Programmable Realtime Unit

Content is no longer maintained and is being kept for reference only!
For the most up to date PRU-ICSS collateral click here (http://processors.wiki.ti.com/index.php/PRU-ICSS)

^ Up to main **Programmable Realtime Unit Subsystem** Table of Contents

This arcticle is part of a collection of articles describing the PRU subsystem included in OMAP-L1x8/C674m/AM18xx devices (where m is an even number). To navigate to the main page for the PRU subsystem click on the link above.

Contents

Overview

