x0							
15	14	13	12	11	10	9	8
FB1CA							
R/W-0x0							
7	6	5	4	3	2	1	0
FB1CA						RESERVED	
R/W-0x0						R-0x0	

Table 20-29. LCDDMACAFB1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	FB1CA	R/W	0x0	Frame buffer 1 ceiling address pointer. For raster mode (MODESEL = 1), this register cannot be the same value as FB1BA.
1-0	RESERVED	R	0x0	

20.7.21 LCDSYSCFG Register (Offset = 0x54) [reset = 0x28]

LCD System Configuration Register (LCDSYSCFG)

LCDSYSCFG is shown in [Figure 20-36](#) and described in [Table 20-30](#).

Return to [Summary Table](#).

Figure 20-36. LCDSYSCFG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		STDBY		IDLEMODE		RESERVED	
R-0x0		R/W-0x2		R/W-0x2		R-0x0	

Table 20-30. LCDSYSCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-4	STDBY	R/W	0x2	Standby mode. Configuration of the local initiator state management mode. By definition, initiator may generate read/write transaction as long as it is out of STANDBY state IP module may generate (master-related) wakeup events when in standby state. 0x0 = Force-standby mode: local initiator is unconditionally placed in standby state. Backup mode, for debug only 0x1 = No-standby mode: local initiator is unconditionally placed out of standby state. Backup mode, for debug only 0x2 = Smart-standby mode: local initiator standby status depends on local conditions, that is the module's functional requirement from the initiator. IP module shall not generate (initiator-related) wakeup events 0x3 = Reserved
3-2	IDLEMODE	R/W	0x2	Idle mode. Configuration of the local target state management mode. By definition, target can handle read/write transaction as long as it is out of IDLE state 0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, that regardless of the IP module's internal requirements. Backup mode, for debug only 0x1 = No-idle mode: local target never enters idle state. Backup mode, for debug only 0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-requestrelated) wakeup events 0x3 = Reserved
1-0	RESERVED	R	0x0	

20.7.22 LCDRISSET Register (Offset = 0x58) [reset = 0x0]

LCD Interrupt Raw Status and Set Register (LCDRISSET)

This register contains the raw interrupt status. In addition to providing the Raw Interrupt Status on a read, a 1 to a bit will set the associated interrupt.

LCDRISSET is shown in [Figure 20-37](#) and described in [Table 20-31](#).

Return to [Summary Table](#).

Figure 20-37. LCDRISSET Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						EOF1	EOF0
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	PALLOAD	FIFOU	RESERVED	ACBS	SYNCS	RRASTRDONE	DONE
R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 20-31. LCDRISSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	EOF1	R/W	0x0	DMA End-of-Frame 1 Raw Interrupt Status and Set. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status 0x0 = Inactive 0x1 = Active
8	EOF0	R/W	0x0	DMA End-of-Frame 0 Raw Interrupt Status and Set. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status 0x0 = Inactive 0x1 = Active
7	RESERVED	R	0x0	
6	PALLOAD	R/W	0x0	DMA Palette Loaded Raw Interrupt Status and Set. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status 0x0 = Inactive 0x1 = Active
5	FIFOU	R/W	0x0	DMA FIFO Underflow Raw Interrupt Status and Set. Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status 0x0 = Inactive 0x1 = Active
4	RESERVED	R	0x0	
3	ACBS	R/W	0x0	AC Bias Count Raw Interrupt Status and Set. For Passive Matrix Panels Only AC bias transition counter has decremented to zero, indicating that the lcd_ac_o line has transitioned the number of times which is specified by the ACBI control bit-field in the LCDRASTRIMn register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status 0x0 = Inactive 0x1 = Active

Table 20-31. LCDRISSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	SYNCS	R/W	0x0	Frame Synchronization Lost Raw Interrupt Status and Set. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status. 0x0 = Inactive 0x1 = Active
1	RRASTRDONE	R/W	0x0	Raster Mode Frame Done interrupt. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status. 0x0 = Inactive 0x1 = Active
0	DONE	R/W	0x0	Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Raw Interrupt Status and Set. Writing 1 will set status. Writing 0 has no effect. Read indicates raw status. 0x0 = Inactive 0x1 = Active

20.7.23 LCDMISCLR Register (Offset = 0x5C) [reset = 0x0]

LCD Interrupt Status and Clear (LCDMISCLR)

This register contains the Interrupt Enable Status. This register returns Masked (Enabled) status interrupts on a Read. Writing a 1 to a bit will clear the associated interrupt.

LCDMISCLR is shown in [Figure 20-38](#) and described in [Table 20-32](#).

Return to [Summary Table](#).

Figure 20-38. LCDMISCLR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						EOF1	EOF0
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	PALLOAD	FIFOU	RESERVED	ACBS	SYNCS	RRASTRDONE	DONE
R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 20-32. LCDMISCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	EOF1	R/W	0x0	DMA End-of-Frame 1 Enabled Interrupt and Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
8	EOF0	R/W	0x0	DMA End-of-Frame 0 Raw Interrupt and Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
7	RESERVED	R	0x0	
6	PALLOAD	R/W	0x0	DMA Palette Loaded Enabled Interrupt and Clear. Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
5	FIFOU	R/W	0x0	DMA FIFO Underflow Enabled Interrupt and Clear. Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO. Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
4	RESERVED	R	0x0	

Table 20-32. LCDMISCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	ACBS	R/W	0x0	AC Bias Count Enabled Interrupt and Clear. For Passive Matrix Panels Only AC bias transition counter has decremented to zero, indicating that the LCDAC line has transitioned the number of times which is specified by the ACBI control bit-field in the LCDRASTRTIMn register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS. Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
2	SYNCS	R/W	0x0	Frame Synchronization Lost Enabled Interrupt and Clear. Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
1	RRASTRDONE	R/W	0x0	Raster Mode Frame Done interrupt. Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active
0	DONE	R/W	0x0	Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Enabled Interrupt and Clear. Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Inactive 0x1 = Active

20.7.24 LCDIM Register (Offset = 0x60) [reset = 0x0]

LCD Interrupt Mask (LCDIM)

This register provides for the setting of interrupt enables (Mask bits).

LCDIM is shown in [Figure 20-39](#) and described in [Table 20-33](#).

Return to [Summary Table](#).

Figure 20-39. LCDIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						EOF1	EOF0
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	PALLOAD	FIFOU	RESERVED	ACBS	SYNCS	RRASTRDONE	DONE
R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 20-33. LCDIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	EOF1	R/W	0x0	DMA End-of-Frame 1 Interrupt Enable Set. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (mask) status. 0x0 = Disabled 0x1 = Enabled
8	EOF0	R/W	0x0	DMA End-of-Frame 0 Interrupt Enable Set. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled
7	RESERVED	R	0x0	
6	PALLOAD	R/W	0x0	DMA Palette Loaded Interrupt Enable Set. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled
5	FIFOU	R/W	0x0	DMA FIFO Underflow Interrupt Enable Set. Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled
4	RESERVED	R	0x0	

Table 20-33. LCDIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	ACBS	R/W	0x0	AC Bias Count Interrupt Enable Set. For Passive Matrix Panels Only AC bias transition counter has decremented to zero, indicating that the LCDAC line has transitioned the number of times which is specified by the ACBI control bit-field in the LCDRASTRTIMn register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled
2	SYNCS	R/W	0x0	Frame Synchronization Lost Interrupt Enable Set. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled
1	RRASTRDONE	R/W	0x0	Raster Mode Frame Done Interrupt Enable Set. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled
0	DONE	R/W	0x0	Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Interrupt Enable Set. Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status. 0x0 = Disabled 0x1 = Enabled

20.7.25 LCDIENC Register (Offset = 0x64) [reset = 0x0]

LCD Interrupt Enable Clear (LCDIENC)

This register provides for the clearing of interrupt enables (Mask bits).

LCDIENC is shown in [Figure 20-40](#) and described in [Table 20-34](#).

Return to [Summary Table](#).

Figure 20-40. LCDIENC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						EOF1	EOF0
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED	PALLOAD	FIFOU	RESERVED	ACBS	SYNCS	RRASTRDONE	DONE
R-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 20-34. LCDIENC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	EOF1	R/W	0x0	DMA End-of-Frame 1 Interrupt Enable Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
8	EOF0	R/W	0x0	DMA End-of-Frame 0 Interrupt Enable Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
7	RESERVED	R	0x0	
6	PALLOAD	R/W	0x0	DMA Palette Loaded Interrupt Enable Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
5	FIFOU	R/W	0x0	DMA FIFO Underflow Interrupt Enable Clear. Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
4	RESERVED	R	0x0	

Table 20-34. LCDIENC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	ACBS	R/W	0x0	AC Bias Count Interrupt Enable Clear. For Passive Matrix Panels Only AC bias transition counter has decremented to zero, indicating that the LCDAC line has transitioned the number of times which is specified by the ACBI control bit-field in the LCDRASTRTIMn register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
2	SYNCS	R/W	0x0	Frame Synchronization Lost Interrupt Enable Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
1	RRASTRDONE	R/W	0x0	Raster Mode Frame Done Interrupt Enable Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled
0	DONE	R/W	0x0	Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Interrupt Enable Clear. Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status. 0x0 = Disabled 0x1 = Enabled

20.7.26 LCDCLKEN Register (Offset = 0x6C) [reset = 0x0]

LCD Clock Enable (LCDCLKEN)

This register contains the Clock enables for each major domain within the LCD.

LCDCLKEN is shown in [Figure 20-41](#) and described in [Table 20-35](#).

Return to [Summary Table](#).

Figure 20-41. LCDCLKEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DMA	LIDD	CORE
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 20-35. LCDCLKEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DMA	R/W	0x0	DMA Clock Enable. Software Clock Enable for the LCD DMA 0x0 = Clock Disabled 0x1 = Clock Enabled
1	LIDD	R/W	0x0	LIDD Submodule Clock Enable. Software Clock Enable for the LIDD submodule (character displays). The LIDD submodule runs on the System Clock domain 0x0 = Clock Disabled 0x1 = Clock Enabled
0	CORE	R/W	0x0	LCD Core Clock Enable. Software Clock Enable for the LCD core, which encompasses the Raster Active Matrix and Passive Matrix logic. The Core runs on the System Clock domain. 0x0 = Clock Disabled 0x1 = Clock Enabled

20.7.27 LCDCLKRESET Register (Offset = 0x70) [reset = 0x0]

LCD Clock Resets (LCDCLKRESET)

This register contains the Software Resets for each major domain within the LCD.

LCDCLKRESET is shown in [Figure 20-42](#) and described in [Table 20-36](#).

Return to [Summary Table](#).

Figure 20-42. LCDCLKRESET Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				MAIN	DMA	LIDD	CORE
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 20-36. LCDCLKRESET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	MAIN	R/W	0x0	Software Reset for the entire LCD module. This reset affects the L3 clk and lcd_clk domain. 0x0 = Reset Disabled 0x1 = Reset Enabled
2	DMA	R/W	0x0	Software Reset for the DMA submodule. This reset affects the L3 clk domain. 0x0 = Reset Disabled 0x1 = Reset Enabled
1	LIDD	R/W	0x0	Software Reset for the LIDD submodule (character displays). This reset affects the LIDD logic in the lcd_clk domain. 0x0 = Reset Disabled 0x1 = Reset Enabled
0	CORE	R/W	0x0	Software Reset for the Core, which encompasses the Raster Active Matrix and Passive Matrix logic. This reset affects the Core (active matrix and passive matrix raster mode) logic in the lcd_clk domain. 0x0 = Reset Disabled 0x1 = Reset Enabled

Pulse Width Modulator (PWM)

Pulse width modulation (PWM) is a powerful technique for digitally encoding analog signal levels. High-resolution counters are used to generate a square wave, and the duty cycle of the square wave is modulated to encode an analog signal. Typical applications include switching power supplies and motor control.

Topic	Page
21.1 Introduction	1436
21.2 Block Diagram	1437
21.3 Functional Description	1438
21.4 Initialization and Configuration	1443
21.5 PWM Registers	1445

21.1 Introduction

The MSP432E4 microcontroller contains one PWM module, with four PWM generator blocks and a control block, for a total of eight PWM outputs. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that share the same timer and frequency and can either be programmed with independent actions or as a single pair of complementary signals with dead-band delays inserted. The output signals, pwmA' and pwmB', of the PWM generation blocks are managed by the output control block before being passed to the device pins as MnPWM0 and MnPWM1 or MnPWM2 and MnPWM3, and so on.

The PWM module provides a great deal of flexibility and can generate simple PWM signals, such as those required by a simple charge pump as well as paired PWM signals with dead-band delays, such as those required by a half-H bridge driver. Three generator blocks can also generate the full six channels of gate controls required by a three-phase inverter bridge.

Each PWM generator block has the following features:

- Four fault-condition handling inputs to quickly provide low-latency shutdown and prevent damage to the motor being controlled
- One 16-bit counter:
 - Runs in Down or Up/Down mode
 - Output frequency controlled by a 16-bit load value
 - Load value updates can be synchronized
 - Produces output signals at zero and load value
- Two PWM comparators:
 - Comparator value updates can be synchronized
 - Produces output signals on match
- PWM signal generator:
 - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
 - Produces two independent PWM signals
- Dead-band generator:
 - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
 - Can be bypassed, leaving input PWM signals unmodified
- Can initiate an ADC sample sequence

The control block determines the polarity of the PWM signals and which signals are passed through to the pins. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins. The PWM control block has the following options:

- PWM output enable of each PWM signal
- Optional output inversion of each PWM signal (polarity control)
- Optional fault handling for each PWM signal
- Synchronization of timers in the PWM generator blocks
- Synchronization of timer/comparator updates across the PWM generator blocks
- Extended PWM synchronization of timer/comparator updates across the PWM generator blocks
- Interrupt status summary of the PWM generator blocks
- Extended PWM fault handling, with multiple fault signals, programmable polarities, and filtering
- PWM generators can be operated independently or synchronized with other generators

21.2 Block Diagram

Figure 21-1 shows the PWM module block diagram, and Figure 21-2 shows a more detailed diagram of a PWM generator. The controller contains four generator blocks that generate eight independent PWM signals or four paired PWM signals with dead-band delays inserted.

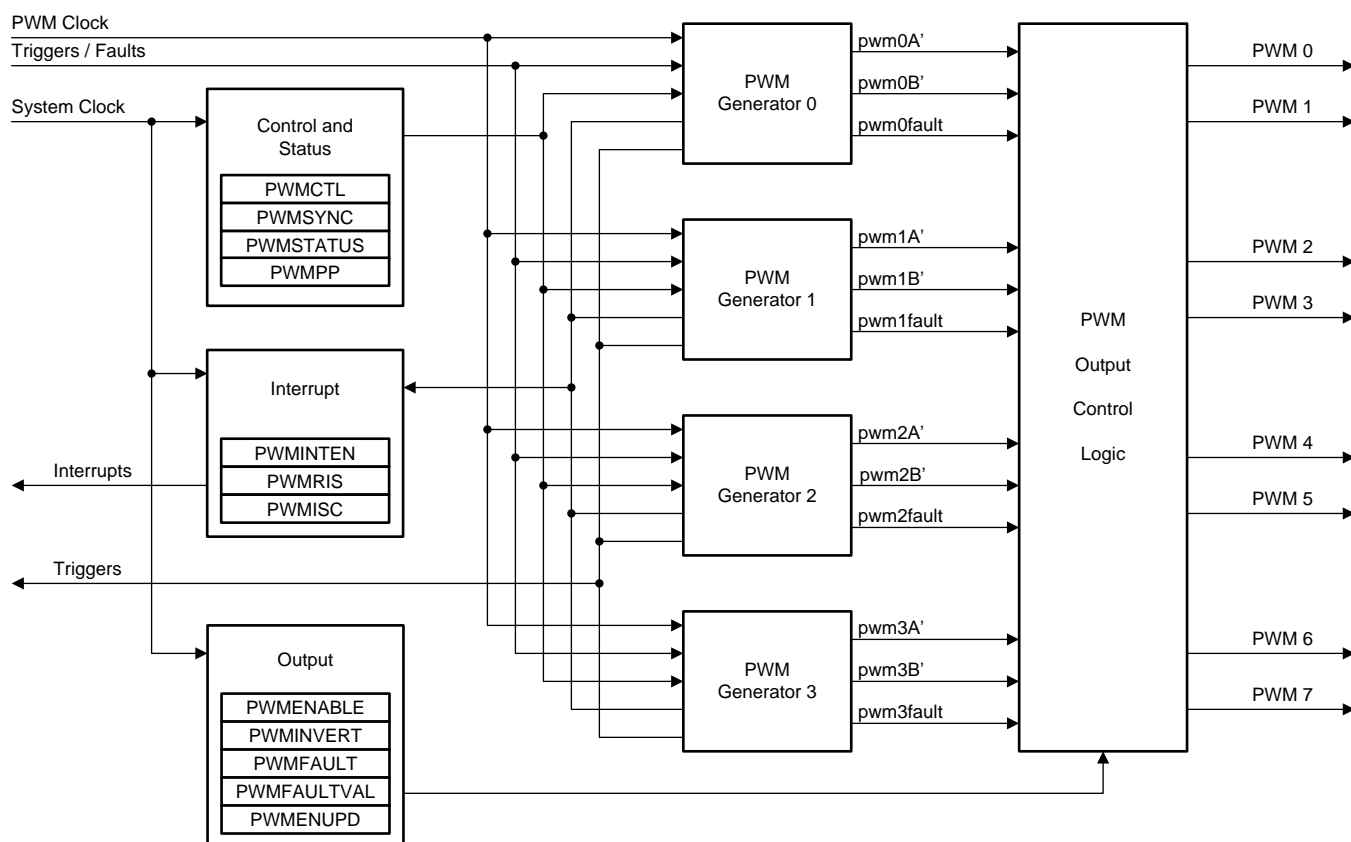


Figure 21-1. PWM Module Diagram

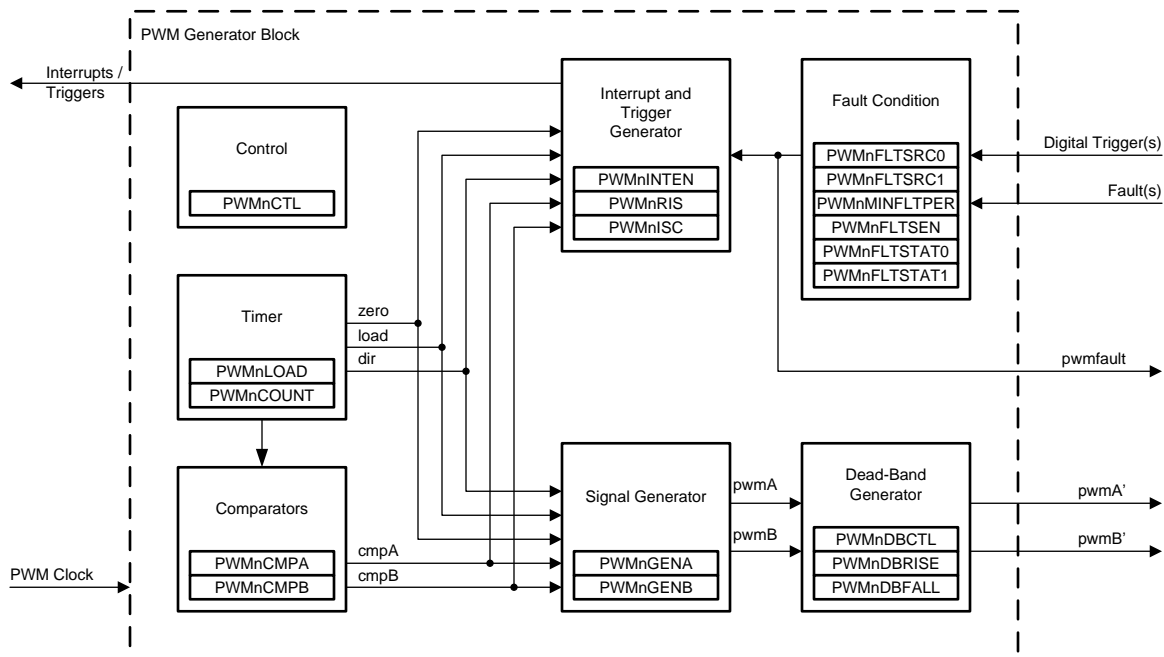


Figure 21-2. PWM Generator Block Diagram

21.3 Functional Description

21.3.1 Clock Configuration

The PWM has two clock source options:

- The system clock
- A predivided system clock

The clock source is selected by programming the USEPWM bit in the PWM Clock Configuration (PWMCC) register. The PWMDIV bit field specifies the divisor of the System Clock that is used to create the PWM Clock.

21.3.2 PWM Timer

The timer in each PWM generator runs in one of two modes: Count-Down mode or Count-Up/Down mode. In Count-Down mode, the timer counts from the load value to zero, goes back to the load value, and continues counting down. In Count-Up/Down mode, the timer counts from zero up to the load value, back down to zero, back up to the load value, and so on. Generally, Count-Down mode is used for generating left- or right-aligned PWM signals, while the Count-Up/Down mode is used for generating center-aligned PWM signals.

The timers output three signals that are used in the PWM generation process: the direction signal (this is always Low in Count-Down mode, but alternates between Low and High in Count-Up/Down mode), a single-clock-cycle-width High pulse when the counter is zero, and a single-clock-cycle-width High pulse when the counter is equal to the load value. Note that in Count-Down mode, the zero pulse is immediately followed by the load pulse. In the figures in this chapter, these signals are labelled *dir*, *zero*, and *load*.

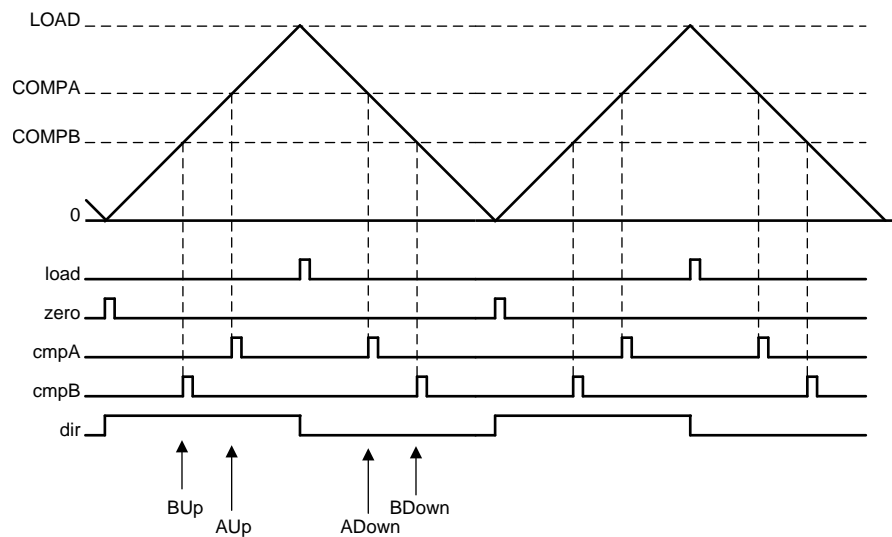


Figure 21-4. PWM Count Up/Down Mode

21.3.4 PWM Signal Generator

Each PWM generator takes the load, zero, cmpA, and cmpB pulses (qualified by the dir signal) and generates two internal PWM signals, pwmA and pwmB. In Count-Down mode, there are four events that can affect these signals: zero, load, match A down, and match B down. In Count-Up/Down mode, there are six events that can affect these signals: zero, load, match A down, match A up, match B down, and match B up. The match A or match B events are ignored when they coincide with the zero or load events. If the match A and match B events coincide, the first signal, pwmA, is generated based only on the match A event, and the second signal, pwmB, is generated based only on the match B event.

For each event, the effect on each output PWM signal is programmable: it can be left alone (ignoring the event), it can be toggled, it can be driven low, or it can be driven high. These actions can be used to generate a pair of PWM signals of various positions and duty cycles, which do or do not overlap.

[Figure 21-5](#) shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles. This figure shows the pwmA and pwmB signals before they have passed through the dead-band generator.

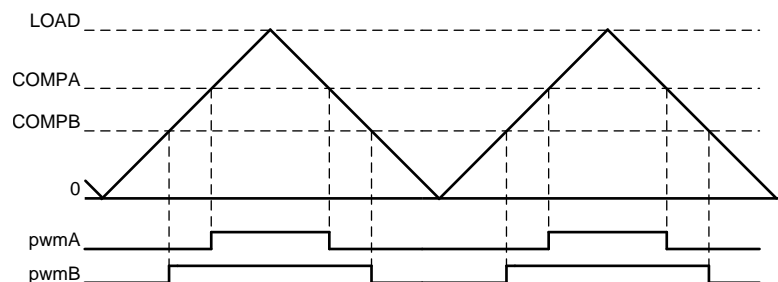


Figure 21-5. PWM Generation Example In Count-Up/Down Mode

In this example, the first generator is set to drive High on match A up, drive low on match A down, and ignore the other four events. The second generator is set to drive high on match B up, drive low on match B down, and ignore the other four events. Changing the value of comparator A changes the duty cycle of the pwmA signal, and changing the value of comparator B changes the duty cycle of the pwmB signal.

21.3.5 Dead-Band Generator

The pwmA and pwmB signals produced by each PWM generator are passed to the dead-band generator. If the dead-band generator is disabled, the PWM signals simply pass through to the pwmA' and pwmB' signals unmodified. If the dead-band generator is enabled, the pwmB signal is lost and two PWM signals are generated based on the pwmA signal. The first output PWM signal, pwmA' is the pwmA signal with the rising edge delayed by a programmable amount. The second output PWM signal, pwmB', is the inversion of the pwmA signal with a programmable delay added between the falling edge of the pwmA signal and the rising edge of the pwmB' signal.

The resulting signals are a pair of active High signals where one is always High, except for a programmable amount of time at transitions where both are Low. These signals are therefore suitable for driving a half-H bridge, with the dead-band delays preventing shoot-through current from damaging the power electronics. [Figure 21-6](#) shows the effect of the dead-band generator on the pwmA signal and the resulting pwmA' and pwmB' signals that are transmitted to the output control block.

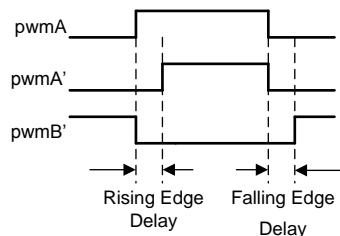


Figure 21-6. PWM Dead-Band Generator

21.3.6 Interrupt or ADC-Trigger Selector

Each PWM generator also takes the same four (or six) counter events and uses them to generate an interrupt or an ADC trigger. Any of these events or a set of these events can be selected as a source for an interrupt; when any of the selected events occur, an interrupt is generated. Additionally, the same event, a different event, the same set of events, or a different set of events can be selected as a source for an ADC trigger; when any of these selected events occur, an ADC trigger pulse is generated. The selection of events allows the interrupt or ADC trigger to occur at a specific position within the pwmA or pwmB signal. Interrupts and ADC triggers are based on the raw events; delays in the PWM signal edges caused by the dead-band generator are not taken into account.

21.3.7 Synchronization Methods

The PWM module provides four PWM generators, each providing two PWM outputs that may be used in a wide variety of applications. Generally speaking, the PWM is used in one of two categories of operation:

- **Unsynchronized**
The PWM generator and its two output signals are used alone, independent of other PWM generators.
- **Synchronized**
The PWM generator and its two outputs signals are used in conjunction with other PWM generators using a common, unified time base. If multiple PWM generators are configured with the same counter load value, synchronization can be used to guarantee that they also have the same count value (the PWM generators must be configured before they are synchronized). With this feature, more than two MnPWMn signals can be produced with a known relationship between the edges of those signals because the counters always have the same values. Other states in the module provide mechanisms to maintain the common time base and mutual synchronization.

The counter in a PWM generator can be reset to zero by writing the PWM Time Base Sync (PWMSYNC) register and setting the SYNCn bit associated with the generator. Multiple PWM generators can be synchronized together by setting all necessary SYNCn bits in one access. For example, setting the SYNC0 and SYNC1 bits in the PWMSYNC register causes the counters in PWM generators 0 and 1 to reset together.

Additional synchronization can occur between multiple PWM generators by updating register contents in one of the following three ways:

- Immediately
 - The write value has immediate effect, and the hardware reacts immediately.
- Locally Synchronized
 - The write value does not affect the logic until the counter reaches the value zero at the end of the PWM cycle. In this case, the effect of the write is deferred, providing a guaranteed defined behavior and preventing overly short or overly long output PWM pulses.
- Globally Synchronized
 - The write value does not affect the logic until two sequential events have occurred:
 1. The Update mode for the generator function is programmed for global synchronization in the PWMnCTL register
 2. The counter reaches zero at the end of the PWM cycle. In this case, the effect of the write is deferred until the end of the PWM cycle following the end of all updates. This mode allows multiple items in multiple PWM generators to be updated simultaneously without odd effects during the update; everything runs from the old values until a point at which they all run from the new values. The Update mode of the load and comparator match values can be individually configured in each PWM generator block. It typically makes sense to use the synchronous update mechanism across PWM generator blocks when the timers in those blocks are synchronized, although this is not required for this mechanism to function properly.

The following registers provide either local or global synchronization based on the state of various Update mode bits and fields in the PWMnCTL register (LOADUPD, CMPAUPD, and CMPBUPD):

- Generator Registers: PWMnLOAD, PWMnCMPA, and PWMnCMPB

The following registers default to immediate update, but are provided with the optional functionality of synchronously updating rather than having all updates take immediate effect:

- Module-Level Register: PWMENABLE (based on the state of the ENUPDn bits in the PWMENUPD register).
- Generator Register: PWMnGENA, PWMnGENB, PWMnDBCTL, PWMnDBRISE, and PWMnDBFALL (based on the state of various Update mode bits and fields in the PWMnCTL register [GENAUPD, GENBUPD, DBCTLUPD, DBRISEUPD, and DBFALLUPD]).

All other registers are considered statically provisioned for the execution of an application or are used dynamically for purposes unrelated to maintaining synchronization and therefore do not need synchronous update functionality.

21.3.8 Fault Conditions

A fault condition is one in which the controller must be signaled to stop normal PWM function and then set the MnPWMn signals to a safe state. Two basic situations cause fault conditions:

- The microcontroller is stalled and cannot perform the necessary computation in the time required for motion control
- An external error or event is detected

The PWM generator can use the following inputs to generate a fault condition, including:

- MnFAULTn pin assertion
- A stall of the controller generated by the debugger
- The trigger of an ADC digital comparator

Fault conditions are calculated on a per-PWM generator basis. Each PWM generator configures the necessary conditions to indicate a fault condition exists. This method allows the development of applications with dependent and independent control.

Four fault input pins (MnFAULTn) are available. These inputs may be used with circuits that generate an active High or active Low signal to indicate an error condition. A MnFAULTn pins may be individually programmed for the appropriate logic sense using the PWMnFLTSEN register.

The PWM generator's mode control, including fault condition handling, is provided in the PWMnCTL register. This register determines whether the input or a combination of MnFAULTn input signals and/or digital comparator triggers (as configured by the PWMnFLTSRC0 and PWMnFLTSRC1 registers) is used to generate a fault condition. The PWMnCTL register also selects whether the fault condition is maintained as long as the external condition lasts or if it is latched until the fault condition until cleared by software. Finally, this register also enables a counter that may be used to extend the period of a fault condition for external events to assure that the duration is a minimum length. The minimum fault period count is specified in the PWMnMINFLTPER register.

NOTE: When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the PWMnCTL register should be set to 1 to ensure trigger assertions are captured.

Status regarding the specific fault cause is provided in the PWMnFLTSTAT0 and PWMnFLTSTAT1 registers. Note that the fault status registers, PWMnFLTSTAT0 and PWMnFLTSTAT1, reflect the status of all fault sources, regardless of what fault sources are enabled for that particular generator.

PWM generator fault conditions may be promoted to a controller interrupt using the PWMINTEN register.

21.3.9 Output Control Block

The output control block performs the final conditioning of the pwmA' and pwmB' signals before they are output on the pins as the MnPWMn signals. Through a single register, the PWM Output Enable (PWNENABLE) register, the set of PWM signals that are actually enabled to the pins can be modified. This function can be used, for example, to perform commutation of a brushless DC motor with a single register write (and without modifying the individual PWM generators, which are modified by the feedback control loop). In addition, the updating of the bits in the PWNENABLE register can be configured to be immediate or locally or globally synchronized to the next synchronous update using the PWM Enable Update (PWMENUPD) register.

During fault conditions, the PWM output signals, MnPWMn, usually must be driven to safe values so that external equipment may be safely controlled. The PWMFAULT register specifies whether during a fault condition, the generated signal continues to be passed driven or to an encoding specified in the PWMFAULTVAL register.

A final inversion can be applied to any of the MnPWMn signals, making them active Low instead of the default active High using the PWM Output Inversion (PWMINVERT). The inversion is applied even if a value has been enabled in the PWMFAULT register and specified in the PWMFAULTVAL register. In other words, if a bit is set in the PWMFAULT, PWMFAULTVAL, and PWMINVERT registers, the output on the MnPWMn signal is 0, not 1 as specified in the PWMFAULTVAL register.

21.4 Initialization and Configuration

The following example shows how to initialize PWM Generator 0 with a 25-kHz frequency, a 25% duty cycle on the MnPWM0 pin, and a 75% duty cycle on the MnPWM1 pin. This example assumes the system clock is 20 MHz.

1. Enable the PWM clock by setting its corresponding bit in the RCGCPWM register in the System Control module (RCGCPWM) (see [Section 4.2.99](#)).
2. Enable the clock to the appropriate GPIO module using the RCGCGPIO register in the System Control module (RCGCGPIO) (see [Section 4.2.87](#)).
3. Enable the appropriate pins in the GPIO module for their alternate function using the GPIOAFSEL register. To determine which GPIOs to configure, see the device-specific data sheet.
4. Configure the PMCN fields in the GPIOCTL register to assign the PWM signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).
5. Configure the PWM Clock Configuration (PWMCC) register to use the PWM divide (USEPWMDIV) and set the divider (PWMDIV) to divide by 2 (0x0).
6. Configure the PWM generator for countdown mode with immediate updates to the parameters.
 - a. Write the PWM0CTL register with a value of 0x0000.0000.
 - b. Write the PWM0GENA register with a value of 0x0000.008C.

- c. Write the PWM0GENB register with a value of 0x0000.080C.
7. Set the period. For a 25-kHz frequency, the period = $1 / 25000$, or 40 μ s. The PWM clock source is 10 MHz, and the system clock divided by 2. Thus, there are 400 clock ticks per period. Use this value to set the PWM0LOAD register. In count-down mode, set the LOAD field in the PWM0LOAD register to the requested period minus one.
 - Write the PWM0LOAD register with a value of 0x0000.018F.
8. Set the pulse width of the MnPWM0 pin for a 25% duty cycle.
 - Write the PWM0CMPA register with a value of 0x0000.012B.
9. Set the pulse width of the MnPWM1 pin for a 75% duty cycle.
 - Write the PWM0CMPB register with a value of 0x0000.0063.
10. Start the timers in PWM generator 0.
 - Write the PWM0CTL register with a value of 0x0000.0001.
11. Enable PWM outputs.
 - Write the PWMENABLE register with a value of 0x0000.0003.

21.5 PWM Registers

[Table 21-1](#) lists the memory-mapped registers for the PWM. All register offset addresses not listed in [Table 21-1](#) should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the base address of the PWM module: 0x40028000.

The PWM module clock must be enabled before the registers can be programmed. There must be a delay of 3 system clock cycles after the PWM module clock is enabled before any PWM module registers are accessed.

Table 21-1. PWM Registers

Offset	Acronym	Register Name	Section
0x0	PWMCTL	PWM Master Control	Section 21.5.1
0x4	PWMSYNC	PWM Time Base Sync	Section 21.5.2
0x8	PWMENABLE	PWM Output Enable	Section 21.5.3
0xC	PWMINVERT	PWM Output Inversion	Section 21.5.4
0x10	PWMFAULT	PWM Output Fault	Section 21.5.5
0x14	PWMINTEN	PWM Interrupt Enable	Section 21.5.6
0x18	PWMRIS	PWM Raw Interrupt Status	Section 21.5.7
0x1C	PWMISC	PWM Interrupt Status and Clear	Section 21.5.8
0x20	PWMSTATUS	PWM Status	Section 21.5.9
0x24	PWMFAULTVAL	PWM Fault Condition Value	Section 21.5.10
0x28	PWMENUPD	PWM Enable Update	Section 21.5.11
0x40	PWM0CTL	PWM0 Control	Section 21.5.12
0x44	PWM0INTEN	PWM0 Interrupt and Trigger Enable	Section 21.5.13
0x48	PWM0RIS	PWM0 Raw Interrupt Status	Section 21.5.14
0x4C	PWM0ISC	PWM0 Interrupt Status and Clear	Section 21.5.15
0x50	PWM0LOAD	PWM0 Load	Section 21.5.16
0x54	PWM0COUNT	PWM0 Counter	Section 21.5.17
0x58	PWM0CMPA	PWM0 Compare A	Section 21.5.18
0x5C	PWM0CMPB	PWM0 Compare B	Section 21.5.19
0x60	PWM0GENA	PWM0 Generator A Control	Section 21.5.20
0x64	PWM0GENB	PWM0 Generator B Control	Section 21.5.21
0x68	PWM0DBCTL	PWM0 Dead-Band Control	Section 21.5.22
0x6C	PWM0DBRISE	PWM0 Dead-Band Rising-Edge Delay	Section 21.5.23
0x70	PWM0DBFALL	PWM0 Dead-Band Falling-Edge-Delay	Section 21.5.24
0x74	PWM0FLTSRC0	PWM0 Fault Source 0	Section 21.5.25
0x78	PWM0FLTSRC1	PWM0 Fault Source 1	Section 21.5.26
0x7C	PWM0MINFLTPER	PWM0 Minimum Fault Period	Section 21.5.27
0x080	PWM1CTL	PWM1 Control	Section 21.5.12
0x084	PWM1INTEN	PWM1 Interrupt and Trigger Enable	Section 21.5.13
0x088	PWM1RIS	PWM1 Raw Interrupt Status	Section 21.5.14
0x08C	PWM1ISC	PWM1 Interrupt Status and Clear	Section 21.5.15
0x090	PWM1LOAD	PWM1 Load	Section 21.5.16
0x094	PWM1COUNT	PWM1 Counter	Section 21.5.17
0x098	PWM1CMPA	PWM1 Compare A	Section 21.5.18
0x09C	PWM1CMPB	PWM1 Compare B	Section 21.5.19
0x0A0	PWM1GENA	PWM1 Generator A Control	Section 21.5.20
0x0A4	PWM1GENB	PWM1 Generator B Control	Section 21.5.21
0x0A8	PWM1DBCTL	PWM1 Dead-Band Control	Section 21.5.22
0x0AC	PWM1DBRISE	PWM1 Dead-Band Rising-Edge Delay	Section 21.5.23

Table 21-1. PWM Registers (continued)

Offset	Acronym	Register Name	Section
0x0B0	PWM1DBFALL	PWM1 Dead-Band Falling-Edge-Delay	Section 21.5.24
0x0B4	PWM1FLTSRC0	PWM1 Fault Source 0	Section 21.5.25
0x0B8	PWM1FLTSRC1	PWM1 Fault Source 1	Section 21.5.26
0x0BC	PWM1MINFLTPER	PWM1 Minimum Fault Period	Section 21.5.27
0x0C0	PWM2CTL	PWM2 Control	Section 21.5.12
0x0C4	PWM2INTEN	PWM2 Interrupt and Trigger Enable	Section 21.5.13
0x0C8	PWM2RIS	PWM2 Raw Interrupt Status	Section 21.5.14
0x0CC	PWM2ISC	PWM2 Interrupt Status and Clear	Section 21.5.15
0x0D0	PWM2LOAD	PWM2 Load	Section 21.5.16
0x0D4	PWM2COUNT	PWM2 Counter	Section 21.5.17
0x0D8	PWM2CMPA	PWM2 Compare A	Section 21.5.18
0x0DC	PWM2CMPB	PWM2 Compare B	Section 21.5.19
0x0E0	PWM2GENA	PWM2 Generator A Control	Section 21.5.20
0x0E4	PWM2GENB	PWM2 Generator B Control	Section 21.5.21
0x0E8	PWM2DBCTL	PWM2 Dead-Band Control	Section 21.5.22
0x0EC	PWM2DBRISE	PWM2 Dead-Band Rising-Edge Delay	Section 21.5.23
0x0F0	PWM2DBFALL	PWM2 Dead-Band Falling-Edge-Delay	Section 21.5.24
0x0F4	PWM2FLTSRC0	PWM2 Fault Source 0	Section 21.5.25
0x0F8	PWM2FLTSRC1	PWM2 Fault Source 1	Section 21.5.26
0x0FC	PWM2MINFLTPER	PWM2 Minimum Fault Period	Section 21.5.27
0x100	PWM3CTL	PWM3 Control	Section 21.5.12
0x104	PWM3INTEN	PWM3 Interrupt and Trigger Enable	Section 21.5.13
0x108	PWM3RIS	PWM3 Raw Interrupt Status	Section 21.5.14
0x10C	PWM3ISC	PWM3 Interrupt Status and Clear	Section 21.5.15
0x110	PWM3LOAD	PWM3 Load	Section 21.5.16
0x114	PWM3COUNT	PWM3 Counter	Section 21.5.17
0x118	PWM3CMPA	PWM3 Compare A	Section 21.5.18
0x11C	PWM3CMPB	PWM3 Compare B	Section 21.5.19
0x120	PWM3GENA	PWM3 Generator A Control	Section 21.5.20
0x124	PWM3GENB	PWM3 Generator B Control	Section 21.5.21
0x128	PWM3DBCTL	PWM3 Dead-Band Control	Section 21.5.22
0x12C	PWM3DBRISE	PWM3 Dead-Band Rising-Edge Delay	Section 21.5.23
0x130	PWM3DBFALL	PWM3 Dead-Band Falling-Edge-Delay	Section 21.5.24
0x134	PWM3FLTSRC0	PWM3 Fault Source 0	Section 21.5.25
0x138	PWM3FLTSRC1	PWM3 Fault Source 1	Section 21.5.26
0x13C	PWM3MINFLTPER	PWM3 Minimum Fault Period	Section 21.5.27
0x800	PWM0FLTSEN	PWM0 Fault Pin Logic Sense	Section 21.5.28
0x804	PWM0FLTSTAT0	PWM0 Fault Status 0	Section 21.5.29
0x808	PWM0FLTSTAT1	PWM0 Fault Status 1	Section 21.5.30
0x880	PWM1FLTSEN	PWM1 Fault Pin Logic Sense	Section 21.5.28
0x884	PWM1FLTSTAT0	PWM1 Fault Status 0	Section 21.5.29
0x888	PWM1FLTSTAT1	PWM1 Fault Status 1	Section 21.5.30
0x900	PWM2FLTSEN	PWM2 Fault Pin Logic Sense	Section 21.5.28
0x904	PWM2FLTSTAT0	PWM2 Fault Status 0	Section 21.5.29
0x908	PWM2FLTSTAT1	PWM2 Fault Status 1	Section 21.5.30
0x980	PWM3FLTSEN	PWM3 Fault Pin Logic Sense	Section 21.5.28
0x984	PWM3FLTSTAT0	PWM3 Fault Status 0	Section 21.5.29

Table 21-1. PWM Registers (continued)

Offset	Acronym	Register Name	Section
0x988	PWM3FLTSTAT1	PWM3 Fault Status 1	Section 21.5.30
0xFC0	PWMPP	PWM Peripheral Properties	Section 21.5.31
0xFC8	PWMCC	PWM Clock Configuration	Section 21.5.32

Complex bit access types are encoded to fit into small table cells. [Table 21-2](#) shows the codes that are used for access types in this section.

Table 21-2. PWM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

21.5.1 PWMCTL Register (Offset = 0x0) [reset = 0x0]

PWM Master Control (PWMCTL)

This register provides master control over the PWM generation blocks.

PWMCTL is shown in [Figure 21-7](#) and described in [Table 21-3](#).

Return to [Summary Table](#).

Figure 21-7. PWMCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				GLOBALSYNC 3	GLOBALSYNC 2	GLOBALSYNC 1	GLOBALSYNC 0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-3. PWMCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	GLOBALSYNC3	R/W	0x0	Update PWM Generator 3. This bit automatically clears when the updates have completed; it cannot be cleared by software. 0x0 = No effect. 0x1 = Any queued update to a load or comparator register in PWM generator 3 is applied the next time the corresponding counter becomes zero.
2	GLOBALSYNC2	R/W	0x0	Update PWM Generator 2. This bit automatically clears when the updates have completed; it cannot be cleared by software. 0x0 = No effect. 0x1 = Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.
1	GLOBALSYNC1	R/W	0x0	Update PWM Generator 1. This bit automatically clears when the updates have completed; it cannot be cleared by software. 0x0 = No effect. 0x1 = Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.
0	GLOBALSYNC0	R/W	0x0	Update PWM Generator 0. This bit automatically clears when the updates have completed; it cannot be cleared by software. 0x0 = No effect. 0x1 = Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.

21.5.2 PWMSYNC Register (Offset = 0x4) [reset = 0x0]

PWM Time Base Sync (PWMSYNC)

This register provides a method to perform synchronization of the counters in the PWM generation blocks. Setting a bit in this register causes the specified counter to reset back to 0; setting multiple bits resets multiple counters simultaneously. The bits auto-clear after the reset has occurred; reading them back as zero indicates that the synchronization has completed.

PWMSYNC is shown in [Figure 21-8](#) and described in [Table 21-4](#).

Return to [Summary Table](#).

Figure 21-8. PWMSYNC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				SYNC3	SYNC2	SYNC1	SYNC0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-4. PWMSYNC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	SYNC3	R/W	0x0	Reset Generator 3 Counter. 0x0 = No effect. 0x1 = Resets the PWM generator 3 counter.
2	SYNC2	R/W	0x0	Reset Generator 2 Counter. 0x0 = No effect. 0x1 = Resets the PWM generator 2 counter.
1	SYNC1	R/W	0x0	Reset Generator 1 Counter. 0x0 = No effect. 0x1 = Resets the PWM generator 1 counter.
0	SYNC0	R/W	0x0	Reset Generator 0 Counter. 0x0 = No effect. 0x1 = Resets the PWM generator 0 counter.

21.5.3 PWMENABLE Register (Offset = 0x8) [reset = 0x0]

PWM Output Enable (PWMENABLE)

This register provides a master control of which generated pwmA' and pwmB' signals are output to the MnPWMn pins. By disabling a PWM output, the generation process can continue (for example, when the time bases are synchronized) without driving PWM signals to the pins. When bits in this register are set, the corresponding pwmA' or pwmB' signal is passed through to the output stage. When bits are clear, the pwmA' or pwmB' signal is replaced by a zero value which is also passed to the output stage. The PWMINVERT register controls the output stage, so if the corresponding bit is set in that register, the value seen on the MnPWMn signal is inverted from what is configured by the bits in this register. Updates to the bits in this register can be immediate or locally or globally synchronized to the next synchronous update as controlled by the ENUPDn fields in the PWMENUPD register.

PWMENABLE is shown in [Figure 21-9](#) and described in [Table 21-5](#).

Return to [Summary Table](#).

Figure 21-9. PWMENABLE Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
PWM7EN	PWM6EN	PWM5EN	PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-5. PWMENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	PWM7EN	R/W	0x0	MnPWM7 Output Enable. 0x0 = The MnPWM7 signal has a zero value. 0x1 = The generated pwm3B' signal is passed to the MnPWM7 pin.
6	PWM6EN	R/W	0x0	MnPWM6 Output Enable. 0x0 = The MnPWM6 signal has a zero value. 0x1 = The generated pwm3A' signal is passed to the MnPWM6 pin.
5	PWM5EN	R/W	0x0	MnPWM5 Output Enable. 0x0 = The MnPWM5 signal has a zero value. 0x1 = The generated pwm2B' signal is passed to the MnPWM5 pin.
4	PWM4EN	R/W	0x0	MnPWM4 Output Enable. 0x0 = The MnPWM4 signal has a zero value. 0x1 = The generated pwm2A' signal is passed to the MnPWM4 pin.
3	PWM3EN	R/W	0x0	MnPWM3 Output Enable. 0x0 = The MnPWM3 signal has a zero value. 0x1 = The generated pwm1B' signal is passed to the MnPWM3 pin.
2	PWM2EN	R/W	0x0	MnPWM2 Output Enable. 0x0 = The MnPWM2 signal has a zero value. 0x1 = The generated pwm1A' signal is passed to the MnPWM2 pin.

Table 21-5. PWMENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	PWM1EN	R/W	0x0	MnPWM1 Output Enable. 0x0 = The MnPWM1 signal has a zero value. 0x1 = The generated pwm0B' signal is passed to the MnPWM1 pin.
0	PWM0EN	R/W	0x0	MnPWM0 Output Enable. 0x0 = The MnPWM0 signal has a zero value. 0x1 = The generated pwm0A' signal is passed to the MnPWM0 pin.

21.5.4 PWMINVERT Register (Offset = 0xC) [reset = 0x0]

PWM Output Inversion (PWMINVERT)

This register provides a master control of the polarity of the MnPWMn signals on the device pins. The pwmA' and pwmB' signals generated by the PWM generator are active High; but can be made active Low via this register. Disabled PWM channels are also passed through the output inverter (if so configured) so that inactive signals can be high. In addition, if the PWMFAULT register enables a specific value to be placed on the MnPWMn signals during a fault condition, that value is inverted if the corresponding bit in this register is set.

PWMINVERT is shown in [Figure 21-10](#) and described in [Table 21-6](#).

Return to [Summary Table](#).

Figure 21-10. PWMINVERT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
PWM7INV	PWM6INV	PWM5INV	PWM4INV	PWM3INV	PWM2INV	PWM1INV	PWM0INV
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-6. PWMINVERT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	PWM7INV	R/W	0x0	Invert MnPWM7 Signal. 0x0 = The MnPWM7 signal is not inverted. 0x1 = The MnPWM7 signal is inverted.
6	PWM6INV	R/W	0x0	Invert MnPWM6 Signal. 0x0 = The MnPWM6 signal is not inverted. 0x1 = The MnPWM6 signal is inverted.
5	PWM5INV	R/W	0x0	Invert MnPWM5 Signal. 0x0 = The MnPWM5 signal is not inverted. 0x1 = The MnPWM5 signal is inverted.
4	PWM4INV	R/W	0x0	Invert MnPWM4 Signal. 0x0 = The MnPWM4 signal is not inverted. 0x1 = The MnPWM4 signal is inverted.
3	PWM3INV	R/W	0x0	Invert MnPWM3 Signal. 0x0 = The MnPWM3 signal is not inverted. 0x1 = The MnPWM3 signal is inverted.
2	PWM2INV	R/W	0x0	Invert MnPWM2 Signal. 0x0 = The MnPWM2 signal is not inverted. 0x1 = The MnPWM2 signal is inverted.
1	PWM1INV	R/W	0x0	Invert MnPWM1 Signal. 0x0 = The MnPWM1 signal is not inverted. 0x1 = The MnPWM1 signal is inverted.

Table 21-6. PWMINVERT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PWM0INV	R/W	0x0	Invert MnPWM0 Signal. 0x0 = The MnPWM0 signal is not inverted. 0x1 = The MnPWM0 signal is inverted.

21.5.5 PWMFAULT Register (Offset = 0x10) [reset = 0x0]

PWM Output Fault (PWMFAULT)

This register controls the behavior of the MnPWMn outputs in the presence of fault conditions. Both the fault inputs (MnFAULTn pins and digital comparator outputs) and debug events are considered fault conditions. On a fault condition, each pwmA' or pwmB' signal can be passed through unmodified or driven to the value specified by the corresponding bit in the PWMFAULTVAL register. For outputs that are configured for pass-through, the debug event handling on the corresponding PWM generator also determines if the pwmA' or pwmB' signal continues to be generated.

Fault condition control occurs before the output inverter, so PWM signals driven to a specified value on fault are inverted if the channel is configured for inversion (therefore, the pin is driven to the logical complement of the specified value on a fault condition).

PWMFAULT is shown in [Figure 21-11](#) and described in [Table 21-7](#).

Return to [Summary Table](#).

Figure 21-11. PWMFAULT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
FAULT7	FAULT6	FAULT5	FAULT4	FAULT3	FAULT2	FAULT1	FAULT0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-7. PWMFAULT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	FAULT7	R/W	0x0	MnPWM7 Fault. 0x0 = The generated pwm3B' signal is passed to the MnPWM7 pin. 0x1 = The MnPWM7 output signal is driven to the value specified by the PWM7 bit in the PWMFAULTVAL register.
6	FAULT6	R/W	0x0	MnPWM6 Fault. 0x0 = The generated pwm3A' signal is passed to the MnPWM6 pin. 0x1 = The MnPWM6 output signal is driven to the value specified by the PWM6 bit in the PWMFAULTVAL register.
5	FAULT5	R/W	0x0	MnPWM5 Fault. 0x0 = The generated pwm2B' signal is passed to the MnPWM5 pin. 0x1 = The MnPWM5 output signal is driven to the value specified by the PWM5 bit in the PWMFAULTVAL register.
4	FAULT4	R/W	0x0	MnPWM4 Fault. 0x0 = The generated pwm2A' signal is passed to the MnPWM4 pin. 0x1 = The MnPWM4 output signal is driven to the value specified by the PWM4 bit in the PWMFAULTVAL register.
3	FAULT3	R/W	0x0	MnPWM3 Fault. 0x0 = The generated pwm1B' signal is passed to the MnPWM3 pin. 0x1 = The MnPWM3 output signal is driven to the value specified by the PWM3 bit in the PWMFAULTVAL register.

Table 21-7. PWMFAULT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	FAULT2	R/W	0x0	MnPWM2 Fault. 0x0 = The generated pwm1A' signal is passed to the MnPWM2 pin. 0x1 = The MnPWM2 output signal is driven to the value specified by the PWM2 bit in the PWMFAULTVAL register.
1	FAULT1	R/W	0x0	MnPWM1 Fault. 0x0 = The generated pwm0B' signal is passed to the MnPWM1 pin. 0x1 = The MnPWM1 output signal is driven to the value specified by the PWM1 bit in the PWMFAULTVAL register.
0	FAULT0	R/W	0x0	MnPWM0 Fault. 0x0 = The generated pwm0A' signal is passed to the MnPWM0 pin. 0x1 = The MnPWM0 output signal is driven to the value specified by the PWM0 bit in the PWMFAULTVAL register.

21.5.6 PWMINTEN Register (Offset = 0x14) [reset = 0x0]

PWM Interrupt Enable (PWMINTEN)

This register controls the global interrupt generation capabilities of the PWM module. The events that can cause an interrupt are the fault input and the individual interrupts from the PWM generators.

NOTE: The "n" in the INTFAULTn and INTPWMn bits in this register correspond to the PWM generators, not to the FAULTn signals.

PWMINTEN is shown in [Figure 21-12](#) and described in [Table 21-8](#).

Return to [Summary Table](#).

Figure 21-12. PWMINTEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				INTFAULT3	INTFAULT2	INTFAULT1	INTFAULT0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				INTPWM3	INTPWM2	INTPWM1	INTPWM0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-8. PWMINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	INTFAULT3	R/W	0x0	Interrupt Fault 3. 0x0 = The fault condition for PWM generator 3 is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the fault condition for PWM generator 3 is asserted.
18	INTFAULT2	R/W	0x0	Interrupt Fault 2. 0x0 = The fault condition for PWM generator 2 is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the fault condition for PWM generator 2 is asserted.
17	INTFAULT1	R/W	0x0	Interrupt Fault 1. 0x0 = The fault condition for PWM generator 1 is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the fault condition for PWM generator 1 is asserted.
16	INTFAULT0	R/W	0x0	Interrupt Fault 0. 0x0 = The fault condition for PWM generator 0 is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the fault condition for PWM generator 0 is asserted.
15-4	RESERVED	R	0x0	

Table 21-8. PWMINTEN Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	INTPWM3	R/W	0x0	PWM3 Interrupt Enable. 0x0 = The PWM generator 3 interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PWM generator 3 block asserts an interrupt.
2	INTPWM2	R/W	0x0	PWM2 Interrupt Enable. 0x0 = The PWM generator 2 interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PWM generator 2 block asserts an interrupt.
1	INTPWM1	R/W	0x0	PWM1 Interrupt Enable. 0x0 = The PWM generator 1 interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PWM generator 1 block asserts an interrupt.
0	INTPWM0	R/W	0x0	PWM0 Interrupt Enable. 0x0 = The PWM generator 0 interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PWM generator 0 block asserts an interrupt.

21.5.7 PWMRS Register (Offset = 0x18) [reset = 0x0]

PWM Raw Interrupt Status (PWMRS)

This register provides the current set of interrupt sources that are asserted, regardless of whether they are enabled to cause an interrupt to be asserted to the interrupt controller. The fault interrupt is asserted based on the fault condition source that is specified by the PWMnCTL, PWMnFLTSRC0 and PWMnFLTSRC1 registers. The fault interrupt is latched on detection and must be cleared through the PWM Interrupt Status and Clear (PWMISC) register. The actual value of the MnFAULTn signals can be observed using the PWMSTATUS register.

The PWM generator interrupts simply reflect the status of the PWM generators and are cleared via the interrupt status register in the PWM generator blocks. If a bit is set, the event is active; if a bit is clear the event is not active.

PWMRS is shown in [Figure 21-13](#) and described in [Table 21-9](#).

Return to [Summary Table](#).

Figure 21-13. PWMRS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				INTFAULT3	INTFAULT2	INTFAULT1	INTFAULT0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				INTPWM3	INTPWM2	INTPWM1	INTPWM0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 21-9. PWMRS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	INTFAULT3	R	0x0	Interrupt Fault PWM 3. If the LATCH bit is set in the PWM3CTL register, the INTFAULT3 bit in this register can be cleared by writing a 1 to the INTFAULT3 bit in the PWMISC register. If the LATCH bit is 0 in the PWM3CTL register, writing a 1 to the INTFAULT3 bit in the PWMISC register has no effect. 0x0 = The fault condition for PWM generator 3 has not been asserted. 0x1 = The fault condition for PWM generator 3 is asserted.
18	INTFAULT2	R	0x0	Interrupt Fault PWM 2. If the LATCH bit is set in the PWM2CTL register, the INTFAULT2 bit in this register can be cleared by writing a 1 to the INTFAULT2 bit in the PWMISC register. If the LATCH bit is 0 in the PWM2CTL register, writing a 1 to the INTFAULT2 bit in the PWMISC register has no effect. 0x0 = The fault condition for PWM generator 2 has not been asserted. 0x1 = The fault condition for PWM generator 2 is asserted.
17	INTFAULT1	R	0x0	Interrupt Fault PWM 1. If the LATCH bit is set in the PWM1CTL register, the INTFAULT1 bit in this register can be cleared by writing a 1 to the INTFAULT1 bit in the PWMISC register. If the LATCH bit is 0 in the PWM1CTL register, writing a 1 to the INTFAULT1 bit in the PWMISC register has no effect. 0x0 = The fault condition for PWM generator 1 has not been asserted. 0x1 = The fault condition for PWM generator 1 is asserted.

Table 21-9. PWMRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
16	INTFAULT0	R	0x0	<p>Interrupt Fault PWM 0. If the LATCH bit is set in the PWM0CTL register, the INTFAULT0 bit in this register can be cleared by writing a 1 to the INTFAULT0 bit in the PWMISC register. If the LATCH bit is 0 in the PWM0CTL register, writing a 1 to the INTFAULT0 bit in the PWMISC register has no effect.</p> <p>0x0 = The fault condition for PWM generator 0 has not been asserted.</p> <p>0x1 = The fault condition for PWM generator 0 is asserted.</p>
15-4	RESERVED	R	0x0	
3	INTPWM3	R	0x0	<p>PWM3 Interrupt Asserted. The PWM3RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM3ISC register.</p> <p>0x0 = The PWM generator 3 block interrupt has not been asserted.</p> <p>0x1 = The PWM generator 3 block interrupt is asserted.</p>
2	INTPWM2	R	0x0	<p>PWM2 Interrupt Asserted. The PWM2RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM2ISC register.</p> <p>0x0 = The PWM generator 2 block interrupt has not been asserted.</p> <p>0x1 = The PWM generator 2 block interrupt is asserted.</p>
1	INTPWM1	R	0x0	<p>PWM1 Interrupt Asserted. The PWM1RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM1ISC register.</p> <p>0x0 = The PWM generator 1 block interrupt has not been asserted.</p> <p>0x1 = The PWM generator 1 block interrupt is asserted.</p>
0	INTPWM0	R	0x0	<p>PWM0 Interrupt Asserted. The PWM0RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM0ISC register.</p> <p>0x0 = The PWM generator 0 block interrupt has not been asserted.</p> <p>0x1 = The PWM generator 0 block interrupt is asserted.</p>

21.5.8 PWMISC Register (Offset = 0x1C) [reset = 0x0]

PWM Interrupt Status and Clear (PWMISC)

This register provides a summary of the interrupt status of the individual PWM generator blocks. If a fault interrupt is set, the corresponding MnFAULTn input has caused an interrupt. For the fault interrupt, a write of 1 to that bit position clears the latched interrupt status. If an block interrupt bit is set, the corresponding generator block is asserting an interrupt. The individual interrupt status registers, PWMnISC, in each block must be consulted to determine the reason for the interrupt and used to clear the interrupt.

PWMISC is shown in [Figure 21-14](#) and described in [Table 21-10](#).

Return to [Summary Table](#).

Figure 21-14. PWMISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				INTFAULT3	INTFAULT2	INTFAULT1	INTFAULT0
R-0x0				R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				INTPWM3	INTPWM2	INTPWM1	INTPWM0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 21-10. PWMISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19	INTFAULT3	R/W1C	0x0	FAULT3 Interrupt Asserted. Writing a 1 to this bit clears it and the INTFAULT3 bit in the PWMRI register. 0x0 = The fault condition for PWM generator 3 has not been asserted or is not enabled. 0x1 = An enabled interrupt for the fault condition for PWM generator 3 is asserted or is latched.
18	INTFAULT2	R/W1C	0x0	FAULT2 Interrupt Asserted. Writing a 1 to this bit clears it and the INTFAULT2 bit in the PWMRI register. 0x0 = The fault condition for PWM generator 2 has not been asserted or is not enabled. 0x1 = An enabled interrupt for the fault condition for PWM generator 2 is asserted or is latched.
17	INTFAULT1	R/W1C	0x0	FAULT1 Interrupt Asserted. Writing a 1 to this bit clears it and the INTFAULT1 bit in the PWMRI register. 0x0 = The fault condition for PWM generator 1 has not been asserted or is not enabled. 0x1 = An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.
16	INTFAULT0	R/W1C	0x0	FAULT0 Interrupt Asserted. Writing a 1 to this bit clears it and the INTFAULT0 bit in the PWMRI register. 0x0 = The fault condition for PWM generator 0 has not been asserted or is not enabled. 0x1 = An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.
15-4	RESERVED	R	0x0	

Table 21-10. PWMISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	INTPWM3	R	0x0	<p>PWM3 Interrupt Status. The PWM3RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM3ISC register.</p> <p>0x0 = The PWM generator 3 block interrupt is not asserted or is not enabled.</p> <p>0x1 = An enabled interrupt for the PWM generator 3 block is asserted.</p>
2	INTPWM2	R	0x0	<p>PWM2 Interrupt Status. The PWM2RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM2ISC register.</p> <p>0x0 = The PWM generator 2 block interrupt is not asserted or is not enabled.</p> <p>0x1 = An enabled interrupt for the PWM generator 2 block is asserted.</p>
1	INTPWM1	R	0x0	<p>PWM1 Interrupt Status. The PWM1RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM1ISC register.</p> <p>0x0 = The PWM generator 1 block interrupt is not asserted or is not enabled.</p> <p>0x1 = An enabled interrupt for the PWM generator 1 block is asserted.</p>
0	INTPWM0	R	0x0	<p>PWM0 Interrupt Status. The PWM0RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM0ISC register.</p> <p>0x0 = The PWM generator 0 block interrupt is not asserted or is not enabled.</p> <p>0x1 = An enabled interrupt for the PWM generator 0 block is asserted.</p>

21.5.9 PWMSTATUS Register (Offset = 0x20) [reset = 0x0]

PWM Status (PWMSTATUS)

This register provides the unlatched status of the PWM generator fault condition.

PWMSTATUS is shown in [Figure 21-15](#) and described in [Table 21-11](#).

Return to [Summary Table](#).

Figure 21-15. PWMSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				FAULT3	FAULT2	FAULT1	FAULT0
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 21-11. PWMSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	FAULT3	R	0x0	Generator 3 Fault Status. 0x0 = The fault condition for PWM generator 3 is not asserted. 0x1 = The fault condition for PWM generator 3 is asserted.If the FLTSRC bit in the PWM3CTL register is clear, the input is the source of the fault condition, and is therefore asserted.
2	FAULT2	R	0x0	Generator 2 Fault Status. 0x0 = The fault condition for PWM generator 2 is not asserted. 0x1 = The fault condition for PWM generator 2 is asserted.If the FLTSRC bit in the PWM2CTL register is clear, the input is the source of the fault condition, and is therefore asserted.
1	FAULT1	R	0x0	Generator 1 Fault Status. 0x0 = The fault condition for PWM generator 1 is not asserted. 0x1 = The fault condition for PWM generator 1 is asserted.If the FLTSRC bit in the PWM1CTL register is clear, the input is the source of the fault condition, and is therefore asserted.
0	FAULT0	R	0x0	Generator 0 Fault Status. 0x0 = The fault condition for PWM generator 0 is not asserted. 0x1 = The fault condition for PWM generator 0 is asserted.If the FLTSRC bit in the PWM0CTL register is clear, the input is the source of the fault condition, and is therefore asserted.

21.5.10 PWMFAULTVAL Register (Offset = 0x24) [reset = 0x0]

PWM Fault Condition Value (PWMFAULTVAL)

This register specifies the output value driven on the MnPWMn signals during a fault condition if enabled by the corresponding bit in the PWMFAULT register. Note that if the corresponding bit in the PWMINVERT register is set, the output value is driven to the logical NOT of the bit value in this register.

PWMFAULTVAL is shown in [Figure 21-16](#) and described in [Table 21-12](#).

Return to [Summary Table](#).

Figure 21-16. PWMFAULTVAL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-12. PWMFAULTVAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	PWM7	R/W	0x0	MnPWM7 Fault value. 0x0 = The MnPWM7 output signal is driven Low during fault conditions if the FAULT7 bit in the PWMFAULT register is set. 0x1 = The MnPWM7 output signal is driven High during fault conditions if the FAULT7 bit in the PWMFAULT register is set.
6	PWM6	R/W	0x0	MnPWM6 Fault value. 0x0 = The MnPWM6 output signal is driven Low during fault conditions if the FAULT6 bit in the PWMFAULT register is set. 0x1 = The MnPWM6 output signal is driven High during fault conditions if the FAULT 6 bit in the PWMFAULT register is set.
5	PWM5	R/W	0x0	MnPWM5 Fault value. 0x0 = The MnPWM5 output signal is driven Low during fault conditions if the FAULT5 bit in the PWMFAULT register is set. 0x1 = The MnPWM5 output signal is driven High during fault conditions if the FAULT5 bit in the PWMFAULT register is set.
4	PWM4	R/W	0x0	MnPWM4 Fault value. 0x0 = The MnPWM4 output signal is driven Low during fault conditions if the FAULT4 bit in the PWMFAULT register is set. 0x1 = The MnPWM4 output signal is driven High during fault conditions if the FAULT4 bit in the PWMFAULT register is set.
3	PWM3	R/W	0x0	MnPWM3 Fault value. 0x0 = The MnPWM3 output signal is driven Low during fault conditions if the FAULT3 bit in the PWMFAULT register is set. 0x1 = The MnPWM3 output signal is driven High during fault conditions if the FAULT3 bit in the PWMFAULT register is set.

Table 21-12. PWMFAULTVAL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	PWM2	R/W	0x0	MnPWM2 Fault value. 0x0 = The MnPWM2 output signal is driven Low during fault conditions if the FAULT2 bit in the PWMFAULT register is set. 0x1 = The MnPWM2 output signal is driven High during fault conditions if the FAULT2 bit in the PWMFAULT register is set.
1	PWM1	R/W	0x0	MnPWM1 Fault value. 0x0 = The MnPWM1 output signal is driven Low during fault conditions if the FAULT1 bit in the PWMFAULT register is set. 0x1 = The MnPWM1 output signal is driven High during fault conditions if the FAULT1 bit in the PWMFAULT register is set.
0	PWM0	R/W	0x0	MnPWM0 Fault value. 0x0 = The MnPWM0 output signal is driven Low during fault conditions if the FAULT0 bit in the PWMFAULT register is set. 0x1 = The MnPWM0 output signal is driven High during fault conditions if the FAULT0 bit in the PWMFAULT register is set.

21.5.11 PWMENUPD Register (Offset = 0x28) [reset = 0x0]

PWM Enable Update (PWMENUPD)

This register specifies when updates to the PWMnEN bit in the PWMENABLE register are performed. The PWMnEN bit enables the pwmA' or pwmB' output to be passed to the microcontroller's pin. Updates can be immediate or locally or globally synchronized to the next synchronous update.

PWMENUPD is shown in [Figure 21-17](#) and described in [Table 21-13](#).

Return to [Summary Table](#).

Figure 21-17. PWMENUPD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENUPD7		ENUPD6		ENUPD5		ENUPD4		ENUPD3		ENUPD2		ENUPD1		ENUPD0	
R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	

Table 21-13. PWMENUPD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-14	ENUPD7	R/W	0x0	MnPWM7 Enable Update Mode. 0x0 = ImmediateWrites to the PWM7EN bit in the PWMENABLE register are used by the PWM generator immediately. 0x1 = Reserved 0x2 = Locally SynchronizedWrites to the PWM7EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0. 0x3 = Globally SynchronizedWrites to the PWM7EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
13-12	ENUPD6	R/W	0x0	MnPWM6 Enable Update Mode. 0x0 = ImmediateWrites to the PWM6EN bit in the PWMENABLE register are used by the PWM generator immediately. 0x1 = Reserved 0x2 = Locally SynchronizedWrites to the PWM6EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0. 0x3 = Globally SynchronizedWrites to the PWM6EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
11-10	ENUPD5	R/W	0x0	MnPWM5 Enable Update Mode. 0x0 = ImmediateWrites to the PWM5EN bit in the PWMENABLE register are used by the PWM generator immediately. 0x1 = Reserved 0x2 = Locally SynchronizedWrites to the PWM5EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0. 0x3 = Globally SynchronizedWrites to the PWM5EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.

Table 21-13. PWMENUPD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	ENUPD4	R/W	0x0	<p>MnPWM4 Enable Update Mode.</p> <p>0x0 = ImmediateWrites to the PWM4EN bit in the PWMENABLE register are used by the PWM generator immediately.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedWrites to the PWM4EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedWrites to the PWM4EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</p>
7-6	ENUPD3	R/W	0x0	<p>MnPWM3 Enable Update Mode.</p> <p>0x0 = ImmediateWrites to the PWM3EN bit in the PWMENABLE register are used by the PWM generator immediately.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedWrites to the PWM3EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedWrites to the PWM3EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</p>
5-4	ENUPD2	R/W	0x0	<p>MnPWM2 Enable Update Mode.</p> <p>0x0 = ImmediateWrites to the PWM2EN bit in the PWMENABLE register are used by the PWM generator immediately.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedWrites to the PWM2EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedWrites to the PWM2EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</p>
3-2	ENUPD1	R/W	0x0	<p>MnPWM1 Enable Update Mode.</p> <p>0x0 = ImmediateWrites to the PWM1EN bit in the PWMENABLE register are used by the PWM generator immediately.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedWrites to the PWM1EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedWrites to the PWM1EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</p>
1-0	ENUPD0	R/W	0x0	<p>MnPWM0 Enable Update Mode.</p> <p>0x0 = ImmediateWrites to the PWM0EN bit in the PWMENABLE register are used by the PWM generator immediately.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedWrites to the PWM0EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedWrites to the PWM0EN bit in the PWMENABLE register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</p>

21.5.12 PWMnCTL Register [reset = 0x0]

PWM0 Control (PWM0CTL), offset 0x040

PWM1 Control (PWM1CTL), offset 0x080

PWM2 Control (PWM2CTL), offset 0x0C0

PWM3 Control (PWM3CTL), offset 0x100

These registers configure the PWM signal generation blocks (PWM0CTL controls the PWM generator 0 block, and so on). The Register Update mode, Debug mode, Counting mode, and Block Enable mode are all controlled via these registers. The blocks produce the PWM signals, which can be either two independent PWM signals (from the same counter), or a paired set of PWM signals with dead-band delays added.

The PWM0 block produces the MnPWM0 and MnPWM1 outputs, the PWM1 block produces the MnPWM2 and MnPWM3 outputs, the PWM2 block produces the MnPWM4 and MnPWM5 outputs, and the PWM3 block produces the MnPWM6 and MnPWM7 outputs.

PWMnCTL is shown in [Figure 21-18](#) and described in [Table 21-14](#).

Return to [Summary Table](#).

Figure 21-18. PWMnCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				LATCH		MINFLTPER	FLTSRC
R-0x0				R/W-0x0		R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
DBFALLUPD		DBRISEUPD		DBCTLUPD		GENBUPD	
R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	
7	6	5	4	3	2	1	0
GENAUPD		CMPBUPD	CMPAUPD	LOADUPD	DEBUG	MODE	ENABLE
R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-14. PWMnCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0x0	
18	LATCH	R/W	0x0	Latch fault input. When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the PWMnCTL register should be set to 1 to ensure trigger assertions are captured. 0x0 = Fault Condition Not LatchedA fault condition is in effect for as long as the generating source is asserting. 0x1 = Fault Condition LatchedA fault condition is set as the result of the assertion of the faulting source and is held (latched) while the PWMISC INTFAULTn bit is set. Clearing the INTFAULTn bit clears the fault condition.

Table 21-14. PWMnCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	MINFLTPER	R/W	0x0	<p>Minimum fault period. This bit specifies that the PWM generator enables a one-shot counter to provide a minimum fault condition period. The timer begins counting on the rising edge of the fault condition to extend the condition for a minimum duration of the count value. The timer ignores the state of the fault condition while counting. The minimum fault delay is in effect only when the MINFLTPER bit is set. If a detected fault is in the process of being extended when the MINFLTPER bit is cleared, the fault condition extension is aborted. The delay time is specified by the PWMnMINFLTPER register MFP field value. The effect of this is to pulse stretch the fault condition input. The delay value is defined by the PWM clock period. Because the fault input is not synchronized to the PWM clock, the period of the time is $PWMClock * (MFP\ value + 1)$ or $PWMClock * (MFP\ value + 2)$. The delay function makes sense only if the fault source is unlatched. A latched fault source makes the fault condition appear asserted until cleared by software and negates the utility of the extend feature. It applies to all fault condition sources as specified in the FLTSRC field. When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the PWMnCTL register should be set to 1 to ensure trigger assertions are captured.</p> <p>0x0 = The FAULT input deassertion is unaffected.</p> <p>0x1 = The PWMnMINFLTPER one-shot counter is active and extends the period of the fault condition to a minimum period.</p>
16	FLTSRC	R/W	0x0	<p>Fault condition source.</p> <p>0x0 = The Fault condition is determined by the Fault0 input.</p> <p>0x1 = The Fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.</p>
15-14	DBFALLUPD	R/W	0x0	<p>PWMnDBFALL update mode</p> <p>0x0 = ImmediateThe PWMnDBFALL register value is immediately updated on a write.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedUpdates to the register are reflected to the generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
13-12	DBRISEUPD	R/W	0x0	<p>PWMnDBRISE update mode.</p> <p>0x0 = ImmediateThe PWMnDBRISE register value is immediately updated on a write.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedUpdates to the register are reflected to the generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
11-10	DBCTLUPD	R/W	0x0	<p>PWMnDBCTL update mode.</p> <p>0x0 = ImmediateThe PWMnDBCTL register value is immediately updated on a write.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedUpdates to the register are reflected to the generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>

Table 21-14. PWMnCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	GENBUPD	R/W	0x0	<p>PWMnGENB update mode.</p> <p>0x0 = ImmediateThe PWMnGENB register value is immediately updated on a write.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally SynchronizedUpdates to the register are reflected to the generator the next time the counter is 0.</p> <p>0x3 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
7-6	GENAUPD	R/W	0x0	<p>PWMnGENA update mode.</p> <p>0x0 = Immediate. The PWMnGENA register value is immediately updated on a write.</p> <p>0x1 = Reserved</p> <p>0x2 = Locally Synchronized. Updates to the register are reflected to the generator the next time the counter is 0.</p> <p>0x3 = Globally Synchronized. Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
5	CMPBUPD	R/W	0x0	<p>Comparator B update mode.</p> <p>0x0 = Locally SynchronizedUpdates to the PWMnCMPB register are reflected to the generator the next time the counter is 0.</p> <p>0x1 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
4	CMPAUPD	R/W	0x0	<p>Comparator A update mode.</p> <p>0x0 = Locally SynchronizedUpdates to the PWMnCMPA register are reflected to the generator the next time the counter is 0.</p> <p>0x1 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
3	LOADUPD	R/W	0x0	<p>Load register update mode.</p> <p>0x0 = Locally SynchronizedUpdates to the PWMnLOAD register are reflected to the generator the next time the counter is 0.</p> <p>0x1 = Globally SynchronizedUpdates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL register.</p>
2	DEBUG	R/W	0x0	<p>Debug mode.</p> <p>0x0 = The counter stops running when it next reaches 0 and continues running again when no longer in Debug mode.</p> <p>0x1 = The counter always runs when in Debug mode.</p>
1	MODE	R/W	0x0	<p>Counter mode.</p> <p>0x0 = The counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode).</p> <p>0x1 = The counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).</p>
0	ENABLE	R/W	0x0	<p>PWM block enable. Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the PWMnCOUNT register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the SRPWM register in the System Control Module.</p> <p>0x0 = The entire PWM generation block is disabled and not clocked.</p> <p>0x1 = The PWM generation block is enabled and produces PWM signals.</p>

21.5.13 PWMnINTEN Register [reset = 0x0]

PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044

PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084

PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4

PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104

These registers control the interrupt and ADC trigger generation capabilities of the PWM generators (PWM0INTEN controls the PWM generator 0 block, and so on). The events that can cause an interrupt, or an ADC trigger are:

- The counter being equal to the load register
- The counter being equal to zero
- The counter being equal to the PWMnCMPA register while counting up
- The counter being equal to the PWMnCMPA register while counting down
- The counter being equal to the PWMnCMPB register while counting up
- The counter being equal to the PWMnCMPB register while counting down

Any combination of these events can generate either an interrupt or an ADC trigger, though no determination can be made as to the actual event that caused an ADC trigger if more than one is specified. The PWMnRIS register provides information about which events have caused raw interrupts.

PWMnINTEN is shown in [Figure 21-19](#) and described in [Table 21-15](#).

Return to [Summary Table](#).

Figure 21-19. PWMnINTEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED		TRCMPBD	TRCMPBU	TRCMPAD	TRCMPAU	TRCNTLOAD	TRCNTZERO
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED		INTCMPBD	INTCMPBU	INTCMPAD	INTCMPAU	INTCNTLOAD	INTCNTZERO
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-15. PWMnINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0x0	
13	TRCMPBD	R/W	0x0	Trigger for Counter= PWMnCMPB down. 0x0 = No ADC trigger is output. 0x1 = An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB register value while counting down.
12	TRCMPBU	R/W	0x0	Trigger for Counter= PWMnCMPB up. 0x0 = No ADC trigger is output. 0x1 = An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB register value while counting up.
11	TRCMPAD	R/W	0x0	Trigger for Counter= PWMnCMPA down. 0x0 = No ADC trigger is output. 0x1 = An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA register value while counting down.

Table 21-15. PWMnINTEN Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	TRCMPAU	R/W	0x0	Trigger for Counter= PWMnCMPA up. 0x0 = No ADC trigger is output. 0x1 = An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA register value while counting up.
9	TRCNTLOAD	R/W	0x0	Trigger for Counter= PWMnLOAD. 0x0 = No ADC trigger is output. 0x1 = An ADC trigger pulse is output when the counter matches the PWMnLOAD register.
8	TRCNTZERO	R/W	0x0	Trigger for Counter=0. 0x0 = No ADC trigger is output. 0x1 = An ADC trigger pulse is output when the counter is 0.
7-6	RESERVED	R	0x0	
5	INTCMPBD	R/W	0x0	Interrupt for Counter= PWMnCMPB down. 0x0 = No interrupt. 0x1 = A raw interrupt occurs when the counter matches the value in the PWMnCMPB register value while counting down.
4	INTCMPBU	R/W	0x0	Interrupt for Counter= PWMnCMPB up. 0x0 = No interrupt. 0x1 = A raw interrupt occurs when the counter matches the value in the PWMnCMPB register value while counting up.
3	INTCMPAD	R/W	0x0	Interrupt for Counter= PWMnCMPA down. 0x0 = No interrupt. 0x1 = A raw interrupt occurs when the counter matches the value in the PWMnCMPA register value while counting down.
2	INTCMPAU	R/W	0x0	Interrupt for Counter= PWMnCMPA up. 0x0 = No interrupt. 0x1 = A raw interrupt occurs when the counter matches the value in the PWMnCMPA register value while counting up.
1	INTCNTLOAD	R/W	0x0	Interrupt for Counter= PWMnLOAD. 0x0 = No interrupt. 0x1 = A raw interrupt occurs when the counter matches the value in the PWMnLOAD register value.
0	INTCNTZERO	R/W	0x0	Interrupt for Counter=0. 0x0 = No interrupt. 0x1 = A raw interrupt occurs when the counter is zero.

21.5.14 PWMnRIS Register [reset = 0x0]

PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048

PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088

PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8

PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108

These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (PWM0RIS controls the PWM generator 0 block, and so on). If a bit is set, the event has occurred; if a bit is clear, the event has not occurred. Bits in this register are cleared by writing a 1 to the corresponding bit in the PWMnISC register.

PWMnRIS is shown in [Figure 21-20](#) and described in [Table 21-16](#).

Return to [Summary Table](#).

Figure 21-20. PWMnRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		INTCMPBD	INTCMPBU	INTCMPAD	INTCMPAU	INTCNTLOAD	INTCNTZERO
R-0x0		R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 21-16. PWMnRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5	INTCMPBD	R	0x0	Comparator B down interrupt status. This bit is cleared by writing a 1 to the INTCMPBD bit in the PWMnISC register. 0x0 = An interrupt has not occurred. 0x1 = The counter has matched the value in the PWMnCMPB register while counting down.
4	INTCMPBU	R	0x0	Comparator B up interrupt status. This bit is cleared by writing a 1 to the INTCMPBU bit in the PWMnISC register. 0x0 = An interrupt has not occurred. 0x1 = The counter has matched the value in the PWMnCMPB register while counting up.
3	INTCMPAD	R	0x0	Comparator A down interrupt status. This bit is cleared by writing a 1 to the INTCMPAD bit in the PWMnISC register. 0x0 = An interrupt has not occurred. 0x1 = The counter has matched the value in the PWMnCMPA register while counting down.
2	INTCMPAU	R	0x0	Comparator A up interrupt status. This bit is cleared by writing a 1 to the INTCMPAU bit in the PWMnISC register. 0x0 = An interrupt has not occurred. 0x1 = The counter has matched the value in the PWMnCMPA register while counting up.

Table 21-16. PWMnRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	INTCNTLOAD	R	0x0	Counter=Load interrupt status. This bit is cleared by writing a 1 to the INCNTLOAD bit in the PWMnISC register. 0x0 = An interrupt has not occurred. 0x1 = The counter has matched the value in the PWMnLOAD register.
0	INTCNTZERO	R	0x0	Counter=0 interrupt status. This bit is cleared by writing a 1 to the INCNTZERO bit in the PWMnISC register. 0x0 = An interrupt has not occurred. 0x1 = The counter has matched zero.

21.5.15 PWMnISC Register [reset = 0x0]

PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C

PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C

PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC

PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C

These registers provide the current set of interrupt sources that are asserted to the interrupt controller (PWM0ISC controls the PWM generator 0 block, and so on). A bit is set if the event has occurred and is enabled in the PWMnINTEN register; if a bit is clear, the event has not occurred or is not enabled. These are RW1C registers; writing a 1 to a bit position clears the corresponding interrupt reason.

NOTE: The interrupt status can only be cleared one PWM Clock cycle after the interrupt occurs. The larger the PWM Clock Divider (PWMDIV) value in PWMCC register, the longer the system delay is to clear the interrupt.

PWMnISC is shown in [Figure 21-21](#) and described in [Table 21-17](#).

Return to [Summary Table](#).

Figure 21-21. PWMnISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED		INTCMPBD	INTCMPBU	INTCMPAD	INTCMPAU	INTCNTLOAD	INTCNTZERO
R-0x0		R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 21-17. PWMnISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5	INTCMPBD	R/W1C	0x0	Comparator B down interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBD bit in the PWMnRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTCMPBD bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.
4	INTCMPBU	R/W1C	0x0	Comparator B up interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBU bit in the PWMnRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTCMPBU bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.
3	INTCMPAD	R/W1C	0x0	Comparator A down interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAD bit in the PWMnRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTCMPAD bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.

Table 21-17. PWMnISC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	INTCMPAU	R/W1C	0x0	Comparator A up interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAU bit in the PWMnRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTCMPAU bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.
1	INTCNTLOAD	R/W1C	0x0	Counter=Load interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTLOAD bit in the PWMnRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTCNTLOAD bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.
0	INTCNTZERO	R/W1C	0x0	Counter=0 interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTZERO bit in the PWMnRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTCNTZERO bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.

21.5.16 PWMnLOAD Register [reset = 0x0]

PWM0 Load (PWM0LOAD), offset 0x050

PWM1 Load (PWM1LOAD), offset 0x090

PWM2 Load (PWM2LOAD), offset 0x0D0

PWM3 Load (PWM3LOAD), offset 0x110

These registers contain the load value for the PWM counter (PWM0LOAD controls the PWM generator 0 block, and so on). Based on the counter mode configured by the MODE bit in the PWMnCTL register, this value is either loaded into the counter after it reaches zero or is the limit of up-counting after which the counter decrements back to zero. When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and/or pwmB signal (via the PWMnGENA / PWMnGENB register) or drive an interrupt or ADC trigger (via the PWMnINTEN register).

If the Load Value Update mode is locally synchronized (based on the LOADUPD field encoding in the PWMnCTL register), the 16-bit LOAD value is used the next time the counter reaches zero. If the update mode is globally synchronized, it is used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is re-written before the actual update occurs, the previous value is never used and is lost.

PWMnLOAD is shown in [Figure 21-22](#) and described in [Table 21-18](#).

Return to [Summary Table](#).

Figure 21-22. PWMnLOAD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																LOAD															
R-0x0																R/W-0x0															

Table 21-18. PWMnLOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	LOAD	R/W	0x0	Counter load value. The counter load value.

21.5.17 PWMnCOUNT Register [reset = 0x0]

PWM0 Counter (PWM0COUNT), offset 0x054

PWM1 Counter (PWM1COUNT), offset 0x094

PWM2 Counter (PWM2COUNT), offset 0x0D4

PWM3 Counter (PWM3COUNT), offset 0x114

These registers contain the current value of the PWM counter (PWM0COUNT is the value of the PWM generator 0 block, and so on). When this value matches zero or the value in the PWMnLOAD, PWMnCMPA, or PWMnCMPB registers, a pulse is output which can be configured to drive the generation of a PWM signal or drive an interrupt or ADC trigger.

NOTE: Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the PWMnCOUNT register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the SRPWM register in the System Control Module.

PWMnCOUNT is shown in [Figure 21-23](#) and described in [Table 21-19](#).

Return to [Summary Table](#).

Figure 21-23. PWMnCOUNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																COUNT															
R-0x0																R-0x0															

Table 21-19. PWMnCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	COUNT	R	0x0	Counter value. The current value of the counter.

21.5.18 PWMnCMPA Register [reset = 0x0]

PWM0 Compare A (PWM0CMPA), offset 0x058

PWM1 Compare A (PWM1CMPA), offset 0x098

PWM2 Compare A (PWM2CMPA), offset 0x0D8

PWM3 Compare A (PWM3CMPA), offset 0x118

These registers contain a value to be compared against the counter (PWM0CMPA controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and pwmB signals (via the PWMnGENA and PWMnGENB registers) or drive an interrupt or ADC trigger (via the PWMnINTEN register). If the value of this register is greater than the PWMnLOAD register (see [Section 21.5.16](#)), then no pulse is ever output.

If the comparator A update mode is locally synchronized (based on the CMPAUPD bit in the PWMnCTL register), the 16-bit COMPA value is used the next time the counter reaches zero. If the update mode is globally synchronized, it is used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnCMPA is shown in [Figure 21-24](#) and described in [Table 21-20](#).

Return to [Summary Table](#).

Figure 21-24. PWMnCMPA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																COMPA															
R-0x0																R/W-0x0															

Table 21-20. PWMnCMPA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	COMPA	R/W	0x0	Comparator A value. The value to be compared against the counter.

21.5.19 PWMnCMPB Register [reset = 0x0]

PWM0 Compare B (PWM0CMPB), offset 0x05C

PWM1 Compare B (PWM1CMPB), offset 0x09C

PWM2 Compare B (PWM2CMPB), offset 0x0DC

PWM3 Compare B (PWM3CMPB), offset 0x11C

These registers contain a value to be compared against the counter (PWM0CMPB controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and pwmB signals (via the PWMnGENA and PWMnGENB registers) or drive an interrupt or ADC trigger (via the PWMnINTEN register). If the value of this register is greater than the PWMnLOAD register, no pulse is ever output.

If the comparator B update mode is locally synchronized (based on the CMPBUPD bit in the PWMnCTL register), the 16-bit COMPB value is used the next time the counter reaches zero. If the update mode is globally synchronized, it is used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnCMPB is shown in [Figure 21-25](#) and described in [Table 21-21](#).

Return to [Summary Table](#).

Figure 21-25. PWMnCMPB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																COMPB															
R-0x0																R/W-0x0															

Table 21-21. PWMnCMPB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	COMPB	R/W	0x0	Comparator B value. The value to be compared against the counter.

21.5.20 PWMnGENA Register [reset = 0x0]

PWM0 Generator A Control (PWM0GENA), offset 0x060

PWM1 Generator A Control (PWM1GENA), offset 0x0A0

PWM2 Generator A Control (PWM2GENA), offset 0x0E0

PWM3 Generator A Control (PWM3GENA), offset 0x120

These registers control the generation of the pwmA signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (PWM0GENA controls the PWM generator 0 block, and so on). When the counter is running in Count-Down mode, only four of these events occur; when running in Count-Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the resulting PWM signal.

The PWM0GENA register controls generation of the pwm0A signal; PWM1GENA, the pwm1A signal; PWM2GENA, the pwm2A signal; and PWM3GENA, the pwm3A signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare A action is taken and the compare B action is ignored.

If the Generator A update mode is immediate (based on the GENAUPD field encoding in the PWMnCTL register), the ACTCMPBD, ACTCMPBU, ACTCMPAD, ACTCMPAU, ACTLOAD, and ACTZERO values are used immediately. If the update mode is locally synchronized, these values are used the next time the counter reaches zero. If the update mode is globally synchronized, these values are used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnGENA is shown in [Figure 21-26](#) and described in [Table 21-22](#).

Return to [Summary Table](#).

Figure 21-26. PWMnGENA Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				ACTCMPBD		ACTCMPBU	
R-0x0				R/W-0x0		R/W-0x0	
7	6	5	4	3	2	1	0
ACTCMPAD		ACTCMPAU		ACTLOAD		ACTZERO	
R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	

Table 21-22. PWMnGENA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11-10	ACTCMPBD	R/W	0x0	Action for Comparator B Down. This field specifies the action to be taken when the counter matches comparator B while counting down. 0x0 = Do nothing 0x1 = Invert pwmA 0x2 = Drive pwmA low 0x3 = Drive pwmA high

Table 21-22. PWMnGENA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	ACTCMPBU	R/W	0x0	Action for Comparator B Up. This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set. 0x0 = Do nothing 0x1 = Invert pwmA 0x2 = Drive pwmA low 0x3 = Drive pwmA high
7-6	ACTCMPAD	R/W	0x0	Action for Comparator A Down. This field specifies the action to be taken when the counter matches comparator A while counting down. 0x0 = Do nothing 0x1 = Invert pwmA 0x2 = Drive pwmA low 0x3 = Drive pwmA high
5-4	ACTCMPAU	R/W	0x0	Action for Comparator A Up. This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set. 0x0 = Do nothing 0x1 = Invert pwmA 0x2 = Drive pwmA low 0x3 = Drive pwmA high
3-2	ACTLOAD	R/W	0x0	Action for Counter= LOAD. This field specifies the action to be taken when the counter matches the value in the PWMnLOAD register. 0x0 = Do nothing 0x1 = Invert pwmA 0x2 = Drive pwmA low 0x3 = Drive pwmA high
1-0	ACTZERO	R/W	0x0	Action for Counter=0. This field specifies the action to be taken when the counter is zero. 0x0 = Do nothing 0x1 = Invert pwmA 0x2 = Drive pwmA low 0x3 = Drive pwmA high

21.5.21 PWMnGENB Register [reset = 0x0]

PWMn Generator B Control (PWMnGENB), offset 0x064

PWM0 Generator B Control (PWM0GENB), offset 0x064

PWM1 Generator B Control (PWM1GENB), offset 0x0A4

PWM2 Generator B Control (PWM2GENB), offset 0x0E4

PWM3 Generator B Control (PWM3GENB), offset 0x124

These registers control the generation of the pwmB signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (PWM0GENB controls the PWM generator 0 block, and so on). When the counter is running in Count-Down mode, only four of these events occur; when running in Count-Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the resulting PWM signal.

The PWM0GENB register controls generation of the pwm0B signal; PWM1GENB, the pwm1B signal; PWM2GENB, the pwm2B signal; and PWM3GENB, the pwm3B signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare B action is taken and the compare A action is ignored.

If the Generator B update mode is immediate (based on the GENBUPD field encoding in the PWMnCTL register), the ACTCMPBD, ACTCMPBU, ACTCMPAD, ACTCMPAU, ACTLOAD, and ACTZERO values are used immediately. If the update mode is locally synchronized, these values are used the next time the counter reaches zero. If the update mode is globally synchronized, these values are used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnGENB is shown in [Figure 21-27](#) and described in [Table 21-23](#).

Return to [Summary Table](#).

Figure 21-27. PWMnGENB Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				ACTCMPBD		ACTCMPBU	
R-0x0				R/W-0x0		R/W-0x0	
7	6	5	4	3	2	1	0
ACTCMPAD		ACTCMPAU		ACTLOAD		ACTZERO	
R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	

Table 21-23. PWMnGENB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11-10	ACTCMPBD	R/W	0x0	Action for Comparator B Down This field specifies the action to be taken when the counter matches comparator B while counting down. 0x0 = Do nothing 0x1 = Invert pwmB 0x2 = Drive pwmB low 0x3 = Drive pwmB high

Table 21-23. PWMnGENB Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	ACTCMPBU	R/W	0x0	Action for Comparator B Up This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set. 0x0 = Do nothing 0x1 = Invert pwmB 0x2 = Drive pwmB low 0x3 = Drive pwmB high
7-6	ACTCMPAD	R/W	0x0	Action for Comparator A Down This field specifies the action to be taken when the counter matches comparator A while counting down. 0x0 = Do nothing 0x1 = Invert pwmB 0x2 = Drive pwmB low 0x3 = Drive pwmB high
5-4	ACTCMPAU	R/W	0x0	Action for Comparator A Up This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set. 0x0 = Do nothing 0x1 = Invert pwmB 0x2 = Drive pwmB low 0x3 = Drive pwmB high
3-2	ACTLOAD	R/W	0x0	Action for Counter= LOAD This field specifies the action to be taken when the counter matches the load value. 0x0 = Do nothing 0x1 = Invert pwmB 0x2 = Drive pwmB low 0x3 = Drive pwmB high
1-0	ACTZERO	R/W	0x0	Action for Counter=0 This field specifies the action to be taken when the counter is 0. 0x0 = Do nothing 0x1 = Invert pwmB 0x2 = Drive pwmB low 0x3 = Drive pwmB high

21.5.22 PWMnDBCTL Register [reset = 0x0]

PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068

PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8

PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8

PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128

The PWMnDBCTL register controls the dead-band generator, which produces the MnPWMn signals based on the pwmA and pwmB signals. When disabled, the pwmA signal passes through to the pwmA' signal and the pwmB signal passes through to the pwmB' signal. When dead-band control is enabled, the pwmB signal is ignored, the pwmA' signal is generated by delaying the rising edge(s) of the pwmA signal by the value in the PWMnDBRISE register (see [Section 21.5.23](#)), and the pwmB' signal is generated by inverting the pwmA signal and delaying the falling edge(s) of the pwmA signal by the value in the PWMnDBFALL register (see [Section 21.5.24](#)). The Output Control block outputs the pwm0A' signal on the MnPWM0 signal and the pwm0B' signal on the MnPWM1 signal. In a similar manner, MnPWM2 and MnPWM3 are produced from the pwm1A' and pwm1B' signals, MnPWM4 and MnPWM5 are produced from the pwm2A' and pwm2B' signals, and MnPWM6 and MnPWM7 are produced from the pwm3A' and pwm3B' signals.

If the Dead-Band Control mode is immediate (based on the DBCTLUPD field encoding in the PWMnCTL register), the ENABLE bit value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value is used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnDBCTL is shown in [Figure 21-28](#) and described in [Table 21-24](#).

Return to [Summary Table](#).

Figure 21-28. PWMnDBCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R-0x0							R/W-0x0

Table 21-24. PWMnDBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	ENABLE	R/W	0x0	Dead-Band Generator Enable. 0x0 = The pwmA and pwmB signals pass through to the pwmA' and pwmB' signals unmodified. 0x1 = The dead-band generator modifies the pwmA signal by inserting dead bands into the pwmA' and pwmB' signals.

21.5.23 PWMnDBRISE Register [reset = 0x0]

PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C

PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC

PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC

PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C

The PWMnDBRISE register contains the number of clock cycles to delay the rising edge of the pwmA signal when generating the pwmA' signal. If the dead-band generator is disabled through the PWMnDBCTL register, this register is ignored. If the value of this register is larger than the width of a High pulse on the pwmA signal, the rising-edge delay consumes the entire High time of the signal, resulting in no High time on the output. Care must be taken to ensure that the pwmA High time always exceeds the rising-edge delay.

If the Dead-Band Rising-Edge Delay mode is immediate (based on the DBRISEUPD field encoding in the PWMnCTL register), the 12-bit RISEDELAY value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value is used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnDBRISE is shown in [Figure 21-29](#) and described in [Table 21-25](#).

Return to [Summary Table](#).

Figure 21-29. PWMnDBRISE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
RESERVED																				RISEDELAY																	
R-0x0																				R/W-0x0																	

Table 21-25. PWMnDBRISE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11-0	RISEDELAY	R/W	0x0	Dead-Band Rise Delay. The number of clock cycles to delay the rising edge of pwmA' after the rising edge of pwmA.

21.5.24 PWMnDBFALL Register [reset = 0x0]

PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070

PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0

PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0

PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130

The PWMnDBFALL register contains the number of clock cycles to delay the rising edge of the pwmB' signal from the falling edge of the pwmA signal. If the dead-band generator is disabled through the PWMnDBCTL register, this register is ignored. If the value of this register is larger than the width of a Low pulse on the pwmA signal, the falling-edge delay consumes the entire Low time of the signal, resulting in no Low time on the output. Care must be taken to ensure that the pwmA Low time always exceeds the falling-edge delay.

If the Dead-Band Falling-Edge-Delay mode is immediate (based on the DBFALLUP field encoding in the PWMnCTL register), the 12-bit FALLDELAY value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value is used the next time the counter reaches zero after a synchronous update has been requested through the PWM Master Control (PWMCTL) register (see [Section 21.5.1](#)). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMnDBFALL is shown in [Figure 21-30](#) and described in [Table 21-26](#).

Return to [Summary Table](#).

Figure 21-30. PWMnDBFALL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												FALLDELAY																			
R-0x0												R/W-0x0																			

Table 21-26. PWMnDBFALL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11-0	FALLDELAY	R/W	0x0	Dead-Band Fall Delay. The number of clock cycles to delay the falling edge of pwmB' from the rising edge of pwmA.

21.5.25 PWMnFLTSRC0 Register [reset = 0x0]

PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074

PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4

PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4

PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134

This register specifies which fault pin inputs are used to generate a fault condition. Each bit in the following register indicates whether the corresponding fault pin is included in the fault condition. All enabled fault pins are ORed together to form the PWMnFLTSRC0 portion of the fault condition. The PWMnFLTSRC0 fault condition is then ORed with the PWMnFLTSRC1 fault condition to generate the final fault condition for the PWM generator.

If the FLTSRC bit in the PWMnCTL register (see [Section 21.5.12](#)) is clear, only the Fault0 signal affects the fault condition generated. Otherwise, sources defined in PWMnFLTSRC0 and PWMnFLTSRC1 affect the fault condition generated.

PWMnFLTSRC0 is shown in [Figure 21-31](#) and described in [Table 21-27](#).

Return to [Summary Table](#).

Figure 21-31. PWMnFLTSRC0 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
reserved_2							
R-0x0							
7	6	5	4	3	2	1	0
reserved_2				FAULT3	FAULT2	FAULT1	FAULT0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-27. PWMnFLTSRC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-4	RESERVED	R	0x0	
3	FAULT3	R/W	0x0	Fault3 Input. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The Fault3 signal is suppressed and cannot generate a fault condition. 0x1 = The Fault3 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
2	FAULT2	R/W	0x0	Fault2 Input. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The Fault2 signal is suppressed and cannot generate a fault condition. 0x1 = The Fault2 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
1	FAULT1	R/W	0x0	Fault1 Input. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The Fault1 signal is suppressed and cannot generate a fault condition. 0x1 = The Fault1 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).

Table 21-27. PWMnFLTSRC0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	FAULT0	R/W	0x0	<p>Fault0 Input. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.</p> <p>0x0 = The Fault0 signal is suppressed and cannot generate a fault condition.</p> <p>0x1 = The Fault0 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</p>

21.5.26 PWMnFLTSRC1 Register [reset = 0x0]

PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078

PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8

PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8

PWM3 Fault Source 1 (PWM3FLTSRC1), offset 0x138

This register specifies which digital comparator triggers from the ADC are used to generate a fault condition. Each bit in the following register indicates whether the corresponding digital comparator trigger is included in the fault condition. All enabled digital comparator triggers are ORed together to form the PWMnFLTSRC1 portion of the fault condition. The PWMnFLTSRC1 fault condition is then ORed with the PWMnFLTSRC0 fault condition to generate the final fault condition for the PWM generator.

If the FLTSRC bit in the PWMnCTL register (see [Section 21.5.12](#)) is clear, only the PWM Fault0 pin affects the fault condition generated. Otherwise, sources defined in PWMnFLTSRC0 and PWMnFLTSRC1 affect the fault condition generated.

PWMnFLTSRC1 is shown in [Figure 21-32](#) and described in [Table 21-28](#).

Return to [Summary Table](#).

Figure 21-32. PWMnFLTSRC1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-28. PWMnFLTSRC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	DCMP7	R/W	0x0	Digital Comparator 7. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 7 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 7 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
6	DCMP6	R/W	0x0	Digital Comparator 6. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 6 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 6 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).

Table 21-28. PWMnFLTSRC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	DCMP5	R/W	0x0	Digital Comparator 5. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 5 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 5 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
4	DCMP4	R/W	0x0	Digital Comparator 4. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 4 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 4 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
3	DCMP3	R/W	0x0	Digital Comparator 3. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 3 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 3 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
2	DCMP2	R/W	0x0	Digital Comparator 2. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 2 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 2 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
1	DCMP1	R/W	0x0	Digital Comparator 1. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 1 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 1 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
0	DCMP0	R/W	0x0	Digital Comparator 0. The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation. 0x0 = The trigger from digital comparator 0 is suppressed and cannot generate a fault condition. 0x1 = The trigger from digital comparator 0 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).

21.5.27 PWMnMINFLTPER Register [reset = 0x0]

PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C

PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC

PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC

PWM3 Minimum Fault Period (PWM3MINFLTPER), offset 0x13C

If the MINFLTPER bit in the PWMnCTL register is set, this register specifies the 16-bit time-extension value to be used in extending the fault condition. The value is loaded into a 16-bit down counter, and the counter value is used to extend the fault condition. The fault condition is released in the clock immediately after the counter value reaches 0. The fault condition is asynchronous to the PWM clock; and the delay value is the product of the PWM clock period and the (MFP field value + 1) or (MFP field value + 2) depending on when the fault condition asserts with respect to the PWM clock. The counter decrements at the PWM clock rate, without pause or condition.

PWMnMINFLTPER is shown in [Figure 21-33](#) and described in [Table 21-29](#).

Return to [Summary Table](#).

Figure 21-33. PWMnMINFLTPER Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MFP															
R-0x0																R/W-0x0															

Table 21-29. PWMnMINFLTPER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	MFP	R/W	0x0	Minimum Fault Period. The number of PWM clocks by which a fault condition is extended when the delay is enabled by PWMnCTL MINFLTPER.

21.5.28 PWMnFLTSEN Register [reset = 0x0]

PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800

PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880

PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900

PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980

This register defines the PWM fault pin logic sense.

PWMnFLTSEN is shown in [Figure 21-34](#) and described in [Table 21-30](#).

Return to [Summary Table](#).

Figure 21-34. PWMnFLTSEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				FAULT3	FAULT2	FAULT1	FAULT0
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 21-30. PWMnFLTSEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	FAULT3	R/W	0x0	Fault3 Sense. 0x0 = An error is indicated if the Fault3 signal is high 0x1 = An error is indicated if the Fault3 signal is low
2	FAULT2	R/W	0x0	Fault2 Sense. 0x0 = An error is indicated if the Fault2 signal is high 0x1 = An error is indicated if the Fault2 signal is low
1	FAULT1	R/W	0x0	Fault1 Sense. 0x0 = An error is indicated if the Fault1 signal is high 0x1 = An error is indicated if the Fault1 signal is low
0	FAULT0	R/W	0x0	Fault0 Sense. 0x0 = An error is indicated if the Fault0 signal is high 0x1 = An error is indicated if the Fault0 signal is low

21.5.29 PWMnFLTSTAT0 Register [reset = 0x0]

PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804

PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884

PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904

PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984

Along with the PWMnFLTSTAT1 register, this register provides status regarding the fault condition inputs.

If the LATCH bit in the PWMnCTL register is clear, the contents of the PWMnFLTSTAT0 register are read-only (R) and provide the current state of the MnFAULTn inputs.

If the LATCH bit in the PWMnCTL register is set, the contents of the PWMnFLTSTAT0 register are read / write 1 to clear (RW1C) and provide a latched version of the MnFAULTn inputs. In this mode, the register bits are cleared by writing a 1 to a set bit. The MnFAULTn inputs are recorded after their sense is adjusted in the generator.

The contents of this register can only be written if the fault source extensions are enabled (the FLTSRC bit in the PWMnCTL register is set).

NOTE: The fault status registers, PWMnFLTSTAT0 and PWMnFLTSTAT1, reflect the status of all fault sources, regardless of what fault sources are enabled for that particular generator.

PWMnFLTSTAT0 is shown in [Figure 21-35](#) and described in [Table 21-31](#).

Return to [Summary Table](#).

Figure 21-35. PWMnFLTSTAT0 Register

15	14	13	12	11	10	9	8
reserved_1							
R-0x0							
7	6	5	4	3	2	1	0
reserved_1				FAULT3	FAULT2	FAULT1	FAULT0
R-0x0				0x0	0x0	0x0	0x0

Table 21-31. PWMnFLTSTAT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0x0	
3	FAULT3		0x0	<p>Fault Input 3.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is R and represents the current state of the MnFAULT3 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT3 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> - If FAULT3 is set, the input transitioned to the active state previously. - If FAULT3 is clear, the input has not transitioned to the active state since the last time it was cleared. - The FAULT3 bit is cleared by writing it with the value 1.

Table 21-31. PWMnFLTSTAT0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	FAULT2		0x0	<p>Fault Input 2.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is R and represents the current state of the MnFAULT2 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT2 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> - If FAULT2 is set, the input transitioned to the active state previously. - If FAULT2 is clear, the input has not transitioned to the active state since the last time it was cleared. - The FAULT2 bit is cleared by writing it with the value 1.
1	FAULT1		0x0	<p>Fault Input 1.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is R and represents the current state of the MnFAULT1 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT1 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> - If FAULT1 is set, the input transitioned to the active state previously. - If FAULT1 is clear, the input has not transitioned to the active state since the last time it was cleared. - The FAULT1 bit is cleared by writing it with the value 1.
0	FAULT0		0x0	<p>Fault Input 0.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is R and represents the current state of the input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is RW1C and represents a sticky version of the input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> - If FAULT0 is set, the input transitioned to the active state previously. - If FAULT0 is clear, the input has not transitioned to the active state since the last time it was cleared. - The FAULT0 bit is cleared by writing it with the value 1.

21.5.30 PWMnFLTSTAT1 Register [reset = 0x0]

PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808

PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888

PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908

PWM3 Fault Status 1 (PWM3FLTSTAT1), offset 0x988

Along with the PWMnFLTSTAT0 register, this register provides status regarding the fault condition inputs.

If the LATCH bit in the PWMnCTL register is clear, the contents of the PWMnFLTSTAT1 register are read-only (R) and provide the current state of the digital comparator triggers.

If the LATCH bit in the PWMnCTL register is set, the contents of the PWMnFLTSTAT1 register are read / write 1 to clear (RW1C) and provide a latched version of the digital comparator triggers. In this mode, the register bits are cleared by writing a 1 to a set bit. The contents of this register can only be written if the fault source extensions are enabled (the FLTSRC bit in the PWMnCTL register is set).

NOTE: The fault status registers, PWMnFLTSTAT0 and PWMnFLTSTAT1, reflect the status of all fault sources, regardless of what fault sources are enabled for that particular generator.

PWMnFLTSTAT1 is shown in [Figure 21-36](#) and described in [Table 21-32](#).

Return to [Summary Table](#).

Figure 21-36. PWMnFLTSTAT1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

Table 21-32. PWMnFLTSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	DCMP7		0x0	<p>Digital Comparator 7 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 7 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP7 is set, the trigger transitioned to the active state previously. - If DCMP7 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP7 bit is cleared by writing it with the value 1.

Table 21-32. PWMnFLTSTAT1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	DCMP6		0x0	<p>Digital Comparator 6 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 6 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP6 is set, the trigger transitioned to the active state previously. - If DCMP6 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP6 bit is cleared by writing it with the value 1.
5	DCMP5		0x0	<p>Digital Comparator 5 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 5 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP5 is set, the trigger transitioned to the active state previously. - If DCMP5 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP5 bit is cleared by writing it with the value 1.
4	DCMP4		0x0	<p>Digital Comparator 4 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 4 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP4 is set, the trigger transitioned to the active state previously. - If DCMP4 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP4 bit is cleared by writing it with the value 1.
3	DCMP3		0x0	<p>Digital Comparator 3 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 3 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP3 is set, the trigger transitioned to the active state previously. - If DCMP3 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP3 bit is cleared by writing it with the value 1.
2	DCMP2		0x0	<p>Digital Comparator 2 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 2 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP2 is set, the trigger transitioned to the active state previously. - If DCMP2 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP2 bit is cleared by writing it with the value 1.

Table 21-32. PWMnFLTSTAT1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	DCMP1		0x0	<p>Digital Comparator 1 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 1 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP1 is set, the trigger transitioned to the active state previously. - If DCMP1 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP1 bit is cleared by writing it with the value 1.
0	DCMP0		0x0	<p>Digital Comparator 0 Trigger.</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 0 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> - If DCMP0 is set, the trigger transitioned to the active state previously. - If DCMP0 is clear, the trigger has not transitioned to the active state since the last time it was cleared. - The DCMP0 bit is cleared by writing it with the value 1.

21.5.31 PWMPP Register (Offset = 0xFC0) [reset = 0x344]

PWM Peripheral Properties (PWMPP)

The PWMPP register provides information regarding the properties of the PWM module.

PWMPP is shown in [Figure 21-37](#) and described in [Table 21-33](#).

Return to [Summary Table](#).

Figure 21-37. PWMPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED					ONE	EFAULT	ESYNC
R-0x0					R-0x0	R-0x1	R-0x1
7	6	5	4	3	2	1	0
FCNT				GCNT			
R-0x4				R-0x4			

Table 21-33. PWMPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0x0	
10	ONE	R	0x0	One-Shot Mode. 0x0 = One-shot modes are not available. 0x1 = One-shot modes are available.
9	EFAULT	R	0x1	Extended Fault. 0x0 = Extended fault capabilities are not available. 0x1 = Extended fault capabilities are available.
8	ESYNC	R	0x1	Extended Synchronization. 0x0 = Extended synchronization is not available. 0x1 = Extended synchronization is available.
7-4	FCNT	R	0x4	Fault Inputs. 0x5 to 0xF = Reserved 0x0 = No fault inputs 0x1 = 1 fault input 0x2 = 2 fault input 0x3 = 3 fault input 0x4 = 4 fault input
3-0	GCNT	R	0x4	Generators. The number of PWM outputs is 2 times the number of PWM generators. 0x5 to 0xF = Reserved 0x0 = No generators 0x1 = 1 generator 0x2 = 2 generators 0x3 = 3 generators 0x4 = 4 generators

21.5.32 PWMCC Register (Offset = 0xFC8) [reset = 0x5]

PWM Clock Configuration (PWMCC), offset 0xFC8

The PWMCC register controls the clock source for the PWM module.

PWMCC is shown in [Figure 21-38](#) and described in [Table 21-34](#).

Return to [Summary Table](#).

Figure 21-38. PWMCC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							USEPWM
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
reserved_2					PWMDIV		
R-0x0					R/W-0x5		

Table 21-34. PWMCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	USEPWM	R/W	0x0	Use PWM Clock Divisor. 0x0 = The system clock is the source of PWM unit clock. 0x1 = The PWM clock divider is the source of PWM unit clock.
7-3	RESERVED	R	0x0	
2-0	PWMDIV	R/W	0x5	PWM Clock Divider. This field specifies the PWM clock frequency as a division of the system clock. 0x0 = /2 0x1 = /4 0x2 = /8 0x3 = /16 0x4 = /32 0x5 = /64 0x6 = Reserved 0x7 = Reserved

1-Wire Master Module

This chapter describes the 1-Wire Master module.

Topic	Page
22.1 Introduction	1501
22.2 Block Diagram	1501
22.3 Functional Description	1502
22.4 Initialization and Configuration	1508
22.5 One-Wire Master Registers	1510

22.1 Introduction

The 1-Wire Master module is a bidirectional serial communication interface that implements the protocol functions of the Dallas Semiconductor 1-Wire protocol and provides both power and data over a single wire. The 1-Wire Master module can interface with a multiple variety of slaves such as thermometers, mixed-signal devices, memory, and authentication devices.

Features of the 1-Wire Master module include:

- Support for standard and overdrive speeds, including a late-sample mechanism
- Allows transfers of send, receive and bidirectional bits
- Data size transfers of 1, 2, 3, or 4 bytes with subbyte search and enumeration support
- Interrupt capability for transaction pacing and line error
- Optional 2-wire support for isolated lines and high voltage use
- Efficient transfers using the μ DMA

22.2 Block Diagram

Figure 22-1 details the components of the 1-Wire Master module. The clock source for the 1-Wire Master module is the 16-MHz precision internal oscillator (PIOSC).

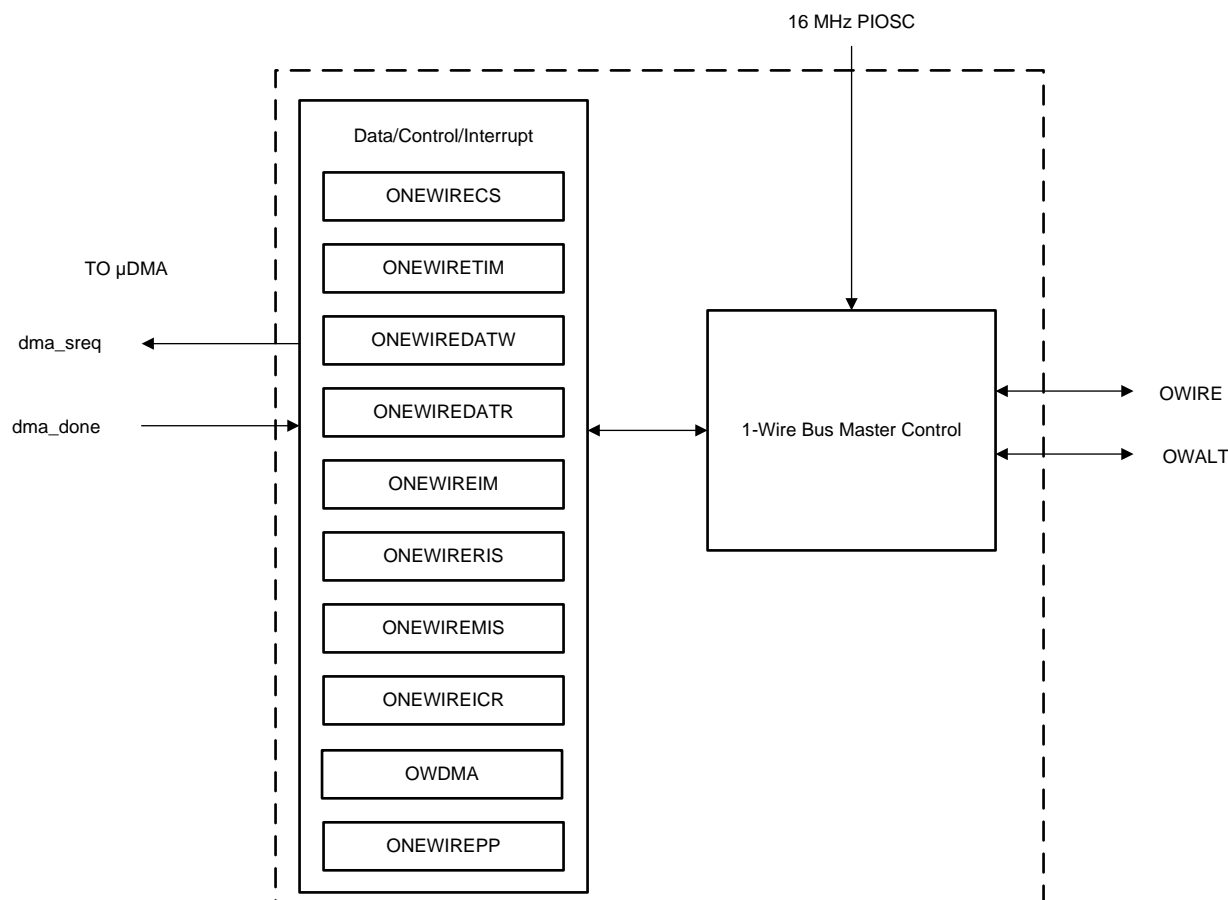


Figure 22-1. 1-Wire Block Diagram

22.3 Functional Description

1-Wire is a simple single-wire communication interface comprising a wire protocol, transport protocol, and base session protocols (as well as some basic commands). The wire protocol generates 0 s and 1 s by varying the time the line is held low (it is pulled up by a resistor). The MSP43E microcontroller is the master and controls the line at all times. A pull-up keeps the line parked high and the microcontroller driver is assumed to be normal open-drain. No special GPIO configuration is needed here since the open-drain support is taken care of outside of the microcontroller. See [Table 17-2](#) for pin configuration information. Data can be both read and written on the same 1-Wire pin. The 1-Wire module supports the most basic aspects of the protocol, including wire protocol, byte transport control and line reset. Software is expected to handle the session protocol, including selection of a slave (when more than one is on the same line) and higher level commands.

NOTE: All time values by the 1-Wire Master are stated in μs , but the actual time on the wire may be less by 62.5 ns.

22.3.1 1-Wire Protocol

The 1-Wire protocol signals 1 s and 0 s by holding the line low for varying lengths of time as described below. For details on the operation of this device, see [Section 22.3.3](#) Before a command is sent, a reset is issued on the line. This is a two-part operation:

- The 1-Wire Master module drives and holds line low for $> 480 \mu\text{s}$.
- The controller waits for an answer-to-reset from one or more slaves. A slave signals a reset by pulling the line low for $60 \mu\text{s}$ to $240 \mu\text{s}$. If the line is not sampled low, there is no slave on the bus and the NOATR bit is set in the 1-Wire Control and Status (ONEWIRECS) register. An interrupt mask can also be set to trigger an interrupt on this condition. If the line is sampled low, it indicates there are one or more slaves on the 1-Wire bus. The Master samples the line some period after releasing the line, taking into consideration the time needed for the slave to respond. The time from reset release to first sample is programmed through the ATRSAM bit in the 1-Wire Timing Override (ONEWIRETIM) register. For example, the Master could sample $10 \mu\text{s}$ after releasing the reset for $240 \mu\text{s}$. Caution should be exercised to make sure the line has been pulled high (by a pull-up) before the Master samples to avert a bogus answer-to-reset. Because the slave may hold the line low for a longer duration than the sample time, the Master must wait for the line to go high before starting a new command. This reset protocol is used to ensure that all slaves are in a known state.

[Figure 22-2](#) shows the details of the 1-wire reset.

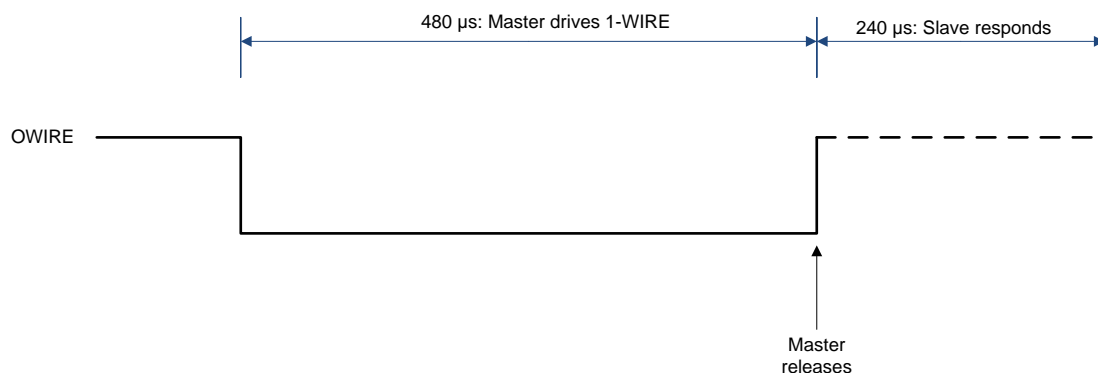


Figure 22-2. 1-Wire Reset Protocol

If the Master is transmitting data to the slave, the signalling is as follows:

- A 1 is signaled by the master driving and holding the line low for $< 15 \mu\text{s}$. Generally, about $6 \mu\text{s}$ is used for normal mode. The slave samples and measures the signal from falling edge and checks the line $15 \mu\text{s}$ later (or more). If line has reverted to high, the slave registers a 1 value.
- A 0 is signaled by the Master driving and holding the line low for $60 \mu\text{s}$ or more. Although the slave reads just past $15 \mu\text{s}$, the normal mode requires the line to be low for $60 \mu\text{s}$. If the line is still low after

15 μ s, a value of 0 is registered.

Figure 22-3 depicts a 1-Wire Master transmitting a 1 to a slave. Figure 22-4 shows a 1-wire Master transmitting a 0 to a slave.

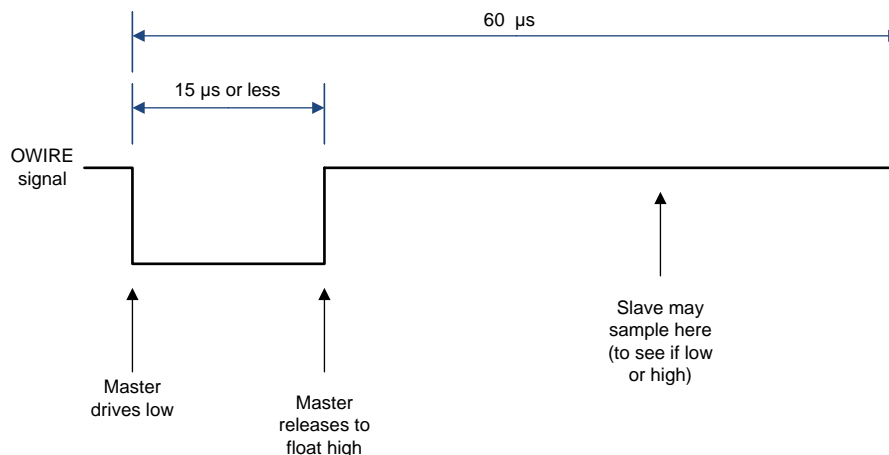


Figure 22-3. 1-Wire Master Transmitting a 1

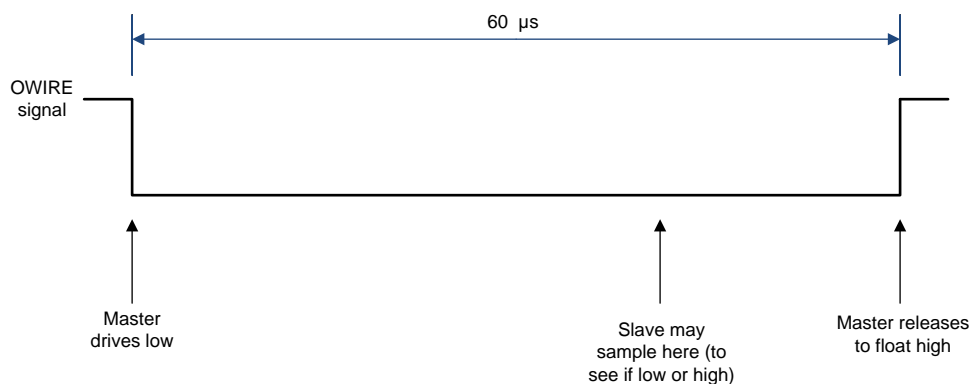


Figure 22-4. 1-Wire Master Transmitting a 0'

For a read from the slave, the Master drives and holds the line low for at least 1 μ s (but less than 15 μ s) and then releases. A typical hold time is 6 μ s; in other words, it uses a transmitted 1. The read is performed as:

- A 1 is signaled by the slave when "does nothing." Thus, the slave just allows the line to come back up when the master releases it. The master samples the line at 15 μ s after the low edge and the line is still high, signaling a 1.
- If the slave holds the line down for more than 5 μ s (and no more than 60 μ s), a 0 is signaled. The master samples the line at 15 μ s after the low edge, and because the line is low still, it is treated as a 0.

Figure 22-5 depicts a Master receiving a 1 value from a slave. Figure 22-6 shows the slave sending a 0 value to a Master.

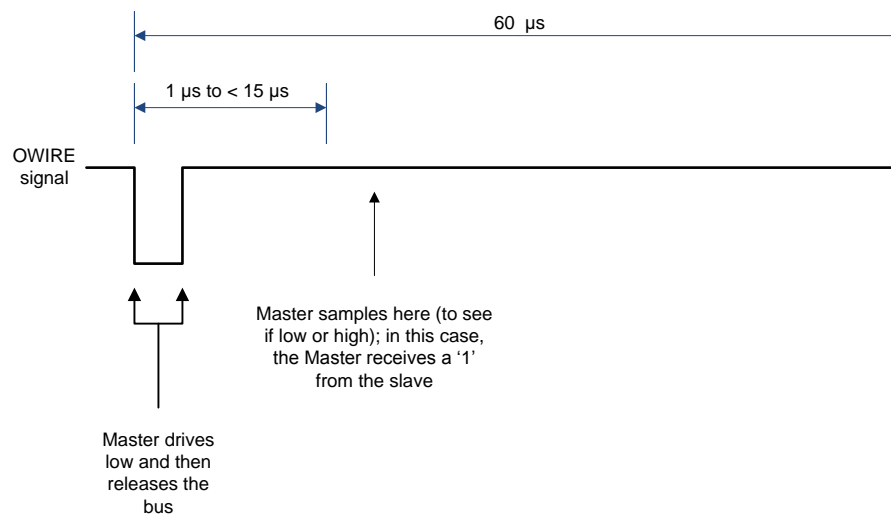


Figure 22-5. 1-Wire Master Receiving a 1 Signal from a Slave

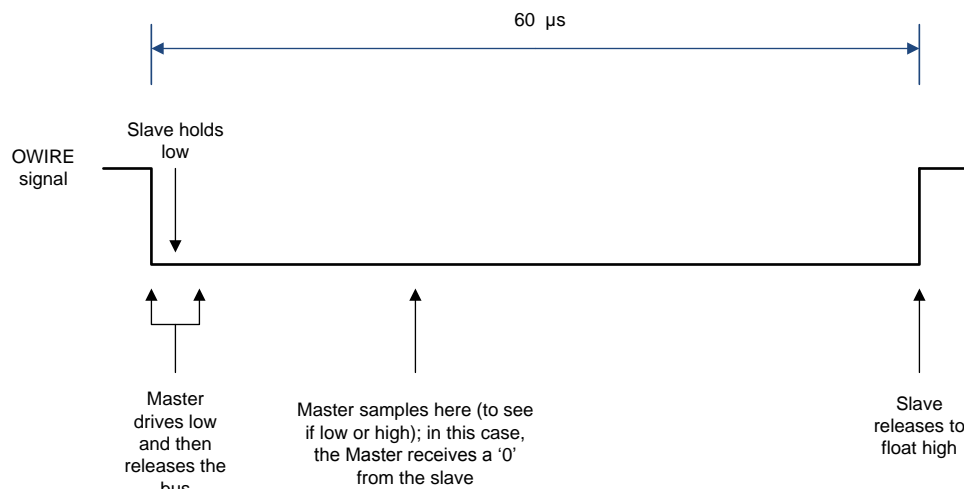


Figure 22-6. 1-Wire Master Receiving a 0 Signal from a Slave

22.3.2 Transport Protocol

The transport level protocol follows a simple model, in part based on treating most 1-wire devices as memory like (IDs, ROM, EEPROM, etc) even if some are sensors or actuators. All commands and data are in 8-bit (byte) quantities and the post-reset commands are pre-defined. The basic protocol is:

- Reset the line by driving and holding the OWIRE signal low for > 480 μ s.
- Wait for answer-to-reset from one or more slaves. Answer-to-reset (ATR) is defined as each slave pulling the line low for 60 μ s to 240 μ s. If the line does not go low, there is no slave on the bus. If it does, there is one or more.
- At this point the Master either figures out the ID of the slaves on the bus using the enumeration scheme, or selects a slave to issue a command. The select command to use is based on whether there is one slave or more than one. If there is only one slave device, the Master skips the selection process. If more than one are present, the Master selects a slave by sending out the ID for the selected slave, indicating to the remaining slaves that any further commands should be ignored until after the next reset.
- The Master sends a command followed by a sending or receiving of data. While most commands are implemented using bytes read or written, some commands use bytes where some bits are written and some read. Since a read looks like a write 1, if the line is held and driven low for less than 15 μ s and

the slave does not know it is supposed to reply, the line remains untouched and is read as a 1 (and written as a 1).

- Usually at the end of one or more commands to that slave, a reset is issued again.

The 1-Wire module supports reset and answer-to-reset on request (although answer-to-reset can be suppressed). It also supports byte operations. The software chooses which read, write, or mixed operation to use. Partial byte (bit or multi-bit) is also supported.

22.3.3 Overdrive

The 1-Wire Module allows for an overdrive speed to be selected for shorter line lengths. Overdrive is about ten times faster because it uses smaller time periods. It is agreed by both master and slave before enabling. Overdrive is selected using the ODRV bit of the 1-Wire Control and Status (ONEWIRECS) register.

In overdrive mode, the time periods shrink. [Table 22-1](#) shows the active time for 1-Wire operations in overdrive mode. For example, the Master typically drives and holds the line low for 15 μ s for a write 1, but in overdrive mode this is 1 μ s.

Table 22-1. Active Time Periods in Overdrive Mode

Operation	Time	Notes
Write 1	1 μ s low	Less than 2 μ s low
Write 0	7.5 μ s low	Up to 10 μ s low
Read 1	1 μ s low	Master samples at 2 μ s
Read 0	9 μ s low	Slave holds at least 7 μ s
Reset	70 μ s low	Note 2 μ s delay before reset for safety
Answer to Reset	4 μ s to 48 μ s	Master samples at 8.5 μ s

22.3.4 Timing Override

The 1-Wire module can manually override timing for any main action. Each non-zero value in the 1-Wire Timing Override (ONEWIRETIM) register replaces the default normal or overdrive time rule.

The time factors that can be controlled:

Table 22-2. Bit Field Definitions for 1-Wire Timing and Override (ONEWIRETIM) Register

Name	Meaning	Range (if used)	Scale	Default (normal)	Default (overdrive)
W1TIM	Amount of time to drive and hold line low for a write 1	1 to 15 μ s	1 μ s units	6 μ s	1 μ s
W0TIM	Amount of time to drive and hold line low for write 0 ⁽¹⁾	4 to 124 μ s	4 μ s units	60 μ s	7.5 μ s
W0REST	Amount of time to hold line high after a write 0 (rest)	1 to 15 μ s	1 μ s units	10 μ s	2.5 μ s
W1SAM	Amount of time after 1-Wire has released the line to sample slave holding line	1 to 15 μ s	1 μ s units	10 μ s	1 μ s
RSTTIM	Amount of time to drive and hold line low for a reset. Allow for an additional 2- μ s prereset period	32 to 2016 μ s	32 μ s units	480 μ s	70 μ s
ATRTIM	Period of answer-to-reset or rest. The ATRSAM bit value must be considered when programming this field. ATRTIM must be greater than ATRSAM to avoid a STUCK fault generation	16 to 496 μ s	16 μ s units	240 μ s	48 μ s
ATRSAM	Period in which first sample occurs in answer-to-reset (after line goes high). If the SKATR bit is set in the ONEWIRECS register, this value indicates the rest period after reset	8 to 120 μ s	8 μ s units	10 μ s	2 μ s

⁽¹⁾ If W0TIM has been programmed to a non-default value, W0REST must be programmed for correct functionality.

22.3.5 Command Protocol

The standard 1-Wire protocol has commands to be used after a reset. These commands normally include slave selection and detection. However, many 1-Wire buses only have one slave, and in that case, those mechanisms can be skipped. If slave management is needed, a small software stack can be used to handle these operations. Other protocols can also be easily supported, whether byte-oriented (as normal) or bit-oriented. To support an alternative protocol, it is usually necessary to override some or all of the timing and may be necessary to disable the answer-to-reset mechanism using the SKATR bit in the ONEWIRECS register. The module is designed to support the most common uses of 1-Wire, including a mechanism to allow a reset followed by an operation of 1, 2, 3, or 4 bytes through the SZ bit in the ONEWIRECS register. Additionally, the OP bit supports a write/read model to allow a mix of write and read in one transaction. Although nominally mixed mode allows for a mix of write and read bytes, it can be used down to the bit level.

The following represents some typical operation flows:

1. To only send a reset, the RST bit of the ONEWIRECS register is set, performing a reset and triggering an interrupt if enabled. If there is no answer-to-reset, the NOATR status bit is set and the corresponding interrupt bit can be set as well.
2. The SZ and OP fields of the ONEWIRECS register are used to perform a write, a read, or a combination. For small transactions, the SZ encoding can be programmed for processing of up to 4 bytes in one transaction. For larger transactions, the OP field can be used multiple times. The OPC raw interrupt bit is used to notify when a transaction is complete.
3. To perform a reset and then a transaction, both the RST and OP and SZ fields of the ONEWIRECS register are configured. The reset runs to completion and then the transaction starts. The interrupt is used to notify when both are done.

To setup a read, write or mixed transaction, the following configuration must occur:

- For writes, the 1-Wire Data Write (ONEWIREDATW) register must first be written with one, two, three or four bytes to send (as a word). Next the ONEWIRECS register is configured as described above.
- For reads, the ONEWIRECS register is written as described above. Then, when the transaction is complete, the 1-Wire Data Read (ONEWIREDATR) register is read to collect the one, two, three, or four bytes that were received (as a word).
- For a R/W combination, the ONEWIREDATW register is written with one, two, three or four bytes to write or read. Then, the ONEWIRECS register is configured as described above. As the data is transferred from the ONEWIREDATW register to the 1-Wire line, any 1 that is transmitted creates a read that is registered in the ONEWIREDATR register. When the entire transaction is complete, the ONEWIREDATR register can be read to collect the bytes that were received by the 1-Wire Master. A ONEWIREDATR register bit contains a 0 for any bit that was written as a 0, a 1 for any bit that was written as a 1 or any bit not modified by the slave (may have been intentional 1 or ignored) and a 0 for any bit written as a 1 but read back as a '0.'

22.3.6 Search (Enumeration) and Sub-Byte

Although 1-Wire is a byte protocol, there are times when less than a byte must to be written or read. By programming the BSIZE field of the ONEWIRECS register, the number of bits to write and or read in the last byte can be configured. The BSIZE field can be used for a 1-Wire search (enumeration) to allow detect and select processing. Enumeration begins by executing the standard SEARCH (0xF0) command (and if available, the ALARM SEARCH (0xEC)) and then reading two bits for the ID, which is the first bit and its complement. Subsequently, a bit is emitted to select devices with that first matching bit followed by another read of two bits. This process is repeated for up to all 64 bits. To start over, a reset is used. In this way, a tree of IDs can be broken down until one unique ID is left. This can be repeated for each device. The BSIZE field can also be used for non-standard devices which do not fully follow protocol.

22.3.7 Interrupts

The interrupt model for the 1-Wire module allows interrupt service routines to drive continuations of operations, whether reading the results or processing more bytes. Most command sequences are small, so the module supports transactions of one, two, three, and four bytes at once. Because most commands start with a line reset, the operational model allows for a reset followed by the transaction, with an interrupt occurring when the transaction is complete. There is also support for enabling interrupts for reset complete (RST), no answer-to-reset (NOATR) and line-hold-low error detected (STUCK) in the 1-Wire Interrupt Mask (ONEWIREIM) register.

22.3.8 DMA

The μ DMA model is designed to support both normal μ DMA operation on write or read, as well as scatter-gather operation. The normal mode of operation allows for continued writes of byte, half-word, or word (three-byte size is not supported by μ DMA for repeated operation) as well as continued reads by byte, half-word, or word. Additionally, the first transaction may be started with a reset.

The 1-Wire DMA Control (ONEWIREDMA) register is used to configure the μ DMA. To use μ DMA with 1-Wire transmits:

1. Select 1-Wire to use μ DMA by programming the DMA Channel Map Select n (DMACHMAPn) register in the μ DMA. See for more information.
2. Configure the μ DMA to transfer bytes, half-words, or words from memory to the 1-Wire ONEWIREDATW register or from the ONEWIREDATR register through the DMA Channel Control Word (DMACHCTL) register (see).
3. Set the SZ field of the ONEWIRECS register to byte, half-word, or word (0, 1, or 3). This is the only field programmed in the ONEWIRECS register.
4. Enable the DMA completion interrupt in the ONEWIREIM register. It is also recommended that the ERR and NOATR interrupts should be enabled when using the μ DMA to ensure that the application is notified if the reset or transaction failed.
5. Write the DMAOP field of the ONEWIREDMA register with the encoding 0x2 to enable the 1-Wire module assert a μ DMA request when ONEWIREDATW is empty. The RST bit in the ONEWIREDMA register can be set for reset to be asserted first before requesting a μ DMA access.
6. When the μ DMA transfer is complete, the DMA bit in the 1-Wire Raw Interrupt Status (ONEWIRERIS) register is set and the ONEWIREDMA register is cleared.

A 1-Wire μ DMA receive configuration is as follows:

1. Select 1-Wire to use μ DMA by programming the DMA Channel Map Select n (DMACHMAPn) register in the μ DMA. See for more information.
2. Configure the μ DMA to transfer bytes, half-words, or words from the 1-Wire ONEWIREDATR register through the DMA Channel Control Word (DMACHCTL) register (see).
3. Enable the DMA completion interrupt in the ONEWIREIM register. It is also recommended that the ERR and NOATR interrupts should be enabled when using the μ DMA to ensure that the application is notified if the reset or transaction failed.
4. Write the DMAOP field of the ONEWIREDMA register with the encoding 0x1 or 0x3 to enable a read or read multiple by the μ DMA. The RST bit in the ONEWIREDMA register can be set for reset to be asserted first before requesting a μ DMA access.
5. Write the ONEWIREDATW register with 0xFFFF.FFFF to prime the read operations. If the RST bit is not set in the ONEWIREDMA register, the transaction begins. If RST was set, the read starts when reset is complete. At the end of the read operation, the μ DMA receives a request to transfer the data from the ONEWIREDATR register. If the DMAOP bit was configured for a multiple read (0x3), then the next read transaction will automatically start. If not, then no further μ DMA requests occur unless ONEWIREDATR is written to.
6. When a μ DMA receive is complete, the DMA bit in the 1-Wire Raw Interrupt Status (ONEWIRERIS) register is set and the ONEWIREDMA register is cleared.

The normal flow for writing and reading in one operation is:

1. Configure the DMAOP bit in the ONEWIREDMA register for a read, and optionally set the RST bit. If RST is selected, the reset is performed first, else the module waits for the next step.

2. Write the ONEWIREDATW register with one, two, or four bytes containing the mix of writes and reads. For example, to write 0x46, 0x20, and then read 2 bytes, set size to word (SZ = 3) and write ONEWIREDATW with 0xFFFF2046 to send the 0x46 and 0x20 and then read the last two bytes. On completion, the ONEWIREDATR contains the read values in the upper two bytes. If RST is not set in the ONEWIREDMA register, the transaction starts. If RST is set, the transaction starts when reset completes. At the end of the operation, the μ DMA is requested to transfer the data from the ONEWIREDATR register (read).
3. When the μ DMA is done, the DMA bit in the ONEWIRERIS register is set, allowing interrupt on completion by μ DMA, and the ONEWIREDMA register is cleared.

It is recommended to enable trigger interrupts for ERR and NOATR when using μ DMA so application is notified if reset or the transaction failed.

The natural model for using scatter-gather is:

1. Enable peripheral-scatter-gather in the μ DMA module by programming the DMA Channel Control Word (DMACHCTL) register.
2. Set the DMAOP to read (not read multiple) and enable scatter gather in the 1-Wire module by setting the SG bit to 0x1 in the ONEWIREDMA register. This allows use of ONEWDATW register to push through transactions, with μ DMA request being triggered off completion, except for the first transaction.
3. Scatter gather can then use a combination of write and read as needed.

22.3.9 1-Wire Timing

The 1-Wire module wire control operates using the Precision Internal Oscillator (PIOSC) to provide a reliable frequency with no need for baud dividers. The register interface operates using the system clock, which may be a different clock source. However, there is no visible impact from this separate clock model. Normal rules apply to use of the registers to ensure correct operation. In particular:

- The control register is used live (not buffered) by the communication engine. Once an operation is started, it must be allowed to finish before changes are made, such as starting a new operation or changing speed. The status bits in the ONEWIRECS register notify the application if the transaction is busy or if a line-hold-low error is detected. When the last write, read or read/write has occurred, the OPC bit is set in the ONEWIRERIS register.
- Data read by the 1-Wire Master is not valid or meaningful until the 1-Wire module signals that the data is ready (through interrupt, DMA, and status).
- Data written should be written before the operation is started and then should not be re-written until the 1-Wire module signals the data has been written (through interrupt, DMA, and status).
- The 1-Wire Timing Override (ONEWIRETIM) register should not be changed while a transaction is taking place.

The ONEWIREIM, ONEWIRERIS and ONEWIREMIS registers may be written and read at any time with no risk.

22.4 Initialization and Configuration

The whole block may be enabled or disabled using the system control registers. The following example shows how to enable, force a reset and send a command.

1. Enable the 1-Wire Module by writing a value of 0x0000.0001 to the 1-Wire Run Mode Clock Gating Control (RCGCOWIRE) register in the System Control Module, see [Section 4.2.104](#).
2. Enable the clock to the appropriate GPIO port that is used for the 1-Wire signals using the General-Purpose Input/Output Run Mode Clock gating Control (RCGCGPIO) in the System Control Module; see [Section 4.2.87](#). To find out which GPIO port to enable, see and the device-specific data sheet.
3. In the GPIO module, enable the appropriate pin for its alternate function using the GPIO Alternate Function Select (GPIOAFSEL) register (see [Section 17.5.10](#)).
4. Configure the PMCN fields in the GPIOPCTL register to assign the 1-Wire signals to the appropriate pins. See [Section 17.5.22](#) and the device-specific data sheet.
5. Optionally enable interrupts to be notified when the transactions completes.
6. Write to the ONEWIREDATW register with the data to write (1, 2, 3, or 4 bytes).

7. Write to the ONEWIRECS register to set the RST bit and configure the OP (operation) and SZ fields. When configuring all three bit fields, the reset is performed first, followed by the operation (unless an error occurs) and then any interrupt triggers.
8. If normal interrupts are generated (for example, OPC and RST), clear these interrupts and read the ONEWIREDATR register if a read or read/write transaction occurred.
9. If the 1-Wire master has more data to write and/or read for the same command, repeat the previous two steps but without setting the RST bit. If a new command is required, repeat the previous two steps including the reset enable.

22.5 One-Wire Master Registers

Table 22-3 lists the memory-mapped registers for the OWIRE. All register offset addresses not listed in Table 22-3 should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the base address of the 1-Wire Master module: 0x400B6000.

Table 22-3. One-Wire Master Registers

Offset	Acronym	Register Name	Section
0x0	ONEWIRECS	1-Wire Control and Status	Section 22.5.1
0x4	ONEWIRETIM	1-Wire Timing Override	Section 22.5.2
0x8	ONEWIREDATW	1-Wire Data Write	Section 22.5.3
0xC	ONEWIREDATR	1-Wire Data Read	Section 22.5.4
0x100	ONEWIREIM	1-Wire Interrupt Mask	Section 22.5.5
0x104	ONEWIRERIS	1-Wire Raw Interrupt Status	Section 22.5.6
0x108	ONEWIREMIS	1-Wire Masked Interrupt Status	Section 22.5.7
0x10C	ONEWIREICR	1-Wire Interrupt Clear	Section 22.5.8
0x120	ONEWIREDMA	1-Wire μ DMA Control	Section 22.5.9
0xFC0	ONEWIREPP	1-Wire Peripheral Properties	Section 22.5.10

Complex bit access types are encoded to fit into small table cells. Table 22-4 shows the codes that are used for access types in this section.

Table 22-4. OWIRE Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

22.5.1 ONEWIRECS Register (Offset = 0x0) [reset = 0x0]

1-Wire Control and Status (ONEWIRECS), offset 0x000

The 1-Wire Control and Status (ONEWIRECS) register contains the control and status bits for the 1-Wire bus, including active state and passive detection. The OP field is written to start an operation. If executing a write or a write/read, the Bn field of the 1-Wire Data Write (ONEWIREDATW) register must be written first before programming the OP field.

ONEWIRECS is shown in [Figure 22-7](#) and described in [Table 22-5](#).

Return to [Summary Table](#).

Figure 22-7. ONEWIRECS Register

31	30	29	28	27	26	25	24
USEALT	ALTP	RESERVED					
R/W-0x0	R/W-0x0	R-0x0					
23	22	21	20	19	18	17	16
RESERVED					BSIZE		
R-0x0					R/W-0x0		
15	14	13	12	11	10	9	8
RESERVED					STUCK	NOATR	BUSY
R-0x0					R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
SKATR	LSAM	ODRV	SZ		OP		RST
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0		R/W-0x0

Table 22-5. ONEWIRECS Register Field Descriptions

Bit	Field	Type	Reset	Description
31	USEALT	R/W	0x0	Two Wire Enable. 0x0 = Function in standard 1-Wire mode, using only the OWIRE signal for both input and output. 0x1 = Use the OWIRE signal for the input and the OWALT signal for the output.
30	ALTP	R/W	0x0	Alternate Polarity Enable. 0x0 = Output pin is driven low to drive 1-wire line low. 0x1 = Output pin is driven high to drive 1-Wire line low (that is, the output pin is connected to an NFET).
29-19	RESERVED	R	0x0	
18-16	BSIZE	R/W	0x0	Last Byte Size. This field indicates the bit-size of the last byte. These bits are sent and received least significant bit first. 0x0 = 8 bits (1 byte) 0x1 = 1 bit 0x2 = 2 bits 0x3 = 3 bits 0x4 = 4 bits 0x5 = 5 bits 0x6 = 6 bits 0x7 = 7 bits
15-11	RESERVED	R	0x0	
10	STUCK	R	0x0	STUCK Status. 0x0 = STUCK status is not detected. 0x1 = Indicates line is being held low (other than in normal operation). The STUCK bit in the ONEWIRERIS is asserted when a line-hold-low error is detected.

Table 22-5. ONEWIRECS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	NOATR	R	0x0	Answer-to-Reset Status. This bit is disabled if the SKATR bit is set. 0x0 = Answer-to-reset behavior was as expected. 0x1 = No answer-to-reset was detected after last reset (RST =1), which indicates no slave may be present on the bus.
8	BUSY	R	0x0	Busy Status. 0x0 = No activity on the bus. 0x1 = Activity is taking place on the bus as a result of a reset, read, write or read/write transaction. Because this bit is on the PIOSC clock domain, it does not set immediately when a read, write, read/write or reset transaction begins. The OPC and RST bits in the ONEWIRERIS register should be used to verify when these operations have completed.
7	SKATR	R/W	0x0	Skip Answer-to-Reset Enable. 0x0 = No effect. 0x1 = Transaction goes from reset to first byte transfer without an answer-to-reset (presence detect) after programmed rest period has passed.
6	LSAM	R/W	0x0	Late Sample Enable. This bit is used for long distance lines or when the slave cannot pull low quickly. 0x0 = Late sample disabled. 0x1 = 1-Wire module samples late in the read. The sample point moves to 50 μ s for normal and 7 μ s for overdrive (versus 16 μ s and 2 μ s in normal operation).
5	ODRV	R/W	0x0	Overdrive Enable. 0x0 = Overdrive mode disabled. 0x1 = Overdrive mode is enabled.
4-3	SZ	R/W	0x0	Data Operation Size. This field is used to program the size in bytes of data operations transferred to and from the ONEWIREDATn registers per request. 0x0 = 1 byte 0x1 = 2 bytes 0x2 = 3 bytes 0x3 = 4 bytes
2-1	OP	R/W	0x0	Operation Request. If written with a non-zero value, this field requests data operations on the 1-Wire bus for number of bytes configured by the SZ field. This field does not clear until the operation completes. The operation starts immediately. If using write or write/read, the ONEWIREDATW register must be written first. μ DMA operations are handled by the ONEWIREDMA register. 0x0 = No operation 0x1 = Read 0x2 = Write 0x3 = Write/Read
0	RST	R/W	0x0	Reset Request. If RST = 1, a reset operation begins. This bit clears when the reset operation is complete and the NOATR bit can be read to verify that a slave responded. If this bit is set at the same time as the OP bit is set, the reset runs first and the operation starts when reset has completed. If this bit is written when an operation is already in progress, it cancels the operation and starts the reset. 0x0 = No effect. 0x1 = Reset operation request.

22.5.2 ONEWIRETIM Register (Offset = 0x4) [reset = 0x0]

1-Wire Timing Override (ONEWIRETIM), offset 0x004

The 1-Wire Timing Override (ONEWIRETIM) register allows for overriding of the timing rules for the 1-Wire interface. If a field is 0, the default is used (normal or overdrive).

ONEWIRETIM is shown in [Figure 22-8](#) and described in [Table 22-6](#).

Return to [Summary Table](#).

Figure 22-8. ONEWIRETIM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W1TIM				W0TIM				W0REST				W1SAM			
R/W-0x0				R/W-0x0				R/W-0x0				R/W-0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W1SAM	ATRSAM				ATRTIM				RSTTIM						
R/W-0x0	R/W-0x0				R/W-0x0				R/W-0x0						

Table 22-6. ONEWIRETIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	W1TIM	R/W	0x0	Value 1 Timing. This field indicates the time low for a 1 in μ s units.
27-23	W0TIM	R/W	0x0	Value 0 Timing. Indicates time low for a 0 in 4 μ s units. If W0TIM has been programmed to a non-default value, W0REST must be programmed for correct functionality.
22-19	W0REST	R/W	0x0	Rest Time. This field indicates time to rest after a 0 in 1 μ s units. If W0TIM has been programmed to a non-default value, W0REST must be programmed for correct functionality.
18-15	W1SAM	R/W	0x0	Sample Time. Time to sample for a 1 vs 0 in μ s after the release of the line by 1-Wire. The LSAM bit is ignored if this bit is set.
14-11	ATRSAM	R/W	0x0	Answer-to-Reset Sample. Time to first sample for answer-to-reset in 8 μ s units. Sampling continues until time indicated by ATRTIM bit. When programming ATRTIM and ATRSAM, the value of ATRTIM must be greater than ATRSAM to prevent the line from being flagged as STUCK.
10-6	ATRTIM	R/W	0x0	Answer-to-Reset/Rest Period. This field indicates the answer-to-reset period in 16 μ s units. If the SKATR bit is set in the ONEWIRECS register, this value indicates the rest period after reset. When programming ATRTIM and ATRSAM, the value of ATRTIM must be greater than ATRSAM to prevent the line from being flagged as STUCK.
5-0	RSTTIM	R/W	0x0	Reset Low Time. Indicates the time reset is low in 32 μ s units.

22.5.3 ONEWIREDATW Register (Offset = 0x8) [reset = 0x0]

1-Wire Data Write (ONEWIREDATW), offset 0x008

The 1-Wire Data Write (ONEWIREDATW) and 1-Wire Data Read (ONEWIREDATR) registers are used to transmit or receive data. Data is processed LSB first, which means the lowest (and possibly only) byte is transferred first. Note that for normal processor use of writing and write/read, the ONEWIREDATW register must be written before the OP field of the ONEWIRECS register is programmed. If reading (or write/read), the ONEWIREDATR register is read after completion (when OP returns to value 0). When doing write/read (OP = 0x3), either register can be written. Because bits written as a 1 in the ONEWIREDATR register also act as a read, writing 0xFF is the same as a read of two bytes. Writing 0xF0 allows writing 0 for the lower nibble and reading the upper nibble. If the slave does not know it is a read, each written 1 reads back as 1.

ONEWIREDATW is shown in [Figure 22-9](#) and described in [Table 22-7](#).

Return to [Summary Table](#).

Figure 22-9. ONEWIREDATW Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3								B2								B1								B0							
R/W-0x0								R/W-0x0								R/W-0x0								R/W-0x0							

Table 22-7. ONEWIREDATW Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	B3	R/W	0x0	Upper Data Byte. This data byte is used when SZ field = 3.
23-16	B2	R/W	0x0	Upper Middle Data Byte. Contains upper middle data byte and is used when SZ field = 2 or 3.
15-8	B1	R/W	0x0	Lower Middle Data Byte. Contains lower middle data byte and is used when SZ field = 1, 2, or 3.
7-0	B0	R/W	0x0	Lowest Data Byte. Contains byte data being read or written (depending on the operation). This byte is always used.

22.5.4 ONEWIREDATR Register (Offset = 0xC) [reset = 0x0]

1-Wire Data Read (ONEWIREDATR), offset 0x00C

The 1-Wire Data Write (ONEWIREDATW) and 1-Wire Data Read (ONEWIREDATR) registers are used to transmit or receive data. Data is processed LSB first, which means the lowest (and possibly only) byte is transferred first. Note that for normal processor use of writing and write/read, the ONEWIREDATW register must be written before the OP field of the ONEWIRECS register is programmed. If reading (or write/read), the ONEWIREDATR register is read after completion (when OP returns to value 0). When doing write/read (OP = 0x3), either register can be written. Because bits written as a 1 in the ONEWIREDATR register also act as a read, writing 0xFF is the same as a read of two bytes. Writing 0xF0 allows writing 0 for the lower nibble and reading the upper nibble. If the slave does not know it is a read, each written 1 reads back as 1.

ONEWIREDATR is shown in [Figure 22-10](#) and described in [Table 22-8](#).

Return to [Summary Table](#).

Figure 22-10. ONEWIREDATR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3								B2								B1								B0							
R/W-0x0								R/W-0x0								R/W-0x0								R/W-0x0							

Table 22-8. ONEWIREDATR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	B3	R/W	0x0	Upper Data Byte. This data byte is used when SZ field = 3.
23-16	B2	R/W	0x0	Upper Middle Data Byte. Contains upper middle data byte and is used when SZ field = 2 or 3.
15-8	B1	R/W	0x0	Lower Middle Data Byte. Contains lower middle data byte and is used when SZ field = 1, 2, or 3.
7-0	B0	R/W	0x0	Lowest Data Byte. Contains byte data being read or written (depending on the operation). This byte is always used.

22.5.5 ONEWIREIM Register (Offset = 0x100) [reset = 0x0]

1-Wire Interrupt Mask (ONEWIREIM), offset 0x100

The 1-Wire Interrupt Mask (ONEWIREIM) register enables the interrupt triggers for the 1-Wire Module.

ONEWIREIM is shown in [Figure 22-11](#) and described in [Table 22-9](#).

Return to [Summary Table](#).

Figure 22-11. ONEWIREIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMA	STUCK	NOATR	OPC	RST
R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 22-9. ONEWIREIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMA	R/W	0x0	DMA Done Interrupt Mask. 0x0 = The DMA Done interrupt is suppressed and not sent to the interrupt controller. 0x1 = The DMA done interrupt is sent to the interrupt controller when the DMA bit in the ONEWIRERIS register is set.
3	STUCK	R/W	0x0	Stuck Status Interrupt Mask. When unmasked, this interrupt indicates a line-hold-low error is detected. 0x0 = The Stuck interrupt is suppressed and not sent to the interrupt controller. 0x1 = The Stuck status interrupt is sent to the interrupt controller when the STUCK bit in the ONEWIRERIS register is set.
2	NOATR	R/W	0x0	No Answer-to-Reset Interrupt Mask. 0x0 = The No Answer-to-Reset interrupt is suppressed and not sent to the interrupt controller. 0x1 = The No Answer-to-Reset interrupt is sent to the interrupt controller when the NOATR bit in the ONEWIRERIS register is set.
1	OPC	R/W	0x0	Operation Complete Interrupt Mask. If this bit is set to a 1, an interrupt is sent when a write, read, or write/read completes. If a read or read/write transfer has occurred then the read data is ready to be accessed when this bit is set in the ONEWIRERIS register. 0x0 = The operation complete interrupt is suppressed and not sent to the interrupt controller. 0x1 = The operation complete interrupt is sent to the interrupt controller when the OPC bit in the ONEWIRERIS register is set.
0	RST	R/W	0x0	Reset Interrupt Mask. 0x0 = The reset interrupt is suppressed and not sent to the interrupt controller. 0x1 = The reset interrupt is sent to the interrupt controller when the RST bit in the ONEWIRERIS register is set.

22.5.6 ONEWIRERIS Register (Offset = 0x104) [reset = 0x0]

1-Wire Raw Interrupt Status (ONEWIRERIS), offset 0x104

The 1-Wire Raw Interrupt Status (ONEWIRERIS) register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set.

ONEWIRERIS is shown in [Figure 22-12](#) and described in [Table 22-10](#).

Return to [Summary Table](#).

Figure 22-12. ONEWIRERIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMA	STUCK	NOATR	OPC	RST
R-0x0			R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 22-10. ONEWIRERIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMA	R	0x0	DMA Done Raw Interrupt Status. 0x0 = No interrupt 0x1 = DMA transfer complete and an interrupt is pending.
3	STUCK	R	0x0	Stuck Status Raw Interrupt Status. When unmasked, this interrupt indicates a line-hold-low error is detected. 0x0 = No interrupt 0x1 = Stuck status (line-hold-error) is detected and an interrupt is pending.
2	NOATR	R	0x0	No Answer-to-Reset Raw Interrupt Status. 0x0 = No interrupt 0x1 = A No Answer-to-Reset is detected from the last reset and an interrupt is pending.
1	OPC	R	0x0	Operation Complete Raw Interrupt Status. This bit indicates when a read, write or read/write operation has completed. If a read or read/write transfer has occurred then the read data is ready to be accessed when this bit is set. 0x0 = No interrupt. 0x1 = The last write, read, or write/read has completed and an interrupt is pending.
0	RST	R	0x0	Reset Raw Interrupt Status. 0x0 = No interrupt. 0x1 = The last reset completed and an interrupt is pending.

22.5.7 ONEWIREMIS Register (Offset = 0x108) [reset = 0x0]

1-Wire Masked Interrupt Status (ONEWIREMIS), offset 0x108

The 1-Wire Masked Interrupt Status (ONEWIREMIS) register indicates when an unmasked interrupt has occurred.

ONEWIREMIS is shown in [Figure 22-13](#) and described in [Table 22-11](#).

[Return to Summary Table.](#)

Figure 22-13. ONEWIREMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMA	STUCK	NOATR	OPC	RST
R-0x0			R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 22-11. ONEWIREMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMA	R	0x0	DMA Done Masked Interrupt Status. 0x0 = No interrupt 0x1 = A DMA transfer is complete and an unmasked interrupt is pending.
3	STUCK	R	0x0	Stuck Status Masked Interrupt Status. When unmasked, this interrupt indicates a line-hold-low error is detected. 0x0 = No interrupt 0x1 = An unmasked Stuck status (line-hold-error) interrupt is detected and pending.
2	NOATR	R	0x0	No Answer-to-Reset Masked Interrupt Status. 0x0 = No interrupt 0x1 = A No Answer-to-Reset unmasked interrupt is detected from the last reset and pending.
1	OPC	R	0x0	Operation Complete Masked Interrupt Status. 0x0 = No Interrupt 0x1 = An unmasked operation complete interrupt is pending.
0	RST	R	0x0	Reset Interrupt Mask. 0x0 = No Interrupt. 0x1 = The unmasked reset interrupt is pending.

22.5.8 ONEWIREICR Register (Offset = 0x10C) [reset = 0x0]

1-Wire Interrupt Clear (ONEWIREICR), offset 0x10C

The 1-Wire Interrupt Clear (ONEWIREICR) register is used to clear the ONEWIRERIS register (and by extension, the ONEWIREMIS register). When read, this register contains the same value as the ONEWIRERIS register. To clear current interrupts, read this register and write the results back.

ONEWIREICR is shown in [Figure 22-14](#) and described in [Table 22-12](#).

Return to [Summary Table](#).

Figure 22-14. ONEWIREICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMA	STUCK	NOATR	OPC	RST
R-0x0			W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0

Table 22-12. ONEWIREICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMA	W1C	0x0	DMA Done Interrupt Clear. Writing a 1 to this bit clears the DMA bit in the ONEWIRERIS register and the DMA bit in the ONEWIREMIS register.
3	STUCK	W1C	0x0	Stuck Status Interrupt Clear. Writing a 1 to this bit clears the STUCK bit in the ONEWIRERIS register and the STUCK bit in the ONEWIREMIS register.
2	NOATR	W1C	0x0	No Answer-to-Reset Interrupt Clear. Writing a 1 to this bit clears the NOATR bit in the ONEWIRERIS register and the NOATR bit in the ONEWIREMIS register.
1	OPC	W1C	0x0	Operation Complete Interrupt Clear. Writing a 1 to this bit clears the OPC bit in the ONEWIRERIS register and the OPC bit in the ONEWIREMIS register.
0	RST	W1C	0x0	Reset Interrupt Clear. Writing a 1 to this bit clears the RST bit in the ONEWIRERIS register and the RST bit in the ONEWIREMIS register.

22.5.9 ONEWIREDMA Register (Offset = 0x120) [reset = 0x0]

1-Wire uDMA Control (ONEWIREDMA), offset 0x120

The 1-Wire DMA Control (ONEWIREDMA) register is used to configure the uDMA operation for 1-Wire. This mechanism supports both uDMA write operations, uDMA read operations, small write/read operations, and scatter-gather support of mixed operations.

NOTE: This register is cleared when the dma_done signal to the 1-Wire module is asserted by the uDMA and the DMA bit in the ONEWIRERIS register is set.

ONEWIREDMA is shown in [Figure 22-15](#) and described in [Table 22-13](#).

Return to [Summary Table](#).

Figure 22-15. ONEWIREDMA Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												SG	DMAOP	RST	
R-0x0												R/W-0x0	R/W-0x0	R/W-0x0	

Table 22-13. ONEWIREDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	SG	R/W	0x0	Scatter-Gather Enable. This bit should be enabled when DMAOP = 0x1 and the scatter-gather method is being used. This bit self clears after it has performed the uDMA request. 0x0 = No effect 0x1 = uDMA is requested at start of operation and all requests after are on completion of transactions.
2-1	DMAOP	R/W	0x0	uDMA Operation. The programmed operation starts when reset completes if the RST bit in the ONEWIRECS register is set. If RST is not set, then write requests the uDMA immediately and read starts on a write to the ONEWIREDATW register. 0x0 = uDMA disabled 0x1 = uDMA single read: 1-Wire requests uDMA to read ONEWIREDATR register after each read transaction. 0x2 = uDMA multiple write: 1-Wire requests uDMA to load whenever the ONEWIREDATW register is empty. 0x3 = uDMA multiple read: An initial read occurs and subsequent reads start after uDMA has read the ONEWIREDATR register.
0	RST	R/W	0x0	uDMA Reset. 0x0 = No effect. 0x1 = A reset is issued and no reads and writes should be started until reset is done. Setting this bit sets the RST bit in the ONEWIRECS register. This bit is self-clearing upon reset completion.

22.5.10 ONEWIREPP Register (Offset = 0xFC0) [reset = 0x11]

1-Wire Peripheral Properties (ONEWIREPP), offset 0xFC0

The 1-Wire Peripheral Properties (ONEWIREPP) register contains peripheral properties for the 1-Wire module.

ONEWIREPP is shown in [Figure 22-16](#) and described in [Table 22-14](#).

Return to [Summary Table](#).

Figure 22-16. ONEWIREPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED			DMAP	RESERVED		CNT	
R-0x0			R-0x1	R-0x0		R-0x1	

Table 22-14. ONEWIREPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	DMAP	R	0x1	μDMA Present. Indicates whether μDMA is available. 0x0 = No μDMA 0x1 = μDMA is available to 1-Wire Module.
3-2	RESERVED	R	0x0	
1-0	CNT	R	0x1	1-Wire Bus Count. Indicates number of 1-Wire buses available. 0x0 = Reserved 0x1 = One 1-Wire bus.

Quad Synchronous Serial Interface (QSSI)

This chapter describes the Quad Synchronous Serial Interface (QSSI).

Topic	Page
23.1 Introduction	1523
23.2 Block Diagram	1523
23.3 Functional Description	1524
23.4 Initialization and Configuration	1534
23.5 QSSI Registers	1537

23.1 Introduction

The MSP432E4 microcontroller includes four Quad-Synchronous Serial Interface (QSSI) modules. All four of the modules support Advanced and Bi-SSI interfaces as well as a Quad-SSI enhancement to provide faster throughput of data. The QSSI module acts as a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI or TI synchronous serial interfaces. The QSSI performs serial-to-parallel conversion on data received from a peripheral device. The transmit and receive paths are buffered with internal, independent FIFO memories allowing up to eight 16-bit values in Legacy mode and 8-bit values in advanced, bi-, and quad-modes. The CPU can access data in these FIFOs as well as the control and status information of the QSSI. A μ DMA interface is also provided to allow the transmit and receive FIFOs to be programmed as source and destination addresses in the μ DMA module.

The QSSI modules have the following features:

- Four QSSI channels with advanced, bi-, and quad-SSI functionality
- Programmable interface operation for Freescale SPI or TI synchronous serial interfaces in Legacy Mode. Support for Freescale interface in bi- and quad-SSI mode.
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic or debug testing
- Standard FIFO-based interrupts and End-of-Transmission (EOT) interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
 - Transmit single request asserted when there is space in the FIFO; burst request asserted when four or more entries are available to be written in the FIFO
 - Maskable μ DMA interrupts for receive and transmit complete
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate baud clock.

23.2 Block Diagram

[Figure 23-1](#) shows a block diagram of an QSSI module with advanced, bi- and quad-SSI.

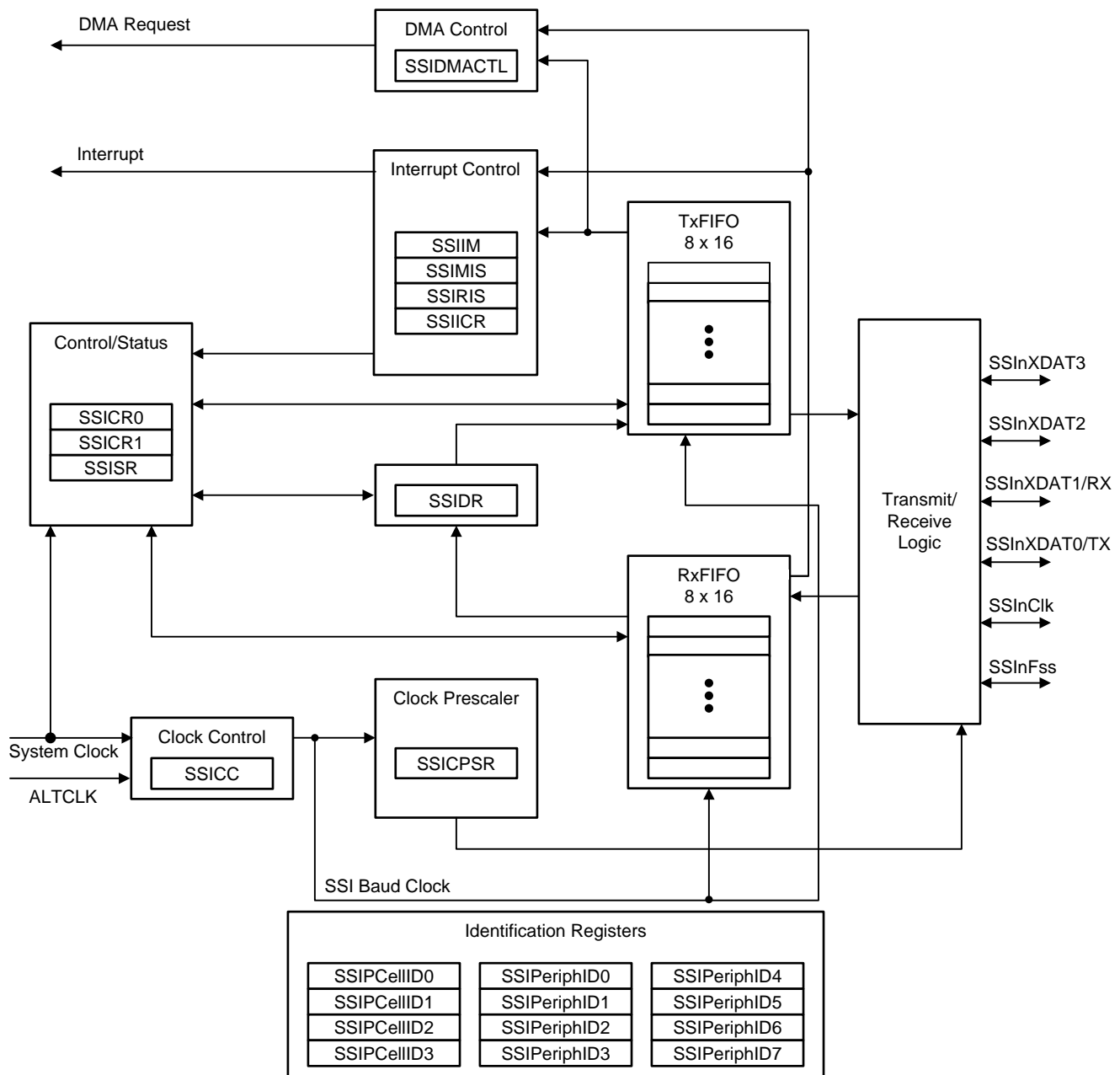


Figure 23-1. QSSI module with Advanced, Bi-SSI and Quad-SSI Support

23.3 Functional Description

The QSSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The QSSI also supports the μ DMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the μ DMA module. μ DMA operation is enabled by setting the appropriate bits in the SSIDMACTL register (see [Section 23.5.10](#)).

23.3.1 Bit Rate Generation

The QSSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (SysClk). The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the SSI Clock Prescale (SSICPSR) register (see [Section 23.5.5](#)). The clock is further divided by a value from 1 to 256, which is 1 + SCR, where SCR is the value programmed in the SSI Control 0 (SSICR0) register (see [Section 23.5.1](#)).

The frequency of the output clock SSInClk is defined by:

$$\text{SSInClk} = \text{SysClk} / (\text{CPSDVSR} \times (1 + \text{SCR}))$$

NOTE: SYSCLK or ALTCLK is used as the source for the SSInClk depending on how the CS field in the SSI Clock Configuration (SSICC) register is configured. For master legacy mode, the SYSCLK or ALTCLK must be at least two times faster than the SSInClk, with the restriction that SSInClk cannot be faster than 60 MHz. For slave mode, SYSCLK or ALTCLK must be at least 12 times faster than the SSInClk. In slave legacy mode, the maximum frequency of SSInClk is 10 MHz.

See the device-specific data sheet for the QSSI timing parameters.

23.3.2 FIFO Operation

23.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the SSI Data (SSIDR) register (see [Section 23.5.3](#)), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to a legacy SSI serial conversion and transmission to the attached slave or master, respectively, through the SSInDAT0/SSInTX pin.

In slave mode, the legacy SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the Rn bit in the RCGCSSI register or if the QSSI is reset using the SRSSI register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The QSSI can be configured to generate an interrupt or a μ DMA request when the FIFO is empty.

NOTE: When operating in Legacy Mode, the QuadSSI's SSInXDAT0 signal functions as SSInTX.

23.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data using the legacy serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the SSIDR register. If the receive FIFO is full when the master or slave receives new data, the data is held off until the receive FIFO has space.

The SSI only provides an SSIClk while transmitting data. When receiving data in master mode, a dummy write to the SSIDR register must be performed before any read so that the SSIClk can be properly received by the slave and allow data to be sent to the receive FIFO of the master.

When configured as a master or slave, serial data received through the SSInDAT1/SSInRX pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

NOTE: When operating in Legacy Mode, the SSInXDAT1 signal of QSSI functions as SSInRX.

23.3.3 Advanced, Bi- and Quad- SSI Function

Bi-SSI uses two data pins, SSInXDAT0 and SSInXDAT1, that can be configured to receive or transmit data. In Quad-SSI mode, SSInXDAT0, SSInXDAT1, SSInXDAT2 and SSInXDAT3 allow four bits of data to be received or transmitted at once. Note that in bi- and quad-SSI data transfers are only half-duplex.

By programming the MODE bits in the SSICR1 register, advanced, bi- or quad-SSI can be enabled. A direction bit, DIR, is provided to program the direction of operation during a bi- or quad-SSI transaction. Because bi- and quad-SSI cannot be full duplex, the DIR bit defines whether or not the RX FIFO is disabled. In Advanced operation, if the QSSI module TX (write) mode is enabled, the RX FIFO is automatically prevented from receiving any data. When Advanced SSI is in RX (read) mode, it operates as a full-duplex interface.

In bi- and quad-SSI mode, because only 8-bit data is allowed, the DSS bit field must be programmed to 0x7 in the SSICR0 register before transferring data to the Rx and TX FIFOs. For a data transmit, the 8-bit data packet is placed in a TX FIFO entry bits [7:0] and the mode of operation is inserted in the three most significant bits of the TX FIFO entry. The mode of operation bits [15:13] in the TX FIFO are used by the QSSI module for configuring the data on the proper pins. The following modes that may be placed on bits [15:13] of the FIFO entry are:

- Bi-SSI mode (0x1)
- Quad-SSI mode (0x2)
- Advanced SSI mode (0x3)

When data is first written to the TX FIFO, a SSInFss is asserted low indicating the start of a frame. At the EOT, bit 12 of the last data entry in the TX FIFO signifies whether a frame is ending. When the EOM bit is 1 it indicates a End of Message (EOM or STOP frame) and SSInFss is subsequently forced high. The EOM bit is cleared in the SSICR1 register on the same clock that the write to TXFIFO is completed. An EOM bit value of 0 indicates no change in transmission. If TX FIFO is emptied and SSInFSS is still asserted low, it remains low but SSInCLK is not pulsed. Likewise, if SSInFss is high when the TX FIFO is empty, it remains high.

During a Bi-SSI transmit frame, data is shifted out by two bits and placed on the corresponding two SSInDATn pins. For a quad-SSI transmit frame data is shifted out by four bits and placed on the corresponding four SSInDATn pins.

In bi-, quad- and advanced SSI, the lower byte of the Rx FIFO contains received data. The upper byte contains no valid information.

NOTE: While the master is in bi- or quad-SSI mode, if the DSS bit in the SSICR0 register is not set to 0x7, the QSSI module reverts to Legacy mode and behavior is not guaranteed.

The SSICR1 register bits DIR and MODE are used to program what operation is needed for the next data bytes that are being loaded into the FIFO. [Table 23-1](#) shows available modes of operation:

Table 23-1. QSSI Transaction Encodings

DIR	MODE	Operation
X	0x0	SSI Legacy operation supporting 4 to 16 data bits
0	0x1	Transmit (TX) bi-SSI with 8-bits of packet data
0	0x2	Transmit (TX) quad-SSI with 8-bits of packet data
0	0x3	Transmit (TX) advanced SSI mode with 8-bits of packet data and write RX FIFO disabled
1	0x1	Receive (RX) bi-SSI with 8-bits of packet data
1	0x2	Receive (RX) quad-SSI with 8-bits of packet data
1	0x3	Full duplex advanced SSI with 8-bits of packet data

NOTE: SPO = 0 and SPH = 0 is the only frame structure allowed for advanced, bi- and quad-SSI modes.

Different transactions can follow one another in the FIFOs. The following transaction combinations are allowed:

- Legacy SSI mode (if configured for this mode, switching to any other alternate mode is not recommended)
- Advanced SSI mode followed by bi-SSI mode
- Advanced SSI mode followed by quad-SSI mode
- Advanced SSI mode followed by bi-SSI mode followed by advanced SSI mode
- Advanced SSI mode followed by quad-SSI mode followed by advanced SSI mode

Note that switching between Quad-SSI and bi-SSI is not encouraged in a single transaction.

23.3.4 SSInFSS Function

For enhanced modes of operation, the SSInFss signal can be programmed to assert low at the start of each byte transfer for one clock or the entire frame. This is configured by programming the FSSHLDfrm bit in the SSICR1 register. The EOM bit is also provided to signify end of frame transmission. This bit is embedded in the TXFIFO entry for use at the interface to deassert SSInFss at the appropriate time. The FSSHLDfrm bit can also be used when operating in 8-bit legacy SSI mode.

The functionality of the FSSHLDfrm bit for both legacy SSI mode and the enhanced modes are as follows:

Table 23-2. SSInFss Functionality

Mode	FSSHLDfrm	Description
Legacy mode	0	For Freescale format, with SPH = 0, the SSInFss signal is deasserted (high) between continuous transfers. For SPH = 1, the SSInFss signal is asserted (low) between continuous transfers. For TI format, the SSInFss signal is deasserted (high) after every data transfer.
	1	Not a valid combination as the SSInFSS signal is forced low even after transmission is completed and requires the FSSHLDfrm bit to be cleared to release the SSInFSS signal.
Advanced, bi-, and quad-SSI mode	0	SSInFss is asserted low after every byte of data
	1	New data written to the TX FIFO notifies SSInFss to assert low until the EOM bit is set with the last transfer, only after which the SSInFSS is asserted high.

23.3.5 High Speed Clock Operation

In master mode, QSSI module can enable a high speed clock by setting the HSCLKEN bit in the SSI Control 1 (SSICR1) register. In this mode of operation, SSInCLK from the QSSI master operation is reflected back as a loopback clock, HSPEEDCLK, to the QSSI module. This allows faster timing since the logic can be used to adjust clock to external data relationships. HSPEEDCLK captures RX data in a separate register. This allows the time between the clock as seen by a remote device and the internal clock to match more closely.

Receive data is captured in a separate register sampled on loop-back clock (HSPEEDCLK) and the RX FIFO write control registered on HSPEEDCLK. If the HSCKEN = 1, the corresponding shift register and FIFO write enable will be selected for use. This supports faster QSSI master speed.

NOTE: For proper functionality of high speed mode, the HSCLKEN bit in the SSICR1 register should be set before any SSI data transfer or after applying a reset to the QSSI module. In addition, the SSE bit must be set to 0x1 before the HSCLKEN bit is set.

23.3.6 Interrupts

The QSSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service (when the transmit FIFO is half full or less)
- Receive FIFO service (when the receive FIFO is half full or more)
- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission
- Receive DMA transfer complete
- Transmit DMA transfer complete

All of the interrupt events are ORed together before being sent to the interrupt controller, so the QSSI generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the seven individual maskable interrupts can be masked by clearing the appropriate bit in the SSI Interrupt Mask (SSIIM) register (see [Section 23.5.6](#)). Setting the appropriate mask bit enables the interrupt.

The individual outputs, along with a combined interrupt output, allow use of either a global interrupt service routine or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the SSI Raw Interrupt Status (SSIRIS) and SSI Masked Interrupt Status (SSIMIS) registers (see [Section 23.5.7](#) and [Section 23.5.8](#), respectively).

The RX FIFO has an associated time-out counter which starts to down count at the same time the RX FIFO is flagged as not empty by the RNE bit in the SSISR register. The counter is reset any time a new or next byte is written to the RX FIFO, thus the counter will continue to count down to zero unless there is new activity. The time-out period is 32 periods based on the period of SSInClk. When the counter reaches zero, a time-out interrupt bit, RTRIS, is set in the SSIRIS register. The time-out interrupt can be cleared by writing a 1 to the RTIC bit of the SSI Interrupt Clear (SSIIC) register or by emptying the RX FIFO. If the interrupt is cleared and there is residual data left in the RX FIFO or new data entries have been written, the timer count down initiates and the interrupt will be reasserted after 32 periods have been counted.

The EOT interrupt indicates that the data has been transmitted completely and is only valid for master mode devices and operations. This interrupt can be used to indicate when it is safe to turn off the QSSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

NOTE: In Freescale SPI mode only, a condition can be created where an EOT interrupt is generated for every byte transferred even if the FIFO is full. If the μ DMA has been configured to transfer data from this QSSI to a Master QSSI on the device using external loopback, an EOT interrupt is generated by the QSSI slave for every byte even if the FIFO is full.

23.3.7 Frame Formats

Each data frame is between 4 and 16 bits long in legacy mode and 8-bits in advanced, bi-, and quad-SSI mode and is transmitted starting with the MSB. There are two basic frame types that can be selected by programming the FRF bit in the SSICR0 register:

- TI synchronous serial
- Freescale SPI

NOTE: Advanced, Bi- and Quad-SSI modules only supports Freescale mode when SPH = 0, SPO = 0, and DDS = 0x8 in the SSI Control 0 (SSICR0) register.

For both formats, the serial clock (SSInClk) is held inactive while the QSSI is idle, and SSInClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSInClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI frame format, the serial frame (SSInFss) pin is active low, and is asserted (pulled down) during the entire transmission of the frame.

For TI synchronous serial frame format, the SSInFss pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the QSSI and the off-chip slave device drive their output data on the rising edge of SSInClk and latch data from the other device on the falling edge.

Table 23-3 is a synopsis of the features supported in each frame format when operating in legacy Mode:

Table 23-3. Legacy Mode TI, Freescale SPI Frame Format Features

Feature	TI Mode	Freescale SPI Mode
Frame hold	Not available	Available
High speed (master RX only)	Not available	Available
SPO and SPH configuration	Not available	Available and can be used in combination with frame hold and high-speed mode
Frequency (system clock: SSInCLK)	Master 1:2 Slave 1:12	Master 1:2 Slave 1:12

For advanced, bi-, and quad-SSI modes using the Freescale SPI format or the bi- and quad-SSi modes using the TI format, the following features are supported:

- Frame hold
- High speed (master RX only)
- SPO and SPH configuration with SPO = 0 and SPH = 0 only allowed
- Frequency (system clock: SSInCLK):
 - Master 1:2
 - Slave 1:12

23.3.7.1 TI Synchronous Serial Frame Format

Figure 23-2 shows the TI synchronous serial frame format for a single transmitted frame.

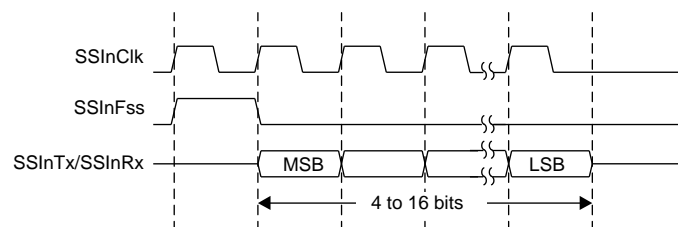


Figure 23-2. TI Synchronous Serial Frame Format (Single Transfer)

In this mode, SSInClk and SSInFss are forced Low, and the transmit data line SSInDAT0/SSInTX is in a tristate condition whenever the QSSI is idle. Once the bottom entry of the transmit FIFO contains data, SSInFss is pulsed High for one SSInClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSInClk, the MSB of the 4- to 16-bit data frame is shifted out on the SSInDAT0 and SSInTX pins. Likewise, the MSB of the received data is shifted onto the SSInDAT1 and SSInRX pins by the off-chip serial slave device.

Both the QSSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SSInClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSInClk after the LSB has been latched.

Figure 23-3 shows the TI synchronous serial frame format when back-to-back frames are transmitted.

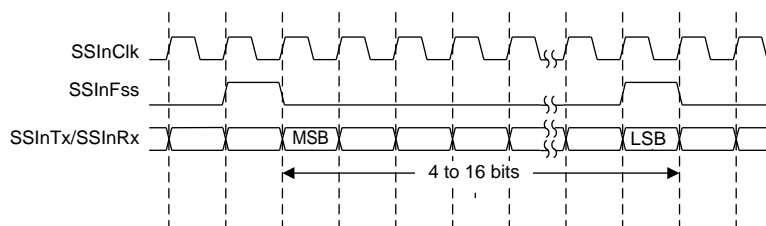


Figure 23-3. TI Synchronous Serial Frame Format (Continuous Transfer)

23.3.7.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSInFss signal behaves as a slave select. If operating in legacy mode and using the Freescale SPI frame format, the inactive state and phase of the SSInClk signal are programmable through the SPO and SPH bits in the SSICR0 control register. If operating in advanced-, bi-, or quad-SSI mode, the SPO and SPH bits must be programmed to 0.

23.3.7.2.1 SPO Clock Polarity Bit

When the SPO clock polarity control bit is clear, it produces a steady state low value on the SSInClk pin. If the SPO bit is set, a steady state high value is placed on the SSInClk pin when data is not being transferred.

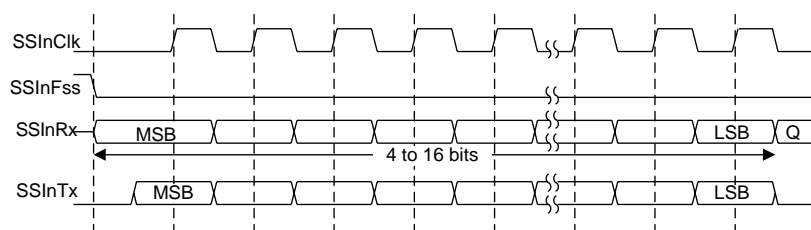
23.3.7.2.2 SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is clear, data is captured on the first clock edge transition. If the SPH bit is set, data is captured on the second clock edge transition.

23.3.7.3 Freescale SPI Frame Format With SPO = 0 and SPH = 0

Single and continuous transmission signal sequences for Freescale SPI format with SPO = 0 and SPH = 0 are shown in [Figure 23-4](#) and [Figure 23-5](#).

NOTE: This is the only Freescale SPI frame format configuration that can be used when operating in advanced, bi-, quad-SSI mode.



NOTE: Q is undefined.

Figure 23-4. Freescale SPI Format (Single Transfer) with SPO = 0 and SPH = 0

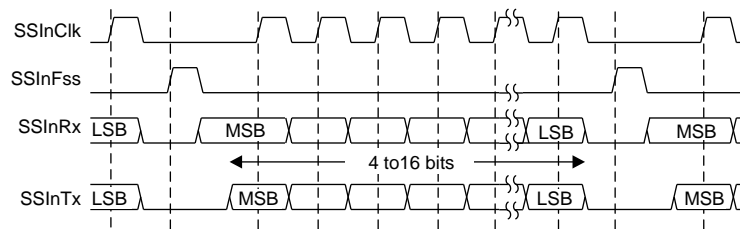


Figure 23-5. Freescale SPI Format (Continuous Transfer) with SPO = 0 and SPH = 0

In this configuration, during idle periods:

- SSInClk is forced low
- SSInFss is forced high
- The transmit data line SSInDAT0 and SSInTX is in a tristate condition
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven low, causing slave data to be enabled onto the SSInDAT1 and SSInRX input line of the master. The master SSInDAT0 and SSInTX output pad is enabled.

One half SSInClk period later, valid master data is transferred to the SSInDAT0 and SSInTX pin. Once both the master and slave data have been set, the SSInClk master clock pin goes High after one additional half SSInClk period.

The data is now captured on the rising and propagated on the falling edges of the SSInClk signal.

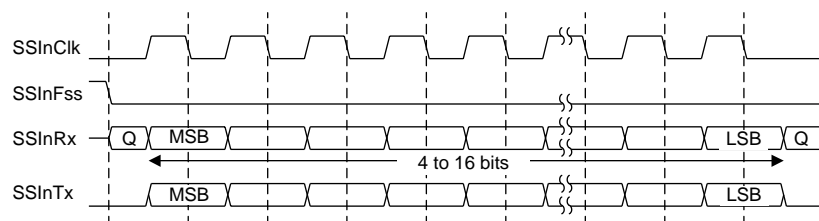
In the case of a single word transmission, after all bits of the data word have been transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSInFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSInFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSInFss pin is returned to its idle state one SSInClk period after the last bit has been captured.

23.3.7.4 Freescale SPI Frame Format with SPO = 0 and SPH = 1

The transfer signal sequence for Freescale SPI format with SPO = 0 and SPH = 1 is shown in [Figure 23-6](#), which covers both single and continuous transfers.

NOTE: This Freescale SPI frame format configuration is only available when operating in legacy SSI mode of operation.



NOTE: Q is undefined.

Figure 23-6. Freescale SPI Frame Format with SPO = 0 and SPH = 1

In this configuration, during idle periods:

- SSInClk is forced low
- SSInFss is forced high
- The transmit data line SSInDAT0/SSInTX is in a tristate condition
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven low. The master SSInDAT0 and SSInTX output is enabled. After an additional one-half SSInClk period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the SSInClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSInClk signal.

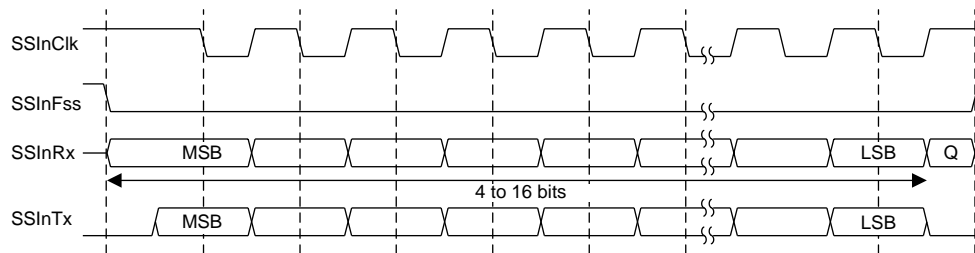
In the case of a single word transfer, after all bits have been transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

For continuous back-to-back transfers, the SSInFss pin is held Low between successive data words, and termination is the same as that of the single word transfer.

23.3.7.5 Freescale SPI Frame Format with SPO = 1 and SPH = 0

Single and continuous transmission signal sequences for Freescale SPI format with SPO = 1 and SPH = 0 are shown in [Figure 23-7](#) and [Figure 23-8](#).

NOTE: This Freescale SPI frame format configuration is only available when operating in legacy SSI mode of operation.



NOTE: Q is undefined.

Figure 23-7. Freescale SPI Frame Format (Single Transfer) With SPO = 1 and SPH = 0

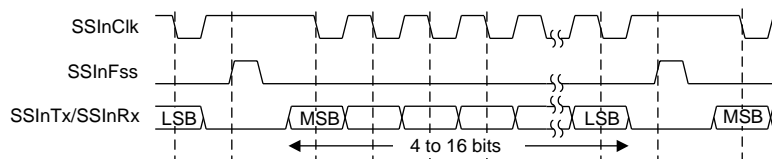


Figure 23-8. Freescale SPI Frame Format (Continuous Transfer) With SPO = 1 and SPH = 0

In this configuration, during idle periods:

- SSInClk is forced high
- SSInFss is forced high
- The transmit data line SSInDAT0/SSInTX is in a tristate condition
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low, causing slave data to be immediately transferred onto the SSInDAT1 and SSInRX line of the master. The master SSInDAT0 and SSInTX output pad is enabled.

One-half period later, valid master data is transferred to the SSInDAT0 and SSInTX line. Once both the master and slave data have been set, the SSInClk master clock pin becomes Low after one additional half SSInClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSInClk signal.

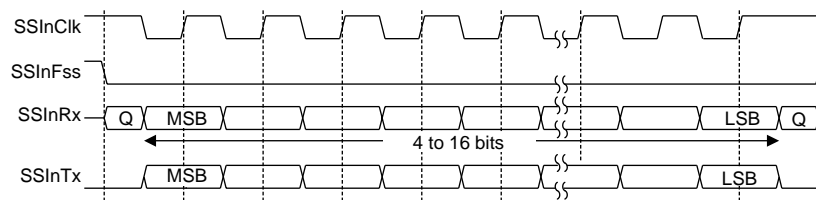
In the case of a single word transmission, after all bits of the data word are transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSInFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSInFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSInFss pin is returned to its idle state one SSInClk period after the last bit has been captured.

23.3.7.6 Freescale SPI Frame Format With SPO = 1 and SPH = 1

Figure 23-9 shows the transfer signal sequence for both single and continuous transfers in Freescale SPI format with SPO = 1 and SPH = 1.

NOTE: This Freescale SPI frame format configuration is only available when operating in Legacy SSI mode of operation.



NOTE: Q is undefined.

Figure 23-9. Freescale SPI Frame Format With SPO = 1 and SPH = 1

In this configuration, during idle periods:

- SSInClk is forced high
- SSInFss is forced high
- The transmit data line SSInDAT0/SSInTX is in a tristate condition
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low. The master SSInDAT0 and SSInTX output pad is enabled. After an additional one-half SSInClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSInClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSInClk signal.

After all bits have been transferred, in the case of a single word transmission, the SSInFss line is returned to its idle high state one SSInClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSInFss pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSInFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

23.3.8 DMA Operation

The QSSI peripheral provides an interface to the μ DMA controller with separate channels for transmit and receive. The μ DMA operation of the QSSI is enabled through the SSI DMA Control (SSIDMACTL) register. When μ DMA operation is enabled, the QSSI asserts a μ DMA request on the receive or transmit channel when the associated FIFO can transfer data.

For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst μ DMA transfer requests are handled automatically by the μ DMA controller depending how the μ DMA channel is configured.

To enable μ DMA operation for the receive channel, the RXDMAE bit of the DMA Control (SSIDMACTL) register should be set after configuring the μ DMA. To enable μ DMA operation for the transmit channel, the TXDMAE bit of SSIDMACTL should be set after configuring the μ DMA.

If the μ DMA is enabled and has completed a data transfer from the Tx FIFO, the DMATXRIS bit is set in the SSIRIS register and cannot be cleared by setting the DMATXIC bit in the SSI Interrupt Clear (SSIICR) register. In the DMA Completion Interrupt Service Routine, software must disable the μ DMA transmit enable to the SSI by clearing the TXDMAE bit in the QSSI DMA Control (SSIDMACTL) register and then setting the DMATXIC bit in the SSIICR register. This clears the DMA completion interrupt. When the μ DMA is needed to transmit more data, the TXDMAE bit must be set (enabled) again.

If a data transfer by the μ DMA from the Rx FIFO completes, the DMARXRIS bit is set. The EOT bit in the SSIRIS register is also provided to indicate when the Tx FIFO is empty and the last bit has been transmitted out of the serializer

NOTE: Wait states are inserted at every byte transfer when using Bi- or Quad-SSI modes as a master with the μ DMA at SSICLK frequencies greater than one sixth of the system clock. These wait states are because of arbitration stall cycles from the μ DMA accesses to SRAM and increased output throughput from the SSI.

See [Chapter 8](#) for more details about programming the μ DMA controller.

23.4 Initialization and Configuration

To enable and initialize the QSSI, the following steps are necessary:

1. Enable the QSSI module using the RCGCSSI register (see [Section 4.2.92](#)).
2. Enable the clock to the appropriate GPIO module through the RCGCGPIO register (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet.
3. Set the GPIO AFSEL bits for the appropriate pins (see [Section 17.5.10](#)). To determine which GPIOs to configure, see the device-specific data sheet.
4. Configure the PMCN fields in the GPIOCTL register to assign the QSSI signals to the appropriate pins. See [Section 17.5.22](#) and the device-specific data sheet.
5. Program the GPIODEN register to enable the pin's digital function. In addition, the drive strength, drain select and pullup and pulldown functions must be configured. See [Chapter 17](#) for more information.

NOTE: Pullups can be used to avoid unnecessary toggles on the QSSI pins, which can take the slave to a wrong state. In addition, if the SSIClk signal is programmed to steady state high through the SPO bit in the SSICR0 register, then software must also configure the GPIO port pin corresponding to the SSIClk signal as a pullup in the GPIO Pullup Select (GPIOPUR) register.

For each of the frame formats, the QSSI is configured using the following steps:

1. If initializing out of reset, ensure that the SSE bit in the SSICR1 register is clear before making any configuration changes. Otherwise, configuration changes for advanced SSI can be made while the SSE bit is set.

2. Select whether the QSSI is a master or slave:
 1. For master operations, set the SSICR1 register to 0x0000.0000.
 2. For slave mode (output enabled), set the SSICR1 register to 0x0000.0004.
 3. For slave mode (output disabled), set the SSICR1 register to 0x0000.000C.
3. Configure the QSSI clock source by writing to the SSICC register.
4. Configure the clock prescale divisor by writing the SSICPSR register.
5. Write the SSICR0 register with the following configuration:
 - Serial clock rate (SCR)
 - Desired clock phase and polarity, if using Freescale SPI mode (SPH and SPO)
 - The protocol mode: Freescale SPI or TI SSF
 - The data size (DSS)
6. (Optional) Configure the SSI module for μ DMA use with the following steps:
 1. Configure a μ DMA for SSI use. See [Chapter 8](#) for more information.
 2. Enable the TX FIFO or RX FIFO of the SSI module by setting the TXDMAE or RXDMAE bit in the SSIDMACTL register.
 3. Optionally, enable the μ DMA completion interrupt by setting the DMATXIM or DMARXIM bit in the SSIIM register.

NOTE: For a TX DMA completion interrupt, software must disable the μ DMA transmit enable to the SSI by clearing the TXDMAE bit in the QSSI DMA Control (SSIDMACTL) register and then setting the DMATXIC bit in the SSICR register. This clears the DMA completion interrupt. When the μ DMA is needed to transmit more data, the TXDMAE bit must be set (enabled) again.

7. If this is the first initialization out of reset, enable the QSSI by setting the SSE bit in the SSICR1 register.

As an example, assume the QSSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO = 1, SPH = 1)
- 1-Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

$$\text{SSInClk} = \text{SysClk} / (\text{CPSDVSR} \times (1 + \text{SCR})) \quad 1 \times 10^6 = 20 \times 10^6 / (\text{CPSDVSR} \times (1 + \text{SCR})) \quad (65)$$

In this case, if CPSDVSR = 0x2, SCR must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the SSICR1 register is clear.
2. Write the SSICR1 register with a value of 0x0000.0000.
3. Write the SSICPSR register with a value of 0x0000.0002.
4. Write the SSICR0 register with a value of 0x0000.09C7.
5. The QSSI is then enabled by setting the SSE bit in the SSICR1 register.

23.4.1 Enhanced Mode Configuration

If the QSSI module supports the advanced, bi, and quad features, then these modes can be enabled after initializing the QSSI module. The following is an example of configuring the QSSI to transmit two data bytes in advanced SSI mode, followed by 2 bytes in bi-SSI mode:

1. Set the MODE bit to 0x3, and the FSSHLDFM bit to 1 in the SSICR1 register. To operate in the master mode, program the MS bit to 0. Program the remaining bits in the SSICR0 and SSICR1 register to relevant values.

2. Write one data byte to the TX FIFO; set the EOM bit to 1 and write the second data byte to the Tx FIFO.
3. Set the MODE bit to 0x1 and the FSSHLDFM bit to 1 in the SSICR1 register. To operate in the master mode, program the MS bit to 0. Program the remaining bits in the SSICR0 and SSICR1 register to relevant values.
4. Fill the Tx FIFO with one data byte.
5. Set the EOM bit in the SSICR1 register.
6. Fill the Tx FIFO with one data byte.

23.5 QSSI Registers

Table 23-4 lists the memory-mapped registers for the QSSI. All register offset addresses not listed in Table 23-4 should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the base address of each QSSI module:

- QSSI0: 0x40008000
- QSSI1: 0x40009000
- QSSI2: 0x4000A000
- QSSI3: 0x4000B000

The QSSI module clock must be enabled before the registers can be programmed (see Section 4.2.92). The Rn bit of the PRSSI register must be read as 0x1 before any QSSI module registers are accessed.

Table 23-4. QSSI Registers

Offset	Acronym	Register Name	Section
0x0	SSICR0	QSSI Control 0	Section 23.5.1
0x4	SSICR1	QSSI Control 1	Section 23.5.2
0x8	SSIDR	QSSI Data	Section 23.5.3
0xC	SSISR	QSSI Status	Section 23.5.4
0x10	SSICPSR	QSSI Clock Prescale	Section 23.5.5
0x14	SSIIM	QSSI Interrupt Mask	Section 23.5.6
0x18	SSIRIS	QSSI Raw Interrupt Status	Section 23.5.7
0x1C	SSIMIS	QSSI Masked Interrupt Status	Section 23.5.8
0x20	SSIICR	QSSI Interrupt Clear	Section 23.5.9
0x24	SSIDMACTL	QSSI DMA Control	Section 23.5.10
0xFC0	SSIPP	QSSI Peripheral Properties	Section 23.5.11
0xFC8	SSICC	QSSI Clock Configuration	Section 23.5.12
0xFD0	SSIPeriphID4	QSSI Peripheral Identification 4	Section 23.5.13
0xFD4	SSIPeriphID5	QSSI Peripheral Identification 5	Section 23.5.14
0xFD8	SSIPeriphID6	QSSI Peripheral Identification 6	Section 23.5.15
0xFDC	SSIPeriphID7	QSSI Peripheral Identification 7	Section 23.5.16
0xFE0	SSIPeriphID0	QSSI Peripheral Identification 0	Section 23.5.17
0xFE4	SSIPeriphID1	QSSI Peripheral Identification 1	Section 23.5.18
0xFE8	SSIPeriphID2	QSSI Peripheral Identification 2	Section 23.5.19
0xFEC	SSIPeriphID3	QSSI Peripheral Identification 3	Section 23.5.20
0xFF0	SSIPCellID0	QSSI PrimeCell Identification 0	Section 23.5.21
0xFF4	SSIPCellID1	QSSI PrimeCell Identification 1	Section 23.5.22
0xFF8	SSIPCellID2	QSSI PrimeCell Identification 2	Section 23.5.23
0xFFC	SSIPCellID3	QSSI PrimeCell Identification 3	Section 23.5.24

Complex bit access types are encoded to fit into small table cells. Table 23-5 shows the codes that are used for access types in this section.

Table 23-5. QSSI Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1CW	1 to clearWrite
Reset or Default Value		

Table 23-5. QSSI Access Type Codes (continued)

Access Type	Code	Description
$-n$		Value after reset or the default value

23.5.1 SSICR0 Register (Offset = 0x0) [reset = 0x0]

QSSI Control 0 (SSICR0), offset 0x000

The SSICR0 register contains bit fields that control various functions within the QSSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSICR0 is shown in [Figure 23-10](#) and described in [Table 23-6](#).

[Return to Summary Table.](#)

Figure 23-10. SSICR0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCR								SPH	SPO	FRF		DSS			
R/W-0x0								R/W-0x0	R/W-0x0	R/W-0x0		R/W-0x0			

Table 23-6. SSICR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-8	SCR	R/W	0x0	QSSI Serial Clock Rate. This bit field is used to generate the transmit and receive bit rate of the QSSI. The bit rate is: $BR = SysClk / (CPSDVSR * (1 + SCR))$ where CPSDVSR is an even value from 2 to 254 programmed in the SSICPSR register, and SCR is a value from 0 to 255.
7	SPH	R/W	0x0	QSSI Serial Clock Phase. This bit is only applicable to the Freescale SPI Format. The SPH control bit selects the clock edge that captures data and allows it to change state. This bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. 0x0 = Data is captured on the first clock edge transition. 0x1 = Data is captured on the second clock edge transition.
6	SPO	R/W	0x0	QSSI Serial Clock Polarity 0x0 = A steady state low value is placed on the SSInClk pin. 0x1 = A steady state high value is placed on the SSInClk pin when data is not being transferred. If this bit is set, then software must also configure the GPIO port pin corresponding to the SSInClk signal as a pullup in the GPIO Pullup Select (GPIOPUR) register.
5-4	FRF	R/W	0x0	QSSI Frame Format Select. When operating in Advanced, Bi-, or Quad-SSI mode these bits must be 0x0. 0x0 = Freescale SPI Frame Format 0x1 = Texas Instruments Synchronous Serial Frame Format 0x02 = Reserved 0x03 = Reserved

Table 23-6. SSICR0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	DSS	R/W	0x0	<p>QSSI Data Size Select. When operating in Advanced, Bi- or Quad-SSI, data size can only be 8-bit. All other fields will be ignored.</p> <p>0x0 = Reserved</p> <p>0x1 = Reserved</p> <p>0x2 = Reserved</p> <p>0x3 = 4-bit data</p> <p>0x4 = 5-bit data</p> <p>0x5 = 6-bit data</p> <p>0x6 = 7-bit data</p> <p>0x7 = 8-bit data</p> <p>0x8 = 9-bit data</p> <p>0x9 = 10-bit data</p> <p>0xA = 11-bit data</p> <p>0xC = 13-bit data</p> <p>0xD = 14-bit data</p> <p>0xE = 15-bit data</p> <p>0xF = 16-bit data</p>

23.5.2 SSICR1 Register (Offset = 0x4) [reset = 0x0]

QSSI Control 1 (SSICR1), offset 0x004

The SSICR1 register contains bit fields that control various functions within the QSSI module. Master and slave mode functionality is controlled by this register.

SSICR1 is shown in [Figure 23-11](#) and described in [Table 23-7](#).

Return to [Summary Table](#).

Figure 23-11. SSICR1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED				EOM	FSSHLD FRM	HSCLKEN	DIR
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
MODE		RESERVED			MS	SSE	LBM
R/W-0x0		R-0x0			R/W-0x0	R/W-0x0	R/W-0x0

Table 23-7. SSICR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	EOM	R/W	0x0	Stop Frame (End of Message). This bit is applicable when MODE is set to Advanced, Bi- or Quad- SSI. This bit is inserted into bit 12 of the TXFIFO data entry by the QSSI module. 0x0 = No change in transmission status. 0x1 = End of message (Stop Frame).
10	FSSHLD FRM	R/W	0x0	FSS Hold Frame 0x0 = Pulse SSInFss at every byte (the DSS bit in the SSICR0 register must be set to 0x7 (data size 8 bits) in this configuration) 0x1 = Hold SSInFss for the whole frame
9	HSCLKEN	R/W	0x0	High Speed Clock Enable. High speed clock enable is available only when operating as a master. For proper functionality of high speed mode, the HSCLKEN bit in the SSICR1 register should be set before any SSI data transfer or after applying a reset to the QSSI module. In addition, the SSE bit must be set to 0x1 before the HSCLKEN bit is set. 0x0 = Use Input Clock 0x1 = Use High Speed Clock
8	DIR	R/W	0x0	QSSI Direction of Operation 0x0 = TX (Transmit Mode) write direction 0x1 = RX (Receive Mode) read direction
7-6	MODE	R/W	0x0	QSSI Mode 0x0 = Legacy SSI mode 0x1 = Bi-SSI mode 0x2 = Quad-SSI Mode 0x3 = Advanced SSI Mode with 8-bit packet size
5-3	RESERVED	R	0x0	

Table 23-7. SSICR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	MS	R/W	0x0	QSSI Master/Slave Select. This bit selects Master or Slave mode and can be modified only when the QSSI is disabled (SSE = 0). 0x0 = The QSSI is configured as a master. 0x1 = The QSSI is configured as a slave.
1	SSE	R/W	0x0	QSSI Synchronous Serial Port Enable 0x0 = QSSI operation is disabled. 0x1 = QSSI operation is enabled. The HSCLKEN bit in the SSICR1 register should be set only after applying reset to the QSSI module and enabling the QSSI by setting the SSE bit, and before any SSI data transfer. All other bits in the SSICR1 register and all bits in SSICR0 register can only be programmed when the SSE is clear.
0	LBM	R/W	0x0	QSSI Loopback Mode 0x0 = Normal serial port operation enabled. 0x1 = Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

23.5.3 SSIDR Register (Offset = 0x8) [reset = 0x0]

QSSI Data (SSIDR), offset 0x008

The SSIDR register is 16-bits wide. When the SSIDR register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the QSSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

When the SSIDR register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted out onto the SSInDAT0/SSInTX pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

SSIDR is shown in [Figure 23-12](#) and described in [Table 23-8](#).

Return to [Summary Table](#).

Figure 23-12. SSIDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DATA															
R-0x0																R/W-0x0															

Table 23-8. SSIDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	DATA	R/W	0x0	QSSI Receive/Transmit Data. A read operation reads the receive FIFO. A write operation writes the transmit FIFO. Software must right-justify data when the QSSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

23.5.4 SSISR Register (Offset = 0xC) [reset = 0x3]

QSSI Status (SSISR), offset 0x00C

The SSISR register contains bits that indicate the FIFO fill status and the QSSI busy status.

SSISR is shown in [Figure 23-13](#) and described in [Table 23-9](#).

Return to [Summary Table](#).

Figure 23-13. SSISR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											BSY	RFF	RNE	TNF	TFE
R-0x0											R-0x0	R-0x0	R-0x0	R-0x1	R-0x1

Table 23-9. SSISR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0x0	
4	BSY	R	0x0	QSSI Busy Bit 0x0 = The QSSI is idle. 0x1 = The QSSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	R	0x0	QSSI Receive FIFO Full 0x0 = The receive FIFO is not full. 0x1 = The receive FIFO is full.
2	RNE	R	0x0	QSSI Receive FIFO Not Empty 0x0 = The receive FIFO is empty. 0x1 = The receive FIFO is not empty.
1	TNF	R	0x1	QSSI Transmit FIFO Not Full 0x0 = The transmit FIFO is full. 0x1 = The transmit FIFO is not full.
0	TFE	R	0x1	QSSI Transmit FIFO Empty 0x0 = The transmit FIFO is not empty. 0x1 = The transmit FIFO is empty.

23.5.5 SSICPSR Register (Offset = 0x10) [reset = 0x0]

QSSI Clock Prescale (SSICPSR), offset 0x010

The SSICPSR register specifies the division factor which is used to derive the SSInClk from the system clock. The clock is further divided by a value from 1 to 256, which is 1 + SCR. SCR is programmed in the SSICR0 register. The frequency of the SSInClk is defined by:

$$\text{SSInClk} = \text{SysClk} / (\text{CPSDVSR} * (1 + \text{SCR}))$$

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSICPSR is shown in [Figure 23-14](#) and described in [Table 23-10](#).

Return to [Summary Table](#).

Figure 23-14. SSICPSR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CPSDVSR							
R-0x0																								R/W-0x0							

Table 23-10. SSICPSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CPSDVSR	R/W	0x0	QSSI Clock Prescale Divisor. This value must be an even number from 2 to 254, depending on the frequency of SSInClk. The LSB always returns 0 on reads.

23.5.6 SSIM Register (Offset = 0x14) [reset = 0x0]

QSSI Interrupt Mask (SSIM), offset 0x014

The SSIM register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared on reset.

On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit clears the mask, enabling the interrupt to be sent to the interrupt controller. Clearing a bit sets the corresponding mask, preventing the interrupt from being signaled to the controller.

SSIM is shown in [Figure 23-15](#) and described in [Table 23-11](#).

Return to [Summary Table](#).

Figure 23-15. SSIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	EOTIM	DMATXIM	DMARXIM	TXIM	RXIM	RTIM	RORIM
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 23-11. SSIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6	EOTIM	R/W	0x0	End of Transmit Interrupt Mask 0x0 = The end of transmit interrupt is masked. 0x1 = The end of transmit interrupt is not masked.
5	DMATXIM	R/W	0x0	QSSI Transmit DMA Interrupt Mask 0x0 = The transmit DMA interrupt is masked. 0x1 = The transmit DMA interrupt is not masked.
4	DMARXIM	R/W	0x0	QSSI Receive DMA Interrupt Mask 0x0 = The receive DMA interrupt is masked. 0x1 = The receive DMA interrupt is not masked.
3	TXIM	R/W	0x0	QSSI Transmit FIFO Interrupt Mask 0x0 = The transmit FIFO interrupt is masked. 0x1 = The transmit FIFO interrupt is not masked.
2	RXIM	R/W	0x0	QSSI Receive FIFO Interrupt Mask 0x0 = The receive FIFO interrupt is masked. 0x1 = The receive FIFO interrupt is not masked.
1	RTIM	R/W	0x0	QSSI Receive Time-Out Interrupt Mask 0x0 = The receive FIFO time-out interrupt is masked. 0x1 = The receive FIFO time-out interrupt is not masked.
0	RORIM	R/W	0x0	QSSI Receive Overrun Interrupt Mask 0x0 = The receive FIFO overrun interrupt is masked. 0x1 = The receive FIFO overrun interrupt is not masked.

23.5.7 SSIRIS Register (Offset = 0x18) [reset = 0x8]

QSSI Raw Interrupt Status (SSIRIS), offset 0x018

The SSIRIS register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSIRIS is shown in [Figure 23-16](#) and described in [Table 23-12](#).

Return to [Summary Table](#).

Figure 23-16. SSIRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	EOTRIS	DMATXRIS	DMARXRIS	TXRIS	RXRIS	RTRIS	RORRIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x1	R-0x0	R-0x0	R-0x0

Table 23-12. SSIRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6	EOTRIS	R	0x0	End of Transmit Raw Interrupt Status. This bit is cleared when a 1 is written to the EOTIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = No interrupt. 0x1 = The transmit FIFO is empty, and the last bit has been transmitted out of the serializer.
5	DMATXRIS	R	0x0	QSSI Transmit DMA Raw Interrupt Status. This bit is cleared when a 1 is written to the DMATXIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = No interrupt. 0x1 = The transmit DMA has completed.
4	DMARXRIS	R	0x0	QSSI Receive DMA Raw Interrupt Status. This bit is cleared when a 1 is written to the DMARXIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = No interrupt. 0x1 = The receive DMA has completed.
3	TXRIS	R	0x1	QSSI Transmit FIFO Raw Interrupt Status. This bit is cleared when the transmit FIFO is more than half full. 0x0 = No interrupt. 0x1 = The transmit FIFO is half empty or less.
2	RXRIS	R	0x0	QSSI Receive FIFO Raw Interrupt Status. This bit is cleared when the receive FIFO is less than half full. 0x0 = No interrupt. 0x1 = The receive FIFO is half full or more.
1	RTRIS	R	0x0	QSSI Receive Time-Out Raw Interrupt Status. This bit is cleared when a 1 is written to the RTIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = No interrupt. 0x1 = The receive time-out has occurred.

Table 23-12. SSIRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RORRIS	R	0x0	QSSI Receive Overrun Raw Interrupt Status. This bit is cleared when a 1 is written to the RORIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = No interrupt. 0x1 = The receive FIFO has overflowed

23.5.8 SSIMIS Register (Offset = 0x1C) [reset = 0x0]

QSSI Masked Interrupt Status (SSIMIS), offset 0x01C

The SSIMIS register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

SSIMIS is shown in [Figure 23-17](#) and described in [Table 23-13](#).

Return to [Summary Table](#).

Figure 23-17. SSIMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	EOTMIS	DMATXMIS	DMARXMIS	TXMIS	RXMIS	RTMIS	RORMIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 23-13. SSIMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6	EOTMIS	R	0x0	End of Transmit Masked Interrupt Status. This bit is cleared when a 1 is written to the EOTIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the transmission of the last data bit.
5	DMATXMIS	R	0x0	QSSI Transmit DMA Masked Interrupt Status. This bit is cleared when a 1 is written to the DMATXIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the completion of the transmit DMA.
4	DMARXMIS	R	0x0	QSSI Receive DMA Masked Interrupt Status. This bit is cleared when a 1 is written to the DMARXIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the completion of the receive DMA.
3	TXMIS	R	0x0	QSSI Transmit FIFO Masked Interrupt Status. This bit is cleared when the transmit FIFO is more than half empty. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the transmit FIFO being half empty or less.
2	RXMIS	R	0x0	QSSI Receive FIFO Masked Interrupt Status. This bit is cleared when the receive FIFO is less than half full. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the receive FIFO being half full or more.

Table 23-13. SSIMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	RTMIS	R	0x0	QSSI Receive Time-Out Masked Interrupt Status. This bit is cleared when a 1 is written to the RTIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the receive time out.
0	RORMIS	R	0x0	QSSI Receive Overrun Masked Interrupt Status. This bit is cleared when a 1 is written to the RORIC bit in the SSI Interrupt Clear (SSIICR) register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the receive FIFO overflowing.

23.5.9 SSICR Register (Offset = 0x20) [reset = 0x0]

QSSI Interrupt Clear (SSICR), offset 0x020

The SSICR register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSICR is shown in [Figure 23-18](#) and described in [Table 23-14](#).

Return to [Summary Table](#).

Figure 23-18. SSICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED	EOTIC	DMATXIC	DMARXIC	RESERVED		RTIC	RORIC
R-0x0	W1C-0x0	W1C-0x0	W1C-0x0	R-0x0		W1C-0x0	W1C-0x0

Table 23-14. SSICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0x0	
6	EOTIC	W1C	0x0	End of Transmit Interrupt Clear. Writing a 1 to this bit clears the EOTRIS bit in the SSIRIS register and the EOTMIS bit in the SSIMIS register.
5	DMATXIC	W1C	0x0	QSSI Transmit DMA Interrupt Clear. Writing a 1 to this bit clears the DMATXRIS bit in the SSIRIS register and the DMATXMIS bit in the SSIMIS register.
4	DMARXIC	W1C	0x0	QSSI Receive DMA Interrupt Clear. Writing a 1 to this bit clears the DMARXRIS bit in the SSIRIS register and the DMARXMIS bit in the SSIMIS register.
3-2	RESERVED	R	0x0	
1	RTIC	W1C	0x0	QSSI Receive Time-Out Interrupt Clear. Writing a 1 to this bit clears the RTRIS bit in the SSIRIS register and the RTMIS bit in the SSIMIS register.
0	RORIC	W1C	0x0	QSSI Receive Overrun Interrupt Clear. Writing a 1 to this bit clears the RORRIS bit in the SSIRIS register and the RORMIS bit in the SSIMIS register.

23.5.10 SSIDMACTL Register (Offset = 0x24) [reset = 0x0]

QSSI DMA Control (SSIDMACTL), offset 0x024

The SSIDMACTL register is the μ DMA control register.

SSIDMACTL is shown in [Figure 23-19](#) and described in [Table 23-15](#).

Return to [Summary Table](#).

Figure 23-19. SSIDMACTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						TXDMAE	RXDMAE
R-0x0						R/W-0x0	R/W-0x0

Table 23-15. SSIDMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	TXDMAE	R/W	0x0	Transmit DMA Enable 0x0 = μ DMA for the transmit FIFO is disabled. 0x1 = μ DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0x0	Receive DMA Enable 0x0 = μ DMA for the receive FIFO is disabled. 0x1 = μ DMA for the receive FIFO is enabled.

23.5.11 SSIPP Register (Offset = 0xFC0) [reset = 0xD]

QSSI Peripheral Properties (SSIPP), offset 0xFC0

The SSIPP register provides information regarding the properties of the QSSI module.

SSIPP is shown in [Figure 23-20](#) and described in [Table 23-16](#).

Return to [Summary Table](#).

Figure 23-20. SSIPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				FSSHLD FRM	MODE		HSCLK
R-0x0				R-0x1	R-0x2		R-0x1

Table 23-16. SSIPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	FSSHLD FRM	R	0x1	SSInFss Hold Frame Capability 0x0 = SSInFss Hold Frame capability disabled. 0x1 = SSInFss Hold Frame capability enabled.
2-1	MODE	R	0x2	Mode of Operation Indicates what QSSI functionality is supported. 0x0 = Legacy SSI mode 0x1 = Legacy mode, Advanced SSI mode and Bi-SSI mode enabled. 0x2 = Legacy mode, Advanced mode, Bi-SSI and Quad-SSI mode enabled. 0x3 = Reserved
0	HSCLK	R	0x1	High Speed Capability 0x0 = High Speed clock capability disabled. 0x1 = High speed clock capability enabled.

23.5.12 SSICC Register (Offset = 0xFC8) [reset = 0x0]

QSSI Clock Configuration (SSICC), offset 0xFC8

The SSICC register controls the baud clock source for the QSSI module.

NOTE: If ALTCLK is used for the QSSI baud clock, the system clock frequency must be at least twice that of the ALTCLK programmed value in Run mode.

SSICC is shown in [Figure 23-21](#) and described in [Table 23-17](#).

Return to [Summary Table](#).

Figure 23-21. SSICC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												CS			
R-0x0																												R/W-0x0			

Table 23-17. SSICC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	CS	R/W	0x0	QSSI Baud Clock Source 0x0 = System clock (based on clock source and divisor factor programmed in RSCLKCFG register in the System Control Module) 0x1-0x4 = Reserved 0x5 = Alternate clock source as defined by ALTCLKCFG register in System Control Module. 0x6-0xF = Reserved

23.5.13 SSIPeriphID4 Register (Offset = 0xFD0) [reset = 0x0]

QSSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID4 is shown in [Figure 23-22](#) and described in [Table 23-18](#).

Return to [Summary Table](#).

Figure 23-22. SSIPeriphID4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID4							
R-0x0																								R-0x0							

Table 23-18. SSIPeriphID4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID4	R	0x0	QSSI Peripheral ID Register [7:0]. Can be used by software to identify the presence of this peripheral.

23.5.14 SSIPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]

QSSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID5 is shown in [Figure 23-23](#) and described in [Table 23-19](#).

Return to [Summary Table](#).

Figure 23-23. SSIPeriphID5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID5							
R-0x0																								R-0x0							

Table 23-19. SSIPeriphID5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID5	R	0x0	QSSI Peripheral ID Register [15:8]. Can be used by software to identify the presence of this peripheral.

23.5.15 SSIPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]

QSSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID6 is shown in [Figure 23-24](#) and described in [Table 23-20](#).

Return to [Summary Table](#).

Figure 23-24. SSIPeriphID6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID6							
R-0x0																								R-0x0							

Table 23-20. SSIPeriphID6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID6	R	0x0	QSSI Peripheral ID Register [23:16]. Can be used by software to identify the presence of this peripheral.

23.5.16 SSIPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]

QSSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID7 is shown in [Figure 23-25](#) and described in [Table 23-21](#).

Return to [Summary Table](#).

Figure 23-25. SSIPeriphID7 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID7							
R-0x0																								R-0x0							

Table 23-21. SSIPeriphID7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID7	R	0x0	QSSI Peripheral ID Register [31:24]. Can be used by software to identify the presence of this peripheral.

23.5.17 SSIPeriphID0 Register (Offset = 0xFE0) [reset = 0x22]

QSSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID0 is shown in [Figure 23-26](#) and described in [Table 23-22](#).

Return to [Summary Table](#).

Figure 23-26. SSIPeriphID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID0							
R-0x0																								R-0x22							

Table 23-22. SSIPeriphID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID0	R	0x22	QSSI Peripheral ID Register [7:0]. Can be used by software to identify the presence of this peripheral.

23.5.18 SSIPeriphID1 Register (Offset = 0xFE4) [reset = 0x0]

QSSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID1 is shown in [Figure 23-27](#) and described in [Table 23-23](#).

Return to [Summary Table](#).

Figure 23-27. SSIPeriphID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID1							
R-0x0																								R-0x0							

Table 23-23. SSIPeriphID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID1	R	0x0	QSSI Peripheral ID Register [15:8]. Can be used by software to identify the presence of this peripheral.

23.5.19 SSIPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]

QSSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID2 is shown in [Figure 23-28](#) and described in [Table 23-24](#).

Return to [Summary Table](#).

Figure 23-28. SSIPeriphID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID2							
R-0x0																								R-0x18							

Table 23-24. SSIPeriphID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID2	R	0x18	QSSI Peripheral ID Register [23:16]. Can be used by software to identify the presence of this peripheral.

23.5.20 SSIPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]

QSSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The SSIPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

SSIPeriphID3 is shown in [Figure 23-29](#) and described in [Table 23-25](#).

Return to [Summary Table](#).

Figure 23-29. SSIPeriphID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID3							
R-0x0																								R-0x1							

Table 23-25. SSIPeriphID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID3	R	0x1	QSSI Peripheral ID Register [31:24]. Can be used by software to identify the presence of this peripheral.

23.5.21 SSIPCellID0 Register (Offset = 0xFF0) [reset = 0xD]

QSSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The SSIPCellIDn registers are hard-coded, and the fields within the register determine the reset value.

SSIPCellID0 is shown in [Figure 23-30](#) and described in [Table 23-26](#).

Return to [Summary Table](#).

Figure 23-30. SSIPCellID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID0							
R-0x0																								R-0xD							

Table 23-26. SSIPCellID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID0	R	0xD	QSSI PrimeCell ID Register [7:0]. Provides software a standard cross-peripheral identification system.

23.5.22 SSIPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]

QSSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The SSIPCellIDn registers are hard-coded, and the fields within the register determine the reset value.

SSIPCellID1 is shown in [Figure 23-31](#) and described in [Table 23-27](#).

Return to [Summary Table](#).

Figure 23-31. SSIPCellID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID1							
R-0x0																								R-0xF0							

Table 23-27. SSIPCellID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID1	R	0xF0	QSSI PrimeCell ID Register [15:8]. Provides software a standard cross-peripheral identification system.

23.5.23 SSIPCellID2 Register (Offset = 0xFF8) [reset = 0x5]

QSSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The SSIPCellIDn registers are hard-coded, and the fields within the register determine the reset value.

SSIPCellID2 is shown in [Figure 23-32](#) and described in [Table 23-28](#).

Return to [Summary Table](#).

Figure 23-32. SSIPCellID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID2							
R-0x0																								R-0x5							

Table 23-28. SSIPCellID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID2	R	0x5	QSSI PrimeCell ID Register [23:16]. Provides software a standard cross-peripheral identification system.

23.5.24 SSIPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]

QSSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The SSIPCellIDn registers are hard-coded, and the fields within the register determine the reset value.

SSIPCellID3 is shown in [Figure 23-33](#) and described in [Table 23-29](#).

Return to [Summary Table](#).

Figure 23-33. SSIPCellID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID3							
R-0x0																								R-0xB1							

Table 23-29. SSIPCellID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID3	R	0xB1	QSSI PrimeCell ID Register [31:24]. Provides software a standard cross-peripheral identification system.

Quadrature Encoder Interface (QEI)

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, users can track the position, direction of rotation, and speed. In addition, a third channel, or index signal, can be used to reset the position counter.

Topic	Page
24.1 Introduction	1568
24.2 Block Diagram	1568
24.3 Functional Description	1570
24.4 Initialization and Configuration	1572
24.5 QEI Registers	1573

24.1 Introduction

The quadrature encoder interface (QEI) module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

The MSP432E4 microcontroller includes one QEI module with the following features:

- Position integrator that tracks the encoder position
- Programmable noise filter on the inputs
- Velocity capture using built-in timer
- The input frequency of the QEI inputs may be as high as 1/4 of the system frequency (for example, 30 MHz for a 120-MHz system)
- Interrupt generation on:
 - Index pulse
 - Velocity-timer expiration
 - Direction change
 - Quadrature error detection

24.2 Block Diagram

[Figure 24-1](#) shows the internal block diagram of a QEI module. The PhA and PhB inputs shown in [Figure 24-1](#) are the internal signals that enter the Quadrature Encoder after the external signals, PhAn and PhBn, have passed through inversion and swapping logic, shown in [Figure 24-2](#). The QEI module has the option of inverting and/or swapping the incoming signals.

NOTE: Any references in this chapter to PhA and PhB refer to the internal PhA and PhB inputs that enter the Quadrature Encoder after the external signals, PhAn and PhBn, have passed through inversion and swapping logic that is enabled through the QEI Control (QEICTL) register.

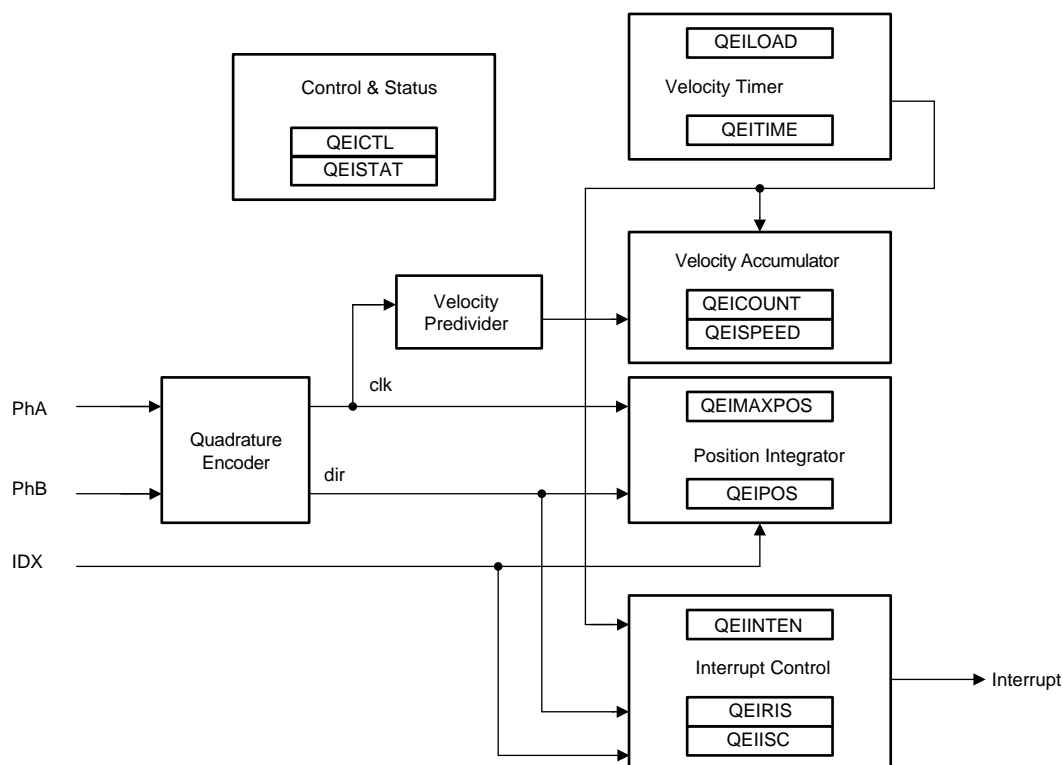


Figure 24-1. QEI Block Diagram

Figure 24-2 shows the logic that is provided to allow the PhAn and PhBn signals to be inverted and/or swapped.

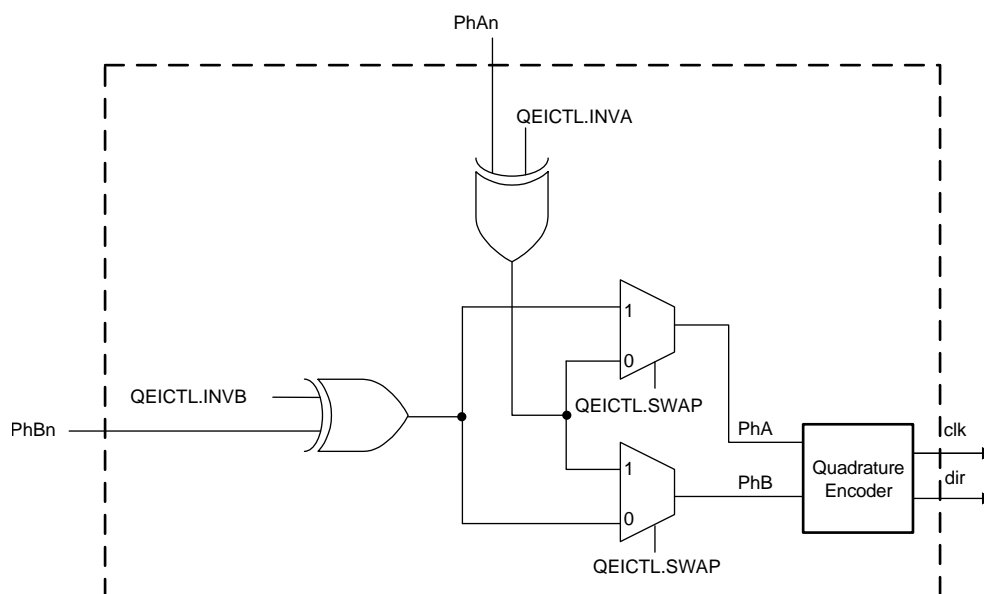


Figure 24-2. QEI Input Signal Logic

24.3 Functional Description

The QEI module interprets the two-bit gray code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

The position integrator and velocity capture can be independently enabled, though the position integrator must be enabled before the velocity capture can be enabled. The two phase signals, PhAn and PhBn, can be swapped before being interpreted by the QEI module to change the meaning of forward and backward and to correct for miswiring of the system. Alternatively, the phase signals can be interpreted as a clock and direction signal as output by some encoders.

The QEI module input signals have a digital noise filter on them that can be enabled to prevent spurious operation. The noise filter requires that the inputs be stable for a specified number of consecutive clock cycles before updating the edge detector. The filter is enabled by the FILTEN bit in the QEI Control (QEICTL) register. The frequency of the input update is programmable using the FILTCNT bit field in the QEICTL register.

The QEI module supports two modes of signal operation: quadrature phase mode and clock and direction mode. In quadrature phase mode, the encoder produces two clocks that are 90 degrees out of phase; the edge relationship is used to determine the direction of rotation. In clock/direction mode, the encoder produces a clock signal to indicate steps and a direction signal to indicate the direction of rotation. This mode is determined by the SIGMODE bit of the QEICTL register (see [Section 24.5.1](#)).

When the QEI module is set to use the quadrature phase mode (SIGMODE bit is clear), the capture mode for the position integrator can be set to update the position counter on every edge of the PhA signal or to update on every edge of both PhA and PhB. Updating the position counter on every PhA and PhB edge provides more positional resolution at the cost of less range in the positional counter.

When edges on PhA lead edges on PhB, the position counter is incremented. When edges on PhB lead edges on PhA, the position counter is decremented. When a rising and falling edge pair is seen on one of the phases without any edges on the other, the direction of rotation has changed.

The positional counter is automatically reset on one of two conditions: sensing the index pulse or reaching the maximum position value. The reset mode is determined by the RESMODE bit of the QEICTL register.

When RESMODE is set, the positional counter is reset when the index pulse is sensed. This mode limits the positional counter to the values [0:N-1], where N is the number of phase edges in a full revolution of the encoder wheel. The QEI Maximum Position (QEIMAXPOS) register must be programmed with N-1 so that the reverse direction from position 0 can move the position counter to N-1. In this mode, the position register contains the absolute position of the encoder relative to the index (or home) position once an index pulse has been seen.

When RESMODE is clear, the positional counter is constrained to the range [0:M], where M is the programmable maximum value. The index pulse is ignored by the positional counter in this mode.

Velocity capture uses a configurable timer and a count register. The timer counts the number of phase edges (using the same configuration as for the position integrator) in a given time period. The edge count from the previous time period is available to the controller via the QEI Velocity (QEISPEED) register, while the edge count for the current time period is being accumulated in the QEI Velocity Counter (QEICOUNT) register. As soon as the current time period is complete, the total number of edges counted in that time period is made available in the QEISPEED register (overwriting the previous value), the QEICOUNT register is cleared, and counting commences on a new time period. The number of edges counted in a given time period is directly proportional to the velocity of the encoder.

[Figure 24-3](#) shows how the MSP432E4 quadrature encoder converts the phase input signals into clock pulses, the direction signal, and how the velocity predivider operates (in Divide by 4 mode).

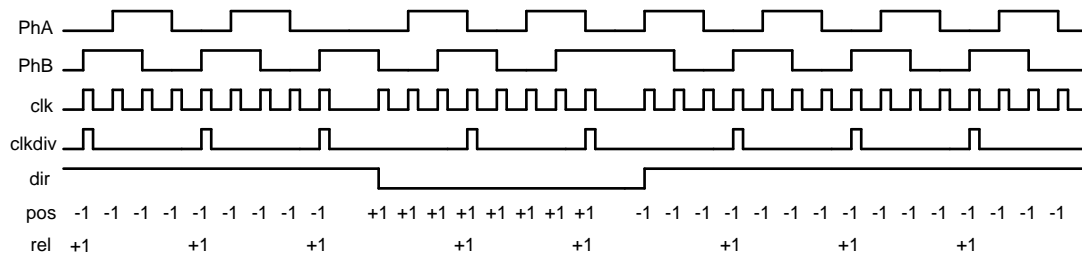


Figure 24-3. Quadrature Encoder and Velocity Predivider Operation

The period of the timer is configurable by specifying the load value for the timer in the QEI Timer Load (QEILOAD) register. When the timer reaches zero, an interrupt can be triggered, and the hardware reloads the timer with the QEILOAD value and continues to count down. Lower encoder speeds require a longer timer period to capture enough edges to have a meaningful result. At higher encoder speeds, both a shorter timer period and/or the velocity predivider can be used.

Equation 66 converts the velocity counter value into an rpm value:

$$\text{rpm} = (\text{clock} \times (2^{\text{VELDIV}}) \times \text{SPEED} \times 60) / (\text{LOAD} \times \text{ppr} \times \text{edges})$$

where

- clock is the controller clock rate
- ppr is the number of pulses per revolution of the physical encoder
- edges is 2 or 4, based on the capture mode set in the QEICTL register (2 for CAPMODE clear and 4 for CAPMODE set)

(66)

For example, consider a motor running at 600 rpm. A 2048 pulse per revolution quadrature encoder is attached to the motor, producing 8192 phase edges per revolution. With a velocity predivider of $\div 1$ (VELDIV is clear) and clocking on both PhA and PhB edges, this results in 81920 pulses per second (the motor turns 10 times per second). If the timer were clocked at 10000 Hz, and the load value was 2500 (1/4 of a second), it would count 20480 pulses per update. Using the above equation:

$$\text{rpm} = (10000 \times 1 \times 20480 \times 60) \div (2500 \times 2048 \times 4) = 600 \text{ rpm}$$

(67)

Now, consider that the motor is sped up to 3000 rpm. This results in 409600 pulses per second, or 102400 every 1/4 of a second. Again, the above equation gives:

$$\text{rpm} = (10000 \times 1 \times 102400 \times 60) \div (2500 \times 2048 \times 4) = 3000 \text{ rpm}$$

(68)

Care must be taken when evaluating this equation because intermediate values may exceed the capacity of a 32-bit integer. In the above examples, the clock is 10000 and the divisor is 2500; both could be predivided by 100 (at compile time if they are constants) and therefore be 100 and 25. In fact, if they were compile-time constants, they could also be reduced to a simple multiply by 4, cancelled by the $\div 4$ for the edge-count factor.

NOTE: Reducing constant factors at compile time is the best way to control the intermediate values of this equation and reduce the processing requirement of computing this equation.

The division can be avoided by selecting a timer load value such that the divisor is a power of 2; a simple shift can therefore be done in place of the division. For encoders with a power of 2 pulses per revolution, the load value can be a power of 2. For other encoders, a load value must be selected such that the product is very close to a power of 2. For example, a 100 pulse-per-revolution encoder could use a load value of 82, resulting in 32800 as the divisor, which is 0.09% above 2^{14} . In this case a shift by 15 would be an adequate approximation of the divide in most cases. If absolute accuracy were required, the microcontroller's divide instruction could be used.

The QEI module can produce a controller interrupt on several events: phase error, direction change, reception of the index pulse, and expiration of the velocity timer. Standard masking, raw interrupt status, interrupt status, and interrupt clear capabilities are provided.

24.4 Initialization and Configuration

Perform these steps to configure the QEI module to read back an absolute position:

1. Enable the QEI clock using the RCGCQEI register in the System Control module (see [Section 4.2.100](#)).
2. Enable the clock to the appropriate GPIO module via the RCGCGPIO register in the System Control module (see [Section 4.2.87](#)).
3. In the GPIO module, enable the appropriate pins for their alternate function using the GPIOAFSEL register. To determine which GPIOs to configure, see the device-specific data sheet.
4. Configure the PMCN fields in the GPIOPCTL register to assign the QEI signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).
5. Configure the quadrature encoder to capture edges on both signals and maintain an absolute position by resetting on index pulses. A 1000-line encoder with four edges per line, results in 4000 pulses per revolution; therefore, set the maximum position to 3999 (0xF9F) as the count is zero-based.
 - a. Write the QEICTL register with the value of 0x0000.0018.
 - b. Write the QEIMAXPOS register with the value of 0x0000.0F9F.
6. Enable the quadrature encoder by setting bit 0 of the QEICTL register.

NOTE: After the QEI module has been enabled by setting the ENABLE bit in the QEICTL register, it cannot be disabled. The only way to clear the ENABLE bit is to reset the module using the Quadrature Encoder Interface Software Reset (SRQEI) register.

7. Delay until the encoder position is required.
8. Read the encoder position by reading the QEI Position (QEIPPOS) register value.

NOTE: If the application requires the quadrature encoder to have a specific initial position, this value must be programmed in the QEIPPOS register after the quadrature encoder has been enabled by setting the ENABLE bit in the QEICTL register.

24.5 QEI Registers

[Table 24-1](#) lists the memory-mapped registers for the QEI. All register offset addresses not listed in [Table 24-1](#) should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the base address of the QEI module: 0x4002C000.

The QEI module clock must be enabled before the registers can be programmed (see [Section 4.2.100](#)). There must be a delay of 3 system clocks after the QEI module clock is enabled before any QEI module registers are accessed.

Table 24-1. QEI Registers

Offset	Acronym	Register Name	Section
0x0	QEICTL	QEI Control	Section 24.5.1
0x4	QEISTAT	QEI Status	Section 24.5.2
0x8	QEIPPOS	QEI Position	Section 24.5.3
0xC	QEIMAXPOS	QEI Maximum Position	Section 24.5.4
0x10	QEILOAD	QEI Timer Load	Section 24.5.5
0x14	QEITIME	QEI Timer	Section 24.5.6
0x18	QEICOUNT	QEI Velocity Counter	Section 24.5.7
0x1C	QEISPEED	QEI Velocity	Section 24.5.8
0x20	QEIINTEN	QEI Interrupt Enable	Section 24.5.9
0x24	QEIRIS	QEI Raw Interrupt Status	Section 24.5.10
0x28	QEISC	QEI Interrupt Status and Clear	Section 24.5.11

Complex bit access types are encoded to fit into small table cells. [Table 24-2](#) shows the codes that are used for access types in this section.

Table 24-2. QEI Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

24.5.1 QEICTL Register (Offset = 0x0) [reset = 0x0]

QEI Control (QEICTL)

This register contains the configuration of the QEI module. Separate enables are provided for the quadrature encoder and the velocity capture blocks; the quadrature encoder must be enabled in order to capture the velocity, but the velocity does not need to be captured in applications that do not need it. The phase signal interpretation, phase swap, Position Update mode, Position Reset mode, and velocity predivider are all set through this register.

QEICTL is shown in [Figure 24-4](#) and described in [Table 24-3](#).

Return to [Summary Table](#).

Figure 24-4. QEICTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED				FILTCNT			
R-0x0				R/W-0x0			
15	14	13	12	11	10	9	8
RESERVED		FILTEN	STALLEN	INVI	INVB	INVA	VELDIV
R-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
VELDIV		VELEN	RESMODE	CAPMODE	SIGMODE	SWAP	ENABLE
R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 24-3. QEICTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0x0	
19-16	FILTCNT	R/W	0x0	Input Filter Prescale Count. This field controls the frequency of the input update. When this field is clear, the input is sampled after 2 system clocks. When this field is 0x1, the input is sampled after 3 system clocks. Similarly, when this field is 0xF, the input is sampled after 17 clocks.
15-14	RESERVED	R	0x0	
13	FILTEN	R/W	0x0	Enable Input Filter. 0x0 = The QEI inputs are not filtered. 0x1 = Enables the digital noise filter on the QEI input signals. Inputs must be stable for 3 consecutive clock edges before the edge detector is updated.
12	STALLEN	R/W	0x0	Stall QEI. 0x0 = The QEI module does not stall when the microcontroller is stopped by a debugger. 0x1 = The QEI module stalls when the microcontroller is stopped by a debugger.
11	INVI	R/W	0x0	Invert Index Pulse. 0x0 = No effect. 0x1 = Inverts the IDX input.
10	INVB	R/W	0x0	Invert PhB. 0x0 = No effect. 0x1 = Inverts the PhBn input.
9	INVA	R/W	0x0	Invert PhA. 0x0 = No effect. 0x1 = Inverts the PhAn input.

Table 24-3. QEICTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8-6	VELDIV	R/W	0x0	Predivide Velocity. This field defines the predivider of the input quadrature pulses before being applied to the QEICOUNT accumulator. 0x0 = /1 0x1 = /2 0x2 = /4 0x3 = /8 0x4 = /16 0x5 = /32 0x6 = /64 0x7 = /128
5	VELEN	R/W	0x0	Capture Velocity. 0x0 = No effect. 0x1 = Enables capture of the velocity of the quadrature encoder.
4	RESMODE	R/W	0x0	Reset Mode. 0x0 = The position counter is reset when it reaches the maximum as defined by the MAXPOS field in the QEIMAXPOS register. 0x1 = The position counter is reset when the index pulse is captured.
3	CAPMODE	R/W	0x0	Capture Mode. When SIGMODE = 1, the CAPMODE setting is not applicable and is reserved. 0x0 = Only the PhA edges are counted. 0x1 = The PhA and PhB edges are counted, providing twice the positional resolution but half the range.
2	SIGMODE	R/W	0x0	Signal Mode. 0x0 = The internal PhA and PhB signals operate as quadrature phase signals. 0x1 = The internal PhA input operates as the clock (CLK) signal and the internal PhB input operates as the direction (DIR) signal.
1	SWAP	R/W	0x0	Swap Signals. Note if the INVA or INVB bit are set, the inversion of the signals occur prior to the swap. 0x0 = No effect. 0x1 = Swaps the PhAn and PhBn signals.
0	ENABLE	R/W	0x0	Enable QEI. After the QEI module has been enabled by setting the ENABLE bit, it cannot be disabled. The only way to clear the ENABLE bit is to reset the module using the Quadrature Encoder Interface Software Reset (SRQEI) register. 0x0 = No effect. 0x1 = Enables the quadrature encoder module.

24.5.2 QEISTAT Register (Offset = 0x4) [reset = 0x0]

QEI Status (QEISTAT)

This register provides status about the operation of the QEI module.

QEISTAT is shown in [Figure 24-5](#) and described in [Table 24-4](#).

Return to [Summary Table](#).

Figure 24-5. QEISTAT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED						DIRECTION	ERROR
R-0x0						R-0x0	R-0x0

Table 24-4. QEISTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0x0	
1	DIRECTION	R	0x0	Direction of Rotation. Indicates the direction the encoder is rotating. 0x0 = The encoder is rotating forward. 0x1 = The encoder is rotating in reverse.
0	ERROR	R	0x0	Error Detected. 0x0 = No error. 0x1 = An error was detected in the gray code sequence (that is, both signals changing at the same time).

24.5.3 QEIPOS Register (Offset = 0x8) [reset = 0x0]

QEI Position (QEIPOS)

This register contains the current value of the position integrator. The value is updated by the status of the QEI phase inputs and can be set to a specific value by writing to it.

QEIPOS is shown in [Figure 24-6](#) and described in [Table 24-5](#).

Return to [Summary Table](#).

Figure 24-6. QEIPOS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSITION																															
R/W-0x0																															

Table 24-5. QEIPOS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	POSITION	R/W	0x0	Current Position Integrator Value. The current value of the position integrator.

24.5.4 QEIMAXPOS Register (Offset = 0xC) [reset = 0x0]

QEI Maximum Position (QEIMAXPOS)

This register contains the maximum value of the position integrator. When moving forward, the position register resets to zero when it increments past this value. When moving in reverse, the position register resets to this value when it decrements from zero.

QEIMAXPOS is shown in [Figure 24-7](#) and described in [Table 24-6](#).

[Return to Summary Table.](#)

Figure 24-7. QEIMAXPOS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXPOS																															
R/W-0x0																															

Table 24-6. QEIMAXPOS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	MAXPOS	R/W	0x0	Maximum Position Integrator Value. The maximum value of the position integrator.

24.5.5 QEILoad Register (Offset = 0x10) [reset = 0x0]

QEI Timer Load (QEILoad)

This register contains the load value for the velocity timer. Because this value is loaded into the timer on the clock cycle after the timer is zero, this value should be one less than the number of clocks in the desired period. So, for example, to have 2000 decimal clocks per timer period, this register should contain 1999 decimal.

QEILoad is shown in [Figure 24-8](#) and described in [Table 24-7](#).

Return to [Summary Table](#).

Figure 24-8. QEILoad Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD																															
R/W-0x0																															

Table 24-7. QEILoad Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LOAD	R/W	0x0	Velocity Timer Load Value. The load value for the velocity timer.

24.5.6 QEITIME Register (Offset = 0x14) [reset = 0x0]

QEI Timer (QEITIME)

This register contains the current value of the velocity timer. This counter does not increment when the VELEN bit in the QEICTL register is clear.

QEITIME is shown in [Figure 24-9](#) and described in [Table 24-8](#).

Return to [Summary Table](#).

Figure 24-9. QEITIME Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIME																															
R-0x0																															

Table 24-8. QEITIME Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TIME	R	0x0	Velocity Timer Current Value. The current value of the velocity timer.

24.5.7 QEICOUNT Register (Offset = 0x18) [reset = 0x0]

QEI Velocity Counter (QEICOUNT)

This register contains the running count of velocity pulses for the current time period. Because this count is a running total, the time period to which it applies cannot be known with precision (that is, a read of this register does not necessarily correspond to the time returned by the QEITIME register because there is a small window of time between the two reads, during which either value may have changed). The QEISPEED register should be used to determine the actual encoder velocity; this register is provided for information purposes only. This counter does not increment when the VELEN bit in the QEICTL register is clear.

QEICOUNT is shown in [Figure 24-10](#) and described in [Table 24-9](#).

Return to [Summary Table](#).

Figure 24-10. QEICOUNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT																															
R-0x0																															

Table 24-9. QEICOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COUNT	R	0x0	Velocity Pulse Count. The running total of encoder pulses during this velocity timer period.

24.5.8 QEISPEED Register (Offset = 0x1C) [reset = 0x0]

QEI Velocity (QEISPEED)

This register contains the most recently measured velocity of the quadrature encoder. This value corresponds to the number of velocity pulses counted in the previous velocity timer period. This register does not update when the VELEN bit in the QEICTL register is clear.

QEISPEED is shown in [Figure 24-11](#) and described in [Table 24-10](#).

Return to [Summary Table](#).

Figure 24-11. QEISPEED Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPEED																															
R-0x0																															

Table 24-10. QEISPEED Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SPEED	R	0x0	Velocity. The measured speed of the quadrature encoder in pulses per period.

24.5.9 QEINTEN Register (Offset = 0x20) [reset = 0x0]

QEI Interrupt Enable (QEINTEN)

This register contains enables for each of the QEI module interrupts. An interrupt is asserted to the interrupt controller if the corresponding bit in this register is set.

QEINTEN is shown in [Figure 24-12](#) and described in [Table 24-11](#).

Return to [Summary Table](#).

Figure 24-12. QEINTEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				INTERROR	INTDIR	INTTIMER	INTINDEX
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 24-11. QEINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	INTERROR	R/W	0x0	Phase Error Interrupt Enable. The INTERROR bit is only applicable when the QEI is operating in quadrature phase mode (SIGMODE =0) and should be masked when SIGMODE =1. 0x0 = The INTERROR interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the INTERROR bit in the QEIRIS register is set.
2	INTDIR	R/W	0x0	Direction Change Interrupt Enable. 0x0 = The INTDIR interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the INTDIR bit in the QEIRIS register is set.
1	INTTIMER	R/W	0x0	Timer Expires Interrupt Enable. 0x0 = The INTTIMER interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the INTTIMER bit in the QEIRIS register is set.
0	INTINDEX	R/W	0x0	Index Pulse Detected Interrupt Enable. 0x0 = The INTINDEX interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the INTINDEX bit in the QEIRIS register is set.

24.5.10 QEIRIS Register (Offset = 0x24) [reset = 0x0]

QEI Raw Interrupt Status (QEIRIS)

This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (configured through the QEINTEN register). If a bit is set, the latched event has occurred; if a bit is clear, the event in question has not occurred.

QEIRIS is shown in [Figure 24-13](#) and described in [Table 24-12](#).

Return to [Summary Table](#).

Figure 24-13. QEIRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				INTERROR	INTDIR	INTTIMER	INTINDEX
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0

Table 24-12. QEIRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	INTERROR	R	0x0	Phase Error Detected. The INTERROR bit is only applicable when the QEI is operating in quadrature phase mode (SIGMODE =0). This bit is cleared by writing a 1 to the INTERROR bit in the QEISC register. 0x0 = An interrupt has not occurred. 0x1 = A phase error has been detected.
2	INTDIR	R	0x0	Direction Change Detected. This bit is cleared by writing a 1 to the INTDIR bit in the QEISC register. 0x0 = An interrupt has not occurred. 0x1 = The rotation direction has changed
1	INTTIMER	R	0x0	Velocity Timer Expired. This bit is cleared by writing a 1 to the INTTIMER bit in the QEISC register. 0x0 = An interrupt has not occurred. 0x1 = The velocity timer has expired.
0	INTINDEX	R	0x0	Index Pulse Asserted. This bit is cleared by writing a 1 to the INTINDEX bit in the QEISC register. 0x0 = An interrupt has not occurred. 0x1 = The index pulse has occurred.

24.5.11 QEISC Register (Offset = 0x28) [reset = 0x0]

QEI Interrupt Status and Clear (QEISC)

This register provides the current set of interrupt sources that are asserted to the controller. If a bit is set, the latched event has occurred and is enabled to generate an interrupt; if a bit is clear the event in question has not occurred or is not enabled to generate an interrupt. This register is RW1C; writing a 1 to a bit position clears the bit and the corresponding interrupt reason.

QEISC is shown in [Figure 24-14](#) and described in [Table 24-13](#).

Return to [Summary Table](#).

Figure 24-14. QEISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				INTERROR	INTDIR	INTTIMER	INTINDEX
R-0x0				R/W1C-0x0	R/W1C-0x0	R/W1C-0x0	R/W1C-0x0

Table 24-13. QEISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	INTERROR	R/W1C	0x0	Phase Error Interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTERROR bit in the QEIRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTERROR bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.
2	INTDIR	R/W1C	0x0	Direction Change Interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTDIR bit in the QEIRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTDIR bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.
1	INTTIMER	R/W1C	0x0	Velocity Timer Expired Interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTTIMER bit in the QEIRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTTIMER bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.
0	INTINDEX	R/W1C	0x0	Index Pulse Interrupt. This bit is cleared by writing a 1. Clearing this bit also clears the INTINDEX bit in the QEIRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The INTINDEX bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.

SHA/MD5 Accelerator

The SHA/MD5 module provides hardware-accelerated hash functions and can run:

- MD5 message digest algorithm developed by Ron Rivest in 1991
- SHA-1 algorithm compliant with the [FIPS 180-3 standard](#)
- SHA-2 (SHA-224 and SHA-256) algorithm compliant with the [FIPS 180-3 standard](#)
- Hash message authentication code (HMAC) operation

The algorithms produce a condensed representation of a message or a data file, called digest or signature, which can then be used to verify the message integrity.

- Hashing of 0 to $2^{33} - 2$ bytes of data (of which $2^{32} - 1$ bytes are in one pass) using the MD5, SHA-1, SHA-224, or SHA-256 hash algorithm (byte granularity only, no support for bit granularity)
- Automatic HMAC key preprocessing for HMAC keys up to 64 bytes
- Host-assisted HMAC key preprocessing for HMAC keys larger than 64 bytes
- HMAC from precomputes (inner and outer digest) for improved performance on small blocks
- Supports μ DMA operation for data and context in as well as result out transfers
- Supports interrupt to read the digest (signature)

Topic	Page
25.1 SHA/MD5 Functional Description	1587
25.2 SHA/MD5 Registers	1600
25.3 SHA/MD5 μDMA Registers	1615

25.1 SHA/MD5 Functional Description

25.1.1 SHA/MD5 Block Diagram

Figure 25-1 shows the module architecture, which consists of four primary blocks: the Hash/HMAC engine, configuration registers, the interface to μ DMA, and the interrupt handler.

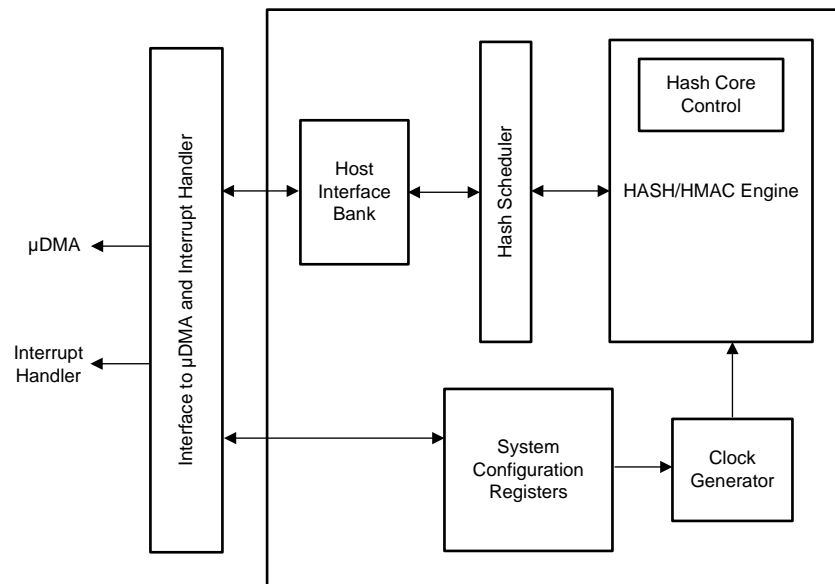


Figure 25-1. SHA/MD5 Module Block Diagram

25.1.1.1 Configuration Registers

The configuration registers contain the following global control and status registers for the SHA/MD5 module:

- System control register that controls the mode of operation (SHA_SYSCONFIG register)
- μ DMA interrupt control registers (SHA_DMAIM, SHA_DMARIS, SHA_DMAMIS, and SHA_DMAIC registers, which reside in the Encryption Control Base address space)
- Interrupt status register (SHA_IRQSTATUS register)
- Enable register (SHA_IRQENABLE register)

25.1.1.2 Hash/HMAC Engine

The Hash/HMAC engine performs the SHA-1, SHA-2, or MD5 hash computation. When loaded with a data block, and optionally an intermediate digest, it independently performs the hash computation (64 or 80 rounds, depending on the algorithm) on that data block.

It can also start from the specified initial digest values instead of a loaded intermediate. Furthermore, it can perform the IPAD and OPAD XORs for MAC operations. The hash core does not perform any hash padding; this is performed in the Host Interface Block, where the data input registers are located. A loaded data block must always be a full 64 bytes (512 bits) long.

25.1.1.3 Hash Core Control

When the hash core is idle or done, a new hash operation can be started. Any additional information needed by the hash core (mode of operation, data to process, input digest if not starting from algorithm constants or continuing) must be provided by programming the SHA registers before the core can accept the operation.

25.1.1.4 Host Interface Bank

The Host Interface Bank can access the hash core for hashing individual 64-byte hash blocks. The Host Interface Bank contains registers such as the data FIFO (SHA Data n Input (SHA_DATA_n_IN) registers), the SHA Inner Digest x (SHA_IDIGEST_X) registers, and several control and status registers.

The Host Interface Block contains all relevant control logic for performing hash and HMAC computations on large (that is, larger than one hash block) blocks of data, including hash padding, final hash, and outer hash. It provides the necessary flow control to the SHA μ DMA and interrupt interface.

25.1.2 Power Management

To save power, the application can disable the clock to the SHA/MD5 module when not in use. The SHA/MD5 is clock gated by setting the SHACFG bit in the Cryptographic Modules Clock Gating Request (CCMCGREQ) register, CCM offset 0x204. The SHA in addition to the AES, DES, and Enhanced CRC can also be clock gated as a group by setting the D0 bit in the CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control (DCGCCCM) register, System Control Module offset 0x874.

25.1.3 Reset Management

To perform a software reset of the SHA module, write a 1 to the SOFTRESET bit in the SHA System Configuration (SHA_SYSCONFIG) register. The RESETDONE bit in the SHA System Status (SHA_SYSSTATUS) register indicates that the software reset is complete when its value is 1. When the software reset completes, the SOFTRESET bit in the SHA_SYSCONFIG register is automatically reset. Software must ensure that the software reset completes before doing any operations.

The behavior of the software reset is the same as the hardware reset, except that the software reset bit resets this module without affecting the reset core domain of the entire device.

25.1.4 μ DMA and Interrupt Requests

The SHA/MD5 module can operate in μ DMA mode where the module can assert a μ DMA request for context in, context out, or data input. The μ DMA signals that can be generated are:

- Context In μ DMA request (SHA/MD5 0 Cin): Request for Key, Digest, Mode and LENGTH information
- Context Out μ DMA request (SHA/MD5 0 Cout): Request for read from HMAC
- Data In μ DMA request (SHA/MD5 0 Din): Request input data in multiples of 16 bytes

The SHA/MD5 Module be programmed to assert an interrupt when the μ DMA has completed its last transfer by programming the SHA DMA Interrupt Mask (SHA_DMAIM), at the CRC and Cryptographic Modules (CCM) offset 0x010. The SHA DMA Raw Interrupt Status (SHA_DMARIS) register, at CCM offset 0x014, indicates when the μ DMA has completed and can be cleared by the SHA DMA Interrupt Clear (SHA_DMAIC) register at CCM offset 0x01C.

NOTE: The SHA module can only be accessed through privileged mode. If the μ DMA is used for SHA transfers, then the μ DMA's DMA Channel Control (DMACHCTL) register also needs to be programmed to allow for privileged accesses.

If context and data transfers are to be handled through software in Interrupt Mode, then the SHA Interrupt Enable (SHA_IRQENABLE), offset 0x11C, can be used to enable interrupt triggering when context out, context in, data in or data out is ready. The SHA Interrupt Status (SHA_IRQSTATUS), offset 0x118, indicates when an interrupt is triggered.

NOTE: If the application uses Interrupt Mode, an interrupt is generated for each block of processed data. To support larger data flow, SHA μ DMA Mode should be used and the bits in the SHA_IRQENABLE register should be cleared.

Table 25-1. Interrupts and Events

Event	Description
SHA_IRQSTATUS [3]: CONTEXT_OUT	Context output interrupt
SHA_IRQSTATUS [1]: DATA_IN	Data input interrupt
SHA_IRQSTATUS [0]: CONTEXT_IN	Context input interrupt

25.1.5 Operation Description

The SHA/MD5 Module can run the SHA-1, SHA-224, SHA-256, and MD5 algorithms, depending on the value of the ALGO bit field in the SHA Mode (SHA_MODE) register, offset 0x044, as listed in [Table 25-2](#).

Table 25-2. SHA/MD5 Module Algorithm Selection

ALGO Field Value in SHA_MODE Register	Description
0x0	MD5 algorithm selected
0x1	SHA-1 algorithm selected
0x2	SHA-224 algorithm selected
0x3	SHA-256 algorithm selected

25.1.5.1 SHA Mode

25.1.5.1.1 Starting a New Hash

To start a new hash, follow these steps:

1. Set the ALGO bitfield in the SHA_MODE register, at offset 0x044, to 0x1, 0x2, or 0x3 to select SHA-1, SHA-224, or SHA-256, respectively.
2. Set the ALGO_CONSTANT bit in the SHA_MODE register to 1 to initialize all SHA Inner and Outer Digest n registers from SHA_ODIGEST_A and SHA_IDIGEST_A to SHA_ODIGEST_H and SHA_IDIGEST_H with their default values specified by the algorithm, and set the SHA_DIGESTCOUNT register to 0.
3. Set the CLOSE_HASH bit of the SHA Mode (SHA_MODE) register to let the SHA engine do the padding. If the Hash is computed in one shot, the length of the message can be any value up to 128 MB. To process an intermediate Hash digest, the CLOSE_HASH bit is set to 0, in which case the packets hashed must be 64 bytes; the last packet must be hashed with the CLOSE_HASH bit set to 1.
4. Specify the LENGTH field in the SHA Length (SHA_LENGTH) register of the hash data to process in bytes.

After the configuration is complete, the INPUT_READY status bit equals 1 in the SHA Interrupt Status (SHA_IRQSTATUS) register (regardless of whether or not the M_INPUT_READY bit in the SHA_IRQENABLE register is set). When this bit is set, it indicates the SHA engine can receive the data to process. Data must be written to the 16 x 32-bit SHA_DATA_n_IN registers that provide storage for one 64-byte block of data. Unless the CLOSE_HASH bit is set, all of the SHA_DATA_n_IN input buffers must be filled. Data can be written by single write accesses to the 16 registers from a processor or by a DMA transfer.

For μ DMA transfers, the DMA_EN bit must be set in the SHA_SYSCONFIG register and the appropriate mask bits must be set in the SHA_DMAIM register before starting the new hash. Note that if the μ DMA is used for transfers, the SHA_IRQENABLE register should be clear so all interrupts are generated through the μ DMA interrupt registers.

The μ DMA must be configured to transfer 16 data words of 32 bits each time it is triggered by a μ DMA request from the SHA/MD5 Module. The 16 data words written are sent to the 16 SHA_DATA_n_IN registers.

The module detects that a 64-byte block is available, and then moves the data to a working register space for processing and asserts the INPUT_READY bit in the SHA_IRQSTATUS register to 1. If the DMA_EN bit in the SHA_SYSCONFIG register has been set to 1, a new μ DMA request triggers a new block transfer; otherwise, the processor polls the INPUT_READY bit and writes the 16 data words of 32 bits when it equals 1.

This operation is repeated until the length of the message to hash is reached. The OUTPUT_READY bit in the SHA_IRQSTATUS register then indicates that the hash operation is complete. If the IT_EN bit in the SHA_SYSCONFIG register is set, an interrupt (active low) is also generated to indicate the hash completion.

The processor can then read the eight digest registers A through H that contain the hash and/or HMAC result. If the hash is an intermediate result of a larger hash, the digest count register must also be read and saved.

NOTE: The number of digest registers used depends on the algorithm selected for the SHA/MD5 Module (MD5, SHA-1, SHA-224, or SHA-256) (see [Table 25-3](#) and [Table 25-4](#)).

Table 25-3. Outer Digest Registers

Register	Address	MD5 (Read/Write)	SHA-1 (Read/Write)	SHA-2 (Read/Write)	HMAC Key Processing (write)
SHA_ODIGEST_A	0x000	Outer digest [127:96]	Outer digest [159:128]	Outer digest [255:224]	HMAC Key [31:0]
SHA_ODIGEST_B	0x004	Outer digest [95:64]	Outer digest [127:96]	Outer digest [223:192]	HMAC key [63:32]
SHA_ODIGEST_C	0x008	Outer digest [63:32]	Outer digest [95:64]	Outer digest [191:160]	HMAC key [95:64]
SHA_ODIGEST_D	0x00C	Outer digest [31:0]	Outer digest [63:32]	Outer digest [159:128]	HMAC key [127:96]
SHA_ODIGEST_E	0x010		Outer digest [31:0]	Outer digest [127:96]	HMAC key [159:128]
SHA_ODIGEST_F	0x014			Outer digest [95:64]	HMAC key [191:160]
SHA_ODIGEST_G	0x018			Outer digest [63:32]	HMAC key [223:192]
SHA_ODIGEST_H	0x01C			Outer digest [31:0]	HMAC key [255:224]

Table 25-4. Inner Digest Registers

Register	Address	MD5 (Read/Write)	SHA-1 (Read/Write)	SHA-2 (Read/Write)	SHA-256 (Read/Write)	HMAC Key Processing (write)
SHA_IDIGEST_A	0x020	Inner digest [127:96]	Inner digest [159:128]	Inner digest [223:192]	Inner digest [255:224]	HMAC key [287:256]
SHA_IDIGEST_B	0x024	Inner digest [95:64]	Inner digest [127:96]	Inner digest [191:160]	Inner digest [223:192]	HMAC key [319:288]
SHA_IDIGEST_C	0x028	Inner digest [63:32]	Inner digest [95:64]	Inner digest [159:128]	Inner digest [191:160]	HMAC key [351:320]
SHA_IDIGEST_D	0x02C	Inner digest [31:0]	Inner digest [63:32]	Inner digest [127:96]	Inner digest [159:128]	HMAC key [383:352]
SHA_IDIGEST_E	0x030		Inner digest [31:0]	Inner digest [95:64]	Inner digest [127:96]	HMAC key [415:384]
SHA_IDIGEST_F	0x034			Inner digest [63:32]	Inner digest [95:64]	HMAC key [447:416]
SHA_IDIGEST_G	0x038			Inner digest [31:0]	Inner digest [63:32]	HMAC key [479:448]

Table 25-4. Inner Digest Registers (continued)

Register	Address	MD5 (Read/Write)	SHA-1 (Read/Write)	SHA-2 (Read/Write)	SHA-256 (Read/Write)	HMAC Key Processing (write)
SHA_IDIGEST_H	0x03C				Inner digest[31:0]	HMAC key [511:480]

NOTE: Inner digests are initial, intermediate, and result digests.

25.1.5.1.1.1 Outer Digest Registers

The SHA_ODIGEST_A to SHA_ODIGEST_H registers are relevant only for HMAC operations; the contents are ignored for hash operations.

Before writing to the digest registers, the operation must be configured in the SHA Mode (SHA_MODE) register. For HMAC operations without key processing, the HMAC_KEY_PROC bit must be clear in the SHA_MODE register before starting operations. Once the algorithm has been programmed in the SHA_MODE register, only the relevant digest registers for the selected algorithm must be written:

- SHA_ODIGEST_A to SHA_ODIGEST_D registers for MD5
- SHA_ODIGEST_A to SHA_ODIGEST_E registers for SHA-1
- SHA_ODIGEST_A to SHA_ODIGEST_H registers for SHA-2 (224 to 256)

When HMAC key processing is enabled (HMAC_KEY_PROC = 1), these registers must be written with the lower 256 bits of the HMAC key to be processed in little-endian format (first byte of key string in bits [7:0]).

NOTE: If the HMAC key is less than 512 bits, it must be properly padded with zeros: all 16 HMAC key registers must be written explicitly; the core does not pad. Additionally, if the HMAC key is larger than 512 bits, the host must perform a preprocessing step to reduce it to one 512-bit block. This involves hashing the large key and padding the hash result with zeros until it is 512 bits wide.

The computed outer digest can be read from these registers when the SHA Interrupt Status (SHA_IRQSTATUS) register when the OUTPUT_READY bit has been set indicating that the operation is done.

NOTE: If no HMAC key processing is performed, the value read is identical to the value written initially. The MD5 outer digest is available from registers SHA_ODIGEST_A to SHA_ODIGEST_D, the SHA-1 outer digest from registers SHA_ODIGEST_A to SHA_ODIGEST_E, and the SHA-224 and SHA-256 outer digest from registers SHA_ODIGEST_A to SHA_ODIGEST_H.

NOTE: The HMAC key is not preserved. If another block must be authenticated using the same key, the key must be reloaded by the host. If the same key must be used many times, it is advisable to do a HMAC key processing-only pass to obtain the inner and outer digest precomputes and load these precomputes for subsequent passes (only the inner digest must be reloaded if the outer digest is not modified by the host), because this saves two hash blocks worth of computation time.

25.1.5.1.1.2 Inner Digest Registers

The SHA_IDIGEST_A to SHA_IDIGEST_H registers are used for HMAC and hash operations.

The inner/initial digest for HMAC and hash continue operations (HMAC_KEY_PROC = 0 and ALGO_CONSTANT = 0) must be written to these registers before starting the operation by writing to the SHA_MODE register. Only the relevant digest registers for the selected algorithm must be written:

- SHA_IDIGEST_A to SHA_IDIGEST_D registers for MD5
- SHA_IDIGEST_A to SHA_IDIGEST_E registers for SHA-1
- SHA_IDIGEST_A to SHA_IDIGEST_H registers for SHA-2

When ALGO_CONSTANT = 1 in the SHA_MODE register, the SHA Inner Digest n (SHA_IDIGEST_n) registers do not need to be written by the application because they are overwritten with the appropriate algorithm constants.

When HMAC_KEY_PROC is 1, these registers must be written with the upper 256 bits of the HMAC key to be processed in little-endian format (first byte of key string in bits [7:0]).

NOTE: If the HMAC key is less than 512 bits, it must be properly padded with zeros: all 16 HMAC key registers must be written explicitly; the core does not pad. Additionally, if the HMAC key is larger than 512 bits, the host must perform a preprocessing step to reduce it to one 512-bit block. This involves hashing the large key and padding the hash result with zeros until it is 512 bits wide.

The order of the bytes within the digest is such that it can be fed back unmodified into the little-endian data input when preprocessing HMAC keys larger than 64 bytes, or it can typically be inserted unmodified into a little-endian data stream (for example, IPSEC packets), regardless of the selected algorithm.

NOTE: The HMAC key or inner digest is not preserved. If another block must be authenticated using the same key, the key or inner digest must be reloaded by the host. If the same key must be used many times, it is advisable to do a HMAC key processing-only pass to obtain the inner and outer digest precomputes and load these precomputes for subsequent passes (only the inner digest must be reloaded if the outer digest is not modified by the host), because this saves two hash blocks worth of computation time.

25.1.5.1.2 Closing a Hash

The amount of data to hash is not necessarily a multiple of 64 bytes. The CLOSE_HASH bit in the SHA_MODE register is set to append padding so that the message size becomes a multiple of 64 bytes. Consequently, a minimum of 9 bytes must be added to the message. Nine bytes is the minimum number of bytes that contains the minimum 65-bit padding specified by FIPS 180-1.

If the size of the last block of data is less than or equal to 55 bytes, no additional 64-byte block is required. However, if the last block of data contains more than 55 bytes, an extra 64-byte block must be added to make the padding as specified by FIPS 180-1. This extra block is added automatically by the hardware; thus, the module is fed with a 64-byte block of data. However, appending a pad on the last block of data can result in the creation of an extra 64-byte block.

The one or two last blocks that contain the padding are processed in the same way as the other blocks. Hash completion is then indicated in the same way as for a new hash, and the hash result can be read in the digest registers. The SHA_DIGESTCOUNT register returns restored Digest Count + Length when it is read, and hashing completes.

Assuming a message of 129 bytes, [Table 25-5](#) shows the SHA digest for three passes. [Table 25-6](#) shows the SHA digest for one pass.

Table 25-5. SHA Digest Processed in Three Passes

	Digest (A to E)	SHA_DIGEST_COUNT	SHA_MODE and SHA_LENGTH	SHA_DATA_n_IN
First pass			WRITE: LENGTH = 64 ALGO (dependent on the algorithm to apply) ALGO_CONSTANT = 1 CLOSE_HASH = 0	First 64 bytes of message

Table 25-5. SHA Digest Processed in Three Passes (continued)

	Digest (A to E)	SHA_DIGEST_COUNT	SHA_MODE and SHA_LENGTH	SHA_DATA_n_IN
Second pass	Round 1 digest calculation	Write: 64	WRITE: LENGTH = 64 ALGO (dependent on the algorithm to apply) ALGO_CONSTANT = 0 CLOSE_HASH = 0	Second 64 bytes of message
Third pass	Round 2 digest calculation	Write: 128	Write: LENGTH = 1 ALGO (dependent on the algorithm to apply) ALGO_CONSTANT = 0 CLOSE_HASH = 1	Last byte of message
–	Final digest	Read: 129		

If the three passes are not performed in succession, the digest registers must be saved and restored for the next use of the SHA/MD5 engine. If the rounds are performed consecutively, there is no need to do anything with the digest registers.

Table 25-6. SHA Digest Processed in One Pass

	Digest (A to E)	SHA_DIGEST_COUNT	SHA_MODE and SHA_LENGTH	SHA_DATA_n_IN
First pass			WRITE: LENGTH = 129 ALGO (dependent on the algorithm to apply) ALGO_CONSTANT = 1 CLOSE_HASH = 1	First 64 bytes of message
	Round 1 digest calculation			Second 64 bytes of message
	Round 2 digest calculation			Last byte of message
	Final digest	Read: 129		

25.1.5.2 MD5 Mode

25.1.5.2.1 Starting a New Hash

To start a new hash, perform the following steps:

1. Set the ALGO bit field in the SHA_MODE register to 0x0 to select the MD5 algorithm.
2. Set the ALGO_CONSTANT bit to 1 in the SHA_MODE register to initialize all digest registers from SHA_ODIGEST_A and SHA_IDIGEST_A to SHA_ODIGEST_H and SHA_IDIGEST_H with default values specified by the algorithm, and set the SHA_DIGESTCOUNT register to 0.
3. Specify the LENGTH field in the SHA_LENGTH register of the hash data to process in bytes.
4. Set the CLOSE_HASH bit in the SHA_MODE register to let the SHA/MD5 engine do the padding. If MD5 is computed in one shot, the length of the message can be any value up to. To process an intermediate MD5 digest, the CLOSE_HASH bit is set to 0, in which case packets to be hashed must be 64 bytes; the last packet must be hashed with the CLOSE_HASH bit set to 1.

After the configuration is complete, the hash engine can receive the data to process (the INPUT_READY bit is 1 in the SHA_IRQSTATUS register). Data must be written to the 16 x 32-bit SHA_DATA_n_IN registers that provide storage for one 64-byte block of data. Unless the CLOSE_HASH bit is set in the SHA_MODE register, the SHA_DATA_n_IN 64-byte input buffer must be filled. Data can be written by single write transactions to the 16 registers from a processor or by a μ DMA transfer.

For a μ DMA transfer, the SDAM_EN bit must be set in the SHA_SYSCONFIG register before starting the new hash and the μ DMA channel for SHA/MD5 0 Data In Request must be configured. The μ DMA must be configured to the appropriate hash transfer size. See [Chapter 8](#) for more information on programming the μ DMA. A μ DMA done is asserted after the last SHA_DATA_n_IN register is filled.

The module detects that a 64-byte block is available, and then moves the data to a working register space for processing and sets the INPUT_READY bit to 1 in the SHA_IRQSTATUS register. If the DMA_EN bit is set in the SHA_SYSCONFIG register, then a new μ DMA request triggers a new block transfer; otherwise, the processor polls the INPUT_READY bit in the SHA_IRQSTATUS register and writes the 16 data words of 32 bits when it equals 1.

This operation repeats until the length of the message to hash is reached. The OUTPUT_READY bit in the SHA_IRQSTATUS register then indicates that the hash operation is complete. If the IT_EN bit in the SHA_SYSCONFIG register is set, an interrupt (active low) is also generated to indicate the hash completion.

25.1.5.2.2 Closing a Hash

The amount of data to hash is not necessarily a multiple of 64 bytes. In this case, the CLOSE_HASH bit in the SHA_MODE register must be set to append padding so that the message size becomes a multiple of 64 bytes. See the previous MD5 algorithm for more information on padding.

The module is fed with a 64-byte block of data, as long as enough data is available. However, a pad is appended on the last block of data. This can result in the creation of an extra 64-byte block.

The one or two last blocks that contain the padding are processed the same way as the other blocks. Hash completion is then indicated the same way as for a new hash, and the 128-bit result can be read in the digest registers. The SHA_DIGESTCOUNT register returns restored digest count and length when it is read, and hashing completes.

25.1.5.3 Generating an Software Interrupt

If the IT_EN bit is 1 in the SHA_SYSCONFIG register, an interrupt is generated at the completion of the hash by the following steps:

1. Receive last block of data (= 64 bytes). (The number of data bytes defined by the SHA_LENGTH register is received in the digest registers, from SHA_ODIGEST_A and SHA_IDIGEST_A to SHA_ODIGEST_H and SHA_IDIGEST_H.
2. If required, apply padding to the last block of data.
3. Hash the last block of data (80 cycles in SHA-1 mode and 64 cycles in MD5, SHA-224, and SHA-256 modes).
4. If required, add an extra 64-byte block of data to complete the padding.
5. Hash this extra block of data (80 cycles in SHA-1 mode and 64 cycles in MD5, SHA-224, and SHA-256 modes).
6. An interrupt is generated (active low).

25.1.6 SHA/MD5 Performance Information

The following table lists the performance for all supported key sizes and modes of operations. It assumed that the engine is kept fully utilized (that is, the host is supplying input block and retrieving output blocks in such a way, that the engine never has to wait for input) and that the previous output has been retrieved before the next output is ready.

Maximum throughput does not include per operation overhead cycles, as the impact on the throughput depends on the size of the block being processed, and would be negligible for large blocks anyway.

Table 25-7. SHA/MD5 Performance

Operation	Algorithm	Cycles per Operation	Cycles per Block
Hash	MD5	0 / 65	65
	SHA-1	0 / 81	81
	SHA-224	0 / 65	65
	SHA-256	0 / 65	65

Table 25-7. SHA/MD5 Performance (continued)

Operation	Algorithm	Cycles per Operation	Cycles per Block
HMAC from key	MD5	196 / 261	65
	SHA-1	244 / 325	81
	SHA-224	196 / 261	65
	SHA-256	196 / 261	65
	MD5	66 / 131	65
HMAC from precomputes	SHA-1	82 / 163	81
	SHA-224	66 / 131	65
	SHA-256	66 / 131	65

NOTE: An extra block needs to be processed if the length of the data block to be hashed module 64 is 0 or equal to 56.

25.1.7 SHA/MD5 Programming Guide

This section covers the hardware programming sequences for configuration and use of the SHA/MD5 module.

25.1.7.1 Global Initialization

25.1.7.1.1 Surrounding Modules Global Initialization

This section identifies the steps required for initializing the surrounding modules when the SHA/MD5 is used for the first time after a device reset.

1. When reset has completed, enable the SHA/MD5 Module by setting the R0 bit in the CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM) register, System Control offset 0x674. When the R0 bit is set in the CRC and Cryptographic Modules (PRCCM) register, System Control offset 0xA74 register, the SHA/MD5 Module is powered and ready to be configured.
2. Configure the SHA μ DMA channels for Context In, Context Out, Data In, and/or Data Out by programming the appropriate encoding value in the DMA Channel Map Select n (DMACHMAPn) register in the μ DMA module, offset 0x510. For more information on how to program channel assignments as well as enabling burst and the configured channels, refer to [Chapter 8](#).
3. Execute a software reset by setting the SOFTRESET bit in the SHA_SYSCONFIG register. When reset is complete, the RESETDONE bit reads as 1 in the SHA_SYSSTATUS register.
4. If the SHA channels are configured in the μ DMA, enable the required SHA DMA requests by programming bits [9:5] of the SHA_SYSCONFIG register, in addition to the completion interrupts in the SHA DMA Interrupt Mask (SHA_DMAIM) register, CRC and Cryptographic Modules offset 0x020.

25.1.7.1.2 Starting a New HMAC Using the SHA-1 Hash Function and HMAC Key Processing

The following procedure is used to begin a new HMAC operation, starting from initial digest values.

1. Load the key value in the SHA_ODIGEST_A/SHA_IDIGEST_A to SHA_ODIGEST_H/SHA_IDIGEST_H registers.
2. Pad the rest of the SHA_ODIGEST_x and SHA_IDIGEST registers with zeros.
3. Load the message in the SHA_DATA_n_IN FIFO registers.
4. Enable HMAC key processing by setting the HMAC_KEY_PROC bit in the SHA_MODE register.
5. Select the SHA-1 hash function by programming the ALGO bit in the SHA_MODE register to 0x1.
6. Select the already loaded key by programming the ALGO_CONSTANT bit in the SHA_MODE register to 0x0.
7. Set the CLOSE_HASH bit and the HMAC_OUTER_HASH bit in the SHA_MODE register so that

appropriate padding is inserted and the outer hash is performed immediately after the inner hash has finished.

8. Program the SHA_LENGTH register with the block length. Writing this register triggers the HMAC engine to begin processing.

NOTE: If more than one pass is used during the process (SHA_MODE[4] CLOSE_HASH == 0x0), the block length value must be a 64-byte multiple. From this point, three operational modes are possible to continue with the processing: polling, interrupt, and DMA. For more information, see the Operational Modes Configuration section.

25.1.7.1.2.1 Subsequence – Continuing a Prior HMAC Using the SHA-1 Hash Function

The following procedure is used to continue a prior HMAC calculation interrupted from a high priority task (see [Table 25-8](#)).

Table 25-8. Continuing a Prior HMAC

Step	Register / Bit Field / Programming Model	Value
Load the initial digest for the used hash algorithm.	SHA_IDIGEST_i[31:0] DATA	-
Restore the digest counter with the value before the switch to the high-priority task.	SHA_DIGEST_COUNT[31:0] COUNT	-
Use the already loaded in the engine key.	SHA_MODE[5] HMAC_KEY_PROC	0x0
Do not use the constants of the selected hash algorithm.	SHA_MODE[3] ALGO_CONSTANT	0x0
Select the SHA-1 hash algorithm.	SHA_MODE[2:1] ALGO	0x1
IF: This is the last 64-byte data block from the input message?	User decision	-
Close the hash; an appropriate padding is added.	SHA_MODE[4] CLOSE_HASH	0x1
ENDIF		-
Load the block length; this is the trigger to start processing.	SHA_LENGTH[31:0] LENGTH	-

NOTE: This initial digest is the intermediate digest from the previous calculation before switching to the high priority task. The value is equal to context1 in.

NOTE: The value is equal to context2 in.

NOTE: The block length is equal to the context3 value in.

25.1.7.1.2.2 Subsequence – Hashing a Key Bigger Than 512 Bits With the SHA-1 Hash Function

The following procedure is used to create a hash value from the key in only one pass (see [Table 25-9](#)).

Table 25-9. SHA-1 Apply on the Key

Step	Register / Bit Field / Programming Model	Value
Load the first part of the key (here, the key is like a message).	SHA_DATA_n_IN (i = 0 to 15)	-
Select the SHA-1 hash function.	SHA_MODE[2:1] ALGO	0x1
Select a new hash operation.	SHA_MODE[3] ALGO_CONSTANT	0x1
Close the hash; the key is processed in single pass.	SHA_MODE[4] CLOSE_HASH	0x1

25.1.7.1.3 Operational Modes Configuration

25.1.7.1.3.1 SHA/MD5 Polling Mode

Figure 25-2 shows the SHA/MD5 polling mode. SHA/MD5 polling mode uses the following registers.

- SHA_IRQSTATUS
- SHA_DATA_n_IN
- SHA_ODIGEST_A
- SHA_DIGEST_COUNT
- SHA_LENGTH

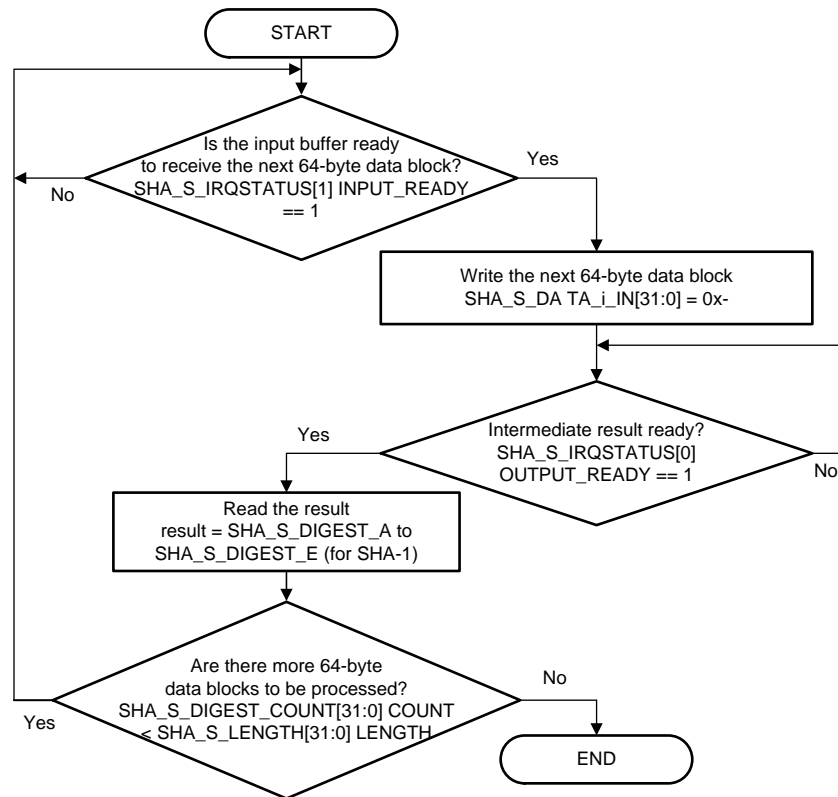


Figure 25-2. SHA/MD5 Polling Mode

25.1.7.1.3.2 SHA/MD5 Interrupt Mode

The following procedure is used to configure the SHA/MD5 module to work in the interrupt-based mode (see Table 25-10). For the interrupt subroutine, see Section 25.1.7.1.4.1.

Table 25-10. Interrupt Mode

Step	Register / Bit Field / Programming Model	Value
Enable the interrupt request to the Cortex-A8 MPU INTC.	SHA_SYSCONFIG[2] IT_EN	0x1
Load the message length; this is the trigger to start processing.	SHA_LENGTH[31:0] LENGTH	-

25.1.7.1.3.3 SHA/MD5 DMA Mode

The following procedure is used to configure the SHA/MD5 module to work in the DMA-based mode.

Table 25-11. DMA Mode

Step	Register / Bit Field / Programming Model	Value
Enable the DMA request to the CDMA controller.	SHA_SYSCONFIG[3] DMA_EN	0x1
Load the message length; this is the trigger to start processing.	SHA_LENGTH[31:0] LENGTH	-

25.1.7.1.4 SHA/MD5 Event Servicing

25.1.7.1.4.1 Interrupt Servicing

This section describes the interrupt event servicing of the module. [Figure 25-3](#) shows the interrupt subroutine. The following registers are used in the SHA/MD5 interrupt routine:

- SHA_IRQSTATUS
- SHA_DATA_n_IN
- SHA_ODIGEST_A
- SHA_DIGEST_COUNT
- SHA_LENGTH
- SHA_IDIGEST_A

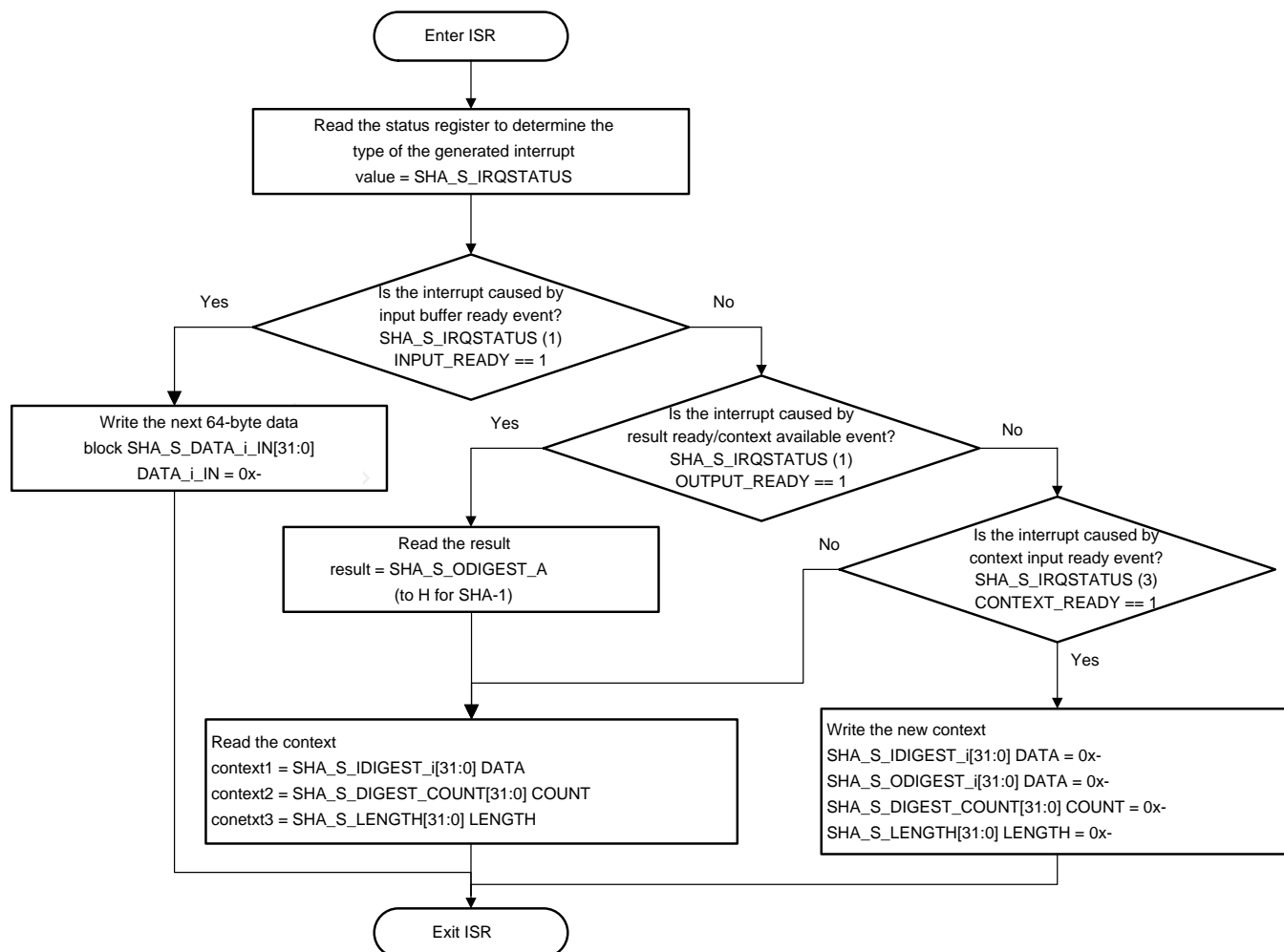


Figure 25-3. SHA/MD5 Interrupt Subroutine

25.2 SHA/MD5 Registers

The SHA Module registers are at an offset relative to the SHA/MD5 Module base address, and a small set of SHA/MD5 μ DMA interrupt registers are at an offset relative to a CRC and Cryptographic module base address.

The SHA/MD5 module register offsets are relative to the base address 0x44034000.

The SHA/MD5 registers are limited to 32-bit data accesses; 8- and 16-bit accesses are not allowed and can corrupt register contents. The first 16 registers of HIB1 are the outer and inner digest registers (see [Table 25-12](#)).

Table 25-12. SHA/MD5 Inner/Outer Digest/HMAC Key Register Mapping

Register Name	Address Offset	MD5 (read/write)	SHA-1 (read/write)	SHA-2 (read/write)		HMAC key proc (write)
SHA_ODIGEST_A	0x0000	Outer digest [127:96]	Outer digest [159:128]	Outer digest [255:224]		HMAC key [31,0]
SHA_ODIGEST_B	0x0004	Outer digest [95:64]	Outer digest [127:96]	Outer digest [223:192]		HMAC key [63,32]
SHA_ODIGEST_C	0x0008	Outer digest [63:32]	Outer digest [95:64]	Outer digest [191:160]		HMAC key [95,64]
SHA_ODIGEST_D	0x000C	Outer digest [31:0]	Outer digest [63:32]	Outer digest [159:128]		HMAC key [127,96]
SHA_ODIGEST_E	0x0010		Outer digest [31:0]	Outer digest [127:96]		HMAC key [159,128]
SHA_ODIGEST_F	0x0014			Outer digest [95:64]		HMAC key [191,160]
SHA_ODIGEST_G	0x0018			Outer digest [63:32]		HMAC key [223,192]
SHA_ODIGEST_H	0x001C			Outer digest [31:0]		HMAC key [255,224]
SHA_IDIGEST_A	0x0020	Inner digest [127:96]	Inner digest [159:128]	Inner digest [223:192]	Inner digest [255:224]	HMAC key [287,256]
SHA_IDIGEST_B	0x0024	Inner digest [95:64]	Inner digest [127:96]	Inner digest [191:160]	Inner digest [223:192]	HMAC key [319,288]
SHA_IDIGEST_C	0x0028	Inner digest [63:32]	Inner digest [95:64]	Inner digest [159:128]	Inner digest [191:160]	HMAC key [351,320]
SHA_IDIGEST_D	0x002C	Inner digest [31:0]	Inner digest [63:32]	Inner digest [127:96]	Inner digest [159:128]	HMAC key [383,352]
SHA_IDIGEST_E	0x0030		Inner digest [31:0]	Inner digest [95:64]	Inner digest [127:96]	HMAC key [415,384]
SHA_IDIGEST_F	0x0034			Inner digest [63:32]	Inner digest [95:64]	HMAC key [447,416]
SHA_IDIGEST_G	0x0038			Inner digest [31:0]	Inner digest [63:32]	HMAC key [479,448]
SHA_IDIGEST_H	0x003C				Inner digest [31:0]	HMAC key [511,480]

Table 25-13 lists the memory-mapped registers for the SHA/MD5. All register offset addresses not listed in Table 25-13 should be considered as reserved locations and the register contents should not be modified.

Table 25-13. SHA/MD5 Registers

Offset	Acronym	Register Name	Section
0x000	SHA_ODIGEST_A	SHA Outer Digest A	Section 25.2.1
0x004	SHA_ODIGEST_B	SHA Outer Digest B	Section 25.2.1
0x008	SHA_ODIGEST_C	SHA Outer Digest C	Section 25.2.1
0x00C	SHA_ODIGEST_D	SHA Outer Digest D	Section 25.2.1
0x010	SHA_ODIGEST_E	SHA Outer Digest E	Section 25.2.1
0x014	SHA_ODIGEST_F	SHA Outer Digest F	Section 25.2.1
0x018	SHA_ODIGEST_G	SHA Outer Digest G	Section 25.2.1
0x01C	SHA_ODIGEST_H	SHA Outer Digest H	Section 25.2.1
0x020	SHA_IDIGEST_A	SHA Inner Digest A	Section 25.2.1
0x024	SHA_IDIGEST_B	SHA Inner Digest B	Section 25.2.1
0x028	SHA_IDIGEST_C	SHA Inner Digest C	Section 25.2.1
0x02C	SHA_IDIGEST_D	SHA Inner Digest D	Section 25.2.1
0x030	SHA_IDIGEST_E	SHA Inner Digest E	Section 25.2.1
0x034	SHA_IDIGEST_F	SHA Inner Digest F	Section 25.2.1
0x038	SHA_IDIGEST_G	SHA Inner Digest G	Section 25.2.1
0x03C	SHA_IDIGEST_H	SHA Inner Digest H	Section 25.2.1
0x040	SHA_DIGEST_COUNT	SHA Digest Count	Section 25.2.2
0x044	SHA_MODE	SHA Mode	Section 25.2.3
0x048	SHA_LENGTH	SHA Length	Section 25.2.4
0x080	SHA_DATA_0_IN	SHA Data 0 Input	Section 25.2.5
0x084	SHA_DATA_1_IN	SHA Data 1 Input	Section 25.2.5
0x088	SHA_DATA_2_IN	SHA Data 2 Input	Section 25.2.5
0x08C	SHA_DATA_3_IN	SHA Data 3 Input	Section 25.2.5
0x090	SHA_DATA_4_IN	SHA Data 4 Input	Section 25.2.5
0x094	SHA_DATA_5_IN	SHA Data 5 Input	Section 25.2.5
0x098	SHA_DATA_6_IN	SHA Data 6 Input	Section 25.2.5
0x09C	SHA_DATA_7_IN	SHA Data 7 Input	Section 25.2.5
0x0A0	SHA_DATA_8_IN	SHA Data 8 Input	Section 25.2.5
0x0A4	SHA_DATA_9_IN	SHA Data 9 Input	Section 25.2.5
0x0A8	SHA_DATA_10_IN	SHA Data 10 Input	Section 25.2.5
0x0AC	SHA_DATA_11_IN	SHA Data 11 Input	Section 25.2.5
0x0B0	SHA_DATA_12_IN	SHA Data 12 Input	Section 25.2.5
0x0B4	SHA_DATA_13_IN	SHA Data 13 Input	Section 25.2.5
0x0B8	SHA_DATA_14_IN	SHA Data 14 Input	Section 25.2.5
0x0BC	SHA_DATA_15_IN	SHA Data 15 Input	Section 25.2.5
0x100	SHA_REVISION	SHA Revision	Section 25.2.6
0x110	SHA_SYSCONFIG	SHA System Configuration	Section 25.2.7
0x114	SHA_SYSSTATUS	SHA System Status	Section 25.2.8
0x118	SHA_IRQSTATUS	SHA Interrupt Status	Section 25.2.9
0x11C	SHA_IRQENABLE	SHA Interrupt Enable	Section 25.2.10

Complex bit access types are encoded to fit into small table cells. Table 25-14 shows the codes that are used for access types in this section.

Table 25-14. SHA/MD5 Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

25.2.1 SHA_ODIGEST_n and SHA_IDIGEST_n Registers (Offset = 0x000 to 0x03C) [reset = 0x0]

SHA Outer Digest A (SHA_ODIGEST_A), offset 0x000

SHA Outer Digest B (SHA_ODIGEST_B), offset 0x004

SHA Outer Digest C (SHA_ODIGEST_C), offset 0x008

SHA Outer Digest D (SHA_ODIGEST_D), offset 0x00C

SHA Outer Digest E (SHA_ODIGEST_E), offset 0x010

SHA Outer Digest F (SHA_ODIGEST_F), offset 0x014

SHA Outer Digest G (SHA_ODIGEST_G), offset 0x018

SHA Outer Digest H (SHA_ODIGEST_H), offset 0x01C

SHA Inner Digest A (SHA_IDIGEST_A), offset 0x020

SHA Inner Digest B (SHA_IDIGEST_B), offset 0x024

SHA Inner Digest C (SHA_IDIGEST_C), offset 0x028

SHA Inner Digest D (SHA_IDIGEST_D), offset 0x02C

SHA Inner Digest E (SHA_IDIGEST_E), offset 0x030

SHA Inner Digest F (SHA_IDIGEST_F), offset 0x034

SHA Inner Digest G (SHA_IDIGEST_G), offset 0x038

SHA Inner Digest H (SHA_IDIGEST_H), offset 0x03C

SHA_ODIGEST_n and SHA_IDIGEST_n are shown in [Figure 25-4](#) and described in [Table 25-15](#).

Return to [Summary Table](#).

Figure 25-4. SHA_ODIGEST_n and SHA_IDIGEST_n Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															
R/W-0x0																															

Table 25-15. SHA_ODIGEST_n and SHA_IDIGEST_n Registers Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0x0	Digest/key data

25.2.2 SHA_DIGEST_COUNT Register (Offset = 0x40) [reset = 0x0]

SHA Digest Count (SHA_DIGEST_COUNT)

This register is written with the initial digest count and can be read to determine the digest count result. Note that for Initial Digest Count writes, bits 5:0 must be zero. This register is written the initial digest byte count when both the HMAC_KEY_PROC bit and the ALGO_CONSTANT bit is zero in the SHA_MODE register. When either the HMAC_KEY_PROC bit or the ALGO_CONSTANT bit is 1, this register does not need to be written because it is overwritten with 64 or 0 automatically. When starting an HMAC operation from precomputes (HMAC_KEY_PROC = 0), the value 64 must be written in the SHA_DIGESTCOUNT register. Note that the value written should always be a 64 byte multiple, the lower 6 bits written are ignored. The updated digest byte count (initial digest byte count + bytes processed) can be read from this register when the status register indicates that the operation is done or when a μ DMA context out is requested.

SHA_DIGEST_COUNT is shown in [Figure 25-5](#) and described in [Table 25-16](#).

Return to [Summary Table](#).

Figure 25-5. SHA_DIGEST_COUNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT																															
R/W-0x0																															

Table 25-16. SHA_DIGEST_COUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COUNT	R/W	0x0	Digest Count. This field is written with the initial digest value for bits [31:6] only ([5:0] are assumed to be zeros). When read, this outputs the result/intermediate digest count.

25.2.3 SHA_MODE Register (Offset = 0x44) [reset = 0x0]

SHA Mode (SHA_MODE)

This register is written to configure the hash algorithm to be used in the hash operation.

SHA_MODE is shown in [Figure 25-6](#) and described in [Table 25-17](#).

Return to [Summary Table](#).

Figure 25-6. SHA_MODE Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
HMAC_OUTER_HASH	RESERVED	HMAC_KEY_PROC	CLOSE_HASH	ALGO_CONSTANT	ALGO		
R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0		

Table 25-17. SHA_MODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	HMAC_OUTER_HASH	R/W	0x0	<p>HMAC Outer Hash Processing Enable.</p> <p>This bit is written to indicate that the outer hash should be performed on the hash digest when the inner hash has finished.</p> <p>The inner hash finishes when the length of hash has been processed the final inner hash is performed (if CLOSE_HASH was set to 1).</p> <p>This bit should normally be set together with CLOSE_HASH to finish the inner hash first, or immediately when block length is zero (HMAC continue with the just outer hash to be done).</p> <p>This bit is auto-cleared when outer hash is finished.</p> <p>0x0 = No operation</p> <p>0x1 = Enable HMAC processing of outer hash. This bit self-clears when outer hash is finished.</p>
6	RESERVED	R	0x0	
5	HMAC_KEY_PROC	R/W	0x0	<p>HMAC Key Processing Enable.</p> <p>This bit enables HMAC key processing on the 512-bit HMAC key loaded into the SHA_IDIGEST_A through SHA_IDIGEST_H registers and the SHA_ODIGEST_A through SHA_ODIGEST_H register block.</p> <p>Once HMAC key processing is finished, this bit is automatically cleared and the resulting Inner and Outer digest is available in the SHA_IDIGEST_x and SHA_ODIGEST_x respectively.</p> <p>After key processing is complete, regular hash processing begins until the block length is exhausted.</p> <p>0x0 = No operation</p> <p>0x1 = Enable HMAC key processing. This bit is automatically cleared once HMAC key processing is finished.</p>

Table 25-17. SHA_MODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	CLOSE_HASH	R/W	0x0	<p>Performs the padding, the Hash/HMAC will be 'closed' at the end of the block, as per MD5/SHA-1/SHA-2 specification (that is, appropriate padding is added), or no padding is done, allowing the hash to be continued later.</p> <p>However, if the Hash/HMAC is not closed, then the Block Length must be a multiple of 64 bytes to ensure correct operation.</p> <p>Auto cleared internally when hash closed.</p> <p>0x0 = No padding, hash computation can be continued.</p> <p>0x1 = Last packet will be padded.</p>
3	ALGO_CONSTANT	R/W	0x0	<p>The initial digest register will be overwritten with the algorithm constants for the selected algorithm when hashing and the initial digest count register will be reset to 0.</p> <p>This will start a normal hash operation.</p> <p>When continuing an existing hash or when performing an HMAC operation, this register must be set to 0 and the intermediate/inner digest or HMAC key and digest count need to be written to the context input registers prior to writing SHA_MODE.</p> <p>Auto cleared internally after first block processed.</p> <p>0x0 = Use precalculated digest (from another operation)</p> <p>0x1 = Use constants of the selected algorithm</p>
2-0	ALGO	R/W	0x0	<p>Hash Algorithm</p> <p>0x0 = MD5</p> <p>0x1 = reserved</p> <p>0x2 = SHA-1</p> <p>0x3 = reserved</p> <p>0x4 = SHA-224</p> <p>0x5 = reserved</p> <p>0x6 = SHA-256</p> <p>0x7 = reserved</p>

25.2.4 SHA_LENGTH Register (Offset = 0x48) [reset = 0x0]

SHA Length (SHA_LENGTH)

WRITE: Block Length/Remaining Byte Count (bytes) READ: Remaining Byte Count. The value programmed MUST be a 64-byte multiple if Close Hash is set to 0. This register is also the trigger to start processing: once this register is written, the core will commence requesting input data via DMA or IRQ (if programmed length > 0) and start processing. The remaining byte count for the active operation can be read from this register when the interrupt status register indicates that the operation is suspended due to a context switch request.

SHA_LENGTH is shown in [Figure 25-7](#) and described in [Table 25-18](#).

Return to [Summary Table](#).

Figure 25-7. SHA_LENGTH Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENGTH																															
R/W-0x0																															

Table 25-18. SHA_LENGTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LENGTH	R/W	0x0	Block Length/Remaining Byte Count. This field is written with the block length in bytes of the data to be processed. When read, this field contains the remaining byte count.

25.2.5 SHA_DATA_n_IN Registers (Offset = 0x080 to 0x0BC) [reset = 0x0]

SHA Data 0 Input (SHA_DATA_0_IN), offset 0x080
SHA Data 1 Input (SHA_DATA_1_IN), offset 0x084
SHA Data 2 Input (SHA_DATA_2_IN), offset 0x088
SHA Data 3 Input (SHA_DATA_3_IN), offset 0x08C
SHA Data 4 Input (SHA_DATA_4_IN), offset 0x090
SHA Data 5 Input (SHA_DATA_5_IN), offset 0x094
SHA Data 6 Input (SHA_DATA_6_IN), offset 0x098
SHA Data 7 Input (SHA_DATA_7_IN), offset 0x09C
SHA Data 8 Input (SHA_DATA_8_IN), offset 0x0A0
SHA Data 9 Input (SHA_DATA_9_IN), offset 0x0A4
SHA Data 10 Input (SHA_DATA_10_IN), offset 0x0A8
SHA Data 11 Input (SHA_DATA_11_IN), offset 0x0AC
SHA Data 12 Input (SHA_DATA_12_IN), offset 0x0B0
SHA Data 13 Input (SHA_DATA_13_IN), offset 0x0B4
SHA Data 14 Input (SHA_DATA_14_IN), offset 0x0B8
SHA Data 15 Input (SHA_DATA_15_IN), offset 0x0BC
Data input message

NOTE: The SHA_DATA_n_IN_0 register acts as a FIFO and shifts data into the other SHA_DATA_n_IN registers.

SHA_DATA_n_IN is shown in [Figure 25-8](#) and described in [Table 25-19](#).

Return to [Summary Table](#).

Figure 25-8. SHA_DATA_n_IN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_IN																															
R/W-0x0																															

Table 25-19. SHA_DATA_n_IN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA_IN	R/W	0x0	Digest/Key Data

25.2.6 SHA_REVISION Register (Offset = 0x100) [reset = 0x40000C03]

SHA Revision (SHA_REVISION)

This register provides the unique revision number of the module. This can be used by the driver to determine the capabilities available.

SHA_REVISION is shown in [Figure 25-9](#) and described in [Table 25-20](#).

Return to [Summary Table](#).

Figure 25-9. SHA_REVISION Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															
R-0x40000C03																															

Table 25-20. SHA_REVISION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REVISION	R	0x40000C03	Revision Number. This field indicates the revision number of the module.

25.2.7 SHA_SYSCONFIG Register (Offset = 0x110) [reset = 0x1]

SHA System Configuration (SHA_SYSCONFIG)

NOTE: After one operation has completed, the SHA_SYSCONFIG register must be cleared and re-configured for the next operation to ensure proper μ DMA and data operation functionality.

SHA_SYSCONFIG is shown in [Figure 25-10](#) and described in [Table 25-21](#).

Return to [Summary Table](#).

Figure 25-10. SHA_SYSCONFIG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
SADVANCED	RESERVED	SIDLE		DMA_EN	IT_EN	SOFTRESET	RESERVED
R/W-0x0	R-0x0	R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R-0x1

Table 25-21. SHA_SYSCONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	SADVANCED	R/W	0x0	Advanced Mode Enable 0x0 = Legacy mode enabled for the Secure World. In Legacy mode, the Secure World, the context input DMA request, and the result output DMA request are masked. This means that neither DMAREQUEST_CTXIN_S and DMAREQUEST_CTXOUT_S are asserted. 0x1 = Advanced mode is enabled. These DMA requests are enabled by bit 3 of this register.
6	RESERVED	R	0x0	
5-4	SIDLE	R/W	0x0	Sidle mode 0x0 = Force-idle mode 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
3	DMA_EN	R/W	0x0	μ DMA Request Enable. This bit controls whether the μ DMA interrupts can be programmed/controlled in the SHA_DMA_IM register. If the μ DMA is used for transferring data, then the IT_EN bit should be set to 0 and the SHA_IRQENABLE register should be clear. 0x0 = μ DMA interrupts are disabled. 0x1 = μ DMA interrupts are enabled.
2	IT_EN	R/W	0x0	Interrupt Enable. This bit controls whether the software interrupts can be programmed/controlled in the SHA_IRQENABLE register. When enabling the interrupts in the SHA_IRQENABLE register, the application should poll these interrupts and configure a software interrupt in the μ DMA to handle a trigger event. 0x0 = SHA software interrupts are disabled. 0x1 = SHA software interrupts are enabled.

Table 25-21. SHA_SYSCONFIG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	SOFTRESET	R/W	0x0	Soft reset 0x0 = No operation 0x1 = Start soft reset sequence
0	RESERVED	R	0x1	

25.2.8 SHA_SYSSTATUS Register (Offset = 0x114) [reset = 0x1]

SHA System Status (SHA_SYSSTATUS)

SHA_SYSSTATUS is shown in [Figure 25-11](#) and described in [Table 25-22](#).

Return to [Summary Table](#).

Figure 25-11. SHA_SYSSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							RESETDONE
R-0x0							R-0x1

Table 25-22. SHA_SYSSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	RESETDONE	R	0x1	Reset done status 0x0 = Reset not complete 0x1 = Reset complete

25.2.9 SHA_IRQSTATUS Register (Offset = 0x118) [reset = 0x8]

SHA Interrupt Status (SHA_IRQSTATUS)

SHA_IRQSTATUS is shown in [Figure 25-12](#) and described in [Table 25-23](#).

Return to [Summary Table](#).

Figure 25-12. SHA_IRQSTATUS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				CONTEXT_READY	RESERVED	INPUT_READY	OUTPUT_READY
R-0x0				R-0x1	R-0x0	R-0x0	R-0x0

Table 25-23. SHA_IRQSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	CONTEXT_READY	R	0x1	Context Ready Status 0x0 = The context registers are not available for a new context. 0x1 = The context input registers are available for a new context for the next packet to be processed.
2	RESERVED	R	0x0	
1	INPUT_READY	R	0x0	Input Ready Status 0x0 = The Data FIFO is not ready to receive the next 64-byte data block. 0x1 = The Data FIFO (SHA_DATA_n_IN registers) is ready to receive the next 64-byte data block.
0	OUTPUT_READY	R	0x0	Output Ready Status 0x0 = No saved context available. 0x1 = A saved context is available from the context output registers.

25.2.10 SHA_IRQENABLE Register (Offset = 0x11C) [reset = 0x3]

SHA Interrupt Enable (SHA_IRQENABLE)

The SHA_IRQENABLE register contains an enable bit for each unique interrupt. An interrupt is enabled when both the global enable, the IT_EN bit, in the SHA_SYSCONFIG register and the bit in this register are both set to 1.

SHA_IRQENABLE is shown in [Figure 25-13](#) and described in [Table 25-24](#).

Return to [Summary Table](#).

Figure 25-13. SHA_IRQENABLE Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				M_CONTEXT_READY	RESERVED	M_INPUT_READY	M_OUTPUT_READY
R-0x0				R/W-0x0	R-0x0	R/W-0x1	R/W-0x1

Table 25-24. SHA_IRQENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	M_CONTEXT_READY	R/W	0x0	Mask for context ready interrupt 0x0 = Context ready interrupt is disabled (masked). 0x1 = Context ready interrupt is enabled.
2	RESERVED	R	0x0	
1	M_INPUT_READY	R/W	0x1	Mask for input ready interrupt 0x0 = Input ready interrupt is disabled (masked). 0x1 = Input ready interrupt is enabled.
0	M_OUTPUT_READY	R/W	0x1	Mask for output ready interrupt 0x0 = Output ready interrupt is disabled (masked). 0x1 = Output ready interrupt is enabled.

25.3 SHA/MD5 μ DMA Registers

Table 25-25 lists the memory-mapped registers for the SHA/MD5 μ DMA. All register offset addresses not listed in Table 25-25 should be considered as reserved locations and the register contents should not be modified. The SHA μ DMA offsets are relative to the base address 0x44030000.

Table 25-25. SHA_MD5_UDMA Registers

Offset	Acronym	Register Name	Section
0x10	SHA_DMAIM	SHA DMA Interrupt Mask	Section 25.3.1
0x14	SHA_DMARIS	SHA DMA Raw Interrupt Status	Section 25.3.2
0x18	SHA_DMAMIS	SHA DMA Masked Interrupt Status	Section 25.3.3
0x1C	SHA_DMAIC	SHA DMA Interrupt Clear	Section 25.3.4

Complex bit access types are encoded to fit into small table cells. Table 25-26 shows the codes that are used for access types in this section.

Table 25-26. SHA/MD5 μ DMA Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

25.3.1 SHA_DMAIM Register (Offset = 0x10) [reset = 0x0]

SHA DMA Interrupt Mask (SHA_DMAIM)

The SHA DMA Interrupt Mask (SHA_DMA_IM) register controls interrupt behavior and are used to program which interrupts are suppressed.

SHA_DMAIM is shown in [Figure 25-14](#) and described in [Table 25-27](#).

Return to [Summary Table](#).

Figure 25-14. SHA_DMAIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					COUT	DIN	CIN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 25-27. SHA_DMAIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	COUT	R/W	0x0	Context Out DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA completes the output context read from the internal register. 0x0 = The COUT interrupt is suppressed and not sent to the interrupt controller. 0x1 = The COUT interrupt is sent to the interrupt controller.
1	DIN	R/W	0x0	Data In DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA writes the last word of input data to the internal FIFO of the engine. 0x0 = The DIN interrupt is suppressed and not sent to the interrupt controller. 0x1 = The DIN interrupt is sent to the interrupt controller.
0	CIN	R/W	0x0	Context In DMA Done Interrupt Mask. If this bit is unmasked, an interrupt is generated when the μ DMA completes a context write to the internal register. 0x0 = The CIN interrupt is suppressed and not sent to the interrupt controller. 0x1 = The CIN interrupt is sent to the interrupt controller.

25.3.2 SHA_DMARIS Register (Offset = 0x14) [reset = 0x0]

SHA DMA Raw Interrupt Status (SHA_DMARIS)

The SHA DMA Raw Interrupt Status (SHA_DMA_RIS) register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set to 1.

SHA_DMARIS is shown in [Figure 25-15](#) and described in [Table 25-28](#).

Return to [Summary Table](#).

Figure 25-15. SHA_DMARIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					COUT	DIN	CIN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 25-28. SHA_DMARIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	COUT	R/W	0x0	Context Out DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has completed the output context read from the internal register and an interrupt has been triggered and is pending.
1	DIN	R/W	0x0	Data In DMA Done Raw Interrupt Status 0x0 = No Interrupt. 0x1 = The μ DMA has written the last word of input data to the internal FIFO of the engine and an interrupt has been triggered and is pending.
0	CIN	R/W	0x0	Context In DMA Done Raw Interrupt Status 0x0 = No interrupt. 0x1 = The μ DMA has completed a context write to the internal register and an interrupt has been triggered and is pending.

25.3.3 SHA_DMAMIS Register (Offset = 0x18) [reset = 0x0]

SHA DMA Masked Interrupt Status (SHA_DMAMIS)

The SHA DMA Masked Interrupt Status (SHA_DMA_MIS) register displays the raw interrupts that are unmasked in the SHA_DMA_RIS register.

SHA_DMAMIS is shown in [Figure 25-16](#) and described in [Table 25-29](#).

Return to [Summary Table](#).

Figure 25-16. SHA_DMAMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					COUT	DIN	CIN
R-0x0					R-0x0	R-0x0	R-0x0

Table 25-29. SHA_DMAMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	COUT	R	0x0	Context Out DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A COUT interrupt has occurred.
1	DIN	R	0x0	Data In DMA Done Masked Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A DIN interrupt has occurred.
0	CIN	R	0x0	Context In DMA Done Raw Interrupt Status 0x0 = An interrupt has not occurred or is masked. 0x1 = A CIN interrupt has occurred.

25.3.4 SHA_DMAIC Register (Offset = 0x1C) [reset = 0x0]

SHA DMA Interrupt Clear (SHA_DMAIC)

The SHA DMA Interrupt Clear register is used to clear the SHA_DMA_RIS and SHA_DMA_MIS registers by writing a 1 to each register bit.

NOTE: This registers always reads as zero.

SHA_DMAIC is shown in [Figure 25-17](#) and described in [Table 25-30](#).

Return to [Summary Table](#).

Figure 25-17. SHA_DMAIC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					COUT	DIN	CIN
R-0x0					W1C-0x0	W1C-0x0	W1C-0x0

Table 25-30. SHA_DMAIC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	COUT	W1C	0x0	Context Out DMA Done Masked Interrupt Status. Writing a 1 to this bit clears the COUT bit in the SHA_DMA_RIS and SHA_DMA_MIS register.
1	DIN	W1C	0x0	Data In DMA Done Interrupt Clear. Writing a 1 to this bit clears the DIN bit in the SHA_DMA_RIS and SHA_DMA_MIS register.
0	CIN	W1C	0x0	Context In DMA Done Raw Interrupt Status. Writing a 1 to this bit clears the CIN bit in the SHA_DMA_RIS and SHA_DMA_MIS register..

Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes the Universal Asynchronous Receivers/Transmitters (UARTs).

Topic	Page
26.1 Introduction	1621
26.2 Block Diagram	1622
26.3 Functional Description	1622
26.4 Initialization and Configuration	1629
26.5 UART Registers	1631

26.1 Introduction

The MSP432E4 controller includes eight Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 7.5 Mbps for regular speed (divide by 16) and 15 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and $\frac{7}{8}$
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics
 - 5, 6, 7, or 8 data bits
 - Even, odd, stick, or no parity bit generation and detection
 - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder and decoder providing
 - Programmable use of IrDA SIR or UART input/output
 - Support of IrDA SIR encoder and decoder functions for data rates up to 115.2 kbps half-duplex
 - Support of normal 3/16 and low-power (1.41 to 2.23 μ s) bit durations
 - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Modem functionality available on the following UARTs:
 - UART0 (modem flow control and modem status)
 - UART1 (modem flow control and modem status)
 - UART2 (modem flow control)
 - UART3 (modem flow control)
 - UART4 (modem flow control)
- EIA-485 9-bit support
- Standard FIFO-level and End-of-Transmission (EOT) interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
 - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate the baud clock.

26.2 Block Diagram

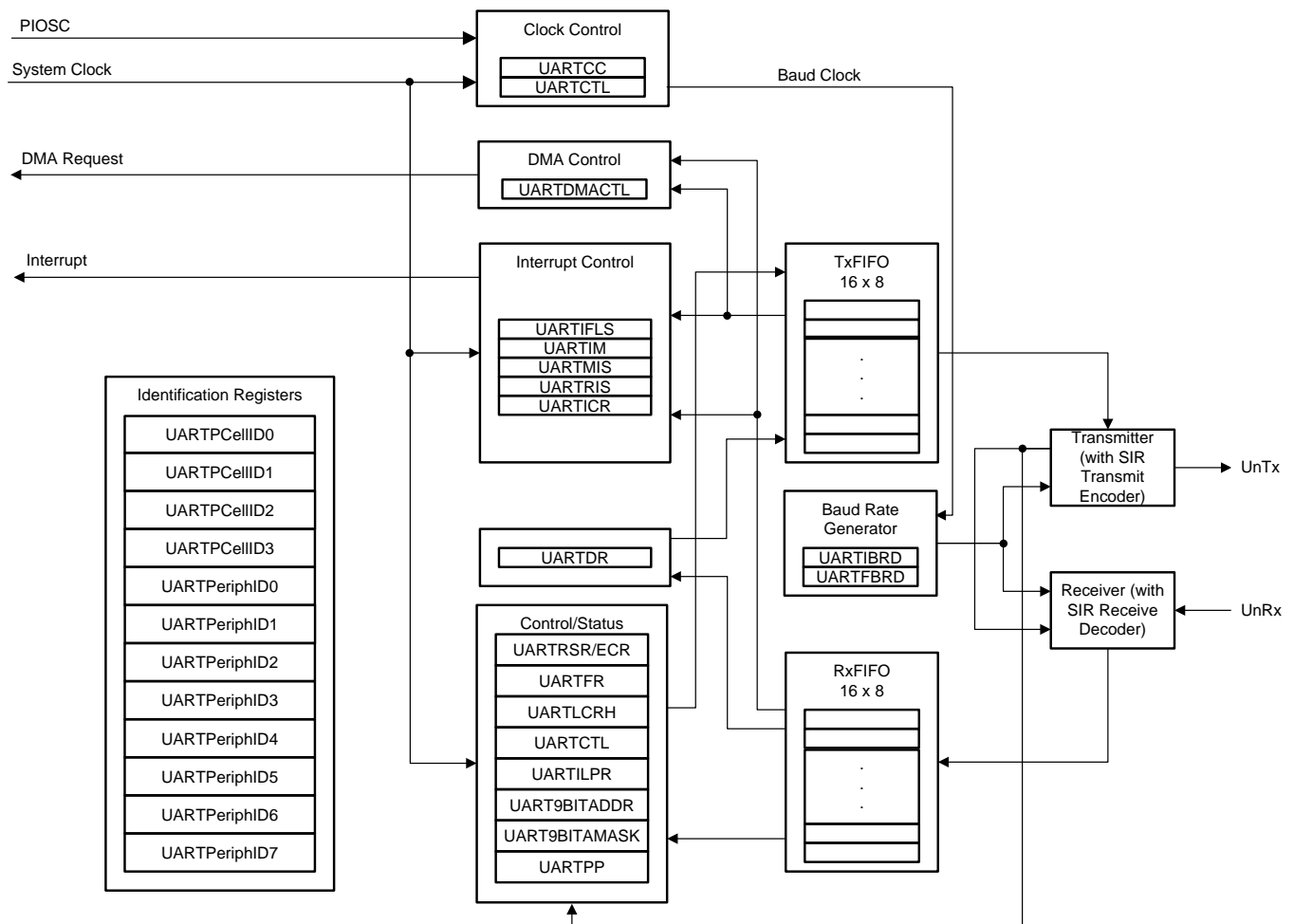


Figure 26-1. UART Module Block Diagram

26.3 Functional Description

Each UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit or receive through the TXE and RXE bits of the UART Control (UARTCTL) register (see [Section 26.5.8](#)). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in UARTCTL. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

The UART module also includes a serial IR (SIR) encoder and decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the UARTCTL register.

26.3.1 Transmit and Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See [Figure 26-2](#) for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

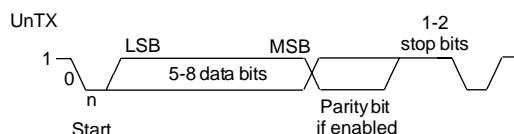


Figure 26-2. UART Character Frame

26.3.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divisor allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the UART Integer Baud-Rate Divisor (UARTIBRD) register (see [Section 26.5.5](#)) and the 6-bit fractional part is loaded with the UART Fractional Baud-Rate Divisor (UARTFBRD) register (see [Section 26.5.6](#)). The baud-rate divisor (BRD) has the following relationship to the system clock (where BRDI is the integer part of the BRD, and BRDF is the fractional part, separated by a decimal place.)

$$\text{BRD} = \text{BRDI} + \text{BRDF} = \text{UARTSysClk} / (\text{ClkDiv} \times \text{Baud Rate})$$

where

- UARTSysClk is the system clock connected to the UART, and ClkDiv is either 16 (if HSE in UARTCTL is clear) or 8 (if HSE is set). (69)

By default, this will be the main system clock described in [Section 4.1.5](#). Alternatively, the UART may be clocked from the internal precision oscillator (PIOSC), independent of the system clock selection. This will allow the UART clock to be programmed independently of the system clock PLL settings. See the UARTCC register for more details.

The 6-bit fractional number (that is to be loaded into the DIVFRAC bit field in the UARTFBRD register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(\text{BRDF} \times 64 + 0.5) \quad (70)$$

The UART generates an internal baud-rate reference clock at 8× or 16× the baud-rate (referred to as Baud8 and Baud16, depending on the setting of the HSE bit [bit 5] in UARTCTL). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during receive operations. Note that the state of the HSE bit has no effect on clock generation in ISO 7816 smart card mode (when the SMART bit in the UARTCTL register is set).

Along with the UART Line Control, High Byte (UARTLCRH) register (see [Section 26.5.7](#)), the UARTIBRD and UARTFBRD registers form an internal 30-bit register. This internal register is only updated when a write operation to UARTLCRH is performed, so any changes to the baud-rate divisor must be followed by a write to the UARTLCRH register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- UARTIBRD write, UARTFBRD write, and UARTLCRH write
- UARTFBRD write, UARTIBRD write, and UARTLCRH write
- UARTIBRD write and UARTLCRH write
- UARTFBRD write and UARTLCRH write

26.3.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the UARTLCRH register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the UART Flag (UARTFR) register (see [Section 26.5.3](#)) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The BUSY bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the UnRx signal is continuously 1), and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of Baud16 or fourth cycle of Baud8 depending on the setting of the HSE bit (bit 5) in UARTCTL (described in [Section 26.3.1](#)).

The start bit is valid and recognized if the UnRx signal is still low on the eighth cycle of Baud16 (HSE clear) or the fourth cycle of Baud8 (HSE set), otherwise it is ignored. After a valid start bit is detected, successive data bits are sampled on every 16th cycle of Baud16 or eighth cycle of Baud8 (that is, one bit period later) according to the programmed length of the data characters and value of the HSE bit in UARTCTL. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the UARTLCRH register.

Lastly, a valid stop bit is confirmed if the UnRx signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

26.3.4 Serial IR (SIR)

The UART peripheral includes an IrDA SIR encoder and decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 μ s, assuming a nominal 1.8432-MHz frequency) by changing the appropriate bit in the UARTCTL register (see [Section 26.5.8](#)).

Whether the device is in normal or low-power IrDA mode, a start bit is deemed valid if the decoder is still low one period of IrLPBaud16 after the low was first detected. This enables a normal-mode UART to receive data from a low-power mode UART that can transmit pulses as small as 1.41 μ s. Thus, for both low-power and normal mode operation, the ILPDVSR field in the UARTILPR register must be programmed such that $1.42 \text{ MHz} < f_{\text{IrLPBaud16}} < 2.12 \text{ MHz}$, resulting in a low-power pulse duration of 1.41 to 2.11 μ s (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 μ s are accepted as valid pulses.

[Figure 26-3](#) shows the UART transmit and receive signals, with and without IrDA modulation.

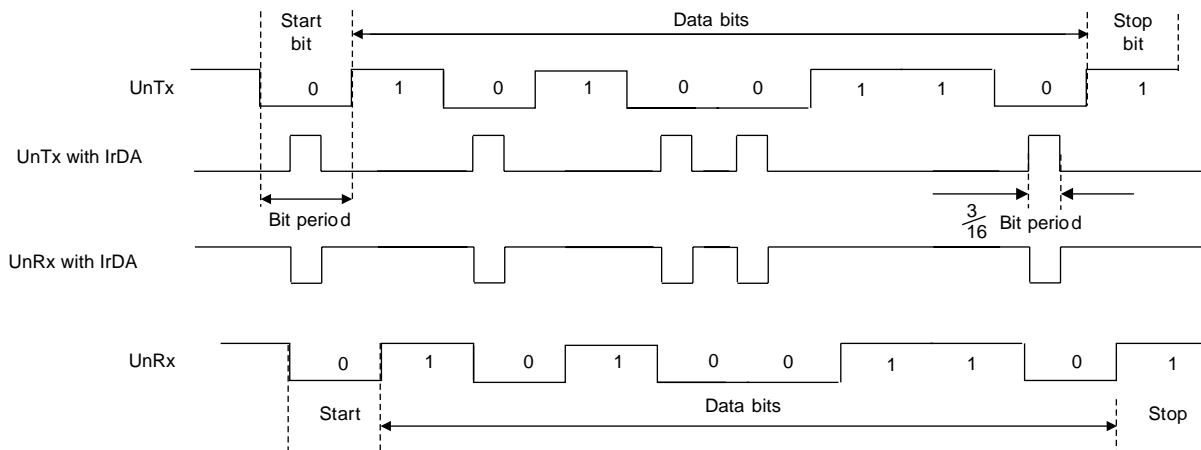


Figure 26-3. IrDA Data Modulation

In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10-ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time.

NOTE: When using SIR mode and the μ DMA, only single transfer mode is supported. To ensure single transfer mode, clear the respective SETn bit DMAUSEBURSTSET register for the μ DMA channel that is mapped to the UART.

26.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (SMART) of the UARTCTL register is set, the UnTx signal is used as a bit clock, and the UnRx signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design. The maximum clock rate in this mode is system clock / 16.

When using ISO 7816 mode, the UARTLCRH register must be set to transmit 8-bit words (WLEN bits 6:5 configured to 0x3) with EVEN parity (PEN set and EPS set). In this mode, the UART automatically uses two stop bits, and the STP2 bit of the UARTLCRH register is ignored.

If a parity error is detected during transmission, UnRx is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

26.3.6 Modem Handshake Support

This section describes how to configure and use the modem flow control and status signals for a UART when connected as a data terminal equipment (DTE) or as a data communications equipment (DCE). In general, a modem is a DCE and a computing device that connects to a modem is the DTE.

26.3.6.1 Signaling

The status signals provided by a UART differ based on whether the UART is used as a DTE or DCE. When used as a DTE, the modem flow control and status signals are defined as:

- UnCTS: clear to send
- UnDSR: data set ready
- UnDCD: data carrier detect
- UnRI: ring indicator
- UnRTS: request to send
- UnDTR: data terminal ready

When used as a DCE, the modem flow control and status signals are defined as:

- UnCTS: request to send
- UnDSR: data terminal ready
- UnRTS: clear to send
- UnDTR: data set ready

Note that the support for DCE functions data carrier detect and ring indicator are not provided. If these signals are required, their function can be emulated by using a GPIO signal and providing software support.

26.3.6.2 Flow Control

Flow control can be accomplished by either hardware or software. The following sections describe the different methods.

26.3.6.2.1 Hardware Flow Control (RTS and CTS)

Hardware flow control between two devices is accomplished by connecting the UnRTS output to the clear-to-send input on the receiving device, and connecting the request-to-send output on the receiving device to the UnCTS input.

The UnCTS input controls the transmitter. The transmitter may only transmit data when the UnCTS input is asserted (active low). The UnRTS output signal indicates the state of the receive FIFO. UnCTS remains asserted (active low) until the preprogrammed watermark level is reached, indicating that the Receive FIFO has no space to store additional characters.

The UARTCTL register bits 15 (CTSEN) and 14 (RTSEN) specify the flow control mode as shown in [Table 26-1](#).

Table 26-1. Flow Control Mode

CTSEN	RTSEN	Description
1	1	RTS and CTS flow control enabled
1	0	Only CTS flow control enabled
0	1	Only RTS flow control enabled
0	0	Both RTS and CTS flow control disabled

When RTSEN is 1, software cannot modify the UnRTS output value through the UARTCTL register request-to-send (RTS) bit, and the status of the RTS bit should be ignored.

26.3.6.2.2 Software Flow Control (Modem Status Interrupts)

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts may be generated for the UnDSR, UnDCD, UnCTS, and UnRI signals using bits 3:0 of the UARTIM register, respectively. The raw and masked interrupt status may be checked using the UARTRIS and UARTMIS register. These interrupts may be cleared using the UARTICR register.

26.3.7 9-Bit UART Mode

The UART provides a 9-bit mode that is enabled with the 9BITEN bit in the UART9BITADDR register. This feature is useful in a multi-drop configuration of the UART where a single master connected to multiple slaves can communicate with a particular slave through its address or set of addresses along with a qualifier for an address byte. All the slaves check for the address qualifier in the place of the parity bit and, if set, then compare the byte received with the preprogrammed address. If the address matches, then it receives or sends further data. If the address does not match, it drops the address byte and any subsequent data bytes. If the UART is in 9-bit mode, then the receiver operates with no parity mode. The address can be predefined to match with the received byte and it can be configured with the UART9BITADDR register. The matching can be extended to a set of addresses using the address mask in the UART9BITAMASK register. By default, the UART9BITAMASK is 0xFF, meaning that only the specified address is matched.

When not finding a match, the rest of the data bytes with the ninth bit cleared are dropped. If a match is found, then an interrupt is generated to the NVIC for further action. The subsequent data bytes with the cleared ninth bit are stored in the FIFO. Software can mask this interrupt in case μ DMA or FIFO operations are enabled for this instance and processor intervention is not required. All the send transactions with 9-bit mode are data bytes and the ninth bit is cleared. Software can override the ninth bit to be set (to indicate address) by overriding the parity settings to sticky parity with odd parity enabled for a particular byte. To match the transmission time with correct parity settings, the address byte can be transmitted as a single then a burst transfer. The transmit FIFO does not hold the address/data bit, hence software should enable the address bit appropriately.

26.3.8 FIFO Operation

The UART has two 16x8 FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the UART Data (UARTDR) register (see [Section 26.5.1](#)). Read operations of the UARTDR register return a 12-bit value consisting of eight data bits and four error flags while write operations place 8-bit data in the transmit FIFO.

Out of reset, both FIFOs are disabled and act as 1-byte-deep holding registers. The FIFOs are enabled by setting the FEN bit in UARTLCRH ([Section 26.5.7](#)).

FIFO status can be monitored through the UART Flag (UARTFR) register (see [Section 26.5.3](#)) and the UART Receive Status (UARTRSR) register. Hardware monitors empty, full, and overrun conditions. The UARTFR register contains empty and full flags (TXFE, TXFF, RXFE, and RXFF bits), and the UARTRSR register shows overrun status with the OE bit. If the FIFOs are disabled, the empty and full flags are set according to the status of the 1-byte-deep holding registers.

The trigger points at which the FIFOs generate interrupts is controlled via the UART Interrupt FIFO Level Select (UARTIFLS) register (see [Section 26.5.9](#)). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and $\frac{7}{8}$. For example, if the $\frac{1}{4}$ option is selected for the receive FIFO, the UART generates a receive interrupt after four data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the $\frac{1}{2}$ mark.

26.3.9 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun error
- Break error
- Parity error
- Framing error
- Receive time-out
- Transmit (when condition defined in the TXIFLSEL bit in the UARTIFLS register is met, or if the EOT bit in UARTCTL is set, when the last bit of all transmitted data leaves the serializer)
- Receive (when condition defined in the RXIFLSEL bit in the UARTIFLS register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the UART Masked Interrupt Status (UARTMIS) register (see [Section 26.5.12](#)).

The interrupt events that can trigger a controller-level interrupt are defined in the UART Interrupt Mask (UARTIM) register (see [Section 26.5.10](#)) by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is visible via the UART Raw Interrupt Status (UARTRIS) register (see [Section 26.5.11](#)).

NOTE: For receive time-out, the RTIM bit in the UARTIM register must be set to see the RTMIS and RTRIS status in the UARTMIS and UARTRIS registers.

Interrupts are always cleared (for the UARTMIS and UARTRIS registers) by writing a 1 to the corresponding bit in the UART Interrupt Clear (UARTICR) register (see [Section 26.5.13](#)).

The receive time-out interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period when the HSE bit is clear or over a 64-bit period when the HSE bit is set. The receive time-out interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the UARTICR register.

The receive interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the RXRIS bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt by writing a 1 to the RXIC bit.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXRIS bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt by writing a 1 to the RXIC bit.

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXRIS bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt by writing a 1 to the TXIC bit.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXRIS bit is set. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt by writing a 1 to the TXIC bit.

26.3.10 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the LBE bit in the UARTCTL register (see [Section 26.5.8](#)). In loopback mode, data transmitted on the UnTx output is received on the UnRx input. Note that the LBE bit should be set before the UART is enabled.

26.3.11 DMA Operation

The UART provides an interface to the μ DMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the UART DMA Control (UARTDMACTL) register. When DMA operation is enabled, the UART asserts a DMA request on the receive or transmit channel when the associated FIFO can transfer data.

For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level configured in the UARTIFLS register.

For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the μ DMA controller depending on how the DMA channel is configured.

NOTE: When using SIR mode (see [Section 26.3.4](#)), transfers can be done only in single transfer mode. To ensure single transfer mode, clear the respective SETn bit DMAUSEBURSTSET register for the μ DMA channel that is mapped to the UART.

To enable DMA operation for the receive channel, set the RXDMAE bit of the DMA Control (UARTDMACTL) register. To enable DMA operation for the transmit channel, set the TXDMAE bit of the UARTDMACTL register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of the UARTDMACR register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

When the μ DMA is finished transferring data to the TX FIFO or from the RX FIFO, a dma_done signal is sent to the UART to indicate completion. The dma_done status is indicated through the DMATXRIS and DMARXIS bits of the UARTRIS register. An interrupt can be generated from these status bits by setting the DMATXIM and DMARXIM bits in the UARTIM register.

NOTE: The DMATXRIS bit can be used to indicate the completion of data transfer from the μ DMA to the TX FIFO. To indicate transfer completion from the serializer of the UART, the EOT bit should be enabled in the UARTCTL register. An interrupt can be generated on an EOT completion by setting the EOTIM bit of the UARTIM register.

See [Chapter 8](#) for more details about programming the μ DMA controller.

26.4 Initialization and Configuration

To enable and initialize the UART, perform the following steps:

1. Enable the UART module using the RCGCUART register (see [Section 4.2.91](#)).
2. Enable the clock to the appropriate GPIO module through the RCGCGPIO register (see [Section 4.2.87](#)). To find out which GPIO port to enable, see the device-specific data sheet.
3. Set the GPIO AFSEL bits for the appropriate pins (see [Section 17.5.10](#)). To determine which GPIOs to configure, see the device-specific data sheet.
4. Configure the GPIO current level or slew rate as specified for the mode selected (see [Section 17.5.11](#) and [Section 17.5.17](#)).
5. Configure the PMCN fields in the GPIOPCTL register to assign the UART signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).

To use the UART, the peripheral clock must be enabled by setting the appropriate bit in the RCGCUART register ([Section 4.2.91](#)). In addition, the clock to the appropriate GPIO module must be enabled through the RCGCGPIO register ([Section 4.2.87](#)) in the system control module. To find out which GPIO port to enable, see the device-specific data sheet.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz, and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One-stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), because the UARTIBRD and UARTRFBRD registers must be written before the UARTRLCRH register. Using the equation described in [Section 26.3.2](#), the BRD can be calculated:

$$\text{BRD} = 20000000 / (16 \times 115200) = 10.8507 \quad (71)$$

This means that the DIVINT field of the UARTIBRD register (see [Section 26.5.5](#)) should be set to 10 decimal or 0xA. The value to be loaded into the UARTFBRD register (see [Section 26.5.6](#)) is calculated by the equation:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(0.8507 \times 64 + 0.5) = 54 \quad (72)$$

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UARTEN bit in the UARTCTL register.
2. Write the integer portion of the BRD to the UARTIBRD register.
3. Write the fractional portion of the BRD to the UARTFBRD register.
4. Write the desired serial parameters to the UARTLCRH register (in this case, a value of 0x0000.0060).
5. Configure the UART clock source by writing to the UARTCC register.
6. Optionally, configure the μ DMA channel (see [Chapter 8](#)) and enable the DMA options in the UARTDMACTL register.
7. Enable the UART by setting the UARTEN bit in the UARTCTL register.

26.5 UART Registers

[Table 26-2](#) lists the memory-mapped registers for the UART. All register offset addresses not listed in [Table 26-2](#) should be considered as reserved locations and the register contents should not be modified.

The offsets are relative to the base address of each instance of the UART:

- UART0: 0x4000C000
- UART1: 0x4000D000
- UART2: 0x4000E000
- UART3: 0x4000F000
- UART4: 0x40010000
- UART5: 0x40011000
- UART6: 0x40012000
- UART7: 0x40013000

The UART module clock must be enabled before the registers can be programmed. There must be a delay of 3 system clocks after the UART module clock is enabled before any UART module registers are accessed.

The UART must be disabled (see the UARTEN bit in the UARTCTL register in [Section 26.5.8](#)) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

NOTE: Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
 - UART1 (modem flow control and modem status)
 - UART2 (modem flow control)
 - UART3 (modem flow control)
 - UART4 (modem flow control)
-

Table 26-2. UART Registers

Offset	Acronym	Register Name	Section
0x0	UARTDR	UART Data	Section 26.5.1
0x4	UARTSR/UARTCR	UART Receive Status/Error Clear	Section 26.5.2
0x18	UARTFR	UART Flag	Section 26.5.3
0x20	UARTILPR	UART IrDA Low-Power Register	Section 26.5.4
0x24	UARTIBRD	UART Integer Baud-Rate Divisor	Section 26.5.5
0x28	UARTFBRD	UART Fractional Baud-Rate Divisor	Section 26.5.6
0x2C	UARTLCRH	UART Line Control	Section 26.5.7
0x30	UARTCTL	UART Control	Section 26.5.8
0x34	UARTIFLS	UART Interrupt FIFO Level Select	Section 26.5.9
0x38	UARTIM	UART Interrupt Mask	Section 26.5.10
0x3C	UARTISR	UART Raw Interrupt Status	Section 26.5.11
0x40	UARTMIS	UART Masked Interrupt Status	Section 26.5.12
0x44	UARTICR	UART Interrupt Clear	Section 26.5.13
0x48	UARTDMACTL	UART DMA Control	Section 26.5.14
0xA4	UART9BITADDR	UART 9-Bit Self Address	Section 26.5.15
0xA8	UART9BITAMASK	UART 9-Bit Self Address Mask	Section 26.5.16
0xFC0	UARTPP	UART Peripheral Properties	Section 26.5.17
0xFC8	UARTCC	UART Clock Configuration	Section 26.5.18
0xFD0	UARTPeriphID4	UART Peripheral Identification 4	Section 26.5.19
0xFD4	UARTPeriphID5	UART Peripheral Identification 5	Section 26.5.20

Table 26-2. UART Registers (continued)

Offset	Acronym	Register Name	Section
0xFD8	UARTPeriphID6	UART Peripheral Identification 6	Section 26.5.21
0xFDC	UARTPeriphID7	UART Peripheral Identification 7	Section 26.5.22
0xFE0	UARTPeriphID0	UART Peripheral Identification 0	Section 26.5.23
0xFE4	UARTPeriphID1	UART Peripheral Identification 1	Section 26.5.24
0xFE8	UARTPeriphID2	UART Peripheral Identification 2	Section 26.5.25
0xFEC	UARTPeriphID3	UART Peripheral Identification 3	Section 26.5.26
0xFF0	UARTPCellID0	UART PrimeCell Identification 0	Section 26.5.27
0xFF4	UARTPCellID1	UART PrimeCell Identification 1	Section 26.5.28
0xFF8	UARTPCellID2	UART PrimeCell Identification 2	Section 26.5.29
0xFFC	UARTPCellID3	UART PrimeCell Identification 3	Section 26.5.30

Complex bit access types are encoded to fit into small table cells. [Table 26-3](#) shows the codes that are used for access types in this section.

Table 26-3. UART Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-n		Value after reset or the default value

26.5.1 UARTDR Register (Offset = 0x0) [reset = 0x0]

UART Data (UARTDR)

NOTE: This register is read-sensitive.

This register is the data register (the interface to the FIFOs).

For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the FIFO is disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If the FIFO is disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UARTDR is shown in [Figure 26-4](#) and described in [Table 26-4](#).

Return to [Summary Table](#).

Figure 26-4. UARTDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				OE	BE	PE	FE	DATA							
R-0x0				R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R/W-0x0						

Table 26-4. UARTDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0x0	
11	OE	R	0x0	UART Overrun Error. 0x0 = No data has been lost due to a FIFO overrun. 0x1 = New data was received when the FIFO was full, resulting in data loss.
10	BE	R	0x0	UART Break Error. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received. 0x0 = No break condition has occurred 0x1 = A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).
9	PE	R	0x0	UART Parity Error. In FIFO mode, this error is associated with the character at the top of the FIFO. 0x0 = No parity error has occurred 0x1 = The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register.
8	FE	R	0x0	UART Framing Error. 0x0 = No framing error has occurred 0x1 = The received character does not have a valid stop bit (a valid stop bit is 1).
7-0	DATA	R/W	0x0	Data Transmitted or Received. Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.

26.5.2 UARTSR/UARTECR Register (Offset = 0x4) [reset = 0x0]

UART Receive Status/Error Clear (UARTSR/UARTECR)

The UARTSR/UARTECR register is the receive status register/error clear register.

In addition to the UARTDR register, receive status can also be read from the UARTSR register. If the status is read from this register, then the status information corresponds to the entry read from UARTDR prior to reading UARTSR. The status information for overrun is set immediately when an overrun condition occurs.

The UARTSR register cannot be written.

A write of any value to the UARTECR register clears the framing, parity, break, and overrun errors. All the bits are cleared on reset.

UARTSR/UARTECR is shown in [Figure 26-5](#) and described in [Table 26-5](#).

Return to [Summary Table](#).

Figure 26-5. UARTSR/UARTECR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OE	BE	PE	FE
R-0x0												0x0	R-0x0	R-0x0	R-0x0

Table 26-5. UARTSR/UARTECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	OE		0x0	<p>UART Overrun Error.</p> <p>This bit is cleared by a write to UARTECR.</p> <p>The FIFO contents remain valid because no further data is written when the FIFO is full, only the contents of the shift register are overwritten.</p> <p>The CPU must read the data in order to empty the FIFO.</p> <p>0x0 = No data has been lost due to a FIFO overrun.</p> <p>0x1 = New data was received when the FIFO was full, resulting in data loss.</p>
2	BE	R	0x0	<p>UART Break Error.</p> <p>This bit is cleared to 0 by a write to UARTECR.</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p> <p>When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.</p> <p>0x0 = No break condition has occurred</p> <p>0x1 = A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).</p>
1	PE	R	0x0	<p>UART Parity Error.</p> <p>This bit is cleared to 0 by a write to UARTECR.</p> <p>0x0 = No parity error has occurred</p> <p>0x1 = The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTECRH register.</p>
0	FE	R	0x0	<p>UART Framing Error.</p> <p>This bit is cleared to 0 by a write to UARTECR.</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p> <p>0x0 = No framing error has occurred</p> <p>0x1 = The received character does not have a valid stop bit (a valid stop bit is 1).</p>

26.5.3 UARTFR Register (Offset = 0x18) [reset = 0x90]

UART Flag (UARTFR)

The UARTFR register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1.

NOTE: Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

UARTFR is shown in [Figure 26-6](#) and described in [Table 26-6](#).

Return to [Summary Table](#).

Figure 26-6. UARTFR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							RI
R-0x0							R-0x0
7	6	5	4	3	2	1	0
TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS
R-0x1	R-0x0	R-0x0	R-0x1	R-0x0	R-0x0	R-0x0	R-0x0

Table 26-6. UARTFR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	RI	R	0x0	Ring Indicator 0x0 = The UnRI signal is not asserted. 0x1 = The UnRI signal is asserted.
7	TXFE	R	0x1	UART Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register. 0x0 = The transmitter has data to transmit. 0x1 = If the FIFO is disabled (FEN is 0), the transmit holding register is empty.If the FIFO is enabled (FEN is 1), the transmit FIFO is empty.
6	RXFF	R	0x0	UART Receive FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register. 0x0 = The receiver can receive data. 0x1 = If the FIFO is disabled (FEN is 0), the receive holding register is full.If the FIFO is enabled (FEN is 1), the receive FIFO is full.
5	TXFF	R	0x0	UART Transmit FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register. 0x0 = The transmitter is not full. 0x1 = If the FIFO is disabled (FEN is 0), the transmit holding register is full.If the FIFO is enabled (FEN is 1), the transmit FIFO is full.

Table 26-6. UARTFR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	RXFE	R	0x1	UART Receive FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register. 0x0 = The receiver is not empty. 0x1 = If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.
3	BUSY	R	0x0	UART Busy. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled). 0x0 = The UART is not busy. 0x1 = The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.
2	DCD	R	0x0	Data Carrier Detect 0x0 = The UnDCD signal is not asserted. 0x1 = The UnDCD signal is asserted.
1	DSR	R	0x0	Data Set Ready 0x0 = The UnDSR signal is not asserted. 0x1 = The UnDSR signal is asserted.
0	CTS	R	0x0	Clear To Send 0x0 = The U1CTS signal is not asserted. 0x1 = The UnCTS signal is asserted.

26.5.4 UARTILPR Register (Offset = 0x20) [reset = 0x0]

UART IrDA Low-Power Register (UARTILPR)

The UARTILPR register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal IrLPBaud16 clock is generated by dividing down SysClk according to the low-power divisor value written to UARTILPR. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the IrLPBaud16 clock. The low-power divisor value is calculated as follows:

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

Because the IrLPBaud16 clock is used to sample transmitted data irrespective of mode, the ILPDVSR field must be programmed in both low power and normal mode, such that $1.42 \text{ MHz} < F_{IrLPBaud16} < 2.12 \text{ MHz}$, resulting in a low-power pulse duration of 1.41-2.11 μs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 μs are accepted as valid pulses.

NOTE: Zero is an illegal value. Programming a zero value results in no IrLPBaud16 pulses being generated.

UARTILPR is shown in [Figure 26-7](#) and described in [Table 26-7](#).

Return to [Summary Table](#).

Figure 26-7. UARTILPR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ILPDVSR							
R-0x0																								R/W-0x0							

Table 26-7. UARTILPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	ILPDVSR	R/W	0x0	IrDA Low-Power Divisor. This field contains the 8-bit low-power divisor value.

26.5.5 UARTIBRD Register (Offset = 0x24) [reset = 0x0]

UART Integer Baud-Rate Divisor (UARTIBRD)

The UARTIBRD register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when UARTIBRD = 0), in which case the UARTFBRD register is ignored. When changing the UARTIBRD register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the UARTLCRH register. See [Section 26.3.2](#) for configuration details.

UARTIBRD is shown in [Figure 26-8](#) and described in [Table 26-8](#).

Return to [Summary Table](#).

Figure 26-8. UARTIBRD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DIVINT															
R-0x0																R/W-0x0															

Table 26-8. UARTIBRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-0	DIVINT	R/W	0x0	Integer Baud-Rate Divisor

26.5.6 UARTFBRD Register (Offset = 0x28) [reset = 0x0]

UART Fractional Baud-Rate Divisor (UARTFBRD)

The UARTFBRD register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the UARTFBRD register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the UARTLCRH register. See [Section 26.3.2](#) for configuration details.

UARTFBRD is shown in [Figure 26-9](#) and described in [Table 26-9](#).

Return to [Summary Table](#).

Figure 26-9. UARTFBRD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										DIVFRAC					
R-0x0										R/W-0x0					

Table 26-9. UARTFBRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-0	DIVFRAC	R/W	0x0	Fractional Baud-Rate Divisor

26.5.7 UARTLCRH Register (Offset = 0x2C) [reset = 0x0]

UART Line Control (UARTLCRH)

The UARTLCRH register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (UARTIBRD and/or UARTIFRD), the UARTLCRH register must also be written. The write strobe for the baud-rate divisor registers is tied to the UARTLCRH register.

UARTLCRH is shown in [Figure 26-10](#) and described in [Table 26-10](#).

Return to [Summary Table](#).

Figure 26-10. UARTLCRH Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
SPS	WLEN		FEN	STP2	EPS	PEN	BRK
R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 26-10. UARTLCRH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7	SPS	R/W	0x0	UART Stick Parity Select. When bits 1, 2, and 7 of UARTLCRH are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1. When this bit is cleared, stick parity is disabled.
6-5	WLEN	R/W	0x0	UART Word Length. The bits indicate the number of data bits transmitted or received in a frame as follows: 0x0 = 5 bits (default) 0x1 = 6 bits 0x2 = 7 bits 0x3 = 8 bits
4	FEN	R/W	0x0	UART Enable FIFOs. 0x0 = The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers. 0x1 = The transmit and receive FIFO buffers are enabled (FIFO mode).
3	STP2	R/W	0x0	UART Two Stop Bits Select. 0x0 = One stop bit is transmitted at the end of a frame. 0x1 = Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received. When in 7816 smartcard mode (the SMART bit is set in the UARTCTL register), the number of stop bits is forced to 2.

Table 26-10. UARTLCRH Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	EPS	R/W	0x0	<p>UART Even Parity Select.</p> <p>This bit has no effect when parity is disabled by the PEN bit.</p> <p>0x0 = Odd parity is performed, which checks for an odd number of 1s.</p> <p>0x1 = Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.</p>
1	PEN	R/W	0x0	<p>UART Parity Enable.</p> <p>0x0 = Parity is disabled and no parity bit is added to the data frame.</p> <p>0x1 = Parity checking and generation is enabled.</p>
0	BRK	R/W	0x0	<p>UART Send Break.</p> <p>0x0 = Normal use.</p> <p>0x1 = A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods).</p>

26.5.8 UARTCTL Register (Offset = 0x30) [reset = 0x300]

UART Control (UARTCTL)

The UARTCTL register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set.

To enable the UART module, the UARTEN bit must be set. If software requires a configuration change in the module, the UARTEN bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

NOTE: Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

NOTE: The UARTCTL register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the UARTCTL register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by clearing bit 4 (FEN) in the line control register (UARTLCRH).
4. Reprogram the control register.
5. Enable the UART.

UARTCTL is shown in [Figure 26-11](#) and described in [Table 26-11](#).

Return to [Summary Table](#).

Figure 26-11. UARTCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
CTSEN	RTSEN	RESERVED		RTS	DTR	RXE	TXE
R/W-0x0	R/W-0x0	R-0x0		R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x1
7	6	5	4	3	2	1	0
LBE	RESERVED	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 26-11. UARTCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	CTSEN	R/W	0x0	Enable Clear To Send. 0x0 = CTS hardware flow control is disabled. 0x1 = CTS hardware flow control is enabled. Data is only transmitted when the UnCTS signal is asserted.

Table 26-11. UARTCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	RTSEN	R/W	0x0	Enable Request to Send. 0x0 = RTS hardware flow control is disabled. 0x1 = RTS hardware flow control is enabled. Data is only requested (by asserting UnRTS) when the receive FIFO has available entries.
13-12	RESERVED	R	0x0	
11	RTS	R/W	0x0	Request to Send. When RTSEN is clear, the status of this bit is reflected on the U1RTS signal. If RTSEN is set, this bit is ignored on a write and should be ignored on read.
10	DTR	R/W	0x0	Data Terminal Ready. This bit sets the state of the UnDTR output.
9	RXE	R/W	0x1	UART Receive Enable. If the UART is disabled in the middle of a receive, it completes the current character before stopping. To enable reception, the UARTEN bit must also be set. 0x0 = The receive section of the UART is disabled. 0x1 = The receive section of the UART is enabled.
8	TXE	R/W	0x1	UART Transmit Enable. If the UART is disabled in the middle of a transmission, it completes the current character before stopping. To enable transmission, the UARTEN bit must also be set. 0x0 = The transmit section of the UART is disabled. 0x1 = The transmit section of the UART is enabled.
7	LBE	R/W	0x0	UART Loop Back Enable. 0x0 = Normal operation. 0x1 = The UnTx path is fed through the UnRx path.
6	RESERVED	R	0x0	
5	HSE	R/W	0x0	High-Speed Enable. System clock used is also dependent on the baud-rate divisor configuration. The state of this bit has no effect on clock generation in ISO 7816 smart card mode (the SMART bit is set). 0x0 = The UART is clocked using the system clock divided by 16. 0x1 = The UART is clocked using the system clock divided by 8.
4	EOT	R/W	0x0	End of Transmission. This bit determines the behavior of the TXRIS bit in the UARTRIS register. 0x0 = The TXRIS bit is set when the transmit FIFO condition specified in UARTIFLS is met. 0x1 = The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer.
3	SMART	R/W	0x0	ISO 7816 Smart Card Support. The application must ensure that it sets 8-bit word length (WLEN set to 0x3) and even parity (PEN set to 1, EPS set to 1, SPS set to 0) in UARTLCRH when using ISO 7816 mode. In this mode, the value of the STP2 bit in UARTLCRH is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message. 0x0 = Normal operation. 0x1 = The UART operates in Smart Card mode.

Table 26-11. UARTCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	SIRLP	R/W	0x0	<p>UART SIR Low-Power Mode. This bit selects the IrDA encoding mode. Setting this bit uses less power, but might reduce transmission distances.</p> <p>0x0 = Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.</p> <p>0x1 = The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate.</p>
1	SIREN	R/W	0x0	<p>UART SIR Enable.</p> <p>0x0 = Normal operation.</p> <p>0x1 = The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.</p>
0	UARTEN	R/W	0x0	<p>UART Enable.</p> <p>If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.</p> <p>0x0 = The UART is disabled.</p> <p>0x1 = The UART is enabled.</p>

26.5.9 UARTIFLS Register (Offset = 0x34) [reset = 0x12]

UART Interrupt FIFO Level Select (UARTIFLS)

The UARTIFLS register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the UARTRIS register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UARTIFLS is shown in [Figure 26-12](#) and described in [Table 26-12](#).

Return to [Summary Table](#).

Figure 26-12. UARTIFLS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										RXIFLSEL			TXIFLSEL		
R-0x0										R/W-0x2			R/W-0x2		

Table 26-12. UARTIFLS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0x0	
5-3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select. The trigger points for the receive interrupt are as follows: 0x0 = RX FIFO \geq 1/8 full 0x1 = RX FIFO \geq 1/4 full 0x2 = RX FIFO \geq 1/2 full (default) 0x3 = RX FIFO \geq 3/4 full 0x4 = RX FIFO \geq 7/8 full 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved
2-0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select. The trigger points for the transmit interrupt are as follows: If the EOT bit in UARTCTL is set, the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of TXIFLSEL is ignored. 0x0 = TX FIFO \leq 7/8 empty 0x1 = TX FIFO \leq 3/4 empty 0x2 = TX FIFO \leq 1/2 empty (default) 0x3 = TX FIFO \leq 1/4 empty 0x4 = TX FIFO \leq 1/8 empty 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved

26.5.10 UARTIM Register (Offset = 0x38) [reset = 0x0]

UART Interrupt Mask (UARTIM)

The UARTIM register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

NOTE: Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

UARTIM is shown in [Figure 26-13](#) and described in [Table 26-13](#).

Return to [Summary Table](#).

Figure 26-13. UARTIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						DMATXIM	DMARXIM
R-0x0						R/W-0x0	R/W-0x0
15	14	13	12	11	10	9	8
RESERVED			9BITIM	EOTIM	OEIM	BEIM	PEIM
R-0x0			R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 26-13. UARTIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	DMATXIM	R/W	0x0	Transmit DMA Interrupt Mask. 0x0 = The DMATXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the DMATXRIS bit in the UARTRIS register is set.
16	DMARXIM	R/W	0x0	Receive DMA Interrupt Mask. 0x0 = The DMARXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the DMARXRIS bit in the UARTRIS register is set.
15-13	RESERVED	R	0x0	
12	9BITIM	R/W	0x0	9-Bit Mode Interrupt Mask. 0x0 = The 9BITRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the 9BITRIS bit in the UARTRIS register is set.

Table 26-13. UARTIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	EOTIM	R/W	0x0	End of Transmission Interrupt Mask. 0x0 = The EOTRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the EOTRIS bit in the UARTRIS register is set.
10	OEIM	R/W	0x0	UART Overrun Error Interrupt Mask. 0x0 = The OERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the OERIS bit in the UARTRIS register is set.
9	BEIM	R/W	0x0	UART Break Error Interrupt Mask. 0x0 = The BERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the BERIS bit in the UARTRIS register is set.
8	PEIM	R/W	0x0	UART Parity Error Interrupt Mask. 0x0 = The PERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the PERIS bit in the UARTRIS register is set.
7	FEIM	R/W	0x0	UART Framing Error Interrupt Mask. 0x0 = The FERIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the FERIS bit in the UARTRIS register is set.
6	RTIM	R/W	0x0	UART Receive Time-Out Interrupt Mask. 0x0 = The RTRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RTRIS bit in the UARTRIS register is set.
5	TXIM	R/W	0x0	UART Transmit Interrupt Mask. 0x0 = The TXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the TXRIS bit in the UARTRIS register is set.
4	RXIM	R/W	0x0	UART Receive Interrupt Mask. 0x0 = The RXRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RXRIS bit in the UARTRIS register is set.
3	DSRIM	R/W	0x0	UART Data Set Ready Modem Interrupt Mask. 0x0 = The DSRRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the DSRRIS bit in the UARTRIS register is set.
2	DCDIM	R/W	0x0	UART Data Carrier Detect Modem Interrupt Mask. 0x0 = The DCDRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the DCDRIS bit in the UARTRIS register is set.
1	CTSIM	R/W	0x0	UART Clear to Send Modem Interrupt Mask. 0x0 = The CTSRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the CTSRIS bit in the UARTRIS register is set.

Table 26-13. UARTIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RIIM	R/W	0x0	UART Ring Indicator Modem Interrupt Mask. 0x0 = The RIRIS interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RIRIS bit in the UARTRIS register is set.

26.5.11 UARTRIS Register (Offset = 0x3C) [reset = 0x0]

UART Raw Interrupt Status (UARTRIS)

The UARTRIS register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UARTRIS is shown in [Figure 26-14](#) and described in [Table 26-14](#).

Return to [Summary Table](#).

Figure 26-14. UARTRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						DMATXRIS	DMARXRIS
R-0x0						R-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED			9BITRIS	EOTRIS	OERIS	BERIS	PERIS
R-0x0			R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
FERIS	RTRIS	TXRIS	RXRIS	DSRRIS	DCDRIS	CTSRIS	RIRIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 26-14. UARTRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	DMATXRIS	R	0x0	Transmit DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMATXIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = The transmit DMA has completed.
16	DMARXRIS	R	0x0	Receive DMA Raw Interrupt Status. This bit is cleared by writing a 1 to the DMARXIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = The receive DMA has completed.
15-13	RESERVED	R	0x0	
12	9BITRIS	R	0x0	9-Bit Mode Raw Interrupt Status. This bit is cleared by writing a 1 to the 9BITIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = A receive address match has occurred.
11	EOTRIS	R	0x0	End of Transmission Raw Interrupt Status. This bit is cleared by writing a 1 to the EOTIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = The last bit of all transmitted data and flags has left the serializer.
10	OERIS	R	0x0	UART Overrun Error Raw Interrupt Status. This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = An overrun error has occurred.

Table 26-14. UARTRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	BERIS	R	0x0	UART Break Error Raw Interrupt Status. This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = A break error has occurred.
8	PERIS	R	0x0	UART Parity Error Raw Interrupt Status This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = A parity error has occurred.
7	FERIS	R	0x0	UART Framing Error Raw Interrupt Status. This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = A framing error has occurred.
6	RTRIS	R	0x0	UART Receive Time-Out Raw Interrupt Status. This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register. For receive timeout, the RTIM bit in the UARTIM register must be set to see the RTRIS status. 0x0 = No interrupt 0x1 = A receive time out has occurred.
5	TXRIS	R	0x0	UART Transmit Raw Interrupt Status. This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled. 0x0 = No interrupt 0x1 = If the EOT bit in the UARTCTL register is clear, the transmit FIFO level has passed through the condition defined in the UARTIFLS register. If the EOT bit is set, the last bit of all transmitted data and flags has left the serializer.
4	RXRIS	R	0x0	UART Receive Raw Interrupt Status. This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled. 0x0 = No interrupt 0x1 = The receive FIFO level has passed through the condition defined in the UARTIFLS register.
3	DSRRIS	R	0x0	UART Data Set Ready Modem Raw Interrupt Status. This bit is cleared by writing a 1 to the DSRIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = Data Set Ready used for software flow control.
2	DCDRIS	R	0x0	UART Data Carrier Detect Modem Raw Interrupt Status. This bit is cleared by writing a 1 to the DCDIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = Data Carrier Detect used for software flow control.
1	CTSRIS	R	0x0	UART Clear to Send Modem Raw Interrupt Status. This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = Clear to Send used for software flow control.
0	RIRIS	R	0x0	UART Ring Indicator Modem Raw Interrupt Status. This bit is cleared by writing a 1 to the RIIC bit in the UARTICR register. 0x0 = No interrupt 0x1 = Ring Indicator used for software flow control.

26.5.12 UARTMIS Register (Offset = 0x40) [reset = 0x0]

UART Masked Interrupt Status (UARTMIS)

The UARTMIS register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UARTMIS is shown in [Figure 26-15](#) and described in [Table 26-15](#).

Return to [Summary Table](#).

Figure 26-15. UARTMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						DMATXMIS	DMARXMIS
R-0x0						R-0x0	R-0x0
15	14	13	12	11	10	9	8
RESERVED			9BITMIS	EOTMIS	OEMIS	BEMIS	PEMIS
R-0x0			R-0x0	R-0x0	R-0x0	R-0x0	R-0x0
7	6	5	4	3	2	1	0
FEMIS	RTMIS	TXMIS	RXMIS	DSRMIS	DCDMIS	CTSMIS	RIMIS
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 26-15. UARTMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	DMATXMIS	R	0x0	Transmit DMA Masked Interrupt Status. This bit is cleared by writing a 1 to the DMATXIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the completion of the transmit DMA.
16	DMARXMIS	R	0x0	Receive DMA Masked Interrupt Status. This bit is cleared by writing a 1 to the DMARXIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the completion of the receive DMA.
15-13	RESERVED	R	0x0	
12	9BITMIS	R	0x0	9-Bit Mode Masked Interrupt Status. This bit is cleared by writing a 1 to the 9BITIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a receive address match.
11	EOTMIS	R	0x0	End of Transmission Masked Interrupt Status. This bit is cleared by writing a 1 to the EOTIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to the transmission of the last data bit.
10	OEMIS	R	0x0	UART Overrun Error Masked Interrupt Status. This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to an overrun error.

Table 26-15. UARTMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	BEMIS	R	0x0	UART Break Error Masked Interrupt Status. This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a break error.
8	PEMIS	R	0x0	UART Parity Error Masked Interrupt Status. This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a parity error.
7	FEMIS	R	0x0	UART Framing Error Masked Interrupt Status. This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a framing error.
6	RTMIS	R	0x0	UART Receive Time-Out Masked Interrupt Status. This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register. For receive timeout, the RTIM bit in the UARTIM register must be set to see the RTMIS status. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to a receive time out.
5	TXMIS	R	0x0	UART Transmit Masked Interrupt Status. This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).
4	RXMIS	R	0x0	UART Receive Masked Interrupt Status. This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to passing through the specified receive FIFO level.
3	DSRMIS	R	0x0	UART Data Set Ready Modem Masked Interrupt Status. This bit is cleared by writing a 1 to the DSRIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to Data Set Ready.
2	DCDMIS	R	0x0	UART Data Carrier Detect Modem Masked Interrupt Status. This bit is cleared by writing a 1 to the DCDIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to Data Carrier Detect.
1	CTSMIS	R	0x0	UART Clear to Send Modem Masked Interrupt Status. This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to Clear to Send.
0	RIMIS	R	0x0	UART Ring Indicator Modem Masked Interrupt Status. This bit is cleared by writing a 1 to the RIIC bit in the UARTICR register. 0x0 = An interrupt has not occurred or is masked. 0x1 = An unmasked interrupt was signaled due to Ring Indicator.

26.5.13 UARTICR Register (Offset = 0x44) [reset = 0x0]

UART Interrupt Clear (UARTICR)

The UARTICR register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UARTICR is shown in [Figure 26-16](#) and described in [Table 26-16](#).

Return to [Summary Table](#).

Figure 26-16. UARTICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED						DMATXIC	DMARXIC
R-0x0						W1C-0x0	W1C-0x0
15	14	13	12	11	10	9	8
RESERVED			9BITIC	EOTIC	OEIC	BEIC	PEIC
R-0x0			R/W-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0
7	6	5	4	3	2	1	0
FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC
W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0	W1C-0x0

Table 26-16. UARTICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0x0	
17	DMATXIC	W1C	0x0	Transmit DMA Interrupt Clear. Writing a 1 to this bit clears the DMATXRIS bit in the UARTRIS register and the DMATXMIS bit in the UARTMIS register.
16	DMARXIC	W1C	0x0	Receive DMA Interrupt Clear. Writing a 1 to this bit clears the DMARXRIS bit in the UARTRIS register and the DMARXMIS bit in the UARTMIS register.
15-13	RESERVED	R	0x0	
12	9BITIC	R/W	0x0	9-Bit Mode Interrupt Clear. Writing a 1 to this bit clears the 9BITRIS bit in the UARTRIS register and the 9BITMIS bit in the UARTMIS register.
11	EOTIC	W1C	0x0	End of Transmission Interrupt Clear. Writing a 1 to this bit clears the EOTRIS bit in the UARTRIS register and the EOTMIS bit in the UARTMIS register.
10	OEIC	W1C	0x0	Overrun Error Interrupt Clear. Writing a 1 to this bit clears the OERIS bit in the UARTRIS register and the OEMIS bit in the UARTMIS register.
9	BEIC	W1C	0x0	Break Error Interrupt Clear. Writing a 1 to this bit clears the BERIS bit in the UARTRIS register and the BEMIS bit in the UARTMIS register.
8	PEIC	W1C	0x0	Parity Error Interrupt Clear. Writing a 1 to this bit clears the PERIS bit in the UARTRIS register and the PEMIS bit in the UARTMIS register.
7	FEIC	W1C	0x0	Framing Error Interrupt Clear. Writing a 1 to this bit clears the FERIS bit in the UARTRIS register and the FEMIS bit in the UARTMIS register.
6	RTIC	W1C	0x0	Receive Time-Out Interrupt Clear. Writing a 1 to this bit clears the RTRIS bit in the UARTRIS register and the RTMIS bit in the UARTMIS register.

Table 26-16. UARTICR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	TXIC	W1C	0x0	Transmit Interrupt Clear. Writing a 1 to this bit clears the TXRIS bit in the UARTRIS register and the TXMIS bit in the UARTMIS register.
4	RXIC	W1C	0x0	Receive Interrupt Clear. Writing a 1 to this bit clears the RXRIS bit in the UARTRIS register and the RXMIS bit in the UARTMIS register.
3	DSRMIC	W1C	0x0	UART Data Set Ready Modem Interrupt Clear Writing a 1 to this bit clears the DSRRIS bit in the UARTRIS register and the DSRMIS bit in the UARTMIS register.
2	DCDMIC	W1C	0x0	UART Data Carrier Detect Modem Interrupt Clear. Writing a 1 to this bit clears the DCDRIS bit in the UARTRIS register and the DCDMIS bit in the UARTMIS register.
1	CTSMIC	W1C	0x0	UART Clear to Send Modem Interrupt Clear. Writing a 1 to this bit clears the CTSRIS bit in the UARTRIS register and the CTSMIS bit in the UARTMIS register.
0	RIMIC	W1C	0x0	UART Ring Indicator Modem Interrupt Clear. Writing a 1 to this bit clears the RIRIS bit in the UARTRIS register and the RIMIS bit in the UARTMIS register.

26.5.14 UARTDMACTL Register (Offset = 0x48) [reset = 0x0]

UART DMA Control (UARTDMACTL)

The UARTDMACTL register is the DMA control register.

UARTDMACTL is shown in [Figure 26-17](#) and described in [Table 26-17](#).

Return to [Summary Table](#).

Figure 26-17. UARTDMACTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					DMAERR	TXDMAE	RXDMAE
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 26-17. UARTDMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2	DMAERR	R/W	0x0	DMA on Error 0x0 = μ DMA receive requests are unaffected when a receive error occurs. 0x1 = μ DMA receive requests are automatically disabled when a receive error occurs.
1	TXDMAE	R/W	0x0	Transmit DMA Enable 0x0 = μ DMA for the transmit FIFO is disabled. 0x1 = μ DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0x0	Receive DMA Enable 0x0 = μ DMA for the receive FIFO is disabled. 0x1 = μ DMA for the receive FIFO is enabled.

26.5.15 UART9BITADDR Register (Offset = 0xA4) [reset = 0x0]

UART 9-Bit Self Address (UART9BITADDR)

The UART9BITADDR register is used to write the specific address that should be matched with the receiving byte when the 9-bit Address Mask (UART9BITAMASK) is set to 0xFF. This register is used in conjunction with UART9BITAMASK to form a match for address-byte received.

UART9BITADDR is shown in [Figure 26-18](#) and described in [Table 26-18](#).

Return to [Summary Table](#).

Figure 26-18. UART9BITADDR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
9BITEN	RESERVED						
R/W-0x0	R-0x0						
7	6	5	4	3	2	1	0
ADDR							
R/W-0x0							

Table 26-18. UART9BITADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15	9BITEN	R/W	0x0	Enable 9-Bit Mode 0x0 = 9-bit mode is disabled. 0x1 = 9-bit mode is enabled.
14-8	RESERVED	R	0x0	
7-0	ADDR	R/W	0x0	Self Address for 9-Bit Mode. This field contains the address that should be matched when UART9BITAMASK is 0xFF.

26.5.16 UART9BITAMASK Register (Offset = 0xA8) [reset = 0xFF]

UART 9-Bit Self Address Mask (UART9BITAMASK)

The UART9BITAMASK register is used to enable the address mask for 9-bit mode. The address bits are masked to create a set of addresses to be matched with the received address byte.

UART9BITAMASK is shown in [Figure 26-19](#) and described in [Table 26-19](#).

Return to [Summary Table](#).

Figure 26-19. UART9BITAMASK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MASK															
R-0x0																R/W-0xFF															

Table 26-19. UART9BITAMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	MASK	R/W	0xFF	Self Address Mask for 9-Bit Mode. This field contains the address mask that creates a set of addresses that should be matched.

26.5.17 UARTPP Register (Offset = 0xFC0) [reset = 0xF]

UART Peripheral Properties (UARTPP)

The UARTPP register provides information regarding the properties of the UART module.

UARTPP is shown in [Figure 26-20](#) and described in [Table 26-20](#).

Return to [Summary Table](#).

Figure 26-20. UARTPP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												MSE	MS	NB	SC
R-0x0												R-0x1	R-0x1	R-0x1	R-0x1

Table 26-20. UARTPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3	MSE	R	0x1	Modem Support Extended 0x0 = The UART module does not provide extended support for modem control. 0x1 = The UART module provides extended support for modem control including UARTnDTR, UARTnDSR, UARTnDCD, and UARTnRI.
2	MS	R	0x1	Modem Support 0x0 = The UART module does not provide support for modem control. 0x1 = The UART module provides support for modem control including UARTnRTS and UARTnCTS.
1	NB	R	0x1	9-Bit Support 0x0 = The UART module does not provide support for the transmission of 9-bit data for RS-485 support. 0x1 = The UART module provides support for the transmission of 9-bit data for RS-485 support.
0	SC	R	0x1	Smart Card Support 0x0 = The UART module does not provide smart card support. 0x1 = The UART module provides smart card support.

26.5.18 UARTCC Register (Offset = 0xFC8) [reset = 0x0]

UART Clock Configuration (UARTCC)

The UARTCC register controls the baud clock source for the UART module. For more information, see [Section 4.1.5.2.1](#).

NOTE: If the PIOSC is used for the UART baud clock, the system clock frequency must be at least 9 MHz in Run mode.

UARTCC is shown in [Figure 26-21](#) and described in [Table 26-21](#).

Return to [Summary Table](#).

Figure 26-21. UARTCC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												CS			
R-0x0																												R/W-0x0			

Table 26-21. UARTCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0x0	
3-0	CS	R/W	0x0	UART Baud Clock Source. 0x0 = reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved

26.5.19 UARTPeriphID4 Register (Offset = 0xFD0) [reset = 0x60]

UART Peripheral Identification 4 (UARTPeriphID4)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID4 is shown in [Figure 26-22](#) and described in [Table 26-22](#).

Return to [Summary Table](#).

Figure 26-22. UARTPeriphID4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID4															
R-0x0																R-0x60															

Table 26-22. UARTPeriphID4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID4	R	0x60	UART Peripheral ID Register [7:0]. Can be used by software to identify the presence of this peripheral.

26.5.20 UARTPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]

UART Peripheral Identification 5 (UARTPeriphID5)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID5 is shown in [Figure 26-23](#) and described in [Table 26-23](#).

Return to [Summary Table](#).

Figure 26-23. UARTPeriphID5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID5															
R-0x0																R-0x0															

Table 26-23. UARTPeriphID5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID5	R	0x0	UART Peripheral ID Register [15:8]. Can be used by software to identify the presence of this peripheral.

26.5.21 UARTPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]

UART Peripheral Identification 6 (UARTPeriphID6)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID6 is shown in [Figure 26-24](#) and described in [Table 26-24](#).

Return to [Summary Table](#).

Figure 26-24. UARTPeriphID6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID6															
R-0x0																R-0x0															

Table 26-24. UARTPeriphID6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID6	R	0x0	UART Peripheral ID Register [23:16]. Can be used by software to identify the presence of this peripheral.

26.5.22 UARTPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]

UART Peripheral Identification 7 (UARTPeriphID7)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID7 is shown in [Figure 26-25](#) and described in [Table 26-25](#).

Return to [Summary Table](#).

Figure 26-25. UARTPeriphID7 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID7							
R-0x0																								R-0x0							

Table 26-25. UARTPeriphID7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID7	R	0x0	UART Peripheral ID Register [31:24]. Can be used by software to identify the presence of this peripheral.

26.5.23 UARTPeriphID0 Register (Offset = 0xFE0) [reset = 0x11]

UART Peripheral Identification 0 (UARTPeriphID0)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID0 is shown in [Figure 26-26](#) and described in [Table 26-26](#).

Return to [Summary Table](#).

Figure 26-26. UARTPeriphID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PID0																	
R-0x0														R-0x11																	

Table 26-26. UARTPeriphID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID0	R	0x11	UART Peripheral ID Register [7:0]. Can be used by software to identify the presence of this peripheral.

26.5.24 UARTPeriphID1 Register (Offset = 0xFE4) [reset = 0x0]

UART Peripheral Identification 1 (UARTPeriphID1)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID1 is shown in [Figure 26-27](#) and described in [Table 26-27](#).

Return to [Summary Table](#).

Figure 26-27. UARTPeriphID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PID1															
R-0x0																R-0x0															

Table 26-27. UARTPeriphID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID1	R	0x0	UART Peripheral ID Register [15:8]. Can be used by software to identify the presence of this peripheral.

26.5.25 UARTPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]

UART Peripheral Identification 2 (UARTPeriphID2)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID2 is shown in [Figure 26-28](#) and described in [Table 26-28](#).

[Return to Summary Table.](#)

Figure 26-28. UARTPeriphID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID2							
R-0x0																								R-0x18							

Table 26-28. UARTPeriphID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID2	R	0x18	UART Peripheral ID Register [23:16]. Can be used by software to identify the presence of this peripheral.

26.5.26 UARTPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]

UART Peripheral Identification 3 (UARTPeriphID3)

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPeriphID3 is shown in [Figure 26-29](#) and described in [Table 26-29](#).

Return to [Summary Table](#).

Figure 26-29. UARTPeriphID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID3							
R-0x0																								R-0x1							

Table 26-29. UARTPeriphID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID3	R	0x1	UART Peripheral ID Register [31:24]. Can be used by software to identify the presence of this peripheral.

26.5.27 UARTPCelIID0 Register (Offset = 0xFF0) [reset = 0xD]

UART PrimeCell Identification 0 (UARTPCelIID0)

The UARTPCelIIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPCelIID0 is shown in [Figure 26-30](#) and described in [Table 26-30](#).

Return to [Summary Table](#).

Figure 26-30. UARTPCelIID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID0							
R-0x0																								R-0xD							

Table 26-30. UARTPCelIID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID0	R	0xD	UART PrimeCell ID Register [7:0]. Provides software a standard cross-peripheral identification system.

26.5.28 UARTPCelIID1 Register (Offset = 0xFF4) [reset = 0xF0]

UART PrimeCell Identification 1 (UARTPCelIID1)

The UARTPCelIIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPCelIID1 is shown in [Figure 26-31](#) and described in [Table 26-31](#).

Return to [Summary Table](#).

Figure 26-31. UARTPCelIID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID1							
R-0x0																								R-0xF0							

Table 26-31. UARTPCelIID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID1	R	0xF0	UART PrimeCell ID Register [15:8]. Provides software a standard cross-peripheral identification system.

26.5.29 UARTPCelIID2 Register (Offset = 0xFF8) [reset = 0x5]

UART PrimeCell Identification 2 (UARTPCelIID2)

The UARTPCelIIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPCelIID2 is shown in [Figure 26-32](#) and described in [Table 26-32](#).

Return to [Summary Table](#).

Figure 26-32. UARTPCelIID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID2							
R-0x0																								R-0x5							

Table 26-32. UARTPCelIID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID2	R	0x5	UART PrimeCell ID Register [23:16]. Provides software a standard cross-peripheral identification system.

26.5.30 UARTPCelIID3 Register (Offset = 0xFFC) [reset = 0xB1]

UART PrimeCell Identification 3 (UARTPCelIID3)

The UARTPCelIIDn registers are hard-coded and the fields within the registers determine the reset values.

UARTPCelIID3 is shown in [Figure 26-33](#) and described in [Table 26-33](#).

Return to [Summary Table](#).

Figure 26-33. UARTPCelIID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID3							
R-0x0																								R-0xB1							

Table 26-33. UARTPCelIID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID3	R	0xB1	UART PrimeCell ID Register [31:24]. Provides software a standard cross-peripheral identification system.

Universal Serial Bus (USB) Controller

This chapter describes the USB controller.

NOTE: Portions of this chapter are excerpted from Mentor Graphics documentation. Mentor Graphics Proprietary. Used with permission.

Topic	Page
27.1 Introduction	1673
27.2 Block Diagram	1673
27.3 Functional Description	1674
27.4 Initialization and Configuration	1692
27.5 USB Registers	1694

27.1 Introduction

The USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB host, device, or OTG functions. If the integrated ULPI interface is used, the USB can operate at high speed. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. 16 endpoints, including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) and 14 endpoints defined by firmware along with a dynamic sizable FIFO, support multiple packet queueing. USB DMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device start-up. The controller complies with the OTG Session Request Protocol (SRP) and Host Negotiation Protocol (HNP).

The USB module has the following features:

- Complies with USB Implementer's Forum (USB-IF) certification standards
- USB 2.0 high-speed (480 Mbps) operation with the integrated ULPI interface communicating with an external PHY
- Link power management support which uses link-state awareness to reduce power usage
- Four transfer types: control, interrupt, bulk, and isochronous
- 16 endpoints
 - One dedicated control IN endpoint and one dedicated control OUT endpoint
 - Seven configurable IN endpoints and seven configurable OUT endpoints
- 4KB of dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop detection and interrupt
- Integrated USB DMA with bus master capability
 - Up to eight RX Endpoint channels and up to eight TX Endpoint channels are available.
 - Each channel can be separately programmed to operate in different modes
 - Incremental burst transfers of 4, 8, 16, or unspecified length supported

27.2 Block Diagram

Figure 27-1 shows the USB module block diagram.

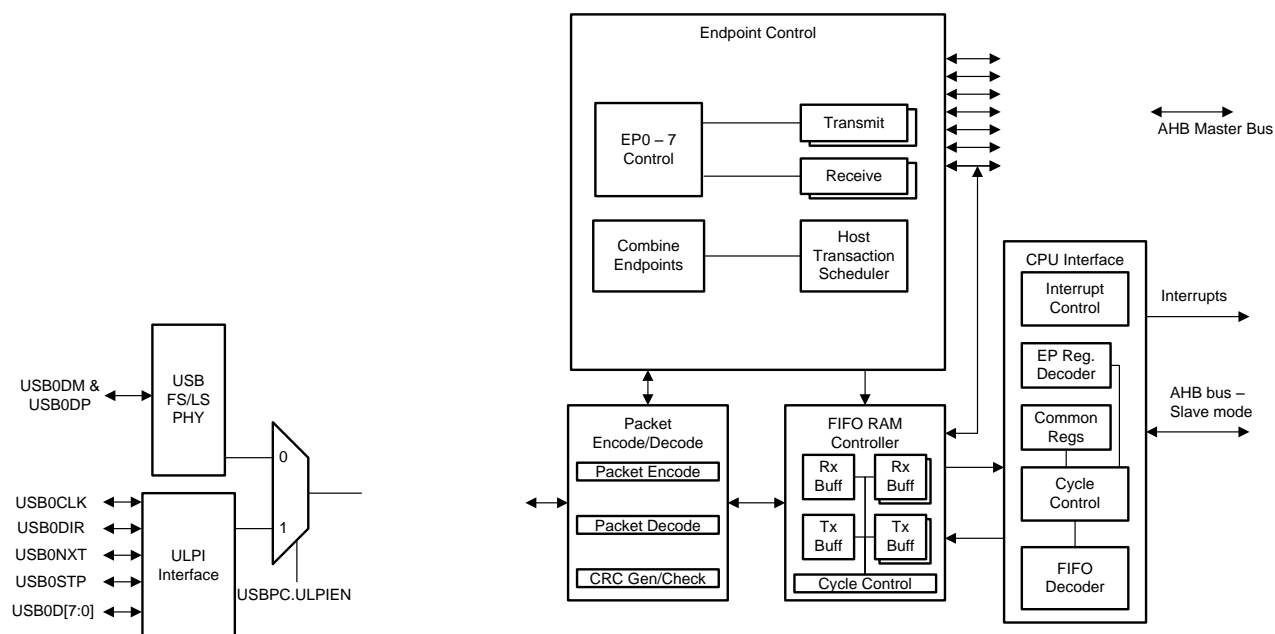


Figure 27-1. USB Module Block Diagram

27.3 Functional Description

The USB controller provides full OTG negotiation by supporting both the Session Request Protocol (SRP) and the Host Negotiation Protocol (HNP). The session request protocol allows devices on the B side of a cable to request the A side device turn on VBUS. The host negotiation protocol is used after the initial session request protocol has powered the bus and provides a method to determine which end of the cable will act as the host controller. When the device is connected to non-OTG peripherals or devices, the controller can detect which cable end was used and provides a register to indicate if the controller should act as the host or the device controller. This indication and the mode of operation are handled automatically by the USB controller. This auto-detection allows the system to use a single A/B connector instead of having both A and B connectors in the system and supports full OTG negotiations with other OTG devices.

In addition, the USB controller provides support for connecting to non-OTG peripherals or host controllers. The USB controller can be configured to act as either a dedicated host or device, in which case, the USB0VBUS and USB0ID signals can be used as GPIOs or any corresponding alternate functions. However, when the USB controller is acting as a self-powered device, a GPIO input or analog comparator input must be connected to VBUS and configured to generate an interrupt when the VBUS level drops. This interrupt is used to disable the pullup resistor on the USB0DP signal.

NOTE: When the USB module is in operation, MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 30 MHz.

27.3.1 Operation as a Device

This section describes the USB controller when it is being used as a USB device. Before the USB controller's operating mode is changed from device to host or host to device, software must reset the USB controller by setting the USB0 bit in the USB Software Reset (SRUSB) register. IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of Start of Frame (SOF) are all described.

When in device mode, IN transactions are controlled by an endpoint's transmit interface and use the transmit endpoint registers for the given endpoint. OUT transactions are handled with an endpoint's receive interface and use the receive endpoint registers for the given endpoint.

When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- Bulk
 - Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- Interrupt
 - Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- Isochronous
 - Isochronous endpoints are more flexible and can be up to 1023 bytes.
- Control
 - It is also possible to specify a separate control endpoint for a USB device. However, in most cases the USB device should use the dedicated control endpoint on the USB controller's endpoint 0.

27.3.1.1 Endpoints

When operating as a device, the USB controller provides two dedicated control endpoints (IN and OUT) and 14 configurable endpoints (seven IN and seven OUT) that can be used for communications with a host controller. The endpoint number and direction associated with an endpoint is directly related to its register designation. For example, when the host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit register interface.

Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the FIFO RAM of the USB controller as a shared memory for both IN and OUT transactions.

The remaining 14 endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as seven configurable IN and seven configurable OUT endpoints. The endpoint pairs are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

27.3.1.2 IN Transactions as a Device

When operating as a USB device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the seven configurable IN endpoints are determined by the USB Transmit FIFO Start Address (USBTXFIFOADD) register. The maximum size of a data packet that may be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the USB Maximum Transmit Data Endpoint n (USBTXMAXPn) register for that endpoint. The FIFO of the endpoint can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size.

NOTE: The maximum packet size set for any endpoint must not exceed the FIFO size. The USBTXMAXPn register should not be written to while data is in the FIFO as unexpected results may occur.

27.3.1.2.1 Single-Packet Buffering

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ) register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the TXRDY bit in the USB Transmit Control and Status Endpoint n Low (USBTXCSRLn) register must be set. If the AUTOSET bit in the USB Transmit Control and Status Endpoint n High (USBTXCSRHn) register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the TXRDY bit must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been successfully sent, both TXRDY and FIFONE are cleared, and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

27.3.1.2.2 Double-Packet Buffering

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the TXRDY bit in the USBTXCSRLn register must be set. If the AUTOSET bit in the USBTXCSRHn register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, TXRDY must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, TXRDY is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and TXRDY set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, TXRDY is automatically cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the FIFONE bit in the USBTXCSRLn register at this point indicates how many packets may be loaded. If the FIFONE bit is set, then another packet is in the FIFO and only one more packet can be loaded. If the FIFONE bit is clear, then no packets are in the FIFO and two more packets can be loaded.

NOTE: Double-packet buffering is disabled if a corresponding EPn bit of an endpoint is set in the USB Transmit Double Packet Buffer Disable (USBTXDPKTBUDIS) register. This bit is set by default, so it must be cleared to enable double-packet buffering.

27.3.1.3 OUT Transactions as a Device

When in device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the seven configurable OUT endpoints are determined by the USB Receive FIFO Start Address (USBRXFIFOADD) register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the USB Maximum Receive Data Endpoint n (USBRXMAXPn) register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size.

NOTE: In all cases, the maximum packet size must not exceed the FIFO size.

27.3.1.3.1 Single-Packet Buffering

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the RXRDY and FULL bits in the USB Receive Control and Status Endpoint n Low (USBRXCSSLn) register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the RXRDY bit must be cleared in order to allow further packets to be received. This action also generates the acknowledge signaling to the host controller. If the AUTOCL bit in the USB Receive Control and Status Endpoint n High (USBRXCSSLn) register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY and FULL bits are cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually.

27.3.1.3.2 Double-Packet Buffering

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the RXRDY bit in the USBXCSSLn register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

NOTE: The FULL bit in USBXCSSLn is not set when the first packet is received. It is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the RXRDY bit must be cleared to allow further packets to be received. If the AUTOCL bit in the USBXCSSLn register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY bit is cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually. If the FULL bit is set when RXRDY is cleared, the USB controller first clears the FULL bit, then sets RXRDY again to indicate that there is another packet waiting in the FIFO to be unloaded.

NOTE: Double-packet buffering is disabled if the corresponding EPn bit of an endpoint is set in the USB Receive Double Packet Buffer Disable (USBRXDPKTBUDIS) register. This bit is set by default, so it must be cleared to enable double-packet buffering.

27.3.1.4 Scheduling

The device has no control over the scheduling of transactions as scheduling is determined by the host controller. The USB controller can set up a transaction at any time. The USB controller waits for the request from the host controller and generates an interrupt when the transaction is complete or if it was terminated due to some error. If the host controller makes a request and the device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

27.3.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the host controller such as when the USB controller automatically stalls a control transfer or unexpected zero length OUT data packets.

27.3.1.5.1 Stalled Control Transfer

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The host sends more data during an OUT data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the DATAEND bit in the USB Control and Status Endpoint 0 Low (USBCTRL0) register has been set.
2. The host requests more data during an IN data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an IN token (instead of an OUT token) after the CPU has cleared TXRDY and set DATAEND in response to the ACK issued by the host to what should have been the last packet.
3. The host sends more than USBRXMAXPn bytes of data with an OUT data token.
4. The host sends more than a zero length data packet for the OUT STATUS phase.

27.3.1.5.2 Zero Length OUT Data Packets

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred.

However, if the host sends a zero-length OUT data packet before the entire length of device request has been transferred, it is signaling the premature end of the transfer. In this case, the USB controller automatically flushes any IN token ready for the data phase from the FIFO and sets the DATAEND bit in the USBCTRL0 register.

27.3.1.5.3 Setting the Device Address

When a host is attempting to enumerate the USB device, it requests that the device change its address from zero to some other value. The address is changed by writing the value that the host requested to the USB Device Functional Address (USBFADDR) register. However, care should be taken when writing to USBFADDR to avoid changing the address before the transaction is complete. This register should only be set after the SET_ADDRESS command is complete. Like all control transactions, the transaction is only complete after the device has left the STATUS phase. In the case of a SET_ADDRESS command, the transaction is completed by responding to the IN request from the host with a zero-byte packet. Once the device has responded to the IN request, the USBFADDR register should be programmed to the new value as soon as possible to avoid missing any new commands sent to the new address.

NOTE: If the USBFADDR register is set to the new value as soon as the device receives the OUT transaction with the SET_ADDRESS command in the packet, it changes the address during the control transfer. In this case, the device does not receive the IN request that allows the USB transaction to exit the STATUS phase of the control transfer because it is sent to the old address. As a result, the host does not get a response to the IN request, and the host fails to enumerate the device.

27.3.1.6 Device Mode SUSPEND

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters SUSPEND mode. If the SUSPEND interrupt has been enabled in the USB Interrupt Enable (USBIE) register, an interrupt is generated at this time. When in SUSPEND mode, the PHY also goes into SUSPEND mode. When RESUME signaling is detected, the USB controller exits SUSPEND mode and takes the PHY out of SUSPEND. If the RESUME interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit SUSPEND mode by setting the RESUME bit in the USB Power (USBPOWER) register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The RESUME bit must be cleared after 10 ms (a maximum of 15 ms) to end RESUME signaling.

To meet USB power requirements, the controller can be put into Deep Sleep mode which keeps the controller in a static state. Hibernation mode should not be used for SUSPEND mode because all internal state information is lost in hibernation.

NOTE: When configured as a self-powered device, the USB module meets the response timing and power draw requirements for USB compliance of SUSPEND mode. When configured as a bus-powered device, the USB can operate in SUSPEND mode but produces a higher power draw than required to be compliant.

27.3.1.7 Start-of-Frame

When the USB controller is operating in device mode, it receives a Start-Of-Frame (SOF) packet from the host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the USB Frame Value (USBFRAME) register, and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, the packet is assumed to have been lost, and the USBFRAME register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

27.3.1.8 USB RESET

When the USB controller is in device mode and a RESET condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the USBFADDR register
- Clears the USB Endpoint Index (USBEPIDX) register
- Flushes all endpoint FIFOs
- Clears all control and status registers
- Enables all endpoint interrupts
- Generates a RESET interrupt

When the application software driving the USB controller receives a RESET interrupt, any open pipes are closed and the USB controller waits for bus enumeration to begin.

27.3.1.9 Connect and Disconnect

The USB controller connection to the USB bus is handled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the SOFTCONN bit of the USBPOWER register. When the SOFTCONN bit is set, the PHY is placed in its normal mode, and the USB0DP and USB0DM lines of the USB bus are enabled. At the same time, the USB controller is placed into a state, in which it does not respond to any USB signaling except a USB RESET.

When the SOFTCONN bit is cleared, the PHY is put into non-driving mode, USB0DP and USB0DM are in a tri-state condition, and the USB controller appears to other devices on the USB bus as if it has been disconnected. The non-driving mode is the default so the USB controller appears disconnected until the SOFTCONN bit has been set. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete, and the system is ready to perform enumeration before connecting to the USB bus. Once the SOFTCONN bit has been set, the USB controller can be disconnected by clearing this bit.

NOTE: The USB controller does not generate an interrupt when the device is connected to the host. However, an interrupt is generated when the host terminates a session.

27.3.2 Operation as a Host

When the USB controller is operating in host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Before the operating mode of the USB controller is changed from host-to-device or device-to-host, software must reset the USB controller by setting the USB0 bit in the USB Software Reset (SRUSB) register. Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB host. Configuration of IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and RESET are all described.

When in host mode, IN transactions are controlled by an endpoint's receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- Bulk
Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- Interrupt
Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- Isochronous
Isochronous endpoints are more flexible and can be up to 1023 bytes.
- Control
It is also possible to specify a separate control endpoint to communicate with a device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with an endpoint 0 of a device.

27.3.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces which communicate with devices that are connected. The endpoints consist of a dedicated control IN endpoint, a dedicated control OUT endpoint, seven configurable OUT endpoints, and seven configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous device endpoints.

These USB interfaces can be used to simultaneously schedule as many as seven independent OUT and seven independent IN transactions to any endpoints on any device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different types of endpoints and different endpoints on devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a bulk OUT endpoint 1 of a device, while the IN portion is communicating with an interrupt IN endpoint 2 of a device.

Before accessing any device, whether for point-to-point communications or for communications through a hub, the relevant USB Receive Functional Address Endpoint *n* (USBRXFUNCADDR_{*n*}) or USB Transmit Functional Address Endpoint *n* (USBTXFUNCADDR_{*n*}) registers must be set for each receive or transmit endpoint to record the address of the device being accessed.

The USB controller also supports connections to devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

27.3.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in device mode except that the transaction first must be initiated by setting the REQPKT bit in the USBCSRL0 register, indicating to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target device. When the packet is received and placed in the receive FIFO, the RXRDY bit in the USBCSRL0 register is set, and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RXRDY must be cleared. The AUTOCL bit in the USBRXCSRH_{*n*} register can be used to have RXRDY automatically cleared when a maximum-sized packet has been unloaded from the FIFO. The AUTORQ bit in USBRXCSRH_{*n*} causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. The AUTOCL and AUTORQ bits can be used with USB DMA accesses to perform complete bulk transfers without main processor intervention. When the RXRDY bit is cleared, the controller sends an acknowledge to the device. When there is a known number of packets to be transferred, the USB Request Packet Count in Block Transfer Endpoint *n* (USBRQPKTCOUNT_{*n*}) register associated with the endpoint should be configured to the number of packets to be transferred. The USB controller decrements the value in the USBRQPKTCOUNT_{*n*} register following each request. When the USBRQPKTCOUNT_{*n*} value decrements to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, USBRQPKTCOUNT_{*n*} should be cleared. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than the MAXLOAD value in the USBRXMAXP_{*n*} register) such as may occur at the end of a bulk transfer.

If the device responds to a bulk or interrupt IN token with a NAK, the USB host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target device responds with a STALL, however, the USB host controller does not retry the transaction but sets the STALLED bit in the USBCSRL0 register. If the target device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, the USB host controller retries the transaction. If after three attempts the target device has still not responded, the USB host controller clears the REQPKT bit and sets the ERROR bit in the USBCSRL0 register.

27.3.2.3 OUT Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in device mode. The TXRDY bit in the USBTXCSRL_{*n*} register must be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the USBTXCSRH_{*n*} register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO. Furthermore, AUTOSET can be used with the USB DMA controller to perform complete bulk transfers without software intervention.

If the target device responds to the OUT token with a NAK, the USB host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the STALLED bit in the USBTXCSRLn register. If the target device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, the USB host controller retries the transaction. If after three attempts the target device has still not responded, the USB controller flushes the FIFO and sets the ERROR bit in the USBTXCSRLn register.

27.3.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB host controller. The host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in one-frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a device is not responding. Isochronous endpoints can be scheduled from every frame to every 2^{16} frames, in powers of 2.

The USB controller maintains a frame counter. If the target device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target device is a low-speed device, aK state is transmitted on the bus to act as a keep-alive to stop the low-speed device from going into SUSPEND mode.

After the SOF packet has been transmitted, the USB host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the REQPKT bit is set or a transmit endpoint for which the TXRDY bit and the FIFONE bit is set.

An isochronous or interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set through the USB host Transmit Interval Endpoint n (USBTXINTERVALn) or USB host Receive Interval Endpoint n (USBRXINTERVALn) register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must be retried (for example, because a NAK was received or the target device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This process ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller also allows the user to specify a limit to the length of time for NAKs to be received from a target device before the endpoint times out.

27.3.2.5 USB Hubs

The following setup requirements apply to the USB host controller only if it is used with a USB hub. When a full- or low-speed device is connected to the USB controller through a USB 2.0 hub, details of the hub address and the hub port also must be recorded in the corresponding USB Receive Hub Address Endpoint n (USBRXHUBADDRn) and USB Receive Hub Port Endpoint n (USBRXHUBPORTn) or the USB Transmit Hub Address Endpoint n (USBTXHUBADDRn) and USB Transmit Hub Port Endpoint n (USBTXHUBPORTn) registers. In addition, the speed at which the device operates (full or low) must be recorded in the USB Type Endpoint 0 (USBTTYPE0) (endpoint 0), USB Host Configure Transmit Type Endpoint n (USBTXTYPEn), or USB Host Configure Receive Type Endpoint n (USBRXTYPEn) registers for each endpoint that is accessed by the device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB devices. To maximize the number of devices supported, the USB host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions must be made following the completion of any on-going transactions on the endpoints affected.

27.3.2.6 Babble

The USB host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB host controller assumes that the target device to which it is connected has malfunctioned, and the USB controller suspends all transactions and generates a babble interrupt.

27.3.2.7 Host SUSPEND

If the SUSPEND bit in the USBPOWER register is set, the USB host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit SUSPEND mode, set the RESUME bit and clear the SUSPEND bit. While the RESUME bit is set, the USB host controller generates RESUME signaling on the bus. After 20 ms, the RESUME bit must be cleared, at which point the frame counter and transaction scheduler start. The host supports the detection of a remote wake-up.

27.3.2.8 USB RESET

If the RESET bit in the USBPOWER register is set, the USB host controller generates USB RESET signaling on the bus. The RESET bit must be set for at least 20 ms to ensure correct resetting of the target device. After the CPU has cleared the bit, the USB host controller starts its frame counter and transaction scheduler.

27.3.2.9 Connect and Disconnect

A session is started by setting the SESSION bit in the USB Device Control (USBDEVCTL) register, enabling the USB controller to wait for a device to be connected. When a device is detected, a connect interrupt is generated. The speed of the device that has been connected can be determined by reading the USBDEVCTL register where the FSDEV bit is set for a full-speed device, and the LSDEV bit is set for a low-speed device. The USB controller must generate a RESET to the device, and then the USB host controller can begin device enumeration. If the device is disconnected while a session is in progress, a disconnect interrupt is generated.

27.3.3 OTG Mode

To conserve power, the USB On-The-Go (OTG) supplement allows VBUS to only be powered up when required and to be turned off when the bus is not in use. VBUS is always supplied by the A device on the bus. The USB OTG controller determines whether it is the A device or the B device by sampling the ID input from the PHY. This signal is pulled LOW when an A-type plug is sensed (signifying that the USB OTG controller should act as the A device) but taken HIGH when a B-type plug is sensed (signifying that the USB controller is a B device). Note that when switching between OTG A and OTG B, the USB controller retains all register contents.

27.3.3.1 Starting a Session

When the USB OTG controller is ready to start a session, the SESSION bit must be set in the USBDEVCTL register. The USB OTG controller then enables ID pin sensing. The ID input is either taken Low if an A-type connection is detected or High if a B-type connection is detected. The DEV bit in the USBDEVCTL register is also set to indicate whether the USB OTG controller has adopted the role of the A device or the B device. As soon as the USB controller has detected that it is on the A side of the cable, it must enable VBUS power within 100ms or the USB controller reverts to device mode.

If the USB OTG controller is the A device, then the USB OTG controller enters host mode (the A device is always the default host), turns on VBUS, and waits for VBUS to go above the VBUS Valid threshold, as indicated by the VBUS bit in the USBDEVCTL register going to 0x3. The USB OTG controller then waits for a peripheral to be connected. When a peripheral is detected, a Connect interrupt is signaled and either the FSDEV or LSDEV bit in the USBDEVCTL register is set, depending whether a full-speed or a low-speed peripheral is detected. The USB controller then issues a RESET to the connected device. The SESSION bit in the USBDEVCTL register can be cleared to end a session. The USB OTG controller also automatically ends the session if babble is detected or if VBUS drops below session valid.

NOTE: The USB OTG controller may not remain in host mode when connected to high-current devices. Some devices draw enough current to momentarily drop VBUS below the VBUS-valid level causing the controller to drop out of host mode. The only way to get back into host mode is to allow VBUS to go below the Session End level. In this situation, the device is causing VBUS to drop repeatedly and pull VBUS back low the next time VBUS is enabled.

In addition, the USB OTG controller may not remain in host mode when a device is told that it can start using its active configuration. At this point the device starts drawing more current and can also drop VBUS below VBUS valid.

If the USB OTG controller is the B device, then the USB OTG controller requests a session using the session request protocol defined in the USB On-The-Go supplement, that is, it first discharges VBUS. Then when VBUS has gone below the Session End threshold (VBUS bit in the USBDEVCTL register goes to 0x0) and the line state has been a single-ended zero for > 2 ms, the USB OTG controller pulses the data line, then pulses VBUS. At the end of the session, the SESSION bit is cleared either by the USB OTG controller or by the application software. The USB OTG controller then causes the PHY to switch out the pullup resistor on D+, signaling the A device to end the session.

27.3.3.2 Detecting Activity

When the other device of the OTG setup wishes to start a session, it either raises VBUS above the Session Valid threshold if it is the A device, or if it is the B device, it pulses the data line then pulses VBUS. Depending on which of these actions happens, the USB controller can determine whether it is the A device or the B device in the current setup and act accordingly. If VBUS is raised above the Session Valid threshold, then the USB controller is the B device. The USB controller sets the SESSION bit in the USBDEVCTL register. When RESET signaling is detected on the bus, a RESET interrupt is signaled, which is interpreted as the start of a session.

The USB controller is in device mode as the B device is the default mode. At the end of the session, the A device turns off the power to VBUS. When VBUS drops below the Session Valid threshold, the USB controller detects this drop and clears the SESSION bit to indicate that the session has ended, causing a disconnect interrupt to be signaled. If data line and VBUS pulsing is detected, then the USB controller is the A device. The controller generates a SESSION REQUEST interrupt to indicate that the B device is requesting a session. The SESSION bit in the USBDEVCTL register must be set to start a session.

27.3.3.3 Host Negotiation

When the USB controller is the A device, ID is Low, and the controller automatically enters host mode when a session starts. When the USB controller is the B device, ID is High, and the controller automatically enters device mode when a session starts. However, software can request that the USB controller become the host by setting the HOSTREQ bit in the USBDEVCTL register. This bit can be set either at the same time as requesting a Session Start by setting the SESSION bit in the USBDEVCTL register or at any time after a session has started. When the USB controller next enters SUSPEND mode and if the HOSTREQ bit remains set, the controller enters host mode and begins host negotiation (as specified in the USB On-The-Go supplement) by causing the PHY to disconnect the pullup resistor on the D+ line, causing the A device to switch to device mode and connect its own pullup resistor. When the USB controller detects this, a Connect interrupt is generated and the RESET bit in the USBPOWER register is set to begin resetting the A device. The USB controller begins this reset sequence automatically to ensure that RESET is started as required within 1 ms of the A device connecting its pullup resistor. The main processor should wait at least 20 ms, then clear the RESET bit and enumerate the A device.

When the USB OTG controller B device has finished using the bus, the USB controller goes into SUSPEND mode by setting the SUSPEND bit in the USBPOWER register. The A device detects this and either terminates the session or reverts to host mode. If the A device is USB OTG controller, it generates a Disconnect interrupt.

27.3.4 ULPI Interface

The ULPI PHY interface is a reduced pin-count interface conforms to the UTMI+ specification. The reduction in pin count is achieved by allowing the relatively static UTMI+ signals to be accessed through registers and by providing a bi-directional data bus both for the USB data and for accessing register data on the ULPI transceiver. The PHY interface supports a 12-pin Single Data Rate (SDR) ULPI Standard.

27.3.4.1 Register Read

To execute a polled read, the steps are as follows:

1. Write the ULPIREGADDR register with the address of the read access.
2. Initiate a read access by setting the RDWR bit and the REGACC bit in the ULPIREGCTL register.
3. Poll the REGCMPLT bit in the ULPIREGCTL register to determine when the access is complete.
4. Clear the REGCMPLT bit and read the return data from the ULPIREGDATA register.

27.3.4.2 Register Write

To execute a polled write:

1. Write the ULPIREGADDR register with the address of the PHY register to be accessed.
2. Initiate the write to the register by setting the REGACC bit in the ULPIREGCTL register.
3. Poll the REGCMPLT bit in the ULPIREGCTL register to determine when the access is complete.
4. Clear the REGCMPLT bit in the ULPIREGCTL register.

27.3.4.3 ULPI PHY Reset Operation

After power-up reset, the ULPI interface performs a write of 0x61 to the external ULPI PHY. This write resets the ULPI PHY.

27.3.5 Link Power Management (LPM)

Link Power Management (LPM) allows the USB to be suspended and resumed by two basic methods.

The procedure in which the USB is suspended and resumed depends on whether the USB is operating as a device or host and the method of suspend desired.

27.3.5.1 LPM Operation for USB as Device

In order for the USB, acting as a device, to be suspended by the host, the LPM feature must be enabled by configuring the USB LPM Control (USBLPMCNTL) register. The LPMEN field is used to enable and support the LPM transactions and the TXLPM field is used to instruct the hardware that it is ready to suspend and respond to the next LPM transaction with an ACK. [Table 27-1](#) describes the response to an LPM transaction by the USB device.

Table 27-1. USB Device LPM Transaction Response

TXLPM	LPMEN	Data Pending (Data Resides in the TX FIFOs)	Response to the Next LPM Transaction
0	0x0	Don't care	Time-out
0	0x2		
1	0x0		
1	0x2		

**Table 27-1. USB Device LPM Transaction
Response (continued)**

TXLPM	LPMEN	Data Pending (Data Resides in the TX FIFOs)	Response to the Next LPM Transaction
0	0x1	Don't care	STALL
1	0x1		
0	0x3	Don't care	NYET
1	0x3	Yes	NYET
1	0x3	No	ACK

For all cases in which the USB device responds (no timeout occurs), an LPM interrupt is generated in the USB LPM Raw Interrupt Status (USBLPMRIS) register. Note that the USB device only responds with an ACK only if there is no data pending in any of the TX Endpoint FIFOs. If there is data pending, it responds with a NYET.

Once an LPM transaction is successfully received, three events occur:

1. The USB LPM Attributes (USBLPMATTR) register is updated with values of the LPM transaction just received. The parameters that are updated are as follows:
 - **LINKSTATE:** This field tells the USB what state to transition after an LPM transaction. The only valid value for this field is 0x1 which indicates that the USB core should suspend. For any other value, the USB device responds with a STALL and the appropriate interrupt is generated. However, in this case the USBLPMATTR register is updated so that software can observe the non-compliant LPM packet payload. In addition, the ERR interrupt bit is set in the USBLPMRIS register and may be used to generated an interrupt to inform software of the non-compliant LPM transaction
 - **HIRD:** Host Initiated Resume Duration. This field tells the USB the minimum duration that the host will drive resume signalling on the bus. This field represents a resume duration range from 50 μ s to 1200 μ s. This value may be different for subsequent LPM transactions.
 - **RMTWAK:** This bit indicates if the remote wakeup by the USB is allowed. This bit applies to the current suspend and resume cycle only and may be different for subsequent LPM transactions. The RMTWAK bit should not supersede the wakeup capability that was previously negotiated on enumeration of the USB.
2. The USB device suspends 9 μ s after transmitting the ACK. Resume signaling can be driven by the host or the USB 50 μ s after this event. During this 9 μ s interval, the host may continue to transmit the LPM transaction. The USB responds with an ACK in this case regardless of the TXLPM value in the USBLPMCNTL register.
3. An interrupt is generated informing software of the response (an ACK in this case). An ACK response is the indication to software that the USB has suspended.

Since the primary purpose of LPM is to save power, the software reads the USBLPMATTR register to determine the attributes of the Suspend. Software must make a determination based on these attributes whether additional power savings in the system can be exploited. In making this determination it is noted that if the host initiates the resume signalling, the USB is required to respond to packet transmissions within the time specified by HIRD + 10 μ s.

When the host resumes the bus, it drives resume signalling for a minimum time specified by the HIRD field in the USBLPMATTR register. The USB must be able to respond to traffic within the time HIRD + 10 μ s. The USB transitions to a normal operating state automatically and a resume interrupt bit is set in the USB LPM Raw Interrupt Status (USBLPMRIS) register. However for this to occur, the System Clock and the input signal USB0CLK must be enabled. To facilitate the resume timing requirement, an additional feature, NAK, is provided in the USBLPMCNTL register. If NAK is set to 0x1, all endpoints respond to any transaction (other than an LPM) with a NAK. This bit only takes effect after the USB has LPM

suspended. Typically, this bit would be asserted when the TXLPM field is also asserted. Using this feature may simplify the resume timing requirement because only XCLK is needed for the USB to respond (with a NAK) to traffic. Software can continue to restore the system to normal operation while the USB responds to all transactions with a NAK. After the system has been completely restored, software can then clear the NAK field in the USBLPMCNTL register.

If the software wants to initiate a remote wakeup while the USB is in Suspend mode, it should write a 0x1 to the RES bit in the USBLPMCNTL register. This bit is self clearing. Writing a 0x1 causes resume signaling to be driven on the bus for 50 μ s. The host responds by driving resume for 60 μ s to 990 μ s. 10 μ s after the host stops driving resume, the USB transitions to its normal operational state and is ready for packet transmission. A resume interrupt bit is set in the USBLPMRIS register.

27.3.5.2 LPM Operation as a Host

When operating as a host, the USB initiates an LPM Suspend (transition from the L0 state to the L1 state) by initiating an LPM transaction as follows:

- Software configures the desired attributes of the Suspend in the USBLPMATTR register. Setting the RMTWAK bit and programming a large HIRD gives the peripheral more opportunity to conserve power.
- All LPM interrupts should be enabled in the USB LPM Interrupt Mask (USBLPMIM) register.
- Software should initiate the transaction by setting the TXLPM bit in the USBLPMCNTL register.
- An interrupt is generated to inform software of the response to the LPM transaction. If an ACK was received then the USB suspends automatically within 8 μ s. This is the indication that the USB has suspended.

If the response from the device has a bit-stuff error or a PID error, then the ERR bit is set in the USBLPMRIS register and an interrupt is generated. The hardware immediately attempts the LPM transaction two more times. The device does not suspend for 8 μ s after the initial LPM so it is able to respond to either of these subsequent LPM transactions. If an LPM timeout has occurred three times, the NC and the ERR interrupt bits are set. At this time, software is unaware of the device state and must deduce it by other means.

Resume signaling should be generated by software as follows:

- All LPM interrupts should be enabled in the USBLPMIM register.
- Software should write the RES bit in the USBLPMCNTL register. This bit is self-clearing and causes the resume signalling to be generated on the bus for the time that is currently specified in the HIRD field of the USBLPLMATTR register. It is assumed by hardware that this value was used in the last LPM transaction that caused the Suspend.
- After HIRD + 10 μ s, the USB transitions to its normal operational state and is ready for packet transmissions. The RES bit is set in the USBLPMRIS register and an interrupt may be generated.

NOTE: Prior to resuming, software must ensure that the system is completely restored from a low-power state and that the System Clock (SYSCLK) and the USB0CLK clock input are enabled.

If the remote wakeup feature was enabled in the LPM transaction that caused the Suspend, then the device may drive resume signaling on the bus. When this occurs, the device drives resume signaling on the bus for 50 μ s. The USB host immediately begins driving resume signaling on the bus and does so for 60 μ s. 10 μ s after completion of the resume signaling, the USB transitions to its normal operating state and is ready for packet transmission. At this time, the RES interrupt bit is set in the USBLPMRIS register and an interrupt may be generated.

27.3.6 USB DMA Controller

The USB includes an integrated eight-channel USB DMA controller for efficient loading and unloading of the endpoint FIFOs. Any Tx Endpoints 1 through 7 and Rx Endpoints 1 through 7 can be assigned to DMA channels 0 through 7. An endpoint and its direction can be assigned to a USB DMA channel through the EP field and DIR bit in the USB DMA Control n (USBDMACTLn) register. The DMA can operate in two modes and can handle packet sizes up to 8K. In addition, the DMA controller can be programmed to conduct transfers using burst of four, eight and sixteen or bursts of unspecified length.

The DMA request lines are individually enabled through the DMAEN bit in the appropriate USB Transmit Control and Status Endpoint n High (USBTXCSRHn) register or USB Receive Control and Status Endpoint n High (USBXRCSRHn) register and operate in two request modes, referred to as DMA Request Mode 0 and DMA Request Mode 1.

When operating in DMA Mode 0, the DMA controller can be only programmed to load and unload one packet, so processor intervention is required for each packet transferred over the USB. This mode can be used with any endpoint, whether it uses Control, Bulk, Isochronous, or Interrupt transactions (that is, including Endpoint 0).

When operating in DMA Mode 1, the DMA controller can be programmed to load and unload a complete bulk transfer (which can be many packets). Once set up, the DMA controller loads and unloads all packets of the transfer, interrupting the processor only when the transfer has completed. DMA Mode 1 can only be used with endpoints that use Bulk transactions.

Each channel can be independently programmed for the selected operating mode. The choice of operating mode must be made through the DMAMOD bit in the USBTXCSRHn and the USBXRCSRHn register. For Rx endpoints operating in Request Mode 0, the DMA request line goes high when a data packet is available in the endpoint FIFO and normally goes low at the end of the cycle in which the 8th from last byte starts to be processed (which happens two transfers minus one clock cycle in advance of the transfer containing this byte). The request line also goes low if the CPU clears the RXRDY bit in the USB Receive Control and Status Endpoint n Low (USBXRCSRLn) register. The behavior of the DMA request lines for Rx endpoints in Request Mode 1 is similar except the request line only goes high when the packet received is of the maximum packet size (as set in the USB Maximum Receive Data Endpoint n (USBRXMAXPn) register). If the packet received is of some other size, the DMA request line stays low. Note, however, that if the Request Mode is switched from Request Mode 1 to Request Mode 0, the request line is asserted if there is a packet in the FIFO in order to allow this prereceived packet to be downloaded.

For Tx endpoints operating in either Request Mode 0 or Request Mode 1, the DMA request line goes high when the endpoint FIFO is able to accept a data packet. It normally goes low one clock cycle after the 8th from last of MAXLOAD bytes in the USBTXMAXPn register have been loaded into the FIFO. The request line also goes low if the CPU sets the TXRDY bit of the USB Transmit Control and Status Endpoint n Low (USBTXCSRLn) register.

NOTE: When operating in host mode, if either the STALLED bit or the ERROR bit in the USBTXCSRLn register becomes set following three failed attempts to transmit a packet, the DMA request line is disabled until the STALLED and ERROR bit has been cleared.

The mode selected also affects the generation of endpoint interrupts. In DMA Request Mode 0, no interrupt is generated when packets are received but the appropriate Endpoint interrupt is generated to prompt the loading of all packets. In DMA Request Mode 1, the Endpoint interrupt is suppressed except following the receipt of a short packet (that is, one of less than MAXLOAD bytes). [Table 27-2](#) and [Table 27-3](#) summarize the conditions under which TX and RX Endpoint interrupts are generated.

Table 27-2. Endpoint Interrupt Associated With USBXRCSRLn.RXRDY = 1

USBRXCSRHn.DMAEN	USBRXCSRHn.DMAMOD	Interrupt Generated?
0	X	Yes
1	0	No
1	1	Only if short packet

Table 27-3. Endpoint Interrupt Associated With USBTXCSRLn.TXRDY = 1

USBTXCSRHn.DMAEN	USBTXCSRHn.DMAMOD	Interrupt Generated?
0	X	Yes
1	0	Yes
1	1	No

DMA Request Mode 0 can be used equally well for Bulk, Interrupt or Isochronous transfers. If the endpoint is configured for Isochronous transfers, DMA Request Mode 0 should always be selected where DMA is used. DMA Request Mode 1 is valuable where large blocks of data are transferred to a Bulk endpoint. The USB protocol requires such packets to be split into a series of packets of the maximum packet size for the endpoint (512 bytes for high speed, 64 bytes for full speed).

NOTE: The MAXLOAD field must be set to an even number of bytes for proper interrupt generation in Mode 1.

DMA Request Mode 1 can be used to avoid the overhead of having to interrupt the processor after each individual packet; instead, the processor is only interrupted after the transfer has completed. In some cases, the block of data transferred comprises a predefined number of these packets, that the controlling software counts through the transfer process. In other cases, the last packet in the series may be less than the maximum packet size and the receiver may use this short packet to signal the end of the transfer. (If the total size of the transfer is an exact multiple of the maximum packet size, the transmitting software should send a null packet for the receiver to detect.)

Further information on using DMA for Bulk transfers is given in [Section 27.3.6.3](#). DMA transfers may be byte, half-word, or word, as required. However, all the transfers associated with one packet (with the exception of the last) must be of the same width so that the data is consistently byte-, word- or double-word-aligned. The last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

NOTE: DMA Requests should be disabled before the DMA Request Mode is changed. In particular, the DMAMODE bit in the USBTXCSRHn register should not be programmed to zero either before or in the same cycle as the corresponding DMAEN bit is cleared.

27.3.6.1 DMA Burst Operation

The DMA uses incrementing bursts for transferring data. The burst transfer begins when the USB DMA is given bus mastership and when the address accesses a new 1-Kb block.

Bursts may be done in increments of 4, 8, 16, or of an unspecified length, depending on how the BRSTM field is configured in the USBDMACLTn register, the size of the packet being transferred and the location relative to the next 1-Kb boundary. For example if BRSTM = 0x2, increments of 8 or 4 bursts and bursts of unspecified length are allowed but not 16-byte bursts. There is no restriction on the BRSTM value for transfers in either DMA Mode 0 or DMA Mode 1.

Each transfer of a packet is generally a word transfer, but there may be additional byte or half-word transfers

27.3.6.2 DMA Bus Errors

If a bus error occurs while the DMA controller is accessing memory, the DMA controller immediately terminates the DMA transfer and interrupts the processor with a bus error by setting the ERR bit in the USBDMACLTn register.

NOTE: The generation of the bus error interrupt is not affected by setting the IE bit in the USBDMACLT register. This bus error interrupt is still generated even if the IE bit is 0.

27.3.6.3 DMA Operation

The DMA may be used in connection with any type of transfer, but it is particularly useful when large blocks of data are to be transferred through a Bulk endpoint. The USB protocol requires that large data blocks are transferred by sending a series of packets of the maximum packet size for the endpoint (512 bytes for high speed, 64 bytes for full speed). The last packet in the series may be less than the maximum packet size. The receiver may use the reception of this short packet to signal the end of the transfer (a

null packet may be sent at the end of the series if the size of the data block is an exact multiple of the maximum packet size). The DMA may be used in either device mode or host mode to avoid the overhead of having to interrupt the processor after each individual packet, and instead, only interrupting the processor after the transfer has completed. The following sections outline the basic actions that are involved in using the DMA alongside some standard types of Bulk Tx and Bulk Rx transfers.

27.3.6.3.1 Using DMA with Bulk Tx Endpoints

For Tx endpoints, the DMA request line is high when the endpoint FIFO is able to accept a data packet, and goes low when MAXLOAD transmit bytes have been loaded into the FIFO. Alternatively, the request line is held low when the TXRDY bit in USBTXCSRLn register is set. To use DMA to send a large block of data to the USB host over a Bulk Tx endpoint, we recommend setting up the DMA controller and the USB as follows:

- The DMA controller should be programmed to perform a burst DMA read of the maximum size of packet for the endpoint (512 bytes for high speed, 64 bytes for full speed) when the DMA request line for the endpoint transitions from low to high. The controller should keep performing these burst reads on each DMA request until the entire data block has been transferred. (The last burst may however be of less than the maximum packet size). It should then interrupt the CPU.
- The USB should be programmed to enable Autoset and DMA Request Mode 1 by setting the AUTOSSET, DMAEN and DMAMOD bits in the USBTXCSRn register.

Programmed like this, the USB DMA request line is held high whenever there is space in its FIFO to accept a packet. Further, the TXRDY bit is automatically set after the DMA controller has loaded the FIFO with a packet of the maximum packet size. The packet is then ready to be sent to the host. When the last packet has been loaded by the DMA controller, the controller interrupts the processor. If the last packet loaded is less than the maximum packet size, the TXRDY bit is not set and therefore needs to be set manually (that is, by the CPU) to allow the last packet to be sent. The TXRDY bit also needs to be set manually if the last packet is of the maximum packet size and a null packet is to be sent to indicate the end of the transfer.

NOTE: If, when operating in host mode, the USB fails to successfully transmit a packet three times, the ERROR bit in the USBTXCSRLn register becomes set and the DMA request line is disabled until this ERROR bit is cleared again. It should also be noted that the DMAMOD bit in the USBTXCSRn register must not be cleared either before or in the same cycle as the corresponding DMAEN bit is cleared.

27.3.6.3.2 Using DMA with Bulk RX Endpoints

The behavior of the DMA request line for an Rx Endpoint depends on the DMA Request Mode (DMAMOD) selected through the USB Receive Control and Status Endpoint n High (USBRXCSRn) register. In DMA Request Mode 0, the Rx DMA request line is held high when a data packet is available in the endpoint FIFO and is low either when the last byte of the data packet has been read, or when the RXRDY bit in USB Receive Control and Status Endpoint n Low (USBRXCSRLn) is cleared. In DMA Request Mode 1, the DMA request line only goes high when the packet received is of the maximum packet size (as set in the USBRXMAXPn register). If the packet received is of some other size, the DMA request line stays low with the result that the packet remains in the FIFO with the RXRDY bit set. This causes an Rx Endpoint interrupt to be generated (if enabled).

The DMA Request Modes are primarily designed to be used where large packets of data are transferred to a Bulk endpoint. The USB protocol requires such packets to be split into a series of packets of maximum packet size (512 bytes for high speed, 64 bytes for full speed). The last packet in the series may be less than the maximum packet size (or a null packet if the total size of the transfer is an exact multiple of the maximum packet size) and the receiver may interpret this short packet as signaling the end of the transfer. DMA Request Mode 1 can be used, with a suitably programmed DMA controller, to avoid the overhead of having to interrupt the processor after each individual packet instead just interrupting the processor after the transfer has completed.

NOTE: If the Request Mode is switched from Request Mode 1 to Request Mode 0, the request line is asserted if there is a packet in the FIFO in order to allow this prereceived packet to be downloaded.

27.3.6.4 DMA Configuration

The use of the internal DMA controller to access the USB FIFOs requires both the DMA controller and the endpoint to be properly programmed. The following section list the standard configuration for the basic actions of transferring individual packets and multiple packets.

27.3.6.4.1 Individual Packet: Rx Endpoint

The transfer of individual packets is normally carried out using DMA Mode 0. For this, the USB Rx endpoint should be programmed as follows:

1. The relevant interrupt enable bit in the USB Receive Interrupt Enable (USBRXIE) register should be set to 1.
2. The DMAEN bit of the appropriate USB Receive Control and Status Endpoint n High (USBRXCSRHn) register is set to 0.

When a packet has been received by the USB, it generates the appropriate Endpoint interrupt. The processor should then program the selected channel of the DMA controller as follows:

1. The USB DMA Address n (USBDMAADDRn) register should be written with the memory address to store the packet.
2. The USB DMA Count n (USBDMACOUNTn) register should be programmed with the size of packet which is determined by reading the USB Receive Byte Count Endpoint n (USBCOUNTn) register.
3. The BRSTM field should be configured in the USB DMA Control n (USBDMACTLn) register and the remaining bits should be programmed with the following:
ENABLE = 0x1; DIR = 0x0; MODE = 0x0; IE = 0x1.

The DMA controller then requests bus mastership and transfers the packet to memory. When it has completed the transfer, it generates a DMA interrupt. The processor should then clear the RXRDY bit in the USB Receive Control and Status Endpoint n Low (USBRXCSSLn) register.

27.3.6.4.2 Individual Packet: Tx Endpoint

To carry out this operation using DMA Mode 0, an USB Tx endpoint should be programmed as follows:

1. The relevant interrupt enable bit in the USB Transmit Interrupt Enable (USBTXIE) register should be set to 1.
2. The DMAEN bit of the appropriate USB Transmit Control and Status Endpoint n High (USBTXCSRHn) register is set to 0.

When the FIFO in the USB becomes available, the USB interrupts the processor with the appropriate Tx Endpoint interrupt. The processor should then program the DMA controller as follows:

1. The USB DMA Address n (USBDMAADDRn) register should be written with the memory address of the packet to send.
2. The USB DMA Count n (USBDMACOUNTn) register should be programmed with the size of packet to be sent.
3. The BRSTM field should be configured in the USB DMA Control n (USBDMACTLn) register and the remaining bits should be programmed with the following:
ENABLE = 0x1, DIR = 0x1, MODE = 0x0, IE = 0x1.

The DMA controller then requests bus mastership and transfers the packet to the USB FIFO. When it has completed the transfer, it generates a DMA interrupt. The processor should then clear the TXRDY bit in the USB Transmit Control and Status Endpoint n Low (USBTXCSSLn) register.

27.3.6.4.3 Multiple Packets: Rx Endpoint

The transfer of multiple packets is normally carried out using DMA Mode 1. Where multiple packets are to be received using DMA Mode 1, the DMA Controller should be programmed as follows:

1. The USB DMA Address n (USBDMAADDRn) register should be written with the memory address of the buffer in which to store the transfer.
2. The USB DMA Count n (USBDMACOUNTn) register should be programmed with the size of the buffer.
3. The BRSTM field should be configured in the USB DMA Control n (USBDMACTLn) register and the remaining bits should be programmed with the following:
ENABLE = 0x1, DIR = 0x0, MODE = 0x1, IE = 0x1.

The USB Rx endpoint should be programmed as follows:

1. The relevant interrupt enable bit in the USB Receive Interrupt Enable (USBRXIE) register should be set to 1.
2. The AUTOCL, DMAEN and DMAMOD bit of the appropriate USB Receive Control and Status Endpoint n High (USBRXCSRHn) register is set to 1. In host mode, the AUTORQ bit should also be set to 1 and the USB Request Packet Count in Block Transfer Endpoint n (USBRQPKTCOUNTn) register should be programmed with the number of packets in the transfer.

As each packet is received by the USB, the DMA controller requests bus mastership and transfers the packet to memory. With AutoClear set, the USB automatically clears the RXRDY bit. In device mode or where Request Packet Count (COUNT) is zero, this process continues automatically until the USB receives a short packet (one of less than the maximum packet size for the endpoint) signifying the end of the transfer. This short packet is not be transferred by the DMA controller; instead, the USB interrupts the processor by generating the appropriate Endpoint interrupt. The processor can then read the USB Receive Byte Count Endpoint n (USBRXCOUNTn) register to see the size of the short packet and either unload it manually or reprogram the DMA controller in Mode 0 to unload the packet. In host mode with AUTORQ set and the USBRQPKTCOUNTn register non-zero, the USB decrements the value in the USBRQPKTCOUNTn register following each request. When the value decrements from 1 to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted.

The USB DMA Address n (USBDMAADDRn) register is incremented as the packets were unloaded so the processor can determine the size of the transfer by comparing the current value of the USB DMA Address n (USBDMAADDRn) register against the start address of the memory buffer.

NOTE: If the size of the transfer exceeds the data buffer size, the DMA controller stops unloading the FIFO and interrupts the processor through a DMA interrupt.

27.3.6.4.4 Multiple Packets: Tx Endpoint

To carry out this operation using DMA Mode 1, the DMA controller should be programmed as follows:

1. The USB DMA Address n (USBDMAADDRn) register should be written with the memory address of the data block to send.
2. The USB DMA Count n (USBDMACOUNTn) register should be programmed with the size of the data block.
3. The BRSTM field should be configured in the USB DMA Control n (USBDMACTLn) register and the remaining bits should be programmed with the following:
ENABLE = 0x1, DIR = 0x1, MODE = 0x1, IE = 0x1.

The USB Tx endpoint should be programmed as follows:

1. The relevant interrupt enable bit in the USB Transmit Interrupt Enable (USBTXIE) register should be set to 1.
2. The AUTOSSET, DMAEN and DMAMOD bit of the appropriate USB Transmit Control and Status Endpoint n High (USBTXCSRHn) register is set to 1.

When the FIFO in the USB becomes available, the DMA controller requests bus mastership and transfer a packet to the FIFO. With the AUTOSSET bit set, the USB automatically sets the TXRDY bit. This process continues until the entire data block has been transferred to the USB. The DMA controller then interrupts the processor. If the last packet to be loaded was less than the maximum packet size for the endpoint, the TXRDY bit is not set for this packet. The processor should therefore respond to the DMA interrupt by setting the TXRDY bit to allow the last short packet to be sent. If the last packet to be loaded was of the maximum packet size, then the action to take depends on whether the transfer is under the control of an application such as the mass storage software on a Windows system that keeps count of the individual packets sent. If the transfer is not under such control, the processor should still respond to the DMA interrupt by setting the TXRDY bit in the USB Transmit Control and Status Endpoint n Low (USBTXCSSLn) register. This has the effect of sending a null packet for the receiving software to interpret as indicating the end of the transfer.

27.3.7 USB Clock Structure

The USB block receives two clocks. The main peripheral is clocked from the system clock (SYSCLK) while the serialization and deserialization circuitry requires a fixed 60 MHz clock source. When using the integrated USB PHY, the 60 MHz clock is constructed by dividing the PLL VCO output by a dedicated programmable divisor. The divisor is controlled by the USBCC register. When using the ULPI interface, the 60-MHz clock is selected either from the internal PLL VCO path or the ULPI signal USB0CLK. If the source is the PLL VCO path, the USB0CLK signal must be an output; otherwise, USB0CLK is an input. To correctly synchronize with the data between the main block and the serialization and deserialization circuitry, the main block requires f_{SYSCLK} of 30 MHz.

Table 27-4. USB0CLK Direction During ULPI Mode

ULPIEN Bit in USBPC Register	CSD Bit in USBCC Register	60 MHz Clock Source ⁽¹⁾	USB0CLK Direction
0	X	VCO PLL (CLKDIV +1)	Not Used
1	0	VCO PLL (CLKDIV +1)	Output
1	1	USBCLK0	Input

⁽¹⁾ The CLKDIV bit resides in the USBCC register.

27.4 Initialization and Configuration

To initialize the USB controller:

1. To enable the USB controller, the peripheral clock must be enabled through the RCGCUSB register.
2. In addition, the clock to the appropriate GPIO module must be enabled through the RCGCGPIO register in the System Control module. To find out which GPIO port to enable, see the device-specific data sheet. Configure the PMCN fields in the GPIOPCTL register to assign the USB signals to the appropriate pins (see [Section 17.5.22](#) and the device-specific data sheet).
3. Program the internal clock configuration through the USBCC register.
4. Perform a software reset of the USB using the SRUSB register.
5. Disable the internal clock source if the USB PHY is driving the clock (as is the case when using the ULPI interface). By setting the CSD bit in the USBCC register, an external USB clock source is selected.
6. Enable the appropriate PHY (external or internal) by programming the ULPIEN bit in the USBPC register.

NOTE: When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated host or device, the DEVMOD field in the USB General-Purpose Control and Status (USBGPCS) register can be used to connect the USB0VBUS and USB0ID inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. Note that PB1 (USB0VBUS) is a 5-V tolerant signal as required. For proper self-powered device operation, the VBUS value must still be monitored to assure that if the host removes VBUS, the self-powered device disables the D+ and D- pullup resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

The termination resistors for the USB PHY have been added internally, and thus there is no need for external resistors. For a device, there is a 1.5-k Ω pullup on the D+ and for a host there are 15-k Ω pulldowns on both D+ and D-.

27.4.1 Pin Configuration

When using the device controller portion of the USB controller in a system that also provides host functionality, the power to VBUS must be disabled to allow the external host controller to supply power. Usually, the USB0EPEN signal is used to control the external regulator and should be negated to avoid having two devices driving the USB0VBUS power pin on the USB connector.

When the USB controller is acting as a host, it is in control of two signals that are attached to an external voltage supply that provides power to VBUS. The host controller uses the USB0EPEN signal to enable or disable power to the USB0VBUS pin on the USB connector. An input pin, USB0PFLT, provides feedback when there has been a power fault on VBUS. The USB0PFLT signal can be configured to either automatically negate the USB0EPEN signal to disable power, and it can generate an interrupt to the interrupt controller to allow software to handle the power fault condition. The polarity and actions related to both USB0EPEN and USB0PFLT are fully configurable in the USB controller. The controller also provides interrupts on device insertion and removal to allow the host controller code to respond to these external events.

27.4.2 Endpoint Configuration

To start communication in host or device mode, the endpoint registers must first be configured. In host mode, this configuration establishes a connection between an endpoint register and an endpoint on a device. In device mode, an endpoint must be configured before enumerating to the host controller.

In both cases, the endpoint 0 configuration is limited because it is a fixed-function, fixed-FIFO-size endpoint. In device and host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the host controller. In host mode, the endpoints must be configured to operate as control, bulk, interrupt or isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring the FIFO of each endpoint involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is Kbytes with the first 64 bytes reserved for endpoint 0. The endpoint's FIFO must be at least as large as the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a device, the soft connect of the USB device controller must be enabled when the device is ready to start communications, indicating to the host controller that the device is ready to start the enumeration process. If operating as a host controller, the device soft connect must be disabled and power must be provided to VBUS through the USB0EPEN signal.

27.5 USB Registers

The USB controller has On-The-Go (OTG) capabilities.

OTG B / Device indicates that the register is used in OTG B or Device mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode.

OTG A / Host indicates that the register is used in OTG A or Host mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode. The USB controller is in OTG B or Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.

OTG indicates that the register is used for OTG-specific functions such as ID detection and negotiation. When OTG negotiation is complete, then the USB controller registers are used according to their Host or Device mode meanings depending on whether the OTG negotiations made the USB controller OTG A (Host) or OTG B (Device).

[Table 27-5](#) lists the memory-mapped registers for the USB. All register offset addresses not listed in [Table 27-5](#) should be considered as reserved locations and the register contents should not be modified.

All offsets are relative to the USB base address of 0x40050000. The USB controller clock must be enabled before the registers can be programmed. There must be a delay of 3 system clock cycles after the USB module clock is enabled before any USB module registers are accessed.

Table 27-5. USB Registers

Offset	Acronym	Register Name	Section
0x0	USBFADDR	USB Device Functional Address	Section 27.5.1
0x1	USBPOWER	USB Power	Section 27.5.2
0x2	USBTXIS	USB Transmit Interrupt Status	Section 27.5.3
0x4	USBRXIS	USB Receive Interrupt Status	Section 27.5.4
0x6	USBTXIE	USB Transmit Interrupt Enable	Section 27.5.5
0x8	USBRXIE	USB Receive Interrupt Enable	Section 27.5.6
0xA	USBIS	USB General Interrupt Status	Section 27.5.7
0xB	USBIE	USB Interrupt Enable	Section 27.5.8
0xC	USBFRAME	USB Frame Value	Section 27.5.9
0xE	USBEPIDX	USB Endpoint Index	Section 27.5.10
0xF	USBTEST	USB Test Mode	Section 27.5.11
0x20	USBFIFO0	USB FIFO Endpoint 0	Section 27.5.12
0x24	USBFIFO1	USB FIFO Endpoint 1	Section 27.5.12
0x28	USBFIFO2	USB FIFO Endpoint 2	Section 27.5.12
0x2C	USBFIFO3	USB FIFO Endpoint 3	Section 27.5.12
0x30	USBFIFO4	USB FIFO Endpoint 4	Section 27.5.12
0x34	USBFIFO5	USB FIFO Endpoint 5	Section 27.5.12
0x38	USBFIFO6	USB FIFO Endpoint 6	Section 27.5.12
0x3C	USBFIFO7	USB FIFO Endpoint 7	Section 27.5.12
0x60	USBDEVCTL	USB Device Control	Section 27.5.13
0x61	USBCCONF	USB Common Configuration	Section 27.5.14
0x62	USBTXFIFOSZ	USB Transmit Dynamic FIFO Sizing	Section 27.5.15
0x63	USBRXFIFOSZ	USB Receive Dynamic FIFO Sizing	Section 27.5.15
0x64	USBTXFIFOADD	USB Transmit FIFO Start Address	Section 27.5.16
0x66	USBRXFIFOADD	USB Receive FIFO Start Address	Section 27.5.16
0x70	ULPIVBUSCTL	USB ULPI VBUS Control	Section 27.5.17
0x74	ULPIREGDATA	USB ULPI Register Data	Section 27.5.18
0x75	ULPIREGADDR	USB ULPI Register Address	Section 27.5.19
0x76	ULPIREGCTL	USB ULPI Register Control	Section 27.5.20
0x78	USBEPINFO	USB Endpoint Information	Section 27.5.21

Table 27-5. USB Registers (continued)

Offset	Acronym	Register Name	Section
0x79	USBRAMINFO	USB RAM Information	Section 27.5.22
0x7A	USBCONTIM	USB Connect Timing	Section 27.5.23
0x7B	USBVPLEN	USB OTG VBUS Pulse Timing	Section 27.5.24
0x7C	USBHSEOF	USB High-Speed Last Transaction to End of Frame Timing	Section 27.5.25
0x7D	USBFSEOF	USB Full-Speed Last Transaction to End of Frame Timing	Section 27.5.26
0x7E	USBLSEOF	USB Low-Speed Last Transaction to End of Frame Timing	Section 27.5.27
0x80	USBTXFUNCADDR0	USB Transmit Functional Address Endpoint 0	Section 27.5.28
0x82	USBTXHUBADDR0	USB Transmit Hub Address Endpoint 0	Section 27.5.29
0x83	USBTXHUBPORT0	USB Transmit Hub Port Endpoint 0	Section 27.5.30
0x88	USBTXFUNCADDR1	USB Transmit Functional Address Endpoint 1	Section 27.5.28
0x8A	USBTXHUBADDR1	USB Transmit Hub Address Endpoint 1	Section 27.5.29
0x8B	USBTXHUBPORT1	USB Transmit Hub Port Endpoint 1	Section 27.5.30
0x8C	USBRXFUNCADDR1	USB Receive Functional Address Endpoint 1	Section 27.5.31
0x8E	USBRXHUBADDR1	USB Receive Hub Address Endpoint 1	Section 27.5.32
0x8F	USBRXHUBPORT1	USB Receive Hub Port Endpoint 1	Section 27.5.33
0x90	USBTXFUNCADDR2	USB Transmit Functional Address Endpoint 2	Section 27.5.28
0x92	USBTXHUBADDR2	USB Transmit Hub Address Endpoint 2	Section 27.5.29
0x93	USBTXHUBPORT2	USB Transmit Hub Port Endpoint 2	Section 27.5.30
0x94	USBRXFUNCADDR2	USB Receive Functional Address Endpoint 2	Section 27.5.31
0x96	USBRXHUBADDR2	USB Receive Hub Address Endpoint 2	Section 27.5.32
0x97	USBRXHUBPORT2	USB Receive Hub Port Endpoint 2	Section 27.5.33
0x98	USBTXFUNCADDR3	USB Transmit Functional Address Endpoint 3	Section 27.5.28
0x9A	USBTXHUBADDR3	USB Transmit Hub Address Endpoint 3	Section 27.5.29
0x9B	USBTXHUBPORT3	USB Transmit Hub Port Endpoint 3	Section 27.5.30
0x9C	USBRXFUNCADDR3	USB Receive Functional Address Endpoint 3	Section 27.5.31
0x9E	USBRXHUBADDR3	USB Receive Hub Address Endpoint 3	Section 27.5.32
0x9F	USBRXHUBPORT3	USB Receive Hub Port Endpoint 3	Section 27.5.33
0xA0	USBTXFUNCADDR4	USB Transmit Functional Address Endpoint 4	Section 27.5.28
0xA2	USBTXHUBADDR4	USB Transmit Hub Address Endpoint 4	Section 27.5.29
0xA3	USBTXHUBPORT4	USB Transmit Hub Port Endpoint 4	Section 27.5.30
0xA4	USBRXFUNCADDR4	USB Receive Functional Address Endpoint 4	Section 27.5.31
0xA6	USBRXHUBADDR4	USB Receive Hub Address Endpoint 4	Section 27.5.32
0xA7	USBRXHUBPORT4	USB Receive Hub Port Endpoint 4	Section 27.5.33
0xA8	USBTXFUNCADDR5	USB Transmit Functional Address Endpoint 5	Section 27.5.28
0xAA	USBTXHUBADDR5	USB Transmit Hub Address Endpoint 5	Section 27.5.29
0xAB	USBTXHUBPORT5	USB Transmit Hub Port Endpoint 5	Section 27.5.30
0xAC	USBRXFUNCADDR5	USB Receive Functional Address Endpoint 5	Section 27.5.31
0xAE	USBRXHUBADDR5	USB Receive Hub Address Endpoint 5	Section 27.5.32
0xAF	USBRXHUBPORT5	USB Receive Hub Port Endpoint 5	Section 27.5.33
0xB0	USBTXFUNCADDR6	USB Transmit Functional Address Endpoint 6	Section 27.5.28
0xB2	USBTXHUBADDR6	USB Transmit Hub Address Endpoint 6	Section 27.5.29
0xB3	USBTXHUBPORT6	USB Transmit Hub Port Endpoint 6	Section 27.5.30
0xB4	USBRXFUNCADDR6	USB Receive Functional Address Endpoint 6	Section 27.5.31
0xB6	USBRXHUBADDR6	USB Receive Hub Address Endpoint 6	Section 27.5.32
0xB7	USBRXHUBPORT6	USB Receive Hub Port Endpoint 6	Section 27.5.33
0xB8	USBTXFUNCADDR7	USB Transmit Functional Address Endpoint 7	Section 27.5.28
0xBA	USBTXHUBADDR7	USB Transmit Hub Address Endpoint 7	Section 27.5.29

Table 27-5. USB Registers (continued)

Offset	Acronym	Register Name	Section
0xBB	USBTXHUBPORT7	USB Transmit Hub Port Endpoint 7	Section 27.5.30
0xBC	USBRXFUNCADDR7	USB Receive Functional Address Endpoint 7	Section 27.5.31
0xBE	USBRXHUBADDR7	USB Receive Hub Address Endpoint 7	Section 27.5.32
0xBF	USBRXHUBPORT7	USB Receive Hub Port Endpoint 7	Section 27.5.33
0x102	USBCSRL0	USB Control and Status Endpoint 0 Low	Section 27.5.34
0x103	USBCSRH0	USB Control and Status Endpoint 0 High	Section 27.5.35
0x108	USBCOUNT0	USB Receive Byte Count Endpoint 0	Section 27.5.36
0x10A	USBTTYPE0	USB Type Endpoint 0	Section 27.5.37
0x10B	USBNAKLMT	USB NAK Limit	Section 27.5.38
0x110	USBTXMAXP1	USB Maximum Transmit Data Endpoint 1	Section 27.5.39
0x112	USBTXCSSL1	USB Transmit Control and Status Endpoint 1 Low	Section 27.5.40
0x113	USBTXCSRH1	USB Transmit Control and Status Endpoint 1 High	Section 27.5.41
0x114	USBRXMAXP1	USB Maximum Receive Data Endpoint 1	Section 27.5.42
0x116	USBRXCSSL1	USB Receive Control and Status Endpoint 1 Low	Section 27.5.43
0x117	USBRXCSRH1	USB Receive Control and Status Endpoint 1 High	Section 27.5.44
0x118	USBRXCOUNT1	USB Receive Byte Count Endpoint 1	Section 27.5.45
0x11A	USBTXTYPE1	USB Host Transmit Configure Type Endpoint 1	Section 27.5.46
0x11B	USBTXINTERVAL1	USB Host Transmit Interval Endpoint 1	Section 27.5.47
0x11C	USBRXTYPE1	USB Host Configure Receive Type Endpoint 1	Section 27.5.48
0x11D	USBRXINTERVAL1	USB Host Receive Polling Interval Endpoint 1	Section 27.5.49
0x120	USBTXMAXP2	USB Maximum Transmit Data Endpoint 2	Section 27.5.39
0x122	USBTXCSSL2	USB Transmit Control and Status Endpoint 2 Low	Section 27.5.40
0x123	USBTXCSRH2	USB Transmit Control and Status Endpoint 2 High	Section 27.5.41
0x124	USBRXMAXP2	USB Maximum Receive Data Endpoint 2	Section 27.5.42
0x126	USBRXCSSL2	USB Receive Control and Status Endpoint 2 Low	Section 27.5.43
0x127	USBRXCSRH2	USB Receive Control and Status Endpoint 2 High	Section 27.5.44
0x128	USBRXCOUNT2	USB Receive Byte Count Endpoint 2	Section 27.5.45
0x12A	USBTXTYPE2	USB Host Transmit Configure Type Endpoint 2	Section 27.5.46
0x12B	USBTXINTERVAL2	USB Host Transmit Interval Endpoint 2	Section 27.5.47
0x12C	USBRXTYPE2	USB Host Configure Receive Type Endpoint 2	Section 27.5.48
0x12D	USBRXINTERVAL2	USB Host Receive Polling Interval Endpoint 2	Section 27.5.49
0x130	USBTXMAXP3	USB Maximum Transmit Data Endpoint 3	Section 27.5.39
0x132	USBTXCSSL3	USB Transmit Control and Status Endpoint 3 Low	Section 27.5.40
0x133	USBTXCSRH3	USB Transmit Control and Status Endpoint 3 High	Section 27.5.41
0x134	USBRXMAXP3	USB Maximum Receive Data Endpoint 3	Section 27.5.42
0x136	USBRXCSSL3	USB Receive Control and Status Endpoint 3 Low	Section 27.5.43
0x137	USBRXCSRH3	USB Receive Control and Status Endpoint 3 High	Section 27.5.44
0x138	USBRXCOUNT3	USB Receive Byte Count Endpoint 3	Section 27.5.45
0x13A	USBTXTYPE3	USB Host Transmit Configure Type Endpoint 3	Section 27.5.46
0x13B	USBTXINTERVAL3	USB Host Transmit Interval Endpoint 3	Section 27.5.47
0x13C	USBRXTYPE3	USB Host Configure Receive Type Endpoint 3	Section 27.5.48
0x13D	USBRXINTERVAL3	USB Host Receive Polling Interval Endpoint 3	Section 27.5.49
0x140	USBTXMAXP4	USB Maximum Transmit Data Endpoint 4	Section 27.5.39
0x142	USBTXCSSL4	USB Transmit Control and Status Endpoint 4 Low	Section 27.5.40
0x143	USBTXCSRH4	USB Transmit Control and Status Endpoint 4 High	Section 27.5.41
0x144	USBRXMAXP4	USB Maximum Receive Data Endpoint 4	Section 27.5.42
0x146	USBRXCSSL4	USB Receive Control and Status Endpoint 4 Low	Section 27.5.43

Table 27-5. USB Registers (continued)

Offset	Acronym	Register Name	Section
0x147	USBRXCSRH4	USB Receive Control and Status Endpoint 4 High	Section 27.5.44
0x148	USBRXCOUNT4	USB Receive Byte Count Endpoint 4	Section 27.5.45
0x14A	USBTXTYPE4	USB Host Transmit Configure Type Endpoint 4	Section 27.5.46
0x14B	USBTXINTERVAL4	USB Host Transmit Interval Endpoint 4	Section 27.5.47
0x14C	USBRXTYPE4	USB Host Configure Receive Type Endpoint 4	Section 27.5.48
0x14D	USBRXINTERVAL4	USB Host Receive Polling Interval Endpoint 4	Section 27.5.49
0x150	USBTXMAXP5	USB Maximum Transmit Data Endpoint 5	Section 27.5.39
0x152	USBTXCSSL5	USB Transmit Control and Status Endpoint 5 Low	Section 27.5.40
0x153	USBTXCSSL5	USB Transmit Control and Status Endpoint 5 High	Section 27.5.41
0x154	USBRXMAXP5	USB Maximum Receive Data Endpoint 5	Section 27.5.42
0x156	USBRXCSSL5	USB Receive Control and Status Endpoint 5 Low	Section 27.5.43
0x157	USBRXCSSL5	USB Receive Control and Status Endpoint 5 High	Section 27.5.44
0x158	USBRXCOUNT5	USB Receive Byte Count Endpoint 5	Section 27.5.45
0x15A	USBTXTYPE5	USB Host Transmit Configure Type Endpoint 5	Section 27.5.46
0x15B	USBTXINTERVAL5	USB Host Transmit Interval Endpoint 5	Section 27.5.47
0x15C	USBRXTYPE5	USB Host Configure Receive Type Endpoint 5	Section 27.5.48
0x15D	USBRXINTERVAL5	USB Host Receive Polling Interval Endpoint 5	Section 27.5.49
0x160	USBTXMAXP6	USB Maximum Transmit Data Endpoint 6	Section 27.5.39
0x162	USBTXCSSL6	USB Transmit Control and Status Endpoint 6 Low	Section 27.5.40
0x163	USBTXCSSL6	USB Transmit Control and Status Endpoint 6 High	Section 27.5.41
0x164	USBRXMAXP6	USB Maximum Receive Data Endpoint 6	Section 27.5.42
0x166	USBRXCSSL6	USB Receive Control and Status Endpoint 6 Low	Section 27.5.43
0x167	USBRXCSSL6	USB Receive Control and Status Endpoint 6 High	Section 27.5.44
0x168	USBRXCOUNT6	USB Receive Byte Count Endpoint 6	Section 27.5.45
0x16A	USBTXTYPE6	USB Host Transmit Configure Type Endpoint 6	Section 27.5.46
0x16B	USBTXINTERVAL6	USB Host Transmit Interval Endpoint 6	Section 27.5.47
0x16C	USBRXTYPE6	USB Host Configure Receive Type Endpoint 6	Section 27.5.48
0x16D	USBRXINTERVAL6	USB Host Receive Polling Interval Endpoint 6	Section 27.5.49
0x170	USBTXMAXP7	USB Maximum Transmit Data Endpoint 7	Section 27.5.39
0x172	USBTXCSSL7	USB Transmit Control and Status Endpoint 7 Low	Section 27.5.40
0x173	USBTXCSSL7	USB Transmit Control and Status Endpoint 7 High	Section 27.5.41
0x174	USBRXMAXP7	USB Maximum Receive Data Endpoint 7	Section 27.5.42
0x176	USBRXCSSL7	USB Receive Control and Status Endpoint 7 Low	Section 27.5.43
0x177	USBRXCSSL7	USB Receive Control and Status Endpoint 7 High	Section 27.5.44
0x178	USBRXCOUNT7	USB Receive Byte Count Endpoint 7	Section 27.5.45
0x17A	USBTXTYPE7	USB Host Transmit Configure Type Endpoint 7	Section 27.5.46
0x17B	USBTXINTERVAL7	USB Host Transmit Interval Endpoint 7	Section 27.5.47
0x17C	USBRXTYPE7	USB Host Configure Receive Type Endpoint 7	Section 27.5.48
0x17D	USBRXINTERVAL7	USB Host Receive Polling Interval Endpoint 7	Section 27.5.49
0x200	USBDMAINTR	USB DMA Interrupt	Section 27.5.50
0x204	USBDMACTL0	USB DMA Control 0	Section 27.5.51
0x208	USBDMAADDR0	USB DMA Address 0	Section 27.5.52
0x20C	USBDMACOUNT0	USB DMA Count 0	Section 27.5.53
0x214	USBDMACTL1	USB DMA Control 1	Section 27.5.51
0x218	USBDMAADDR1	USB DMA Address 1	Section 27.5.52
0x21C	USBDMACOUNT1	USB DMA Count 1	Section 27.5.53
0x224	USBDMACTL2	USB DMA Control 2	Section 27.5.51

Table 27-5. USB Registers (continued)

Offset	Acronym	Register Name	Section
0x228	USBDMAADDR2	USB DMA Address 2	Section 27.5.52
0x22C	USBDMACOUNT2	USB DMA Count 2	Section 27.5.53
0x234	USBDMACTL3	USB DMA Control 3	Section 27.5.51
0x238	USBDMAADDR3	USB DMA Address 3	Section 27.5.52
0x23C	USBDMACOUNT3	USB DMA Count 3	Section 27.5.53
0x244	USBDMACTL4	USB DMA Control 4	Section 27.5.51
0x248	USBDMAADDR4	USB DMA Address 4	Section 27.5.52
0x24C	USBDMACOUNT4	USB DMA Count 4	Section 27.5.53
0x254	USBDMACTL5	USB DMA Control 5	Section 27.5.51
0x258	USBDMAADDR5	USB DMA Address 5	Section 27.5.52
0x25C	USBDMACOUNT5	USB DMA Count 5	Section 27.5.53
0x264	USBDMACTL6	USB DMA Control 6	Section 27.5.51
0x268	USBDMAADDR6	USB DMA Address 6	Section 27.5.52
0x26C	USBDMACOUNT6	USB DMA Count 6	Section 27.5.53
0x274	USBDMACTL7	USB DMA Control 7	Section 27.5.51
0x278	USBDMAADDR7	USB DMA Address 7	Section 27.5.52
0x27C	USBDMACOUNT7	USB DMA Count 7	Section 27.5.53
0x304	USBRQPKTCOUNT1	USB Request Packet Count in Block Transfer Endpoint 1	Section 27.5.54
0x308	USBRQPKTCOUNT2	USB Request Packet Count in Block Transfer Endpoint 2	Section 27.5.54
0x30C	USBRQPKTCOUNT3	USB Request Packet Count in Block Transfer Endpoint 3	Section 27.5.54
0x310	USBRQPKTCOUNT4	USB Request Packet Count in Block Transfer Endpoint 4	Section 27.5.54
0x314	USBRQPKTCOUNT5	USB Request Packet Count in Block Transfer Endpoint 5	Section 27.5.54
0x318	USBRQPKTCOUNT6	USB Request Packet Count in Block Transfer Endpoint 6	Section 27.5.54
0x31C	USBRQPKTCOUNT7	USB Request Packet Count in Block Transfer Endpoint 7	Section 27.5.54
0x340	USBRXDPKTBUFFDIS	USB Receive Double Packet Buffer Disable	Section 27.5.55
0x342	USBTXDPKTBUFFDIS	USB Transmit Double Packet Buffer Disable	Section 27.5.56
0x344	USBCTO	USB Chirp Time-out	Section 27.5.57
0x346	USBHHSRTN	USB High Speed to UTM Operating Delay	Section 27.5.58
0x348	USBHSBT	USB High Speed Time-out Adder	Section 27.5.59
0x360	USBLPMATTR	USB LPM Attributes	Section 27.5.60
0x362	USBLPMCNTL	USB LPM Control	Section 27.5.61
0x363	USBLPMIM	USB LPM Interrupt Mask	Section 27.5.62
0x364	USBLPMRIS	USB LPM Raw Interrupt Status	Section 27.5.63
0x365	USBLPMFADDR	USB LPM Function Address	Section 27.5.64
0x400	USBEPCC	USB External Power Control	Section 27.5.65
0x404	USBEPCCRIS	USB External Power Control Raw Interrupt Status	Section 27.5.66
0x408	USBEPCCIM	USB External Power Control Interrupt Mask	Section 27.5.67
0x40C	USBEPCCISC	USB External Power Control Interrupt Status and Clear	Section 27.5.68
0x410	USBDRRIS	USB Device RESUME Raw Interrupt Status	Section 27.5.69
0x414	USBDRRIM	USB Device RESUME Interrupt Mask	Section 27.5.70
0x418	USBDRRISC	USB Device RESUME Interrupt Status and Clear	Section 27.5.71
0x41C	USBGPCS	USB General-Purpose Control and Status	Section 27.5.72
0x430	USBVDC	USB VBUS Droop Control	Section 27.5.73
0x434	USBVDCRIS	USB VBUS Droop Control Raw Interrupt Status	Section 27.5.74
0x438	USBVDCIM	USB VBUS Droop Control Interrupt Mask	Section 27.5.75
0x43C	USBVDCISC	USB VBUS Droop Control Interrupt Status and Clear	Section 27.5.76
0xFC0	USBPP	USB Peripheral Properties	Section 27.5.77

Table 27-5. USB Registers (continued)

Offset	Acronym	Register Name	Section
0xFC4	USBPC	USB Peripheral Configuration	Section 27.5.78
0xFC8	USBC	USB Clock Configuration	Section 27.5.79

Complex bit access types are encoded to fit into small table cells. [Table 27-6](#) shows the codes that are used for access types in this section.

Table 27-6. USB Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W0	W	Write
W1C	1C W	1 to clear Write
W1S	1S W	1 to set Write
Reset or Default Value		
-n		Value after reset or the default value

27.5.1 USBFADDR Register (Offset = 0x0) [reset = 0x0]

USB Device Functional Address (USBFADDR)

OTG B / Device

USBFADDR is an 8-bit register that contains the 7-bit address of the Device part of the transaction.

When the USB controller is being used in Device mode (the HOST bit in the USBDEVCTL register is clear), this register must be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

NOTE: See [Section 27.3.1.5.3](#) for special considerations when writing this register.

USBFADDR is shown in [Figure 27-2](#) and described in [Table 27-7](#).

Return to [Summary Table](#).

Figure 27-2. USBFADDR Register

7	6	5	4	3	2	1	0
RESERVED	FUNCADDR						
R-0x0	R/W-0x0						

Table 27-7. USBFADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	FUNCADDR	R/W	0x0	Function Address Function Address of Device as received through SET_ADDRESS.

27.5.2 USBPOWER Register (Offset = 0x1) [reset = 0xA0]

USB Power (USBPOWER)

OTG A / Host

OTG B / Device

USBPOWER is an 8-bit register used for controlling SUSPEND and RESUME signaling and some basic operational aspects of the USB controller.

USBPOWER for OTG A / Host is shown in [Figure 27-3](#) and described in [Table 27-8](#).

USBPOWER for OTG B / Device is shown in [Figure 27-4](#) and described in [Table 27-9](#).

Return to [Summary Table](#).

Figure 27-3. USBPOWER Register (OTG A / Host)

7	6	5	4	3	2	1	0
RESERVED		HSENAB	HSMODE	RESET	RESUME	SUSPEND	PWRDNPHY
R-0x2		R/W-0x1	R-0x0	R/W-0x0	R/W-0x0	R/W1S-0x0	R/W-0x0

Table 27-8. USBPOWER Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0x2	
5	HSENAB	R/W	0x1	High Speed Enable. 0x0 = The USB operates in full-speed mode. 0x1 = The USB negotiates for high-speed mode when the device is reset by the hub.
4	HSMODE	R	0x0	High Speed Mode. This bit becomes valid when the RESET bit (bit 3) is cleared. Allowance is made for Tiny-J signaling in determining the transfer speed to select. 0x0 = The USB operates in full-speed mode. 0x1 = USB reset has completed, and the high-speed mode has been successfully negotiated.
3	RESET	R/W	0x0	RESET Signaling. 0x0 = Ends RESET signaling on the bus. 0x1 = Enables RESET signaling on the bus.
2	RESUME	R/W	0x0	RESUME Signaling. This bit must be cleared by software 20 ms after being set. 0x0 = Ends RESUME signaling on the bus. 0x1 = Enables RESUME signaling when the Device is in SUSPEND mode.
1	SUSPEND	R/W1S	0x0	SUSPEND Mode. 0x0 = No effect. 0x1 = Enables SUSPEND mode.
0	PWRDNPHY	R/W	0x0	Power Down PHY. 0x0 = No effect. 0x1 = Powers down the internal USB PHY.

Figure 27-4. USBPOWER Register (OTG B / Device)

7	6	5	4	3	2	1	0
ISOUP	SOFTCONN	HSENAB	HSMODE	RESET	RESUME	SUSPEND	PWRDNPHY
R/W-0x0	R/W-0x0	R/W-0x1	R-0x0	R-0x0	R/W-0x0	R-0x0	R/W-0x0

Table 27-9. USBPOWER Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	ISOUP	R/W	0x0	Isochronous Update. This bit is only valid for isochronous transfers. 0x0 = No effect. 0x1 = The USB controller waits for an SOF token from the time the TXRDY bit is set in the USBTXCSRLn register before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.
6	SOFTCONN	R/W	0x0	Soft Connect/Disconnect. 0x0 = The USB D+/D- lines are tri-stated. 0x1 = The USB D+/D- lines are enabled.
5	HSENAB	R/W	0x1	High Speed Enable. 0x0 = The USB operates in full-speed mode. 0x1 = The USB negotiates for high-speed mode when the device is reset by the hub.
4	HSMODE	R	0x0	High Speed Enable. Allowance is made for Tiny-J signaling in determining the transfer speed to select. 0x0 = The USB operates in full-speed mode. 0x1 = USB reset has completed, and the high-speed mode has been successfully negotiated.
3	RESET	R	0x0	RESET Signaling. 0x0 = RESET signaling is not present on the bus. 0x1 = RESET signaling is present on the bus.
2	RESUME	R/W	0x0	RESUME Signaling. This bit must be cleared by software 10 ms (a maximum of 15 ms) after being set. 0x0 = Ends RESUME signaling on the bus. 0x1 = Enables RESUME signaling when the Device is in SUSPEND mode.
1	SUSPEND	R	0x0	SUSPEND Mode. 0x0 = This bit is cleared when software reads the interrupt register or sets the RESUME bit above. 0x1 = The USB controller is in SUSPEND mode.
0	PWRDNPHY	R/W	0x0	Power Down PHY. 0x0 = No effect. 0x1 = Powers down the internal USB PHY.

27.5.3 USBTXIS Register (Offset = 0x2) [reset = 0x0]

USB Transmit Interrupt Status (USBTXIS)

OTG A / Host

OTG B / Device

USBTXIS is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1â€”7. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP7 bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints. The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

NOTE: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

USBTXIS is shown in [Figure 27-5](#) and described in [Table 27-10](#).

Return to [Summary Table](#).

Figure 27-5. USBTXIS Register

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 27-10. USBTXIS Register Field Descriptions

Bit	Field	Type	Reset	Description
7	EP7	R	0x0	TX Endpoint 7 Interrupt. 0x0 = No interrupt. 0x1 = The Endpoint 7 transmit interrupt is asserted.
6	EP6	R	0x0	TX Endpoint 6 Interrupt. Same description as EP7.
5	EP5	R	0x0	TX Endpoint 5 Interrupt. Same description as EP7.
4	EP4	R	0x0	TX Endpoint 4 Interrupt. Same description as EP7.
3	EP3	R	0x0	TX Endpoint 3 Interrupt. Same description as EP7.
2	EP2	R	0x0	TX Endpoint 2 Interrupt. Same description as EP7.
1	EP1	R	0x0	TX Endpoint 1 Interrupt. Same description as EP7.
0	EP0	R	0x0	TX and RX Endpoint 0 Interrupt. 0x0 = No interrupt. 0x1 = The Endpoint 0 transmit and receive interrupt is asserted.

27.5.4 USBRXIS Register (Offset = 0x4) [reset = 0x0]

USB Receive Interrupt Status (USBRXIS)

OTG A / Host

OTG B / Device

USBRXIS is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1 to 7 are currently active.

NOTE: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

USBRXIS is shown in [Figure 27-6](#) and described in [Table 27-11](#).

Return to [Summary Table](#).

Figure 27-6. USBRXIS Register

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	RESERVED
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 27-11. USBRXIS Register Field Descriptions

Bit	Field	Type	Reset	Description
7	EP7	R	0x0	RX Endpoint 7 Interrupt. 0x0 = No interrupt. 0x1 = The Endpoint 7 transmit interrupt is asserted.
6	EP6	R	0x0	RX Endpoint 6 Interrupt. Same description as EP7.
5	EP5	R	0x0	RX Endpoint 5 Interrupt. Same description as EP7.
4	EP4	R	0x0	RX Endpoint 4 Interrupt. Same description as EP7.
3	EP3	R	0x0	RX Endpoint 3 Interrupt. Same description as EP7.
2	EP2	R	0x0	RX Endpoint 2 Interrupt. Same description as EP7.
1	EP1	R	0x0	RX Endpoint 1 Interrupt. Same description as EP7.
0	RESERVED	R	0x0	

27.5.5 USBTXIE Register (Offset = 0x6) [reset = 0xFF]

USB Transmit Interrupt Enable (USBTXIE)

OTG A / Host

OTG B / Device

USBTXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the USBTXIS register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the USBTXIS register is set. When a bit is cleared, the interrupt in the USBTXIS register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

USBTXIE is shown in [Figure 27-7](#) and described in [Table 27-12](#).

Return to [Summary Table](#).

Figure 27-7. USBTXIE Register

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1

Table 27-12. USBTXIE Register Field Descriptions

Bit	Field	Type	Reset	Description
7	EP7	R/W	0x1	TX Endpoint 7 Interrupt Enable. 0x0 = The EP7 transmit interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the EP7 bit in the USBTXIS register is set.
6	EP6	R/W	0x1	TX Endpoint 6 Interrupt Enable. Same description as EP7.
5	EP5	R/W	0x1	TX Endpoint 5 Interrupt Enable. Same description as EP7.
4	EP4	R/W	0x1	TX Endpoint 4 Interrupt Enable. Same description as EP7.
3	EP3	R/W	0x1	TX Endpoint 3 Interrupt Enable. Same description as EP7.
2	EP2	R/W	0x1	TX Endpoint 2 Interrupt Enable. Same description as EP7.
1	EP1	R/W	0x1	TX Endpoint 1 Interrupt Enable. Same description as EP7.
0	EP0	R/W	0x1	TX and RX Endpoint 0 Interrupt Enable. 0x0 = The EP0 transmit and receive interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the EP0 bit in the USBTXIS register is set.

27.5.6 USBRXIE Register (Offset = 0x8) [reset = 0xFE]

USB Receive Interrupt Enable (USBRXIE)

OTG A / Host

OTG B / Device

USBRXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the USBRXIS register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the USBRXIS register is set. When a bit is cleared, the interrupt in the USBRXIS register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

USBRXIE is shown in [Figure 27-8](#) and described in [Table 27-13](#).

Return to [Summary Table](#).

Figure 27-8. USBRXIE Register

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	RESERVED
R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R/W-0x1	R-0x0

Table 27-13. USBRXIE Register Field Descriptions

Bit	Field	Type	Reset	Description
7	EP7	R/W	0x1	RX Endpoint 7 Interrupt Enable. 0x0 = The EP7 receive interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the EP7 bit in the USBRXIS register is set.
6	EP6	R/W	0x1	RX Endpoint 6 Interrupt Enable. Same description as EP7.
5	EP5	R/W	0x1	RX Endpoint 5 Interrupt Enable. Same description as EP7.
4	EP4	R/W	0x1	RX Endpoint 4 Interrupt Enable. Same description as EP7.
3	EP3	R/W	0x1	RX Endpoint 3 Interrupt Enable. Same description as EP7.
2	EP2	R/W	0x1	RX Endpoint 2 Interrupt Enable. Same description as EP7.
1	EP1	R/W	0x1	RX Endpoint 1 Interrupt Enable. Same description as EP7.
0	RESERVED	R	0x0	

27.5.7 USBIS Register (Offset = 0xA) [reset = 0x0]

USB General Interrupt Status (USBIS)

OTG A / Host

OTG B / Device

USBIS is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

USBIS for OTG A / Host is shown in [Figure 27-9](#) and described in [Table 27-14](#).

USBIS for OTG B / Device is shown in [Figure 27-10](#) and described in [Table 27-15](#).

Return to [Summary Table](#).

Figure 27-9. USBIS Register (OTG A / Host)

7	6	5	4	3	2	1	0
VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	RESERVED
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 27-14. USBIS Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	VBUSERR	R	0x0	VBUS Error. 0x0 = No interrupt. 0x1 = VBUS has dropped below the VBUS Valid threshold during a session.
6	SESREQ	R	0x0	SESSION REQUEST. 0x0 = No interrupt. 0x1 = SESSION REQUEST signaling has been detected.
5	DISCON	R	0x0	Session Disconnect. 0x0 = No interrupt. 0x1 = A Device disconnect has been detected.
4	CONN	R	0x0	Session Connect. 0x0 = No interrupt. 0x1 = A Device connection has been detected.
3	SOF	R	0x0	Start of Frame. 0x0 = No interrupt. 0x1 = A new frame has started.
2	BABBLE	R	0x0	Babble Detected. 0x0 = No interrupt. 0x1 = Babble has been detected. This interrupt is active only after the first SOF has been sent.
1	RESUME	R	0x0	RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. 0x0 = No interrupt. 0x1 = RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.
0	RESERVED	R	0x0	

Figure 27-10. USBIS Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED	DISCON	RESERVED	SOF	RESET	RESUME	SUSPEND	
R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0	R-0x0

Table 27-15. USBIS Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0x0	
5	DISCON	R	0x0	Session Disconnect. 0x0 = No interrupt. 0x1 = The device has been disconnected from the host.
4	RESERVED	R	0x0	
3	SOF	R	0x0	Start of Frame. 0x0 = No interrupt. 0x1 = A new frame has started.
2	RESET	R	0x0	RESET Signaling Detected. 0x0 = No interrupt. 0x1 = RESET signaling has been detected on the bus.
1	RESUME	R	0x0	RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. 0x0 = No interrupt. 0x1 = RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.
0	SUSPEND	R	0x0	SUSPEND Signaling Detected. 0x0 = No interrupt. 0x1 = SUSPEND signaling has been detected on the bus.

27.5.8 USBIE Register (Offset = 0xB) [reset = 0x6]

USB Interrupt Enable (USBIE)

OTG A / Host

OTG B / Device

USBIE is an 8-bit register that provides interrupt enable bits for each of the interrupts in USBIS. At reset interrupts 1 and 2 are enabled in Device mode.

USBIE for OTG A / Host is shown in [Figure 27-11](#) and described in [Table 27-16](#).

USBIE for OTG B / Device is shown in [Figure 27-12](#) and described in [Table 27-17](#).

Return to [Summary Table](#).

Figure 27-11. USBIE Register (OTG A / Host)

7	6	5	4	3	2	1	0
VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	RESERVED
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x1	R/W-0x1	R-0x0

Table 27-16. USBIE Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	VBUSERR	R/W	0x0	Enable VBUS Error Interrupt. 0x0 = The VBUSERR interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the VBUSERR bit in the USBIS register is set.
6	SESREQ	R/W	0x0	Enable Session Request. 0x0 = The SESREQ interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the SESREQ bit in the USBIS register is set.
5	DISCON	R/W	0x0	Enable Disconnect Interrupt. 0x0 = The DISCON interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
4	CONN	R/W	0x0	Enable Connect Interrupt. 0x0 = The CONN interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the CONN bit in the USBIS register is set.
3	SOF	R/W	0x0	Enable Start-of-Frame Interrupt. 0x0 = The SOF interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set.
2	BABBLE	R/W	0x1	Enable Babble Interrupt. 0x0 = The BABBLE interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the BABBLE bit in the USBIS register is set.
1	RESUME	R/W	0x1	Enable RESUME Interrupt. 0x0 = The RESUME interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set.
0	RESERVED	R	0x0	

Figure 27-12. USBIE Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED	DISCON	RESERVED	SOF	RESET	RESUME	SUSPEND	
R-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x1	R/W-0x1	R/W-0x0	

Table 27-17. USBIE Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0x0	
5	DISCON	R/W	0x0	Enable Disconnect Interrupt. 0x0 = The DISCON interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
4	RESERVED	R	0x0	
3	SOF	R/W	0x0	Enable Start-of-Frame Interrupt. 0x0 = The SOF interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set.
2	RESET	R/W	0x1	Enable RESET Interrupt. 0x0 = The RESET interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RESET bit in the USBIS register is set.
1	RESUME	R/W	0x1	Enable RESUME Interrupt. 0x0 = The RESUME interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set.
0	SUSPEND	R/W	0x0	Enable SUSPEND Interrupt. 0x0 = The SUSPEND interrupt is suppressed and not sent to the interrupt controller. 0x1 = An interrupt is sent to the interrupt controller when the SUSPEND bit in the USBIS register is set.

27.5.9 USBFRAME Register (Offset = 0xC) [reset = 0x0]

USB Frame Value (USBFRAME)

OTG A / Host

OTG B / Device

USBFRAME is a 16-bit read-only register that holds the last received frame number.

USBFRAME is shown in [Figure 27-13](#) and described in [Table 27-18](#).

Return to [Summary Table](#).

Figure 27-13. USBFRAME Register

15	14	13	12	11	10	9	8
RESERVED						FRAME	
R-0x0						R-0x0	
7	6	5	4	3	2	1	0
FRAME							
R-0x0							

Table 27-18. USBFRAME Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0x0	
10-0	FRAME	R	0x0	Frame Number.

27.5.10 USBEPIDX Register (Offset = 0xE) [reset = 0x0]

USB Endpoint Index (USBEPIDX)

OTG A / Host

OTG B / Device

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The USBEPIDX 8-bit register is used with the USBTXFIFOSZ, USBRXFIFOSZ, USBTXFIFOADD, and USBRXFIFOADD registers.

USBEPIDX is shown in [Figure 27-14](#) and described in [Table 27-19](#).

Return to [Summary Table](#).

Figure 27-14. USBEPIDX Register

7	6	5	4	3	2	1	0
RESERVED				EPIDX			
R-0x0				R/W-0x0			

Table 27-19. USBEPIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0x0	
3-0	EPIDX	R/W	0x0	Endpoint Index. This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0x7 corresponds to Endpoint 7.

27.5.11 USBTEST Register (Offset = 0xF) [reset = 0x0]

USB Test Mode (USBTEST)

OTG A / Host

OTG B / Device

USBTEST is an 8-bit register that is primarily used to put the USB controller into one of the four test modes for operation described in the USB 2.0 Specification, in response to a SET FEATURE: USBTESTMODE command. This register is not used in normal operation.

NOTE: Only one of these bits should be set at any time.

USBTEST for OTG A / Host is shown in [Figure 27-15](#) and described in [Table 27-20](#).

USBTEST for OTG B / Device is shown in [Figure 27-16](#) and described in [Table 27-21](#).

Return to [Summary Table](#).

Figure 27-15. USBTEST Register (OTG A / Host)

7	6	5	4	3	2	1	0
FORCEH	FIFOACC	FORCEFS	FORCEHS	TESTPKT	TESTK	TESTJ	TESTSE0NAK
R/W-0x0	R/W1S-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-20. USBTEST Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	FORCEH	R/W	0x0	Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit and FORCEHS bit. The operating speed is as follows: FORCEHS = 0 and FORCEFS = 0: Low Speed FORCEHS = 0 and FORCEFS = 1: Full Speed FORCEHS = 1 and FORCEFS = 0: High Speed FORCEHS = 1 and FORCEFS = 1: Undefined 0x0 = No effect. 0x1 = Forces the USB controller to enter Host mode when the SESSION bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the USB0DP and USB0DM signals is ignored. The USB controller then remains in Host mode until the SESSION bit is cleared, even if a Device is disconnected. If the FORCEH bit remains set, the USB controller re-enters Host mode the next time the SESSION bit is set.
6	FIFOACC	R/W1S	0x0	FIFO Access. This bit is cleared automatically. 0x0 = No effect. 0x1 = Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.
5	FORCEFS	R/W	0x0	Force Full-Speed Mode. If FORCEH is set, then this bit is used in combination with FORCEHS to determine the operating speed. If FORCEH is not set, then this operates as: 0x0 = The USB controller operates at Low Speed. 0x1 = Forces the USB controller into Full-Speed mode upon receiving a USB RESET.
4	FORCEHS	R/W	0x0	Force High-Speed Mode. If FORCEH is set, then this bit is used in combination with FORCEFS to determine the operating speed. If FORCEH is not set, then this operates as: 0x0 = The USB controller operates at Low Speed. 0x1 = Forces the USB controller into High-Speed mode upon receiving a USB RESET.

Table 27-20. USBTEST Register Field Descriptions (OTG A / Host) (continued)

Bit	Field	Type	Reset	Description
3	TESTPKT	R/W	0x0	Test Packet Mode Enable. When this mode is entered, the USB repetitively transmits on the bus a 53-byte test packet, the form which is defined in the Universal Serial Bus Specification, Revision 2.0. The test packet has a fixed format and must be loaded into the Endpoint 0 FIFO before test mode is entered. 0x0 = Test packet mode disabled. 0x1 = Test packet mode enabled.
2	TESTK	R/W	0x0	Test_K Mode Enable. In this mode, a continuous K is transmitted on the bus. 0x0 = Test_K mode disabled 0x1 = Test_K mode enabled
1	TESTJ	R/W	0x0	Test_J Mode Enable. In this mode, a continuous J is transmitted on the bus. 0x0 = Test_J mode disabled 0x1 = Test_J Mode enabled.
0	TESTSE0NAK	R/W	0x0	Test_SE0_NAK Test Mode Enable. In this mode, the USB remains in high-speed mode but responds to any valid IN token with a NAK. 0x0 = Test_SE0_NAK test mode disabled. 0x1 = Test_SE0_NAK test mode enabled.

Figure 27-16. USBTEST Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED	FIFOACC	FORCEFS	RESERVED				
R-0x0	R/W1S-0x0	R/W-0x0	R-0x0				

Table 27-21. USBTEST Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6	FIFOACC	R/W1S	0x0	FIFO Access. This bit is cleared automatically. 0x0 = No effect. 0x1 = Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.
5	FORCEFS	R/W	0x0	Force Full-Speed Mode. If FORCEH is set, then this bit is used in combination with FORCEHS to determine the operating speed. If FORCEH is not set, then this operates as: 0x0 = The USB controller operates at Low Speed. 0x1 = Forces the USB controller into Full-Speed mode upon receiving a USB RESET.
4-0	RESERVED	R	0x0	

27.5.12 USBFIFOn Register [reset = 0x0]

USB FIFO Endpoint 0 (USBFIFO0), offset 0x020

USB FIFO Endpoint 1 (USBFIFO1), offset 0x024

USB FIFO Endpoint 2 (USBFIFO2), offset 0x028

USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C

USB FIFO Endpoint 4 (USBFIFO4), offset 0x030

USB FIFO Endpoint 5 (USBFIFO5), offset 0x034

USB FIFO Endpoint 6 (USBFIFO6), offset 0x038

USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C

OTG A / Host

OTG B / Device

These 32-bit registers provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of accesses is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, halfword- or word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering (see [Section 27.3.1.3.1](#)). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1 to 7, the associated FIFO is completely flushed.

USBFIFOn is shown in [Figure 27-17](#) and described in [Table 27-22](#).

Return to [Summary Table](#).

Figure 27-17. USBFIFOn Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDATA																															
R/W-0x0																															

Table 27-22. USBFIFOn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	EPDATA	R/W	0x0	Endpoint Data. Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

27.5.13 USBDEVCTL Register (Offset = 0x60) [reset = 0x80]

USB Device Control (USBDEVCTL)

OTG A / Host

USBDEVCTL is an 8-bit register used for controlling and monitoring the USB VBUS line. If the PHY is suspended, no PHY clock is received and the VBUS is not sampled. In addition, in Host mode, USBDEVCTL provides the status information for the current operating mode (Host or Device) of the USB controller. If the USB controller is in Host mode, this register also indicates if a full- or low-speed Device has been connected.

USBDEVCTL is shown in [Figure 27-18](#) and described in [Table 27-23](#).

Return to [Summary Table](#).

Figure 27-18. USBDEVCTL Register

7	6	5	4	3	2	1	0
DEV	FSDEV	LSDEV	VBUS		HOST	HOSTREQ	SESSION
R-0x1	R-0x0	R-0x0	R-0x0		R-0x0	R/W-0x0	R/W-0x0

Table 27-23. USBDEVCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
7	DEV	R	0x1	Device Mode. This value is only valid while a session is in progress. 0x0 = The USB controller is operating on the OTG A side of the cable. 0x1 = The USB controller is operating on the OTG B side of the cable.
6	FSDEV	R	0x0	Full-Speed Device Detected. 0x0 = A full-speed Device has not been detected on the port. 0x1 = A full-speed Device has been detected on the port.
5	LSDEV	R	0x0	Low-Speed Device Detected. 0x0 = A low-speed Device has not been detected on the port. 0x1 = A low-speed Device has been detected on the port.
4-3	VBUS	R	0x0	VBUS Level. 0x0 = Below SessionEnd VBUS is detected as under 0.5 V. 0x1 = Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V. 0x2 = Above AValid, below VBUSValid VBUS is detected as above 1.5 V and below 4.75 V. 0x3 = Above VBUSValid VBUS is detected as above 4.75 V.
2	HOST	R	0x0	Host Mode. This value is only valid while a session is in progress. 0x0 = The USB controller is acting as a Device. 0x1 = The USB controller is acting as a Host.
1	HOSTREQ	R/W	0x0	Host Request. This bit is cleared when Host Negotiation is completed. 0x0 = No effect. 0x1 = Initiates the Host Negotiation when SUSPEND mode is entered.
0	SESSION	R/W	0x0	Session Start/End. When operating as an OTG A device: When operating as an OTG B device: Clearing this bit when the USB controller is not suspended results in undefined behavior. 0x0 = When cleared by software, this bit ends a session. 0x1 = When set by software, this bit starts a session.

27.5.14 USBCCONF Register (Offset = 0x61) [reset = 0x0]

USB Common Configuration (USBCCONF)

OTG A / Host

OTG B / Device

The USBCCONF register is an 8-bit register that contains various common configuration bits. These bits include the RX/TX early DMA enable bits.

USBCCONF is shown in [Figure 27-19](#) and described in [Table 27-24](#).

Return to [Summary Table](#).

Figure 27-19. USBCCONF Register

7	6	5	4	3	2	1	0
RESERVED						TXEDMA	RXEDMA
R-0x0						R/W-0x0	R/W-0x0

Table 27-24. USBCCONF Register Field Descriptions

Bit	Field	Type	Reset	Description
7-2	RESERVED	R	0x0	
1	TXEDMA	R/W	0x0	TX Early DMA Enable. 0x0 = Late Mode: DMAREQ signal for all IN endpoints is deasserted when the MAXP bytes have been written to an endpoint. 0x1 = Early Mode: DMAREQ signal for all IN endpoints is deasserted when MAXP-8 bytes have been written to an endpoint.
0	RXEDMA	R/W	0x0	TX Early DMA Enable. 0x0 = Late Mode: DMAREQ signal for all OUT endpoints is deasserted when the MAXP bytes have been read to an endpoint. 0x1 = Early Mode: DMAREQ signal for all OUT endpoints is deasserted when MAXP-8 bytes have been read to an endpoint.

27.5.15 USBnXFIFOSZ Register [reset = 0x0]

USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062

USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063

OTG A / Host

OTG B / Device

These 8-bit registers allow the selected TX/RX endpoint FIFOs to be dynamically sized. USBEPIDX is used to configure each transmit endpoint's FIFO size.

USBnXFIFOSZ is shown in [Figure 27-20](#) and described in [Table 27-25](#).

Return to [Summary Table](#).

Figure 27-20. USBnXFIFOSZ Register

7	6	5	4	3	2	1	0
RESERVED			DPB	SIZE			
R-0x0			R/W-0x0	R/W-0x0			

Table 27-25. USBnXFIFOSZ Register Field Descriptions

Bit	Field	Type	Reset	Description
7-5	RESERVED	R	0x0	
4	DPB	R/W	0x0	Double Packet Buffer Support. 0x0 = Only single-packet buffering is supported. 0x1 = Double-packet buffering is supported.
3-0	SIZE	R/W	0x0	Max Packet Size. Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size. If DPB = 1, the FIFO is twice this size. 0x0 = 8-byte packet size 0x1 = 16-byte packet size 0x2 = 32-byte packet size 0x3 = 64-byte packet size 0x4 = 128-byte packet size 0x5 = 256-byte packet size 0x6 = 512-byte packet size 0x7 = 1024-byte packet size 0x8 = 2048-byte packet size 0x9 to 0xF = Reserved

27.5.16 USBnXFIFOADD Register [reset = 0x0]

USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064

USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066

OTG A / Host

OTG B / Device

USBTXFIFOADD and USBRXFIFOADD are 16-bit registers that control the start address of the selected transmit and receive endpoint FIFOs.

USBnXFIFOADD is shown in [Figure 27-21](#) and described in [Table 27-26](#).

Return to [Summary Table](#).

Figure 27-21. USBnXFIFOADD Register

15	14	13	12	11	10	9	8
RESERVED							ADDR
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
ADDR							
R/W-0x0							

Table 27-26. USBnXFIFOADD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0x0	
8-0	ADDR	R/W	0x0	Transmit/Receive Start Address. Start address of the endpoint FIFO = ADDR*8.

27.5.17 ULPIVBUSCTL Register (Offset = 0x70) [reset = 0x0]

USB ULPI VBUS Control (ULPIVBUSCTL)

OTG A / Host

OTG B / Device

ULPI PHYs may use an external charge pump to generate VBUS rather than an internal charge pump. This register allows selection of the external charge pump. It also allows this selection to be displayed through the external USB0VBUS signal.

NOTE: This register is read back from the PHY clock domain. These bits will not therefore return updated values while the PHY is suspended.

ULPIVBUSCTL is shown in [Figure 27-22](#) and described in [Table 27-27](#).

Return to [Summary Table](#).

Figure 27-22. ULPIVBUSCTL Register

7	6	5	4	3	2	1	0
RESERVED						USEEXTVBUSIND	USEEXTVBUS
R-0x0						R/W-0x0	R/W-0x0

Table 27-27. ULPIVBUSCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
7-2	RESERVED	R	0x0	
1	USEEXTVBUSIND	R/W	0x0	Use External VBUS Indicator. 0x0 = An external VBUS indicator is not used in the PHY's VBUS state determination. 0x1 = An external VBUS indicator is used in the PHY's VBUS state determination.
0	USEEXTVBUS	R/W	0x0	Use External VBUS. 0x0 = An external charge pump is not used. 0x1 = An external charge pump is used.

27.5.18 ULPIREGDATA Register (Offset = 0x74) [reset = 0x0]

USB ULPI Register Data (ULPIREGDATA)

OTG A / Host

OTG B / Device

The ULPIREGDATA register contains the data associated with register reads/writes conducted through the ULPI interface.

ULPIREGDATA is shown in [Figure 27-23](#) and described in [Table 27-28](#).

Return to [Summary Table](#).

Figure 27-23. ULPIREGDATA Register

7	6	5	4	3	2	1	0
REGDATA							
R/W-0x0							

Table 27-28. ULPIREGDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	REGDATA	R/W	0x0	Register Data. This field contains the register data for USB PHY register accesses.

27.5.19 ULPIREGADDR Register (Offset = 0x75) [reset = 0x0]

USB ULPI Register Address (ULPIREGADDR)

OTG A / Host

OTG B / Device

The ULPIREGADDR register contains the address of the register being read/written through the ULPI interface.

ULPIREGADDR is shown in [Figure 27-24](#) and described in [Table 27-29](#).

Return to [Summary Table](#).

Figure 27-24. ULPIREGADDR Register

7	6	5	4	3	2	1	0
REGADDR							
R/W-0x0							

Table 27-29. ULPIREGADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	REGADDR	R/W	0x0	Register Address. This field contains the register address for USB PHY register accesses.

27.5.20 ULPIREGCTL Register (Offset = 0x76) [reset = 0x0]

USB ULPI Register Control (ULPIREGCTL)

OTG A / Host

OTG B / Device

The ULPIREGCTL register contains control and status bits relating to the register being read/written through the ULPI interface.

ULPIREGCTL is shown in [Figure 27-25](#) and described in [Table 27-30](#).

Return to [Summary Table](#).

Figure 27-25. ULPIREGCTL Register

7	6	5	4	3	2	1	0
RESERVED					RDWR	REGCMPLT	REGACC
R-0x0					R/W-0x0	R/W0-0x0	R/W-0x0

Table 27-30. ULPIREGCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
7-3	RESERVED	R	0x0	
2	RDWR	R/W	0x0	Read/Write Control. 0x0 = Write access 0x1 = Read access
1	REGCMPLT	R/W0	0x0	Register Access Complete. This bit is set by the link when the register access is complete and must be cleared by software.
0	REGACC	R/W	0x0	Initiate Register Access. This bit is used to initiate a register access and is cleared automatically when the register access is complete and REGCMPLT = 1. 0x0 = Register access complete 0x1 = Initiate register access

27.5.21 USBEPINFO Register (Offset = 0x78) [reset = 0x77]

USB Endpoint Information (USBEPINFO)

This 8-bit read-only register allows read-back of the number of TX and Rx endpoints included in the design.

USBEPINFO is shown in [Figure 27-26](#) and described in [Table 27-31](#).

Return to [Summary Table](#).

Figure 27-26. USBEPINFO Register

7	6	5	4	3	2	1	0
RXEP				TXEP			
R-0x7				R-0x7			

Table 27-31. USBEPINFO Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RXEP	R	0x7	RX Endpoints. The number of Rx endpoints implemented in the design.
3-0	TXEP	R	0x7	TX Endpoints. The number of TX endpoints implemented in the design.

27.5.22 USBRAMINFO Register (Offset = 0x79) [reset = 0x8A]

USB RAM Information (USBAMINFO)

This 8-bit read-only register provides information about the width of the RAM.

USBAMINFO is shown in [Figure 27-27](#) and described in [Table 27-32](#).

Return to [Summary Table](#).

Figure 27-27. USBRAMINFO Register

7	6	5	4	3	2	1	0
DMACHAN				RAMBITS			
R-0x8				R-0xA			

Table 27-32. USBRAMINFO Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	DMACHAN	R	0x8	DMA Channels. This field identifies the number of DMA channels implemented in the design.
3-0	RAMBITS	R	0xA	RAM Address Bus Width. This field displays the width of the RAM address bus.

27.5.23 USBCONTIM Register (Offset = 0x7A) [reset = 0x5C]

USB Connect Timing (USBCONTIM)

OTG A / Host

OTG B / Device

This 8-bit configuration register specifies connection and negotiation delays.

USBCONTIM is shown in [Figure 27-28](#) and described in [Table 27-33](#).

Return to [Summary Table](#).

Figure 27-28. USBCONTIM Register

7	6	5	4	3	2	1	0
WTCON				WTID			
R/W-0x5				R/W-0xC			

Table 27-33. USBCONTIM Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	WTCON	R/W	0x5	Connect Wait. This field configures the wait required to allow for the user's connect/disconnect filter, in units of 533.3 ns. The default corresponds to 2.667 us.
3-0	WTID	R/W	0xC	Wait ID. This field configures the delay required from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. The default corresponds to 52.43 ms.

27.5.24 USBVPLEN Register (Offset = 0x7B) [reset = 0x3C]

USB OTG VBUS Pulse Timing (USBVPLEN)

OTG

This 8-bit configuration register specifies the duration of the VBUS pulsing charge.

USBVPLEN is shown in [Figure 27-29](#) and described in [Table 27-34](#).

Return to [Summary Table](#).

Figure 27-29. USBVPLEN Register

7	6	5	4	3	2	1	0
VPLEN							
R/W-0x3C							

Table 27-34. USBVPLEN Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	VPLEN	R/W	0x3C	VBUS Pulse Length. This field configures the duration of the VBUS pulsing charge in units of 546.1 μ s. The default corresponds to 32.77 ms.

27.5.25 USBHSEOF Register (Offset = 0x7C) [reset = 0x80]

USB High-Speed Last Transaction to End of Frame Timing (USBHSEOF)

OTG A / Host

OTG B / Device

This 8-bit register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for High-speed transactions.

USBHSEOF is shown in [Figure 27-30](#) and described in [Table 27-35](#).

Return to [Summary Table](#).

Figure 27-30. USBHSEOF Register

7	6	5	4	3	2	1	0
HSEOFG							
R/W-0x80							

Table 27-35. USBHSEOF Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	HSEOFG	R/W	0x80	High-Speed End-of-Frame Gap. This field is used during high-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 133.3 ns. The default corresponds to 17.07 us.

27.5.26 USBFSEOF Register (Offset = 0x7D) [reset = 0x77]

USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

OTG A / Host

OTG B / Device

This 8-bit configuration register specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

USBFSEOF is shown in [Figure 27-31](#) and described in [Table 27-36](#).

Return to [Summary Table](#).

Figure 27-31. USBFSEOF Register

7	6	5	4	3	2	1	0
FSEOFG							
R/W-0x77							

Table 27-36. USBFSEOF Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	FSEOFG	R/W	0x77	Full-Speed End-of-Frame Gap. This field is used during full-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 us.

27.5.27 USBLSEOF Register (Offset = 0x7E) [reset = 0x72]

USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

OTG A / Host

OTG B / Device

This 8-bit configuration register specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

USBLSEOF is shown in [Figure 27-32](#) and described in [Table 27-37](#).

Return to [Summary Table](#).

Figure 27-32. USBLSEOF Register

7	6	5	4	3	2	1	0
LSEOFG							
R/W-0x72							

Table 27-37. USBLSEOF Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	LSEOFG	R/W	0x72	Low-Speed End-of-Frame Gap. This field is used during low-speed transactions to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 μ s. The default corresponds to 121.6 μ s.

27.5.28 USBTXFUNCADDRn Register [reset = 0x0]

USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080

USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088

USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090

USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098

USB Transmit Functional Address Endpoint 4 (USBTXFUNCADDR4), offset 0x0A0

USB Transmit Functional Address Endpoint 5 (USBTXFUNCADDR5), offset 0x0A8

USB Transmit Functional Address Endpoint 6 (USBTXFUNCADDR6), offset 0x0B0

USB Transmit Functional Address Endpoint 7 (USBTXFUNCADDR7), offset 0x0B8

OTG A / Host

USBTXFUNCADDRn is an 8-bit read/write register that records the address of the target function to be accessed through the associated endpoint (EPn). USBTXFUNCADDRn must be defined for each transmit endpoint that is used.

NOTE: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

USBTXFUNCADDRn is shown in [Figure 27-33](#) and described in [Table 27-38](#).

Return to [Summary Table](#).

Figure 27-33. USBTXFUNCADDRn Register

7	6	5	4	3	2	1	0
RESERVED	ADDR						
R-0x0	R/W-0x0						

Table 27-38. USBTXFUNCADDRn Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	ADDR	R/W	0x0	Device Address. Specifies the USB bus address for the target Device.

27.5.29 USBTXHUBADDRn Register [reset = 0x0]

USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082

USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A

USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092

USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A

USB Transmit Hub Address Endpoint 4 (USBTXHUBADDR4), offset 0x0A2

USB Transmit Hub Address Endpoint 5 (USBTXHUBADDR5), offset 0x0AA

USB Transmit Hub Address Endpoint 6 (USBTXHUBADDR6), offset 0x0B2

USB Transmit Hub Address Endpoint 7 (USBTXHUBADDR7), offset 0x0BA

OTG A / Host

USBTXHUBADDRn is an 8-bit read/write register that, like USBTXHUBPORTn, only must be written when a USB Device is connected to transmit endpoint EPn through a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

NOTE: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

USBTXHUBADDRn is shown in [Figure 27-34](#) and described in [Table 27-39](#).

Return to [Summary Table](#).

Figure 27-34. USBTXHUBADDRn Register

7	6	5	4	3	2	1	0
RESERVED	ADDR						
R-0x0	R/W-0x0						

Table 27-39. USBTXHUBADDRn Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	ADDR	R/W	0x0	Hub Address. This field specifies the USB bus address for the USB 2.0 hub.

27.5.30 USBTXHUBPORTn Register [reset = 0x0]

USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083

USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B

USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093

USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B

USB Transmit Hub Port Endpoint 4 (USBTXHUBPORT4), offset 0x0A3

USB Transmit Hub Port Endpoint 5 (USBTXHUBPORT5), offset 0x0AB

USB Transmit Hub Port Endpoint 6 (USBTXHUBPORT6), offset 0x0B3

USB Transmit Hub Port Endpoint 7 (USBTXHUBPORT7), offset 0x0BB

OTG A / Host

USBTXHUBPORTn is an 8-bit read/write register that, like USBTXHUBADDRn, only must be written when a full- or low-speed Device is connected to transmit endpoint EPn through a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

NOTE: USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

USBTXHUBPORTn is shown in [Figure 27-35](#) and described in [Table 27-40](#).

Return to [Summary Table](#).

Figure 27-35. USBTXHUBPORTn Register

7	6	5	4	3	2	1	0
RESERVED	PORT						
R-0x0	R/W-0x0						

Table 27-40. USBTXHUBPORTn Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	PORT	R/W	0x0	Hub Port. This field specifies the USB hub port number.

27.5.31 USBRXFUNCADDRn Register [reset = 0x0]

USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C

USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094

USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C

USB Receive Functional Address Endpoint 4 (USBRXFUNCADDR4), offset 0x0A4

USB Receive Functional Address Endpoint 5 (USBRXFUNCADDR5), offset 0x0AC

USB Receive Functional Address Endpoint 6 (USBRXFUNCADDR6), offset 0x0B4

USB Receive Functional Address Endpoint 7 (USBRXFUNCADDR7), offset 0x0BC

OTG A / Host

USBRXFUNCADDRn is an 8-bit read/write register that records the address of the target function accessed through the associated endpoint (EPn). USBRXFUNCADDRn must be defined for each receive endpoint that is used.

NOTE: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

USBRXFUNCADDRn is shown in [Figure 27-36](#) and described in [Table 27-41](#).

Return to [Summary Table](#).

Figure 27-36. USBRXFUNCADDRn Register

7	6	5	4	3	2	1	0
RESERVED	ADDR						
R-0x0	R/W-0x0						

Table 27-41. USBRXFUNCADDRn Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	ADDR	R/W	0x0	Device Address. This field specifies the USB bus address for the target Device.

27.5.32 USBRXHUBADDRn Register [reset = 0x0]

USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E

USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096

USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E

USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6

USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE

USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6

USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE

OTG A / Host

USBRXHUBADDRn is an 8-bit read/write register that, like USBRXHUBPORTn, only must be written when a full- or low-speed Device is connected to receive endpoint EPn through a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

NOTE: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

USBRXHUBADDRn is shown in [Figure 27-37](#) and described in [Table 27-42](#).

Return to [Summary Table](#).

Figure 27-37. USBRXHUBADDRn Register

7	6	5	4	3	2	1	0
RESERVED	ADDR						
R-0x0	R/W-0x0						

Table 27-42. USBRXHUBADDRn Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	ADDR	R/W	0x0	Hub Address. This field specifies the USB bus address for the USB 2.0 hub.

27.5.33 USBRXHUBPORTn Register [reset = 0x0]

USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E

USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096

USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E

USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6

USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE

USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6

USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE

OTG A / Host

USBRXHUBADDRn is an 8-bit read/write register that, like USBRXHUBPORTn, only must be written when a full- or low-speed Device is connected to receive endpoint EPn through a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

NOTE: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

USBRXHUBPORTn is shown in [Figure 27-38](#) and described in [Table 27-43](#).

Return to [Summary Table](#).

Figure 27-38. USBRXHUBPORTn Register

7	6	5	4	3	2	1	0
RESERVED	PORT						
R-0x0	R/W-0x0						

Table 27-43. USBRXHUBPORTn Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	PORT	R/W	0x0	Hub Port. This field specifies the USB hub port number.

27.5.34 USBCSRL0 Register (Offset = 0x102) [reset = 0x0]

USB Control and Status Endpoint 0 Low (USBCSRL0)

OTG A / Host

OTG B / Device

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

USBCSRL0 for OTG A / Host is shown in [Figure 27-39](#) and described in [Table 27-44](#).

USBCSRL0 for OTG B / Device is shown in [Figure 27-40](#) and described in [Table 27-45](#).

Return to [Summary Table](#).

Figure 27-39. USBCSRL0 Register (OTG A / Host)

7	6	5	4	3	2	1	0
NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-44. USBCSRL0 Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	NAKTO	R/W	0x0	NAK Time-out. Software must clear this bit to allow the endpoint to continue. 0x0 = No time-out. 0x1 = Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USBNAKLMT register.
6	STATUS	R/W	0x0	STATUS Packet. Setting this bit ensures that the DT bit is set in the USBCSRH0 register so that a DATA1 packet is used for the STATUS stage transaction. This bit is automatically cleared when the STATUS stage is over. 0x0 = No transaction. 0x1 = Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set.
5	REQPKT	R/W	0x0	Request Packet. This bit is cleared when the RXRDY bit is set. 0x0 = No request. 0x1 = Requests an IN transaction.
4	ERROR	R/W	0x0	Error. Software must clear this bit. 0x0 = No error. 0x1 = Three attempts have been made to perform a transaction with no response from the peripheral. The EP0 bit in the USBTXIS register is also set in this situation.
3	SETUP	R/W	0x0	Setup Packet. Setting this bit always clears the DT bit in the USBCSRH0 register to send a DATA0 packet. 0x0 = Sends an OUT token. 0x1 = Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.
2	STALLED	R/W	0x0	Endpoint Stalled. Software must clear this bit. 0x0 = No handshake has been received. 0x1 = A STALL handshake has been received.

Table 27-44. USBCSRL0 Register Field Descriptions (OTG A / Host) (continued)

Bit	Field	Type	Reset	Description
1	TXRDY	R/W	0x0	Transmit Packet Ready. This bit is cleared automatically when the data packet has been transmitted. 0x0 = No transmit packet is ready. 0x1 = Software sets this bit after loading a data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.
0	RXRDY	R/W	0x0	Receive Packet Ready. Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO. 0x0 = No received packet has been received. 0x1 = Indicates that a data packet has been received in the RX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.

Figure 27-40. USBCSRL0 Register (OTG B / Device)

7	6	5	4	3	2	1	0
SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
W1C-0x0	W1C-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 27-45. USBCSRL0 Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	SETENDC	W1C	0x0	Setup End Clear. Writing a 1 to this bit clears the SETEND bit.
6	RXRDYC	W1C	0x0	RXRDY Clear. Writing a 1 to this bit clears the RXRDY bit.
5	STALL	R/W	0x0	Send Stall. This bit is cleared automatically after the STALL handshake is transmitted. 0x0 = No effect. 0x1 = Terminates the current transaction and transmits the STALL handshake.
4	SETEND	R	0x0	Setup End. This bit is cleared by writing a 1 to the SETENDC bit. 0x0 = A control transaction has not ended or ended after the DATAEND bit was set. 0x1 = A control transaction has ended before the DATAEND bit has been set. The EP0 bit in the USBTXIS register is also set in this situation.
3	DATAEND	R/W	0x0	Data End. This bit is cleared automatically. 0x0 = No effect. 0x1 = Set this bit in the following situations: When setting TXRDY for the last data packet. When clearing RXRDY after unloading the last data packet. When setting TXRDY for a zero-length data packet.
2	STALLED	R/W	0x0	Endpoint Stalled. Software must clear this bit. 0x0 = A STALL handshake has not been transmitted. 0x1 = A STALL handshake has been transmitted.

Table 27-45. USBCSRL0 Register Field Descriptions (OTG B / Device) (continued)

Bit	Field	Type	Reset	Description
1	TXRDY	R/W	0x0	Transmit Packet Ready. This bit is cleared automatically when the data packet has been transmitted. 0x0 = No transmit packet is ready. 0x1 = Software sets this bit after loading an IN data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.
0	RXRDY	R	0x0	Receive Packet Ready. This bit is cleared by writing a 1 to the RXRDYC bit. 0x0 = No data packet has been received. 0x1 = A data packet has been received. The EP0 bit in the USBTXIS register is also set in this situation.

27.5.35 USBCSRH0 Register (Offset = 0x103) [reset = 0x0]

USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

OTG A / Host

OTG B / Device

USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

USBCSRH0 for OTG A / Host is shown in [Figure 27-41](#) and described in [Table 27-46](#).

USBCSRH0 for OTG B / Device is shown in [Figure 27-42](#) and described in [Table 27-47](#).

Return to [Summary Table](#).

Figure 27-41. USBCSRH0 Register (OTG A / Host)

7	6	5	4	3	2	1	0
RESERVED				DISPING	DTWE	DT	FLUSH
R-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-46. USBCSRH0 Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0x0	
3	DISPING	R/W	0x0	PING Disable. This bit is available for devices that do not respond to PING. 0x0 = PING token issues enabled. 0x1 = PING tokens are not issued in data and status phases of high-speed control transfer.
2	DTWE	R/W	0x0	Data Toggle Write Enable. This bit is automatically cleared once the new value is written. 0x0 = The DT bit cannot be written. 0x1 = Enables the current state of the endpoint 0 data toggle to be written (see DT bit).
1	DT	R/W	0x0	Data Toggle. When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0.
0	FLUSH	R/W	0x0	Flush FIFO. This bit is automatically cleared after the flush is performed. This bit should only be set when TXRDY is clear and RXRDY is set. At other times, it may cause data to be corrupted. 0x0 = No effect. 0x1 = Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY / RXRDY bit is cleared.

Figure 27-42. USBCSRH0 Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED							FLUSH
R-0x0							R/W-0x0

Table 27-47. USBCSRH0 Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7-1	RESERVED	R	0x0	
0	FLUSH	R/W	0x0	<p>Flush FIFO.</p> <p>This bit is automatically cleared after the flush is performed.</p> <p>This bit should only be set when TXRDY is clear and RXRDY is set. At other times, it may cause data to be corrupted.</p> <p>0x0 = No effect.</p> <p>0x1 = Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY / RXRDY bit is cleared.</p>

27.5.36 USBCOUNT0 Register (Offset = 0x108) [reset = 0x0]

USB Receive Byte Count Endpoint 0 (USBCOUNT0)

OTG A / Host

OTG B / Device

USBCOUNT0 is an 8-bit read-only register that indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the RXRDY bit is set.

USBCOUNT0 is shown in [Figure 27-43](#) and described in [Table 27-48](#).

Return to [Summary Table](#).

Figure 27-43. USBCOUNT0 Register

7	6	5	4	3	2	1	0
RESERVED	COUNT						
R-0x0	R-0x0						

Table 27-48. USBCOUNT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	COUNT	R	0x0	FIFO Count. COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.

27.5.37 USBTYPE0 Register (Offset = 0x10A) [reset = 0x0]

USB Type Endpoint 0 (USBTYPE0)

OTG A / Host

This is an 8-bit register that must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

USBTYPE0 is shown in [Figure 27-44](#) and described in [Table 27-49](#).

Return to [Summary Table](#).

Figure 27-44. USBTYPE0 Register

7	6	5	4	3	2	1	0
SPEED		RESERVED					
R/W-0x0		R-0x0					

Table 27-49. USBTYPE0 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-6	SPEED	R/W	0x0	Operating Speed. This field specifies the operating speed of the target Device. If selected, the target is assumed to have the same connection speed as the USB controller. 0x0 = Reserved 0x1 = High 0x2 = Full 0x3 = Low
5-0	RESERVED	R	0x0	

27.5.38 USBNAKLMT Register (Offset = 0x10B) [reset = 0x0]

USB NAK Limit (USBNAKLMT)

OTG A / Host

USBNAKLMT is an 8-bit register that sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their USBTXINTERVALn and USBRXINTERVALn registers.)

The number of frames selected is $2^{(m-1)}$ (where m is the value set in the register, with valid values of 2 to 16). If the Host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

NOTE: A value of 0 or 1 disables the NAK time-out function.

USBNAKLMT is shown in [Figure 27-45](#) and described in [Table 27-50](#).

Return to [Summary Table](#).

Figure 27-45. USBNAKLMT Register

7	6	5	4	3	2	1	0
RESERVED				NAKLMT			
R-0x0				R/W-0x0			

Table 27-50. USBNAKLMT Register Field Descriptions

Bit	Field	Type	Reset	Description
7-5	RESERVED	R	0x0	
4-0	NAKLMT	R/W	0x0	EP0 NAK Limit. This field specifies the number of frames after receiving a stream of NAK responses.

27.5.39 USBTXMAXPn Register [reset = 0x0]

USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110

USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120

USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130

USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140

USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150

USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160

USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170

OTG A / Host

OTG B / Device

The USBTXMAXPn 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO must be completely flushed (using the FLUSH bit in USBTXCSRLn) after writing the new value to this register.

NOTE: USBTXMAXPn must be set to an even number of bytes for proper interrupt generation in USB DMA Basic Mode.

USBTXMAXPn is shown in [Figure 27-46](#) and described in [Table 27-51](#).

Return to [Summary Table](#).

Figure 27-46. USBTXMAXPn Register

15	14	13	12	11	10	9	8
RESERVED						MAXLOAD	
R-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
MAXLOAD							
R/W-0x0							

Table 27-51. USBTXMAXPn Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0x0	
10-0	MAXLOAD	R/W	0x0	Maximum Payload. This field specifies the maximum payload in bytes per transaction.

27.5.40 USBTXCSRLn Register [reset = 0x0]

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112

USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122

USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132

USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142

USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152

USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162

USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172

OTG A / Host

OTG B / Device

USBTXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

USBTXCSRLn for OTG A / Host is shown in [Figure 27-47](#) and described in [Table 27-52](#).

USBTXCSRLn for OTG B / Device is shown in [Figure 27-48](#) and described in [Table 27-53](#).

Return to [Summary Table](#).

Figure 27-47. USBTXCSRLn Register (OTG A / Host)

7	6	5	4	3	2	1	0
NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-52. USBTXCSRLn Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	NAKTO	R/W	0x0	NAK Time-out. 0x0 = No time-out. 0x1 = Bulk endpoints only: Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBTXINTERVALn register. Software must clear this bit to allow the endpoint to continue.
6	CLRDT	R/W	0x0	Clear Data Toggle. Writing a 1 to this bit clears the DT bit in the USBTXCSRHn register.
5	STALLED	R/W	0x0	Endpoint Stalled. Software must clear this bit. 0x0 = A STALL handshake has not been received. 0x1 = Indicates that a STALL handshake has been received. When this bit is set, any USB DMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared.
4	SETUP	R/W	0x0	Setup Packet. Setting this bit also clears the DT bit in the USBTXCSRHn register. 0x0 = No SETUP token is sent. 0x1 = Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.

Table 27-52. USBTXCSRLn Register Field Descriptions (OTG A / Host) (continued)

Bit	Field	Type	Reset	Description
3	FLUSH	R/W	0x0	<p>Flush FIFO.</p> <p>This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO.</p> <p>Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.</p> <p>This bit should only be set when the TXRDY bit is clear.</p> <p>At other times, it may cause data to be corrupted.</p> <p>0x0 = No effect.</p> <p>0x1 = Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.</p>
2	ERROR	R/W	0x0	<p>Error.</p> <p>Software must clear this bit.</p> <p>This is valid only when the endpoint is operating in Bulk or Interrupt mode.</p> <p>0x0 = No error.</p> <p>0x1 = Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation.</p>
1	FIFONE	R/W	0x0	<p>FIFO Not Empty.</p> <p>0x0 = The FIFO is empty.</p> <p>0x1 = At least one packet is in the transmit FIFO.</p>
0	TXRDY	R/W	0x0	<p>Transmit Packet Ready.</p> <p>This bit is cleared automatically when a data packet has been transmitted.</p> <p>The EPn bit in the USBTXIS register is also set at this point.</p> <p>TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p> <p>0x0 = No transmit packet is ready.</p> <p>0x1 = Software sets this bit after loading a data packet into the TX FIFO.</p>

Figure 27-48. USBTXCSRLn Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-53. USBTXCSRLn Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6	CLRDT	R/W	0x0	<p>Clear Data Toggle.</p> <p>Writing a 1 to this bit clears the DT bit in the USBTXCSRn register.</p>
5	STALLED	R/W	0x0	<p>Endpoint Stalled.</p> <p>Software must clear this bit.</p> <p>0x0 = A STALL handshake has not been transmitted.</p> <p>0x1 = A STALL handshake has been transmitted. The FIFO is flushed and the TXRDY bit is cleared.</p>
4	STALL	R/W	0x0	<p>Send STALL.</p> <p>Software clears this bit to terminate the STALL condition.</p> <p>This bit has no effect in isochronous transfers.</p> <p>0x0 = No effect.</p> <p>0x1 = Issues a STALL handshake to an IN token.</p>

Table 27-53. USBTXCSRLn Register Field Descriptions (OTG B / Device) (continued)

Bit	Field	Type	Reset	Description
3	FLUSH	R/W	0x0	<p>Flush FIFO.</p> <p>This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO.</p> <p>Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.</p> <p>This bit should only be set when the TXRDY bit is clear.</p> <p>At other times, it may cause data to be corrupted.</p> <p>0x0 = No effect.</p> <p>0x1 = Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.</p>
2	UNDRN	R/W	0x0	<p>Underrun.</p> <p>Software must clear this bit.</p> <p>0x0 = No underrun.</p> <p>0x1 = An IN token has been received when TXRDY is not set.</p>
1	FIFONE	R/W	0x0	<p>FIFO Not Empty.</p> <p>0x0 = The FIFO is empty.</p> <p>0x1 = At least one packet is in the transmit FIFO.</p>
0	TXRDY	R/W	0x0	<p>Transmit Packet Ready.</p> <p>This bit is cleared automatically when a data packet has been transmitted.</p> <p>The EPn bit in the USBTXIS register is also set at this point.</p> <p>TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p> <p>0x0 = No transmit packet is ready.</p> <p>0x1 = Software sets this bit after loading a data packet into the TX FIFO.</p>

27.5.41 USBTXCSRHn Register [reset = 0x0]

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113

USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123

USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133

USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143

USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153

USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163

USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173

OTG A / Host

OTG B / Device

USBTXCSRHn is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

USBTXCSRHn for OTG A / Host is shown in [Figure 27-49](#) and described in [Table 27-54](#).

USBTXCSRHn for OTG B / Device is shown in [Figure 27-50](#) and described in [Table 27-55](#).

Return to [Summary Table](#).

Figure 27-49. USBTXCSRHn Register (OTG A / Host)

7	6	5	4	3	2	1	0
AUTOSET	RESERVED	MODE	DMAEN	FDT	DMAMOD	DTWE	DT
R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-54. USBTXCSRHn Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	AUTOSET	R/W	0x0	Auto Set. 0x0 = The TXRDY bit must be set manually. 0x1 = Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.
6	RESERVED	R	0x0	
5	MODE	R/W	0x0	Mode. This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions. 0x0 = Enables the endpoint direction as RX. 0x1 = Enables the endpoint direction as TX.
4	DMAEN	R/W	0x0	DMA Request Enable. 0x0 = Disables the DMA request for the transmit endpoint. 0x1 = Enables the DMA request for the transmit endpoint.
3	FDT	R/W	0x0	Force Data Toggle. 0x0 = No effect. 0x1 = Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD	R/W	0x0	DMA Request Mode. This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. 0x0 = An interrupt is generated after every DMA packet transfer. 0x1 = An interrupt is generated only after the entire DMA transfer is complete.

Table 27-54. USBTXCSRHn Register Field Descriptions (OTG A / Host) (continued)

Bit	Field	Type	Reset	Description
1	DTWE	R/W	0x0	Data Toggle Write Enable. This bit is automatically cleared once the new value is written. 0x0 = The DT bit cannot be written. 0x1 = Enables the current state of the transmit endpoint data to be written (see DT bit).
0	DT	R/W	0x0	Data Toggle. When read, this bit indicates the current state of the transmit endpoint data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint.

Figure 27-50. USBTXCSRHn Register (OTG B / Device)

7	6	5	4	3	2	1	0
AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	RESERVED	
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	

Table 27-55. USBTXCSRHn Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	AUTOSET	R/W	0x0	Auto Set. 0x0 = The TXRDY bit must be set manually. 0x1 = Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.
6	ISO	R/W	0x0	Isochronous Transfers. 0x0 = Enables the transmit endpoint for bulk or interrupt transfers. 0x1 = Enables the transmit endpoint for isochronous transfers.
5	MODE	R/W	0x0	Mode. This bit only has an effect where the same endpoint FIFO is used for both transmit and receive transactions. 0x0 = Enables the endpoint direction as RX. 0x1 = Enables the endpoint direction as TX.
4	DMAEN	R/W	0x0	DMA Request Enable. 0x0 = Disables the DMA request for the transmit endpoint. 0x1 = Enables the DMA request for the transmit endpoint.
3	FDT	R/W	0x0	Force Data Toggle. 0x0 = No effect. 0x1 = Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD	R/W	0x0	DMA Request Mode. This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. 0x0 = An interrupt is generated after every DMA packet transfer. 0x1 = An interrupt is generated only after the entire DMA transfer is complete.
1-0	RESERVED	R	0x0	

27.5.42 USBRXMAXPn Register [reset = 0x0]

USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114

USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124

USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134

USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144

USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154

USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164

USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174

OTG A / Host

OTG B / Device

The USBRXMAXPn is a 16-bit register which defines the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for bulk, interrupt and isochronous transfers in full-speed operations.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the receive endpoint, and must not exceed half the FIFO size if double-buffering is required.

NOTE: USBRXMAXPn must be set to an even number of bytes for proper interrupt generation in USB DMA Basic mode.

USBRXMAXPn is shown in [Figure 27-51](#) and described in [Table 27-56](#).

Return to [Summary Table](#).

Figure 27-51. USBRXMAXPn Register

15	14	13	12	11	10	9	8
RESERVED						MAXLOAD	
R-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
MAXLOAD							
R/W-0x0							

Table 27-56. USBRXMAXPn Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0x0	
10-0	MAXLOAD	R/W	0x0	Maximum Payload. The maximum payload in bytes per transaction.

27.5.43 USBRXCSRLn Register [reset = 0x0]

USB Receive Control and Status Endpoint 1 Low (USBXCSRL1), offset 0x116

USB Receive Control and Status Endpoint 2 Low (USBXCSRL2), offset 0x126

USB Receive Control and Status Endpoint 3 Low (USBXCSRL3), offset 0x136

USB Receive Control and Status Endpoint 4 Low (USBXCSRL4), offset 0x146

USB Receive Control and Status Endpoint 5 Low (USBXCSRL5), offset 0x156

USB Receive Control and Status Endpoint 6 Low (USBXCSRL6), offset 0x166

USB Receive Control and Status Endpoint 7 Low (USBXCSRL7), offset 0x176

OTG A / Host

OTG B / Device

USBXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

USBXCSRLn for OTG A / Host is shown in [Figure 27-52](#) and described in [Table 27-57](#).

USBXCSRLn for OTG B / Device is shown in [Figure 27-53](#) and described in [Table 27-58](#).

Return to [Summary Table](#).

Figure 27-52. USBXCSRLn Register (OTG A / Host)

7	6	5	4	3	2	1	0
CLRDT	STALLED	REQPKT	FLUSH	DATAERR/NAK TO	ERROR	FULL	RXRDY
W1C-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0

Table 27-57. USBXCSRLn Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	CLRDT	W1C	0x0	Clear Data Toggle. Writing a 1 to this bit clears the DT bit in the USBXCSRn register.
6	STALLED	R/W	0x0	Endpoint Stalled. Software must clear this bit. 0x0 = A STALL handshake has not been received. 0x1 = A STALL handshake has been received. The EPn bit in the USBXIS register is also set.
5	REQPKT	R/W	0x0	Request Packet. This bit is cleared when RXRDY is set. 0x0 = No request. 0x1 = Requests an IN transaction.
4	FLUSH	R/W	0x0	Flush FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted. 0x0 = No effect. 0x1 = Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.

Table 27-57. USBRXCSRLn Register Field Descriptions (OTG A / Host) (continued)

Bit	Field	Type	Reset	Description
3	DATAERR/NAKTO	R/W	0x0	Data Error / NAK Time-out. 0x0 = Normal operation. 0x1 = Isochronous endpoints only: Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. Bulk endpoints only: Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVALn register. Software must clear this bit to allow the endpoint to continue.
2	ERROR	R/W	0x0	Error. Software must clear this bit. This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In Isochronous mode, it always returns zero. 0x0 = No error. 0x1 = Three attempts have been made to receive a packet and no data packet has been received. The EPn bit in the USBRXIS register is set in this situation.
1	FULL	R	0x0	FIFO Full. 0x0 = The receive FIFO is not full. 0x1 = No more packets can be loaded into the receive FIFO.
0	RXRDY	R/W	0x0	Receive Packet Ready. If the AUTOCLR bit in the USBRXCSRHn register is set, then the this bit is automatically cleared when a packet of USBRXMAXPn bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO. 0x0 = No data packet has been received. 0x1 = A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.

Figure 27-53. USBRXCSRLn Register (OTG B / Device)

7	6	5	4	3	2	1	0
CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
W1C-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R-0x0	R/W-0x0

Table 27-58. USBRXCSRLn Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	CLRDT	W1C	0x0	Clear Data Toggle. Writing a 1 to this bit clears the DT bit in the USBRXCSRHn register.
6	STALLED	R/W	0x0	Endpoint Stalled. Software must clear this bit. 0x0 = A STALL handshake has not been transmitted. 0x1 = A STALL handshake has been transmitted.
5	STALL	R/W	0x0	Send STALL. Software must clear this bit to terminate the STALL condition. This bit has no effect where the endpoint is being used for isochronous transfers. 0x0 = No effect. 0x1 = Issues a STALL handshake.

Table 27-58. USBRXCSRLn Register Field Descriptions (OTG B / Device) (continued)

Bit	Field	Type	Reset	Description
4	FLUSH	R/W	0x0	<p>Flush FIFO.</p> <p>The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.</p> <p>Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.</p> <p>This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.</p> <p>0x0 = No effect.</p> <p>0x1 = Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.</p>
3	DATAERR	R	0x0	<p>Data Error.</p> <p>This bit is cleared when RXRDY is cleared.</p> <p>This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.</p> <p>0x0 = Normal operation.</p> <p>0x1 = Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error.</p>
2	OVER	R/W	0x0	<p>Overrun.</p> <p>Software must clear this bit.</p> <p>This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.</p> <p>0x0 = No overrun error.</p> <p>0x1 = Indicates that an OUT packet cannot be loaded into the receive FIFO.</p>
1	FULL	R	0x0	<p>FIFO Full.</p> <p>0x0 = The receive FIFO is not full.</p> <p>0x1 = No more packets can be loaded into the receive FIFO.</p>
0	RXRDY	R/W	0x0	<p>Receive Packet Ready.</p> <p>If the AUTOCLR bit in the USBRXCSRHn register is set, then the this bit is automatically cleared when a packet of USBRXMAXPn bytes has been unloaded from the receive FIFO.</p> <p>If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.</p> <p>0x0 = No data packet has been received.</p> <p>0x1 = A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.</p>

27.5.44 USBRXCSRHn Register [reset = 0x0]

USB Receive Control and Status Endpoint n High (USBRXCSRHn) USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117 USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127 USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137 USB Receive Control and Status Endpoint 4 High (USBRXCSRH4), offset 0x147 USB Receive Control and Status Endpoint 5 High (USBRXCSRH5), offset 0x157 USB Receive Control and Status Endpoint 6 High (USBRXCSRH6), offset 0x167 USB Receive Control and Status Endpoint 7 High (USBRXCSRH7), offset 0x177

OTG A / Host

OTG B / Device

USBRXCSRHn is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

USBRXCSRHn for OTG A / Host is shown in [Figure 27-54](#) and described in [Table 27-59](#).

USBRXCSRHn for OTG B / Device is shown in [Figure 27-55](#) and described in [Table 27-60](#).

Return to [Summary Table](#).

Figure 27-54. USBRXCSRHn Register (OTG A / Host)

7	6	5	4	3	2	1	0
AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	RESERVED
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R-0x0	R-0x0

Table 27-59. USBRXCSRHn Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7	AUTOCL	R/W	0x0	Auto Clear. 0x0 = No effect. 0x1 = Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXPn bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using USB DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXPn register, see .
6	AUTORQ	R/W	0x0	Auto Request. This bit is automatically cleared when a short packet is received. 0x0 = No effect. 0x1 = Enables the REQPKT bit to be automatically set when the RXRDY bit is cleared.
5	DMAEN	R/W	0x0	DMA Request Enable. 0x0 = Disables the USB DMA request for the receive endpoint. 0x1 = Enables the USB DMA request for the receive endpoint.
4	PIDERR	R/W	0x0	PID Error. This bit is ignored in bulk or interrupt transactions. 0x0 = No error. 0x1 = Indicates a PID error in the received packet of an isochronous transaction.
3	DMAMOD	R/W	0x0	DMA Request Mode. This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. 0x0 = An interrupt is generated after every USB DMA packet transfer. 0x1 = An interrupt is generated only after the entire USB DMA transfer is complete.

Table 27-59. USBRXCSRn Register Field Descriptions (OTG A / Host) (continued)

Bit	Field	Type	Reset	Description
2	DTWE	R	0x0	Data Toggle Write Enable. This bit is automatically cleared once the new value is written. 0x0 = The DT bit cannot be written. 0x1 = Enables the current state of the receive endpoint data to be written (see DT bit).
1	DT	R	0x0	Data Toggle. When read, this bit indicates the current state of the receive data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint.
0	RESERVED	R	0x0	

Figure 27-55. USBRXCSRn Register (OTG B / Device)

7	6	5	4	3	2	1	0
AUTOCL	ISO	DMAEN	DISNYET/PIDE RR	DMAMOD	RESERVED		INCOMPRX
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0		R/W0C-0x0

Table 27-60. USBRXCSRn Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7	AUTOCL	R/W	0x0	Auto Clear. 0x0 = No effect. 0x1 = Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXPn bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using USB DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXPn register, see.
6	ISO	R/W	0x0	Isochronous Transfers. 0x0 = Enables the receive endpoint for isochronous transfers. 0x1 = Enables the receive endpoint for bulk/interrupt transfers.
5	DMAEN	R/W	0x0	DMA Request Enable. 0x0 = Disables the USB DMA request for the receive endpoint. 0x1 = Enables the USB DMA request for the receive endpoint.
4	DISNYET/PIDERR	R/W	0x0	Disable NYET / PID Error. 0x0 = No effect. 0x1 = For bulk or interrupt transactions: Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full. For isochronous transactions: Indicates a PID error in the received packet.
3	DMAMOD	R/W	0x0	DMA Request Mode. This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. 0x0 = An interrupt is generated after every USB DMA packet transfer. 0x1 = An interrupt is generated only after the entire USB DMA transfer is complete.
2-1	RESERVED	R	0x0	

Table 27-60. USBRXCSRHn Register Field Descriptions (OTG B / Device) (continued)

Bit	Field	Type	Reset	Description
0	INCOMPRX	R/W0C	0x0	<p>Incomplete RX Transmission Status.</p> <p>In anything other than an Isochronous transfer, this bit returns a 0.</p> <p>0x0 = No effect.</p> <p>0x1 = Indicates that the packet in the RX FIFO is incomplete because parts of the data were not received in the high-bandwidth/Isochronous transfer</p>

27.5.45 USBRXCOUNTn Register [reset = 0x0]

USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118

USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128

USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138

USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148

USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158

USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168

USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178

OTG A / Host

OTG B / Device

NOTE: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the USBRXCSSLn register is set.

USBRXCOUNTn is a 16-bit read-only register that holds the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

USBRXCOUNTn is shown in [Figure 27-56](#) and described in [Table 27-61](#).

Return to [Summary Table](#).

Figure 27-56. USBRXCOUNTn Register

15	14	13	12	11	10	9	8
RESERVED				COUNT			
R-0x0				R-0x0			
7	6	5	4	3	2	1	0
COUNT							
R-0x0							

Table 27-61. USBRXCOUNTn Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0x0	
12-0	COUNT	R	0x0	Receive Packet Count. Indicates the number of bytes in the receive packet.

27.5.46 USBTXTYPE_n Register [reset = 0x0]

USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A

USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A

USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A

USB Host Transmit Configure Type Endpoint 4 (USBTXTYPE4), offset 0x14A

USB Host Transmit Configure Type Endpoint 5 (USBTXTYPE5), offset 0x15A

USB Host Transmit Configure Type Endpoint 6 (USBTXTYPE6), offset 0x16A

USB Host Transmit Configure Type Endpoint 7 (USBTXTYPE7), offset 0x17A

OTG A / Host

USBTXTYPE_n is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

USBTXTYPE_n is shown in [Figure 27-57](#) and described in [Table 27-62](#).

Return to [Summary Table](#).

Figure 27-57. USBTXTYPE_n Register

7	6	5	4	3	2	1	0
SPEED		PROTO		TEP			
R/W-0x0		R/W-0x0		R/W-0x0			

Table 27-62. USBTXTYPE_n Register Field Descriptions

Bit	Field	Type	Reset	Description
7-6	SPEED	R/W	0x0	Operating Speed. This bit field specifies the operating speed of the target Device. 0x0 = Default. The target is assumed to be using the same connection speed as the USB controller. 0x1 = High 0x2 = Full 0x3 = Low
5-4	PROTO	R/W	0x0	Protocol. Software must configure this bit field to select the required protocol for the transmit endpoint: 0x0 = Control 0x1 = Isochronous 0x2 = Bulk 0x3 = Interrupt
3-0	TEP	R/W	0x0	Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.

27.5.47 USBTXINTERVALn Register [reset = 0x0]

USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B

USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B

USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B

USB Host Transmit Interval Endpoint 4 (USBTXINTERVAL4), offset 0x14B

USB Host Transmit Interval Endpoint 5 (USBTXINTERVAL5), offset 0x15B

USB Host Transmit Interval Endpoint 6 (USBTXINTERVAL6), offset 0x16B

USB Host Transmit Interval Endpoint 7 (USBTXINTERVAL7), offset 0x17B

OTG A / Host

USBTXINTERVALn is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBTXINTERVALn register value defines a number of frames:

Table 27-63. USBTXINTERVALn Register Values

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low Speed or Full Speed	0x01 to 0xFF	The polling interval is m frames.
Interrupt	High Speed	0x01 to 0x10	Polling interval is $2^{(m-1)}$ microframes.
Isochronous	Full Speed or High Speed	0x01 to 0x10	The polling interval is $2^{(m-1)}$ frames/microframes.
Bulk	Full-Speed or High Speed	0x02 to 0x10	The NAK Limit is $2^{(m-1)}$ frames/microframes. A value of 0 or 1 disables the NAK time-out function.

USBTXINTERVALn is shown in [Figure 27-58](#) and described in [Table 27-64](#).

Return to [Summary Table](#).

Figure 27-58. USBTXINTERVALn Register

7	6	5	4	3	2	1	0
TXPOLL/NAKLMT							
R/W-0x0							

Table 27-64. USBTXINTERVALn Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	TXPOLL/NAKLMT	R/W	0x0	TX Polling / NAK Limit. The polling interval for interrupt/isochronous transfers or the NAK limit for bulk transfers. See Table 27-63 for valid entries; other values are reserved.

27.5.48 USBRXTYPE_n Register [reset = 0x0]

USB Host Configure Receive Type Endpoint 1 (USBRTYPE1), offset 0x11C

USB Host Configure Receive Type Endpoint 2 (USBRTYPE2), offset 0x12C

USB Host Configure Receive Type Endpoint 3 (USBRTYPE3), offset 0x13C

USB Host Configure Receive Type Endpoint 4 (USBRTYPE4), offset 0x14C

USB Host Configure Receive Type Endpoint 5 (USBRTYPE5), offset 0x15C

USB Host Configure Receive Type Endpoint 6 (USBRTYPE6), offset 0x16C

USB Host Configure Receive Type Endpoint 7 (USBRTYPE7), offset 0x17C

OTG A / Host

USBRTYPE_n is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

USBRTYPE_n is shown in [Figure 27-59](#) and described in [Table 27-65](#).

Return to [Summary Table](#).

Figure 27-59. USBRXTYPE_n Register

7	6	5	4	3	2	1	0
SPEED		PROTO		TEP			
R/W-0x0		R/W-0x0		R/W-0x0			

Table 27-65. USBRXTYPE_n Register Field Descriptions

Bit	Field	Type	Reset	Description
7-6	SPEED	R/W	0x0	Operating Speed. This bit field specifies the operating speed of the target Device: 0x0 = Default. The target is assumed to be using the same connection speed as the USB controller. 0x1 = High 0x2 = Full 0x3 = Low
5-4	PROTO	R/W	0x0	Protocol. Software must configure this bit field to select the required protocol for the receive endpoint: 0x0 = Control 0x1 = Isochronous 0x2 = Bulk 0x3 = Interrupt
3-0	TEP	R/W	0x0	Target Endpoint Number. Software must set this value to the endpoint number contained in the receive endpoint descriptor returned to the USB controller during Device enumeration.

27.5.49 USBRXINTERVALn Register [reset = 0x0]

USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D

USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D

USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D

USB Host Receive Polling Interval Endpoint 4 (USBRXINTERVAL4), offset 0x14D

USB Host Receive Polling Interval Endpoint 5 (USBRXINTERVAL5), offset 0x15D

USB Host Receive Polling Interval Endpoint 6 (USBRXINTERVAL6), offset 0x16D

USB Host Receive Polling Interval Endpoint 7 (USBRXINTERVAL7), offset 0x17D

OTG A / Host

USBRXINTERVALn is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected receive endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBRXINTERVALn register value defines a number of frames:

Table 27-66. USBRXINTERVALn Register Values

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low Speed or Full Speed	0x01 to 0xFF	The polling interval is m frames.
Interrupt	High Speed	0x01 to 0x10	Polling interval is $2^{(m-1)}$ microframes.
Isochronous	Full Speed or High Speed	0x01 to 0x10	The polling interval is $2^{(m-1)}$ frames/microframes.
Bulk	Full-Speed or High Speed	0x02 to 0x10	The NAK Limit is $2^{(m-1)}$ frames/microframes. A value of 0 or 1 disables the NAK time-out function.

USBRXINTERVALn is shown in [Figure 27-60](#) and described in [Table 27-67](#).

Return to [Summary Table](#).

Figure 27-60. USBRXINTERVALn Register

7	6	5	4	3	2	1	0
TXPOLL/NAKLMT							
R/W-0x0							

Table 27-67. USBRXINTERVALn Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	TXPOLL/NAKLMT	R/W	0x0	RX Polling / NAK Limit. The polling interval for interrupt/isochronous transfers or the NAK limit for bulk transfers. See Table 27-66 for valid entries; other values are Reserved.

27.5.50 USBDMAINTR Register (Offset = 0x200) [reset = 0x0]

USB DMA Interrupt (USBDMAINTR)

OTG A / Host

OTG B / Device

This register provides an interrupt for each USB DMA channel. This interrupt register is cleared when read. When any bit of this register is set, an interrupt is sent to the interrupt controller. Bits in this register will only be set if the corresponding enable bit in the DMACTLn register is set.

USBDMAINTR is shown in [Figure 27-61](#) and described in [Table 27-68](#).

Return to [Summary Table](#).

Figure 27-61. USBDMAINTR Register

7	6	5	4	3	2	1	0
CH7DMAINT	CH6DMAINT	CH5DMAINT	CH4DMAINT	CH3DMAINT	CH2DMAINT	CH1DMAINT	CH0DMAINT
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 27-68. USBDMAINTR Register Field Descriptions

Bit	Field	Type	Reset	Description
7	CH7DMAINT	R/W	0x0	Channel 7 DMA Interrupt. 0x0 = No effect 0x1 = Channel 7 DMA interrupt enabled.
6	CH6DMAINT	R/W	0x0	Channel 6 DMA Interrupt. 0x0 = No effect 0x1 = Channel 6 DMA interrupt enabled.
5	CH5DMAINT	R/W	0x0	Channel 5 DMA Interrupt. 0x0 = No effect 0x1 = Channel 5 DMA interrupt enabled.
4	CH4DMAINT	R/W	0x0	Channel 4 DMA Interrupt. 0x0 = No effect 0x1 = Channel 4 DMA interrupt enabled.
3	CH3DMAINT	R/W	0x0	Channel 3 DMA Interrupt. 0x0 = No effect 0x1 = Channel 3 DMA interrupt enabled.
2	CH2DMAINT	R/W	0x0	Channel 2 DMA Interrupt. 0x0 = No effect 0x1 = Channel 2 DMA interrupt enabled.
1	CH1DMAINT	R/W	0x0	Channel 1 DMA Interrupt. 0x0 = No effect 0x1 = Channel 1 DMA interrupt enabled.
0	CH0DMAINT	R	0x0	Channel 0 DMA Interrupt. 0x0 = No effect 0x1 = Channel 0 DMA interrupt enabled.

27.5.51 USBDMACLTn Register [reset = 0x0]

USB DMA Control 0 (USBDMACLT0), offset 0x204

USB DMA Control 1 (USBDMACLT1), offset 0x214

USB DMA Control 2 (USBDMACLT2), offset 0x224

USB DMA Control 3 (USBDMACLT3), offset 0x234

USB DMA Control 4 (USBDMACLT4), offset 0x244

USB DMA Control 5 (USBDMACLT5), offset 0x254

USB DMA Control 6 (USBDMACLT6), offset 0x264

USB DMA Control 7 (USBDMACLT7), offset 0x274

OTG A / Host

OTG B / Device

This register provides the DMA transfer control for each channel. The enabling, transfer direction, transfer mode, the DMA burst modes are all controlled by this register.

USBDMACLTn is shown in [Figure 27-62](#) and described in [Table 27-69](#).

Return to [Summary Table](#).

Figure 27-62. USBDMACLTn Register

15	14	13	12	11	10	9	8
RESERVED					BRSTM		ERR
R-0x0					R-0x0		R/W-0x0
7	6	5	4	3	2	1	0
EP				IE	MODE	DIR	ENABLE
R/W-0x0				R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-69. USBDMACLTn Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0x0	
10-9	BRSTM	R	0x0	Burst Mode. 0x0 = Bursts of unspecified length 0x1 = INCR4 or unspecified length 0x2 = INCR8, INCR4 or unspecified length 0x3 = INCR16, INCR8, INCR4 or unspecified length
8	ERR	R/W	0x0	Bus Error Bit. This bit is cleared by software. 0x0 = No effect 0x1 = A bus error has occurred.
7-4	EP	R/W	0x0	Endpoint number. This field indicates the endpoint number to which this channel is assigned. This value can be 0x1 to 0x7.
3	IE	R/W	0x0	DMA Interrupt Enable. 0x0 = No effect 0x1 = Interrupt is enabled for this channel
2	MODE	R/W	0x0	DMA Transfer Mode. 0x0 = DMA Mode0 Transfer 0x1 = DMA Mode1 Transfer
1	DIR	R/W	0x0	DMA Direction. 0x0 = DMA Write (RX endpoint) 0x1 = DMA Read (TX Endpoint)

Table 27-69. USBDMACLTn Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	ENABLE	R/W	0x0	DMA Transfer Enable. 0x0 = No effect. 0x1 = DMA transfer is initiated.

27.5.52 USBDMAADDRn Register [reset = 0x0]

USB DMA Address 0 (USBDMAADDR0), offset 0x208

USB DMA Address 1 (USBDMAADDR1), offset 0x218

USB DMA Address 2 (USBDMAADDR2), offset 0x228

USB DMA Address 3 (USBDMAADDR3), offset 0x238

USB DMA Address 4 (USBDMAADDR4), offset 0x248

USB DMA Address 5 (USBDMAADDR5), offset 0x258

USB DMA Address 6 (USBDMAADDR6), offset 0x268

USB DMA Address 7 (USBDMAADDR7), offset 0x278

OTG A / Host

OTG B / Device

This register provides the DMA transfer control for each channel. The enabling, transfer direction, transfer mode, the DMA burst modes are all controlled by this register.

USBDMAADDRn is shown in [Figure 27-63](#) and described in [Table 27-70](#).

Return to [Summary Table](#).

Figure 27-63. USBDMAADDRn Register

31	30	29	28	27	26	25	24
DMAADDR							
R/W-0x0							
23	22	21	20	19	18	17	16
DMAADDR							
R/W-0x0							
15	14	13	12	11	10	9	8
DMAADDR							
R/W-0x0							
7	6	5	4	3	2	1	0
DMAADDR						RESERVED	
R/W-0x0						R-0x0	

Table 27-70. USBDMAADDRn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	DMAADDR	R/W	0x0	DMA Address. This field contains the DMA memory address. The memory address written to this register must have a modulo 4 value equal to 0. The lower two bits of this register are read-only and cannot be set by software.
1-0	RESERVED	R	0x0	

27.5.53 USBDMACOUNTn Register [reset = 0x0]

USB DMA Count 0 (USBDMACOUNT0), offset 0x20C

USB DMA Count 1 (USBDMACOUNT1), offset 0x21C

USB DMA Count 2 (USBDMACOUNT2), offset 0x22C

USB DMA Count 3 (USBDMACOUNT3), offset 0x23C

USB DMA Count 4 (USBDMACOUNT4), offset 0x24C

USB DMA Count 5 (USBDMACOUNT5), offset 0x25C

USB DMA Count 6 (USBDMACOUNT6), offset 0x26C

USB DMA Count 7 (USBDMACOUNT7), offset 0x27C

OTG A / Host

OTG B / Device

This register identifies the current DMA count of the transfer. Software will set the initial count of the transfer which identifies the entire transfer length. As the count progresses this count is decremented as bytes are transferred.

USBDMACOUNTn is shown in [Figure 27-64](#) and described in [Table 27-71](#).

Return to [Summary Table](#).

Figure 27-64. USBDMACOUNTn Register

31	30	29	28	27	26	25	24
DMACOUNT							
R/W-0x0							
23	22	21	20	19	18	17	16
DMACOUNT							
R/W-0x0							
15	14	13	12	11	10	9	8
DMACOUNT							
R/W-0x0							
7	6	5	4	3	2	1	0
DMACOUNT						RESERVED	
R/W-0x0						R-0x0	

Table 27-71. USBDMACOUNTn Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	DMACOUNT	R/W	0x0	DMA Count. This field contains the count of the DMA transfer. If the DMA is enabled with a DMACOUNT = 0x0, the bus is not requested and a DMA interrupt is generated.
1-0	RESERVED	R	0x0	

27.5.54 USBRQPKTCOUNTn Register [reset = 0x0]

USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304

USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308

USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C

USB Request Packet Count in Block Transfer Endpoint 4 (USBRQPKTCOUNT4), offset 0x310

USB Request Packet Count in Block Transfer Endpoint 5 (USBRQPKTCOUNT5), offset 0x314

USB Request Packet Count in Block Transfer Endpoint 6 (USBRQPKTCOUNT6), offset 0x318

USB Request Packet Count in Block Transfer Endpoint 7 (USBRQPKTCOUNT7), offset 0x31C

OTG A / Host

This 16-bit read/write register is used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint n. The USB controller uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the USBRXCSRn register has been set. See [Section 27.3.2.2](#).

NOTE: Multiple packets combined into a single bulk packet within the FIFO count as one packet.

USBRQPKTCOUNTn is shown in [Figure 27-65](#) and described in [Table 27-72](#).

Return to [Summary Table](#).

Figure 27-65. USBRQPKTCOUNTn Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT															
R/W-0x0															

Table 27-72. USBRQPKTCOUNTn Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	COUNT	R/W	0x0	Block Transfer Packet Count. Sets the number of packets of the size defined by the MAXLOAD bit field that are to be transferred in a block transfer. This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set.

27.5.55 USBRXDPKTBUFFDIS Register (Offset = 0x340) [reset = 0x0]

USB Receive Double Packet Buffer Disable (USBRXDPKTBUFFDIS)

OTG A / Host

OTG B / Device

USBRXDPKTBUFFDIS is a 16-bit register that indicates which of the receive endpoints have disabled the double-packet buffer functionality (see [Section 27.3.1.3.2](#)).

USBRXDPKTBUFFDIS is shown in [Figure 27-66](#) and described in [Table 27-73](#).

Return to [Summary Table](#).

Figure 27-66. USBRXDPKTBUFFDIS Register

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	RESERVED
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 27-73. USBRXDPKTBUFFDIS Register Field Descriptions

Bit	Field	Type	Reset	Description
7	EP7	R/W	0x0	EP7 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
6	EP6	R/W	0x0	EP6 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
5	EP5	R/W	0x0	EP5 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
4	EP4	R/W	0x0	EP4 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
3	EP3	R/W	0x0	EP3 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
2	EP2	R/W	0x0	EP2 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
1	EP1	R/W	0x0	EP1 RX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
0	RESERVED	R	0x0	

27.5.56 USBTXDPKTBUFDIS Register (Offset = 0x342) [reset = 0x0]

USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)

OTG A / Host

OTG B / Device

USBTXDPKTBUFDIS is a 16-bit register that indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see [Section 27.3.1.2.2](#)).

USBTXDPKTBUFDIS is shown in [Figure 27-67](#) and described in [Table 27-74](#).

Return to [Summary Table](#).

Figure 27-67. USBTXDPKTBUFDIS Register

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	RESERVED
R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0

Table 27-74. USBTXDPKTBUFDIS Register Field Descriptions

Bit	Field	Type	Reset	Description
7	EP7	R/W	0x0	EP7 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
6	EP6	R/W	0x0	EP6 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
5	EP5	R/W	0x0	EP5 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
4	EP4	R/W	0x0	EP4 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
3	EP3	R/W	0x0	EP3 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
2	EP2	R/W	0x0	EP2 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
1	EP1	R/W	0x0	EP1 TX Double-Packet Buffer Disable. 0x0 = Disables double-packet buffering. 0x1 = Enables double-packet buffering.
0	RESERVED	R	0x0	

27.5.57 USBCTO Register (Offset = 0x344) [reset = 0x0]

USB Chirp Time-out (USBCTO)

OTG A / Host

OTG B / Device

This register sets the chirp time-out. This number, when multiplied by four, represents the number of SYSCLK cycles before the time-out occurs. That is, if SYSCLK is 30MHz, this number represents the number of 133 ns time intervals before the time-out occurs. If SYSCLK is 60MHz, this number represents the number of 67ns time intervals before the time-out occurs. Although this bit is written by the host in the CLK domain, the counter that utilizes this value is in the SYSCLK domain. No time domain crossing is provided as the value in this register is a static.

USBCTO is shown in [Figure 27-68](#) and described in [Table 27-75](#).

Return to [Summary Table](#).

Figure 27-68. USBCTO Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCTV															
R/W-0x0															

Table 27-75. USBCTO Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	CCTV	R/W	0x0	Configurable Chirp Time-out Value. This field, when multiplied by four, represents the number of XCLK cycles before a time-out occurs.

27.5.58 USBHHSRTN Register (Offset = 0x346) [reset = 0x0]

USB High Speed to UTM Operating Delay (USBHHSRTN)

OTG A / Host

OTG B / Device

This register sets the delay from the end of High Speed resume signaling (acting as a Host) to enable the UTM normal operating mode. This number when multiplied by 4 represents the number of SYSCLK cycles before the time-out occurs. That is, if SYSCLK is 30MHz, this number represents the number of 33.3 ns time intervals before the time-out occurs. If SYSCLK is 60MHz, this number represents the number of 16.7ns time intervals before the time-out occurs. Although this bit is written by the host in the CLK domain, the counter that utilizes this value is in the SYSCLK domain. No time domain crossing is provided as the value in this register is a static.

USBHHSRTN is shown in [Figure 27-69](#) and described in [Table 27-76](#).

Return to [Summary Table](#).

Figure 27-69. USBHHSRTN Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HHSRTN															
R/W-0x0															

Table 27-76. USBHHSRTN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	HHSRTN	R/W	0x0	High Speed to UTM Operating Delay. This field contains the delay from the end of High Speed resume signaling to enabling UTM normal operating mode.

27.5.59 USBHSBT Register (Offset = 0x348) [reset = 0x0]

USB High Speed Time-out Adder (USBHSBT)

OTG A / Host

OTG B / Device

A high-speed host or device expecting a response to a transmission must not time-out the transaction if the interpacket delay is less than 736 bit times, and it must time-out the transaction if no signaling is seen within 816 bit times. This register represents the value to be added to the minimum high speed time-out period of 736 bit times. The time-out period can be increased in increments of 64 high speed bit times (133 ns). There are 16 possible values. By default, the adder is 0 thus setting the high speed time-out to its minimum value. Use of this register will allow the high speed time-out to be set to values that are greater than the maximum specified in USB 2.0 making the USB non-compliant.

USBHSBT is shown in [Figure 27-70](#) and described in [Table 27-77](#).

Return to [Summary Table](#).

Figure 27-70. USBHSBT Register

15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED				HSBT			
R-0x0				R/W-0x0			

Table 27-77. USBHSBT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0x0	
3-0	HSBT	R/W	0x0	High Speed Time-out Adder. This field contains the value added to the minimum high speed timer period (736 bit timers) in increments of 64 high speed bit times. This allows the turn around time-out period to be set to 16 possible values as follows:

27.5.60 USBLPMATTR Register (Offset = 0x360) [reset = 0x0]

USB LPM Attributes (USBLPMATTR)

OTG A / Host

OTG B / Device

This register is used to define the attributes of an LPM transaction and sleep cycle. In both the Host mode and the Peripheral mode, the meaning of this register is the same however the source of the data is different for Host and Peripheral as follows:

- **Peripheral Mode:** In Peripheral mode, the values in this register will contain the equivalent attributes that were received in the last LPM transaction that was accepted. This register is updated with the LPM packet contents if the response to the LPM transaction was an ACK. This register can be update through software. In all other cases, this register will hold its current value.
- **Host Mode:** In HOST mode software will set-up the values in this register to define the next LPM transaction that will be transmitted. These values will be inserted in the payload of the next LPM Transaction.

USBLPMATTR is shown in [Figure 27-71](#) and described in [Table 27-78](#).

Return to [Summary Table](#).

Figure 27-71. USBLPMATTR Register

15	14	13	12	11	10	9	8
ENDPT				RESERVED			RMTWAK
R-0x0				R-0x0			R-0x0
7	6	5	4	3	2	1	0
HIRD				LINKSTATE			
R-0x0				R/W-0x0			

Table 27-78. USBLPMATTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	ENDPT	R	0x0	Endpoint. This is the Endpoint in the Token Packet of the LPM transaction.
11-9	RESERVED	R	0x0	
8	RMTWAK	R	0x0	Remote Wake. This bit is applied on a temporary basis only and only to the current suspend state. After the current suspend cycle, the remote wakeup capability that was negotiated upon enumeration applies. 0x0 = Remote wakeup is not enabled. 0x1 = Remote wakeup is enabled.
7-4	HIRD	R	0x0	Host Initiated Resume Duration. This value is the minimum time the host drives resume on the Bus. The value in this register corresponds to an actual resume time of: Resume Time = 50 μ s + HIRD * 75 μ s This results in a range from 50 μ s to 1200 us.
3-0	LINKSTATE	R/W	0x0	Link State. This value is provided by the host to the peripheral to indicate what state the peripheral must transition to after the receipt and acceptance of a LPM transaction. 0x0 = RESERVED 0x1 = Sleep State (L1)

27.5.61 USBLPMCNTRL Register (Offset = 0x362) [reset = 0x0]

USB LPM Control (USBLPMCNTRL)

OTG A / Host

OTG B / Device

This register is used to control LPM transactions.

USBLPMCNTRL for OTG A / Host is shown in [Figure 27-72](#) and described in [Table 27-79](#).

USBLPMCNTRL for OTG B / Device is shown in [Figure 27-73](#) and described in [Table 27-80](#).

Return to [Summary Table](#).

Figure 27-72. USBLPMCNTRL Register (OTG A / Host)

7	6	5	4	3	2	1	0
RESERVED						RES	TXLPM
R-0x0						R/W-0x0	R/W-0x0

Table 27-79. USBLPMCNTRL Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7-2	RESERVED	R	0x0	
1	RES	R/W	0x0	LPM Resume. This bit is used by software to initiate a RESUME from the L1 State. This bit differs from the classic RESUME bit in the USBPOWER register (address 0x001) in that the RESUME signal timing is controlled by hardware. 0x0 = No effect 0x1 = Resume signaling is asserted for a time specified by the HIRD field in the LPMATTR register. This bit is self-clearing.
0	TXLPM	R/W	0x0	Transmit LPM Transaction Enable. Software should set this bit to transmit an LPM transaction. This bit is self clearing and is immediately cleared upon receipt of any Token or once three timeouts have occurred.

Figure 27-73. USBLPMCNTRL Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED			NAK	LPMEN		RES	TXLPM
R-0x0			R/W-0x0	R/W-0x0		R/W-0x0	R/W-0x0

Table 27-80. USBLPMCNTRL Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7-5	RESERVED	R	0x0	
4	NAK	R/W	0x0	LPM NAK. This bit only takes effect after the USB has been LPM suspended. In this case, the USB continues to NAK until this bit has been cleared by software. 0x0 = No effect 0x1 = Place all endpoints in a state such that all transactions other than an LPM transaction is a NAK.

Table 27-80. USBLPMCNTRL Register Field Descriptions (OTG B / Device) (continued)

Bit	Field	Type	Reset	Description
3-2	LPMEN	R/W	0x0	<p>LPM Enable.</p> <p>This register is used to enable LPM in the USB. There are three LPM levels which determine the response of the USB to LPM transactions.</p> <p>The three levels are:</p> <p>0x0 = LPM and Extended transactions are not supported. In this case, the USB does not respond to LPM transactions and LPM transactions cause a time-out.</p> <p>0x1 = LPM is not supported but extended transactions are supported. In this case, the USB does respond to an LPM transaction with a STALL.</p> <p>0x2 = LPM and Extended transactions are not supported. In this case, the USB does not respond to LPM transactions and LPM transactions cause a time-out.</p> <p>0x3 = The USB supports LPM extended transactions. In this case, the USB responds with a NYET or an ACK as determined by the value of TXLPM and other conditions.</p>
1	RES	R/W	0x0	<p>LPM Resume.</p> <p>This bit is used by software to initiate resume (remote wakeup). This bit differs from the classic RESUME bit in the USBPOWER register (address 0x001), in that the RESUME signal timing is controlled by hardware.</p> <p>0x0 = No effect</p> <p>0x1 = Resume signaling is asserted for 50 μs. This bit is self clearing.</p>
0	TXLPM	R/W	0x0	<p>Transmit LPM Transaction Enable.</p> <p>This bit is only effective if LPMEN is set to 0x3.</p> <p>This bit can be set in the same cycle as LPMEN.</p> <p>If this bit is set to 0x1 and LPMEN = 0x3, the USB can respond in the following ways: If no data is pending (all TX FIFOs are empty), the USB will respond with an ACK.</p> <p>In this case, this bit self clears and a software interrupt is generated.</p> <p>If data is pending (Data resides in at least one TX FIFO), the USB responds with a NYET.</p> <p>In this case, this bit does NOT self clear, however a software interrupt is generated.</p> <p>0x0 = No effect</p> <p>0x1 = USB transitions to the L1 state upon the receipt of the next LPM transaction.</p>

27.5.62 USBLPMIM Register (Offset = 0x363) [reset = 0x0]

USB LPM Interrupt Mask (USBLPMIM)

USBLPMIM provides enable bits for the interrupts in the USBLPMRIS register.

USBLPMIM is shown in [Figure 27-74](#) and described in [Table 27-81](#).

Return to [Summary Table](#).

Figure 27-74. USBLPMIM Register

7	6	5	4	3	2	1	0
RESERVED	ERR	RES	NC	ACK	NY	STALL	
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-81. USBLPMIM Register Field Descriptions

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0x0	
5	ERR	R/W	0x0	LPM Error Interrupt Mask. 0x0 = The ERR bit in the USBLPMRIS registers is masked and does not cause an interrupt 0x1 = The ERR bit in the USBLPMRIS register is not masked and can trigger an interrupt to the interrupt controller.
4	RES	R/W	0x0	LPM Resume Interrupt Mask. 0x0 = The RES bit in the USBLPMRIS registers is masked and does not cause an interrupt 0x1 = The RES bit in the USBLPMRIS register is not masked and can trigger an interrupt to the interrupt controller.
3	NC	R/W	0x0	LPM NC Interrupt Mask. 0x0 = The NC bit in the USBLPMRIS registers is masked and does not cause an interrupt 0x1 = The NC bit in the USBLPMRIS register is not masked and can trigger an interrupt to the interrupt controller.
2	ACK	R/W	0x0	LPM ACK Interrupt Mask. 0x0 = The ACK bit in the USBLPMRIS registers is masked and does not cause an interrupt 0x1 = The ACK bit in the USBLPMRIS register is not masked and can trigger an interrupt to the interrupt controller.
1	NY	R/W	0x0	LPM NY Interrupt Mask. 0x0 = The NY bit in the USBLPMRIS registers is masked and does not cause an interrupt 0x1 = The NY bit in the USBLPMRIS register is not masked and can trigger an interrupt to the interrupt controller.
0	STALL	R/W	0x0	LPM STALL Interrupt Mask. 0x0 = The STALL bit in the USBLPMRIS registers is masked and does not cause an interrupt 0x1 = The STALL bit in the USBLPMRIS register is not masked and can trigger an interrupt to the interrupt controller.

27.5.63 USBLPMRIS Register (Offset = 0x364) [reset = 0x0]

USB LPM Raw Interrupt Status (USBLPMRIS)

OTG A / Host

OTG B / Device

The USBLPMRIS is a 7 bit register that provides status of the LPM Power state. On reset, all bits in this register are reset to 0. This register is self-clearing on read.

USBLPMRIS for OTG A / Host is shown in [Figure 27-75](#) and described in [Table 27-82](#).

USBLPMRIS for OTG B / Device is shown in [Figure 27-76](#) and described in [Table 27-83](#).

Return to [Summary Table](#).

Figure 27-75. USBLPMRIS Register (OTG A / Host)

7	6	5	4	3	2	1	0
RESERVED	ERR	RES	NC	ACK	NY	LPMST	
R-0x0	R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-82. USBLPMRIS Register Field Descriptions (OTG A / Host)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0x0	
5	ERR	R	0x0	LPM Error Interrupt Status. 0x0 = No status interrupt. 0x1 = An LPM transaction is received with a Bit Stuff error or a PID error. In this case, no suspend occurs and the state of the device is now unknown.
4	RES	R/W	0x0	LPM Resume Interrupt Status. This bit is mutually exclusive from the RESUME bit in the USBPOWER register (0x001). 0x0 = No status interrupt. 0x1 = The USB has been resumed.
3	NC	R/W	0x0	LPM No Completion Interrupt Status. 0x0 = No status interrupt. 0x1 = An LPM transaction has been transmitted and has failed to complete. The transaction has failed because a time-out occurred or there were bit errors in the response for three attempts.
2	ACK	R/W	0x0	LPM ACK Interrupt Status. 0x0 = No status interrupt. 0x1 = An LPM transaction is transmitted and the device responds with an ACK.
1	NY	R/W	0x0	LPM NY Interrupt Status. 0x0 = No status interrupt. 0x1 = An LPM transaction is transmitted and the device responds with a NYET.
0	LPMST	R/W	0x0	LPM STALL Interrupt Status. This condition can only occur if the LPMEN field in the LPMCNTL field is set to 0x1. 0x0 = No status interrupt. 0x1 = An LPM transaction is transmitted and the device has responded with a STALL.

Figure 27-76. USBLPMRIS Register (OTG B / Device)

7	6	5	4	3	2	1	0
RESERVED		ERR	RES	NC	ACK	NY	LPMST
R-0x0		R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R/W-0x0

Table 27-83. USBLPMRIS Register Field Descriptions (OTG B / Device)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0x0	
5	ERR	R	0x0	LPM Interrupt Status. 0x0 = No status interrupt. 0x1 = An LPM transaction is received that has a Link State field that is not supported and an interrupt has triggered and is pending. In this case the response to the transaction is to STALL. However, the LPMATTR register is updated so that software can observe the non-compliant LPM packet payload.
4	RES	R/W	0x0	LPM Resume Interrupt Status. This bit is mutually exclusive from the RESUME bit in the USBPOWER register (0x001). 0x0 = No status interrupt 0x1 = The USB has been resumed.
3	NC	R/W	0x0	LPM NC Interrupt Status. This condition can only occur when the LPMEN field is set to 0x3 and the TXLPM field is set to 0x1 in the LPMCNTL register and there is no data pending in the USB TX FIFOs. 0x0 = No status interrupt. 0x1 = An LPM transaction has been received and the USB has responded with a NYET due to a data pending in the RX FIFOs.
2	ACK	R/W	0x0	LPM ACK Interrupt Status. This condition can only occur when the LPMEN field is set to 0x3 and the TXLPM field is set to 0x1 in the LPMCNTL register and there is no data pending in the USB TX FIFOs. 0x0 = No status interrupt. 0x1 = An LPM transaction is received and the USB responds with an ACK.
1	NY	R/W	0x0	LPM NY Interrupt Status. This condition can only occur when the LPMEN field is set to 0x3 and the TXLPM field is 0x0 in the LPMCNTL register. 0x0 = No status interrupt. 0x1 = An LPM transaction is received and the USB has responded with a NYET.
0	LPMST	R/W	0x0	LPM STALL Interrupt Status. This condition can only occur if the LPMEN field in the LPMCNTL field is set to 0x1. 0x0 = No status interrupt. 0x1 = An LPM transaction is received and the USB has responded with a STALL.

27.5.64 USBLPMFADDR Register (Offset = 0x365) [reset = 0x0]

USB LPM Function Address (USBLPMFADDR)

OTG A / Host

The USBLPMFADDR register is the function address that is placed in the LPM payload.

USBLPMFADDR is shown in [Figure 27-77](#) and described in [Table 27-84](#).

Return to [Summary Table](#).

Figure 27-77. USBLPMFADDR Register

7	6	5	4	3	2	1	0
RESERVED	LPMFADDR						
R-0x0	R/W-0x0						

Table 27-84. USBLPMFADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	
6-0	LPMFADDR	R/W	0x0	LPM Function Address. This field contains the function address placed in the LPM payload.

27.5.65 USBEPC Register (Offset = 0x400) [reset = 0x0]

USB External Power Control (USBEPEN)

OTG A / Host

OTG B / Device

This 32-bit register specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary because the fault condition may require a response faster than one provided by firmware.

USBEPEN is shown in [Figure 27-78](#) and described in [Table 27-85](#).

Return to [Summary Table](#).

Figure 27-78. USBEPC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						PFLTACT	
R-0x0						R/W-0x0	
7	6	5	4	3	2	1	0
RESERVED	PFLTAEN	PFLTSEN	PFLTEN	RESERVED	EPENDE	EPEN	
R-0x0	R/W-0x0	R/W-0x0	R/W-0x0	R-0x0	R/W-0x0	R/W-0x0	

Table 27-85. USBEPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9-8	PFLTACT	R/W	0x0	Power Fault Action. This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault. 0x0 = Unchanged. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. 0x1 = Tristate. USB0EPEN is undriven (tristate). 0x2 = Low. USB0EPEN is driven Low. 0x3 = High. USB0EPEN is driven High.
7	RESERVED	R	0x0	
6	PFLTAEN	R/W	0x0	Power Fault Action Enable. This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the USB0EPEN signal. 0x0 = Disabled. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. 0x1 = Enabled. The USB0EPEN output is automatically changed to the state specified by the PFLTACT field.
5	PFLTSEN	R/W	0x0	Power Fault Sense. This bit specifies the logical sense of the USB0PFLT input signal that indicates an error condition. The complementary state is the inactive state. 0x0 = Low Fault. If USB0PFLT is driven Low, the power fault is signaled internally (if enabled by the PFLTEN bit). 0x1 = High Fault. If USB0PFLT is driven High, the power fault is signaled internally (if enabled by the PFLTEN bit).

Table 27-85. USBEPC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	PFLTEN	R/W	0x0	Power Fault Input Enable. This bit specifies whether the USB0PFLT input signal is used in internal logic. 0x0 = Not Used. The USB0PFLT signal is ignored. 0x1 = Used. The USB0PFLT signal is used internally.
3	RESERVED	R	0x0	
2	EPENDE	R/W	0x0	EPEN Drive Enable. This bit specifies whether the USB0EPEN signal is driven or undriven (tristate). When driven, the signal value is specified by the EPEN field. When not driven, the EPEN field is ignored and the USB0EPEN signal is placed in a high-impedance state. The USB0EPEN signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the high-impedance state, system designers may bias the power supply enable to the disabled state using a large resistor (100 kohm) and later configure and drive the output signal to enable the power supply. 0x0 = Not Driven. The USB0EPEN signal is high impedance. 0x1 = Driven. The USB0EPEN signal is driven to the logical value specified by the value of the EPEN field.
1-0	EPEN	R/W	0x0	External Power Supply Enable Configuration. This bit field specifies and controls the logical value driven on the USB0EPEN signal. 0x0 = Power Enable Active Low. The USB0EPEN signal is driven Low if the EPENDE bit is set. 0x1 = Power Enable Active High. The USB0EPEN signal is driven High if the EPENDE bit is set. 0x2 = Power Enable High if VBUS Low. The USB0EPEN signal is driven High when the A device is not recognized. 0x3 = Power Enable High if VBUS High. The USB0EPEN signal is driven High when the A device is recognized.

27.5.66 USBEPCRIS Register (Offset = 0x404) [reset = 0x0]

USB External Power Control Raw Interrupt Status (USBEPCRIS)

OTG A / Host

OTG B / Device

This 32-bit register specifies the unmasked interrupt status of the two-pin external power interface.

USBEPCRIS is shown in [Figure 27-79](#) and described in [Table 27-86](#).

Return to [Summary Table](#).

Figure 27-79. USBEPCRIS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															PF
R-0x0															R-0x0

Table 27-86. USBEPCRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	PF	R	0x0	USB Power Fault Interrupt Status. This bit is cleared by writing a 1 to the PF bit in the USBEPCISC register. 0x0 = An interrupt has not occurred. 0x1 = A Power Fault status has been detected.

27.5.67 USBEPCIM Register (Offset = 0x408) [reset = 0x0]

USB External Power Control Interrupt Mask (USBEPCIM)

OTG A / Host

OTG B / Device

This 32-bit register specifies the interrupt mask of the two-pin external power interface.

USBEPICIM is shown in [Figure 27-80](#) and described in [Table 27-87](#).

Return to [Summary Table](#).

Figure 27-80. USBEPCIM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															PF
R-0x0															R/W-0x0

Table 27-87. USBEPCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	PF	R/W	0x0	USB Power Fault Interrupt Mask. 0x0 = A detected power fault does not affect the interrupt status. 0x1 = The raw interrupt signal from a detected power fault is sent to the interrupt controller.

27.5.68 USBEPCISC Register (Offset = 0x40C) [reset = 0x0]

USB External Power Control Interrupt Status and Clear (USBEPICISC)

OTG A / Host

OTG B / Device

This 32-bit register specifies the masked interrupt status of the two-pin external power interface. It also provides a method to clear the interrupt state.

USBEPICISC is shown in [Figure 27-81](#) and described in [Table 27-88](#).

Return to [Summary Table](#).

Figure 27-81. USBEPICISC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															PF
R-0x0															R/W1 C-0x0

Table 27-88. USBEPICISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	PF	R/W1C	0x0	<p>USB Power Fault Interrupt Status and Clear.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the PF bit in the USBEPCRIS register.</p> <p>0x0 = No interrupt has occurred or the interrupt is masked.</p> <p>0x1 = The PF bits in the USBEPCRIS and USBEPCIM registers are set, providing an interrupt to the interrupt controller.</p>

27.5.69 USBDRRIS Register (Offset = 0x410) [reset = 0x0]

USB Device RESUME Raw Interrupt Status (USBDRRIS)

OTG A / Host

OTG B / Device

The USBDRRIS 32-bit register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

USBDRRIS is shown in [Figure 27-82](#) and described in [Table 27-89](#).

Return to [Summary Table](#).

Figure 27-82. USBDRRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							RESUME
R-0x0							R-0x0

Table 27-89. USBDRRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	RESUME	R	0x0	RESUME Interrupt Status. This bit is cleared by writing a 1 to the RESUME bit in the USBDRISC register. 0x0 = An interrupt has not occurred. 0x1 = A RESUME status has been detected.

27.5.70 USBDRIM Register (Offset = 0x414) [reset = 0x0]

USB Device RESUME Interrupt Mask (USBDRIM)

OTG A / Host

OTG B / Device

The USBDRIM 32-bit register is the masked interrupt status register. On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

USBDRIM is shown in [Figure 27-83](#) and described in [Table 27-90](#).

Return to [Summary Table](#).

Figure 27-83. USBDRIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							RESUME
R-0x0							R/W-0x0

Table 27-90. USBDRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	RESUME	R/W	0x0	RESUME Interrupt Mask. 0x0 = A detected RESUME does not affect the interrupt status. 0x1 = The raw interrupt signal from a detected RESUME is sent to the interrupt controller. This bit should only be set when a SUSPEND has been detected (the SUSPEND bit in the USBIS register is set).

27.5.71 USBDRISC Register (Offset = 0x418) [reset = 0x0]

USB Device RESUME Interrupt Status and Clear (USBDRISC)

OTG A / Host

OTG B / Device

The USBDRISC 32-bit register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

USBDRISC is shown in [Figure 27-84](#) and described in [Table 27-91](#).

Return to [Summary Table](#).

Figure 27-84. USBDRISC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							RESUME
R-0x0							R/W1C-0x0

Table 27-91. USBDRISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	RESUME	R/W1C	0x0	RESUME Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the USBDRCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The RESUME bits in the USBDRRIS and USBDRCIM registers are set, providing an interrupt to the interrupt controller.

27.5.72 USBGPCS Register (Offset = 0x41C) [reset = 0x0]

USB General-Purpose Control and Status (USBGPCS)

OTG A / Host

OTG B / Device

USBGPCS provides the state of the internal ID signal.

NOTE: When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMOD field in the USB General-Purpose Control and Status (USBGPCS) register can be used to connect the USB0VBUS and /or USB0ID inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. Note that PB1 (USB0VBUS) is a 5-V tolerant signal as required. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pullup resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

The termination resistors for the USB PHY have been added internally, and thus there is no need for external resistors. For a device, there is a 1.5 kohm pullup on the D+ and for a host there are 15 kohm pulldowns on both D+ and D-.

USBGPCS is shown in [Figure 27-85](#) and described in [Table 27-92](#).

Return to [Summary Table](#).

Figure 27-85. USBGPCS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													DEVMOD		
R-0x0													R/W-0x0		

Table 27-92. USBGPCS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0x0	
2-0	DEVMOD	R/W	0x0	Device Mode. This bit field determines whether the USB0VBUS or USB0ID pin are forced to a value to allow these pins to be free for use as GPIO. 0x0 = Use USB0VBUS and USB0ID pin. 0x1 = Reserved 0x2 = Force USB0VBUS and USB0ID low. 0x3 = Force USB0VBUS and USB0ID high. 0x4 = Use USB0VBUS and force USB0ID low. 0x5 = Use USB0VBUS and force USB0ID high. 0x6 = Reserved 0x7 = Reserved

27.5.73 USBVDC Register (Offset = 0x430) [reset = 0x0]

USB VBUS Droop Control (USBVDC)

OTG A / Host

This 32-bit register enables a controlled masking of VBUS to compensate for any in-rush current by a Device that is connected to the Host controller. The in-rush current can cause VBUS to droop, causing the USB controller's behavior to be unexpected. The USB Host controller allows VBUS to fall lower than the VBUS Valid level (4.75 V) but not below AValid (2.0 V) for 65 microseconds without signaling a VBUSERR interrupt in the controller. Without this, any glitch on VBUS would force the USB Host controller to remove power from VBUS and then re-enumerate the Device.

USBVDC is shown in [Figure 27-86](#) and described in [Table 27-93](#).

Return to [Summary Table](#).

Figure 27-86. USBVDC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							VBDEN
R-0x0							R/W-0x0

Table 27-93. USBVDC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	VBDEN	R/W	0x0	VBUS Droop Enable. 0x0 = No effect. 0x1 = Any changes from VBUSVALID are masked when VBUS goes below 4.75 V but not lower than 2.0 V for 65 microseconds. During this time, the VBUS state indicates VBUSVALID.

27.5.74 USBVDCRIS Register (Offset = 0x434) [reset = 0x0]

USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS)

OTG A / Host

This 32-bit register specifies the unmasked interrupt status of the VBUS droop limit of 65 μ s.

USBVDCRIS is shown in [Figure 27-87](#) and described in [Table 27-94](#).

Return to [Summary Table](#).

Figure 27-87. USBVDCRIS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															VD
R-0x0															R-0x0

Table 27-94. USBVDCRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	VD	R	0x0	VBUS Droop Raw Interrupt Status. This bit is cleared by writing a 1 to the VD bit in the USBVDCISC register. 0x0 = An interrupt has not occurred. 0x1 = A VBUS droop lasting for 65 μ s has been detected.

27.5.75 USBVDCIM Register (Offset = 0x438) [reset = 0x0]

USB VBUS Droop Control Interrupt Mask (USBVDCIM)

OTG A / Host

This 32-bit register specifies the interrupt mask of the VBUS droop.

USBVDCIM is shown in [Figure 27-88](#) and described in [Table 27-95](#).

Return to [Summary Table](#).

Figure 27-88. USBVDCIM Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															VD
R-0x0															R/W-0x0

Table 27-95. USBVDCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	VD	R/W	0x0	VBUS Droop Interrupt Mask. 0x0 = A detected VBUS droop does not affect the interrupt status. 0x1 = The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller.

27.5.76 USBVDCISC Register (Offset = 0x43C) [reset = 0x0]

USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC)

OTG A / Host

This 32-bit register specifies the masked interrupt status of the VBUS droop and provides a method to clear the interrupt state.

USBVDCISC is shown in [Figure 27-89](#) and described in [Table 27-96](#).

Return to [Summary Table](#).

Figure 27-89. USBVDCISC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															VD
R-0x0															R/W1 C-0x0

Table 27-96. USBVDCISC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	VD	R/W1C	0x0	VBUS Droop Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the VD bit in the USBVDCRIS register. 0x0 = No interrupt has occurred or the interrupt is masked. 0x1 = The VD bits in the USBVDCRIS and USBVDCIM registers are set, providing an interrupt to the interrupt controller.

27.5.77 USBPP Register (Offset = 0xFC0) [reset = 0x8F1]

USB Peripheral Properties (USBPP)

The USBPP register provides information regarding the properties of the USB module.

USBPP is shown in [Figure 27-90](#) and described in [Table 27-97](#).

Return to [Summary Table](#).

Figure 27-90. USBPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
ECNT							
R-0x8							
7	6	5	4	3	2	1	0
USB		ULPI	PHY	TYPE			
R-0x3		R-0x1	R-0x1	R-0x1			

Table 27-97. USBPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0x0	
15-8	ECNT	R	0x8	Endpoint Count. Encodes the number of endpoint pairs.
7-6	USB	R	0x3	USB Capability 0x0 = NA. USB is not present. 0x1 = DEVICE. Device Only. 0x2 = HOST. Device or Host. 0x3 = OTG. Device, Host, or OTG.
5	ULPI	R	0x1	ULPI Present. 0x0 = ULPI interface (pins) is not present on the device. 0x1 = ULPI interface (pins) is present on the device.
4	PHY	R	0x1	PHY Present. 0x0 = A PHY is not integrated with the USB MAC. 0x1 = A PHY is integrated with the USB MAC.
3-0	TYPE	R	0x1	Controller Type. 0x0 = Reserved 0x1 = USB controller revision. 0x2 to 0xF = Reserved

27.5.78 USBPC Register (Offset = 0xFC4) [reset = 0x0]

USB Peripheral Configuration (USBPC)

The USBPC register provides information regarding the configuration of the USB module.

USBPC is shown in [Figure 27-91](#) and described in [Table 27-98](#).

Return to [Summary Table](#).

Figure 27-91. USBPC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							ULPIEN
R-0x0							R/W-0x0
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							
R-0x0							

Table 27-98. USBPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0x0	
16	ULPIEN	R/W	0x0	ULPI Enable. When this bit is set to 1, the GPIO pin corresponding to the alternate function, USB0CLK, has to be configured for Digital Loop Back (GPIODEN register, GPIO offset 0x51C) 0x0 = Integrated PHY 0x1 = ULPI-attached PHY
15-0	RESERVED	R	0x0	

27.5.79 USBCC Register (Offset = 0xFC8) [reset = 0x0]

USB Clock Configuration (USBCC)

The USBCC register specifies the clock configuration for the USB controller.

NOTE: When the USB module uses the integrated USB PHY, the MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 30 MHz. In addition, only integer divisors should be used to achieve the 60 MHz USB clock source. Fractional divisors may increase jitter and compromise USB function.

USBCC is shown in [Figure 27-92](#) and described in [Table 27-99](#).

Return to [Summary Table](#).

Figure 27-92. USBCC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED						CLKEN	CSD
R-0x0						R/W-0x0	R/W-0x0
7	6	5	4	3	2	1	0
RESERVED				CLKDIV			
R-0x0				R/W-0x0			

Table 27-99. USBCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0x0	
9	CLKEN	R/W	0x0	USB Clock Enable. This bit enables the 60 MHz clock used for the USB PHY. This bit must always be set when using the USB, regardless of whether the PHY is internal or external. 0x0 = USB clock is disabled. 0x1 = USB clock is enabled.
8	CSD	R/W	0x0	Clock Source/Direction. This bit specifies the source of the 60 MHz USB PHY clock when using ULPI. 0x0 = Internal Source. USB0CLK is an output to the external ULPI PHY and is configured using the following equation: $PLL_VCO/(CLKDIV + 1)$ 0x1 = External Source. USB0CLK is an input from the ULPI external PHY.
7-4	RESERVED	R	0x0	
3-0	CLKDIV	R/W	0x0	PLL Clock Divisor. This field specifies the divisor used to reduce the PLL VCO output to the 60 MHz clock source required for the serialization and deserialization module of the USB controller. The divisor value is the CLKDIV value plus 1.

Watchdog Timers

This chapter describes the watchdog timers.

Topic	Page
28.1 Introduction	1798
28.2 Block Diagram	1798
28.3 Functional Description	1799
28.4 Initialization and Configuration	1800
28.5 WDT Registers.....	1801

28.1 Introduction

A watchdog timer can generate a nonmaskable interrupt (NMI), a regular interrupt, or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or the failure of an external device to respond in the expected way. The MSP432E4 microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other module (Watchdog Timer 1) is clocked by the clock source programmed in the ALTCLK field of the Alternate Clock Configuration (ALTCLKCFG) register, System Control offset 0x138. The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the Watchdog Timer Control (WDTCTL) register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

The MSP432E4 controller has two Watchdog Timer modules with the following features:

- 32-bit, down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking and optional NMI function
- Lock register protection from runaway software
- Reset generation logic with an enable or disable
- User-enabled stalling when the microcontroller asserts the CPU Halt flag during debug

The Watchdog Timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. When the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

28.2 Block Diagram

[Figure 28-1](#) shows the WDT module block diagram.

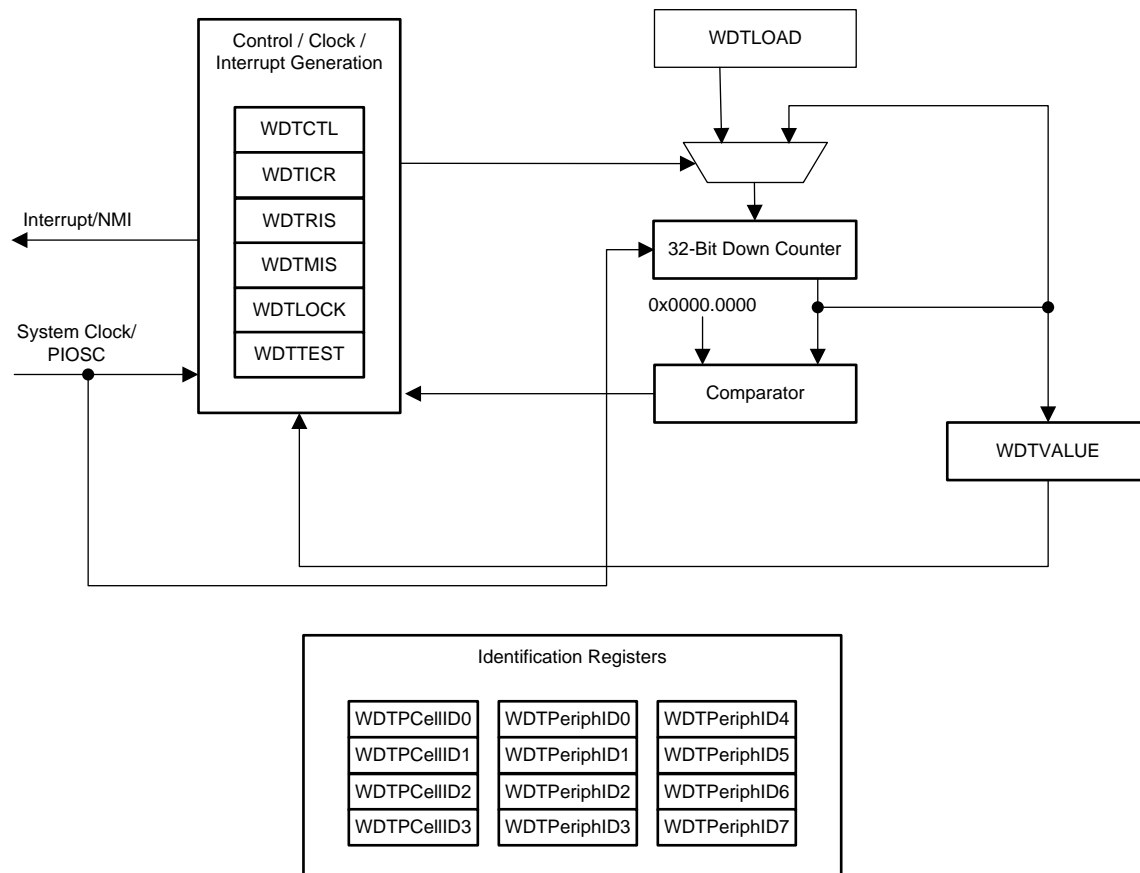


Figure 28-1. WDT Module Block Diagram

28.3 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. The watchdog interrupt can be programmed to be a nonmaskable interrupt (NMI) using the INTTYPE bit in the WDTCTL register. After the first time-out event, the 32-bit counter is reloaded with the value of the Watchdog Timer Load (WDTLOAD) register, and the timer resumes counting down from that value. When the Watchdog Timer has been configured, the Watchdog Timer Lock (WDTLOCK) register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the RESEN bit in the WDTCTL register, the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the WDTLOAD register, and counting resumes from that value.

- If WDTLOAD is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.
- Writing to WDTLOAD does not clear an active interrupt. An interrupt must be specifically cleared by writing to the Watchdog Interrupt Clear (WDTICR) register.
- The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is reenabled, the 32-bit counter is preloaded with the load register value and not its last state.
- The watchdog timer is disabled by default out of reset. To achieve maximum watchdog protection of the device, the watchdog timer can be enabled at the start of the reset vector.

28.3.1 Register Access Timing

Because the WDT1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the Watchdog Control (WDTCTL) register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set when the write completes, indicating to software that another write or read may be started safely. Software should poll WDTCTL for WRC = 1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

28.4 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the Rn bit in the Watchdog Timer Run Mode Clock Gating Control (RCGCWD) register, see [Section 4.2.85](#).

The Watchdog Timer is configured using the following sequence:

1. Load the WDTLOAD register with the desired timer load value.
2. If WDT1, wait for the WRC bit in the WDTCTL register to be set.
3. Set the INTEN bit (if interrupts are required) or the RESEN bit (if a reset is required after two timeouts) in the WDTCTL register. The Watchdog Timer starts when either of them is enabled.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the WDTLOCK register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

To service the watchdog, periodically reload the count value into the WDTLOAD register to restart the count. The interrupt can be enabled using the INTEN bit in the WDTCTL register to allow the processor to attempt corrective action if the watchdog is not serviced often enough. The RESEN bit in WDTCTL can be set so that the system resets if the failure is not recoverable using the ISR.

NOTE: The application should be sure not to modify the ALTCLK encoding in the ALTCLKCFG register while WDT1 is enabled and running.

28.5 WDT Registers

[Table 28-1](#) lists the memory-mapped registers for the WDT. All register offset addresses not listed in [Table 28-1](#) should be considered as reserved locations and the register contents should not be modified.

The offset is relative to the Watchdog Timer base address:

- WDT0: 0x40000000
- WDT1: 0x40001000

The Watchdog Timer module clock must be enabled before the registers can be programmed (see [Section 4.2.85](#)).

Table 28-1. WDT Registers

Offset	Acronym	Register Name	Section
0x0	WDTLOAD	Watchdog Load	Section 28.5.1
0x4	WDTVALUE	Watchdog Value	Section 28.5.2
0x8	WDTCTL	Watchdog Control	Section 28.5.3
0xC	WDTICR	Watchdog Interrupt Clear	Section 28.5.4
0x10	WDTRIS	Watchdog Raw Interrupt Status	Section 28.5.5
0x14	WDTMIS	Watchdog Masked Interrupt Status	Section 28.5.6
0x418	WDTTEST	Watchdog Test	Section 28.5.7
0xC00	WDTLOCK	Watchdog Lock	Section 28.5.8
0xFD0	WDTPeriphID4	Watchdog Peripheral Identification 4	Section 28.5.9
0xFD4	WDTPeriphID5	Watchdog Peripheral Identification 5	Section 28.5.10
0xFD8	WDTPeriphID6	Watchdog Peripheral Identification 6	Section 28.5.11
0xFDC	WDTPeriphID7	Watchdog Peripheral Identification 7	Section 28.5.12
0xFE0	WDTPeriphID0	Watchdog Peripheral Identification 0	Section 28.5.13
0xFE4	WDTPeriphID1	Watchdog Peripheral Identification 1	Section 28.5.14
0xFE8	WDTPeriphID2	Watchdog Peripheral Identification 2	Section 28.5.15
0xFEC	WDTPeriphID3	Watchdog Peripheral Identification 3	Section 28.5.16
0xFF0	WDTPCellID0	Watchdog PrimeCell Identification 0	Section 28.5.17
0xFF4	WDTPCellID1	Watchdog PrimeCell Identification 1	Section 28.5.18
0xFF8	WDTPCellID2	Watchdog PrimeCell Identification 2	Section 28.5.19
0xFFC	WDTPCellID3	Watchdog PrimeCell Identification 3	Section 28.5.20

Complex bit access types are encoded to fit into small table cells. [Table 28-2](#) shows the codes that are used for access types in this section.

Table 28-2. WDT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

28.5.1 WDTLOAD Register (Offset = 0x0) [reset = 0xFFFFFFFF]

Watchdog Load (WDTLOAD)

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the WDTLOAD register is loaded with 0x00000000, an interrupt is immediately generated.

WDTLOAD is shown in [Figure 28-2](#) and described in [Table 28-3](#).

Return to [Summary Table](#).

Figure 28-2. WDTLOAD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTLOAD																															
R/W-0xFFFFFFFF																															

Table 28-3. WDTLOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTLOAD	R/W	0xFFFFFFFF	Watchdog Load Value

28.5.2 WDTVALUE Register (Offset = 0x4) [reset = 0xFFFFFFFF]

Watchdog Value (WDTVALUE)

This register contains the current count value of the timer.

WDTVALUE is shown in [Figure 28-3](#) and described in [Table 28-4](#).

Return to [Summary Table](#).

Figure 28-3. WDTVALUE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTVALUE																															
R-0xFFFFFFFF																															

Table 28-4. WDTVALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTVALUE	R	0xFFFFFFFF	Watchdog Value. Current value of the 32-bit down counter.

28.5.3 WDTCTL Register (Offset = 0x8)

Watchdog Control (WDTCTL)

WDT0, reset = 0x0000.0000

WDT1, reset = 0x8000.0000

This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled by setting the INTEN bit, all subsequent writes to the INTEN bit are ignored. The only mechanisms that can re-enable writes to this bit are a hardware reset or a software reset initiated by setting the appropriate bit in the Watchdog Timer Software Reset (SRWD) register.

NOTE: Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the Watchdog Control (WDTCTL) register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll WDTCTL for WRC = 1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a WRC bit.

WDTCTL is shown in [Figure 28-4](#) and described in [Table 28-5](#).

Return to [Summary Table](#).

Figure 28-4. WDTCTL Register

31	30	29	28	27	26	25	24
WRC	RESERVED						
R-0x1	R-0x0						
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED					INTTYPE	RESEN	INTEN
R-0x0					R/W-0x0	R/W-0x0	R/W-0x0

Table 28-5. WDTCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	WRC	R	0x0	Write Complete. This bit is RESERVED for WDT0 and has a reset value of 0. 0x0 = A write access to one of the WDT1 registers is in progress. 0x1 = A write access is not in progress, and WDT1 registers can be read or written.
30-3	RESERVED	R	0x0	
2	INTTYPE	R/W	0x0	Watchdog Interrupt Type 0x0 = Watchdog interrupt is a standard interrupt. 0x1 = Watchdog interrupt is a non-maskable interrupt.

Table 28-5. WDTCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	RESEN	R/W	0x0	Watchdog Reset Enable 0x0 = Disabled. 0x1 = Enable the Watchdog module reset output. Setting this bit enables the Watchdog Timer.
0	INTEN	R/W	0x0	Watchdog Interrupt Enable 0x0 = Interrupt event disabled. Once this bit is set, it can only be cleared by a hardware reset or a software reset initiated by setting the appropriate bit in the Watchdog Timer Software Reset (SRWD) register. 0x1 = Interrupt event enabled. Once enabled, all writes are ignored. Setting this bit enables the Watchdog Timer.

28.5.4 WDTICR Register (Offset = 0xC) [reset = X]

Watchdog Interrupt Clear (WDTICR)

This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the WDTLOAD register. Write to this register when a watchdog time-out interrupt has occurred to properly service the Watchdog. Value for a read or reset is indeterminate.

NOTE: Locking the watchdog registers by using the WDTLOCK register does not affect the WDTICR register and allows interrupts to always be serviced. Thus, a write at any time of the WDTICR register clears the WDTMIS register and reloads the 32-bit counter from the WDTLOAD register. The WDTICR register should only be written when interrupts have triggered and need to be serviced.

WDTICR is shown in [Figure 28-5](#) and described in [Table 28-6](#).

Return to [Summary Table](#).

Figure 28-5. WDTICR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTINTCLR																															
W-X																															

Table 28-6. WDTICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTINTCLR	W	X	Watchdog Interrupt Clear. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the WDTLOAD register. Write to this register when a watchdog time-out interrupt has occurred to properly service the Watchdog. Value for a read or reset is indeterminate.

28.5.5 WDTRIS Register (Offset = 0x10) [reset = 0x0]

Watchdog Raw Interrupt Status (WDTRIS)

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

WDTRIS is shown in [Figure 28-6](#) and described in [Table 28-7](#).

Return to [Summary Table](#).

Figure 28-6. WDTRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							WDTRIS
R-0x0							R-0x0

Table 28-7. WDTRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	WDTRIS	R	0x0	Watchdog Raw Interrupt Status 0x0 = The watchdog has not timed out. 0x1 = A watchdog time-out event has occurred.

28.5.6 WDTMIS Register (Offset = 0x14) [reset = 0x0]

Watchdog Masked Interrupt Status (WDTMIS)

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

WDTMIS is shown in [Figure 28-7](#) and described in [Table 28-8](#).

Return to [Summary Table](#).

Figure 28-7. WDTMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							
R-0x0							
7	6	5	4	3	2	1	0
RESERVED							WDTMIS
R-0x0							R-0x0

Table 28-8. WDTMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0x0	
0	WDTMIS	R	0x0	Watchdog Masked Interrupt Status 0x0 = The watchdog has not timed out or the watchdog timer interrupt is masked. 0x1 = A watchdog time-out event has been signalled to the interrupt controller.

28.5.7 WDTTEST Register (Offset = 0x418) [reset = 0x0]

Watchdog Test (WDTTEST)

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

WDTTEST is shown in [Figure 28-8](#) and described in [Table 28-9](#).

Return to [Summary Table](#).

Figure 28-8. WDTTEST Register

31	30	29	28	27	26	25	24
RESERVED							
R-0x0							
23	22	21	20	19	18	17	16
RESERVED							
R-0x0							
15	14	13	12	11	10	9	8
RESERVED							STALL
R-0x0							R/W-0x0
7	6	5	4	3	2	1	0
RESERVED							
R-0x0							

Table 28-9. WDTTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0x0	
8	STALL	R/W	0x0	Watchdog Stall Enable 0x0 = The watchdog timer continues counting if the microcontroller is stopped with a debugger. 0x1 = If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
7-0	RESERVED	R	0x0	

28.5.8 WDTLOCK Register (Offset = 0xC00) [reset = 0x0]

Watchdog Lock (WDTLOCK)

Writing 0x1ACCE551 to the WDTLOCK register enables write access to all other registers. Writing any other value to the WDTLOCK register re-enables the locked state for register writes to all the other registers, except for the Watchdog Test (WDTTEST) register. Reading the WDTLOCK register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the WDTLOCK register returns 0x00000001 when locked; otherwise, the returned value is 0x00000000 when unlocked.

WDTLOCK is shown in [Figure 28-9](#) and described in [Table 28-10](#).

Return to [Summary Table](#).

Figure 28-9. WDTLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTLOCK																															
R/W-0x0																															

Table 28-10. WDTLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTLOCK	R/W	0x0	<p>Watchdog Lock.</p> <p>A write of the value 0x1ACCE551 unlocks the watchdog registers for write access.</p> <p>A write of any other value reapplies the lock, preventing any register updates, except for the WDTTEST register.</p> <p>Avoid writes to the WDTTEST register when the watchdog registers are locked.</p> <p>A read of this register returns the following values:</p> <p>0x00000000 = Unlocked</p> <p>0x00000001 = Locked</p>

28.5.9 WDTPeriphID4 Register (Offset = 0xFD0) [reset = 0x0]

Watchdog Peripheral Identification 4 (WDTPeriphID4)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID4 is shown in [Figure 28-10](#) and described in [Table 28-11](#).

Return to [Summary Table](#).

Figure 28-10. WDTPeriphID4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID4							
R-0x0																								R-0x0							

Table 28-11. WDTPeriphID4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID4	R	0x0	WDT Peripheral ID Register [7:0]

28.5.10 WDTPeriphID5 Register (Offset = 0xFD4) [reset = 0x0]

Watchdog Peripheral Identification 5 (WDTPeriphID5)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID5 is shown in [Figure 28-11](#) and described in [Table 28-12](#).

Return to [Summary Table](#).

Figure 28-11. WDTPeriphID5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID5							
R-0x0																								R-0x0							

Table 28-12. WDTPeriphID5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID5	R	0x0	WDT Peripheral ID Register [15:8]

28.5.11 WDTPeriphID6 Register (Offset = 0xFD8) [reset = 0x0]

Watchdog Peripheral Identification 6 (WDTPeriphID6)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID6 is shown in [Figure 28-12](#) and described in [Table 28-13](#).

Return to [Summary Table](#).

Figure 28-12. WDTPeriphID6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID6							
R-0x0																								R-0x0							

Table 28-13. WDTPeriphID6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID6	R	0x0	WDT Peripheral ID Register [23:16]

28.5.12 WDTPeriphID7 Register (Offset = 0xFDC) [reset = 0x0]

Watchdog Peripheral Identification 7 (WDTPeriphID7)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID7 is shown in [Figure 28-12](#) and described in [Table 28-13](#).

Return to [Summary Table](#).

Figure 28-13. WDTPeriphID7 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID6							
R-0x0																								R-0x0							

Table 28-14. WDTPeriphID7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID7	R	0x0	WDT Peripheral ID Register [31:24]

28.5.13 WDTPeriphID0 Register (Offset = 0xFE0) [reset = 0x5]

Watchdog Peripheral Identification 0 (WDTPeriphID0)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID0 is shown in [Figure 28-14](#) and described in [Table 28-15](#).

Return to [Summary Table](#).

Figure 28-14. WDTPeriphID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID0							
R-0x0																								R-0x5							

Table 28-15. WDTPeriphID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID0	R	0x5	Watchdog Peripheral ID Register [7:0]

28.5.14 WDTPeriphID1 Register (Offset = 0xFE4) [reset = 0x18]

Watchdog Peripheral Identification 1 (WDTPeriphID1)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID1 is shown in [Figure 28-15](#) and described in [Table 28-16](#).

Return to [Summary Table](#).

Figure 28-15. WDTPeriphID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID1							
R-0x0																								R-0x18							

Table 28-16. WDTPeriphID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID1	R	0x18	Watchdog Peripheral ID Register [15:8]

28.5.15 WDTPeriphID2 Register (Offset = 0xFE8) [reset = 0x18]

Watchdog Peripheral Identification 2 (WDTPeriphID2)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID2 is shown in [Figure 28-16](#) and described in [Table 28-17](#).

Return to [Summary Table](#).

Figure 28-16. WDTPeriphID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID2							
R-0x0																								R-0x18							

Table 28-17. WDTPeriphID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID2	R	0x18	Watchdog Peripheral ID Register [23:16]

28.5.16 WDTPeriphID3 Register (Offset = 0xFEC) [reset = 0x1]

Watchdog Peripheral Identification 3 (WDTPeriphID3)

The WDTPeriphIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPeriphID3 is shown in [Figure 28-17](#) and described in [Table 28-18](#).

Return to [Summary Table](#).

Figure 28-17. WDTPeriphID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID3							
R-0x0																								R-0x1							

Table 28-18. WDTPeriphID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	PID3	R	0x1	Watchdog Peripheral ID Register [31:24]

28.5.17 WDTPCellID0 Register (Offset = 0xFF0) [reset = 0xD]

Watchdog PrimeCell Identification 0 (WDTPCellID0)

The WDTPCellIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPCellID0 is shown in [Figure 28-18](#) and described in [Table 28-19](#).

Return to [Summary Table](#).

Figure 28-18. WDTPCellID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID0							
R-0x0																								R-0xD							

Table 28-19. WDTPCellID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID0	R	0xD	Watchdog PrimeCell ID Register [7:0]

28.5.18 WDTPCellID1 Register (Offset = 0xFF4) [reset = 0xF0]

Watchdog PrimeCell Identification 1 (WDTPCellID1)

The WDTPCellIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPCellID1 is shown in [Figure 28-19](#) and described in [Table 28-20](#).

Return to [Summary Table](#).

Figure 28-19. WDTPCellID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID1							
R-0x0																								R-0xF0							

Table 28-20. WDTPCellID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID1	R	0xF0	Watchdog PrimeCell ID Register [15:8]

28.5.19 WDTPCellID2 Register (Offset = 0xFF8) [reset = 0x6]

Watchdog PrimeCell Identification 2 (WDTPCellID2)

The WDTPCellIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPCellID2 is shown in [Figure 28-20](#) and described in [Table 28-21](#).

Return to [Summary Table](#).

Figure 28-20. WDTPCellID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID2							
R-0x0																								R-0x6							

Table 28-21. WDTPCellID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID2	R	0x6	Watchdog PrimeCell ID Register [23:16]

28.5.20 WDTPCellID3 Register (Offset = 0xFFC) [reset = 0xB1]

Watchdog PrimeCell Identification 3 (WDTPCellID3)

The WDTPCellIDn registers are hard-coded and the fields within the register determine the reset value.

WDTPCellID3 is shown in [Figure 28-21](#) and described in [Table 28-22](#).

Return to [Summary Table](#).

Figure 28-21. WDTPCellID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID3							
R-0x0																								R-0xB1							

Table 28-22. WDTPCellID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0x0	
7-0	CID3	R	0xB1	Watchdog PrimeCell ID Register [31:24]

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from October 27, 2017 to October 25, 2018	Page
<ul style="list-style-type: none"> Corrected the enumerated values 0x0, 0x1, and 0x2 for the ALTCLK bit in Table 4-24, <i>ALTCLKCFG Register Field Descriptions</i> Updated the first paragraph in Section 7.2.3.5, <i>Execute-Only Protection</i> Updated Figure 9-2, <i>AES – ECB Feedback Mode</i>..... Updated Figure 9-5, <i>AES – CFB Feedback Mode</i>..... Updated Figure 9-7, <i>AES – XTS Operation</i> Corrected the enum 0h in the TYPE bit in Table 13-5, <i>CRCCTRL Register Field Descriptions</i>..... Changed EN0REF_CLK to EN0RREF_CLK throughout "Ethernet Controller" chapter to match data sheets Removed GPIO column from Table 15-1, <i>MII and RMII Signals</i>; see device-specific data sheet for pin assignments .. Added missing arrows and text in Figure 15-11, <i>System Time Update Using Fine Correction Method</i>..... Added comments that RTS and CTS are active low in Section 26.3.6.2.1, <i>Hardware Flow Control (RTS and CTS)</i> .. 	<p>244</p> <p>540</p> <p>664</p> <p>665</p> <p>667</p> <p>848</p> <p>886</p> <p>887</p> <p>917</p> <p>1626</p>

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated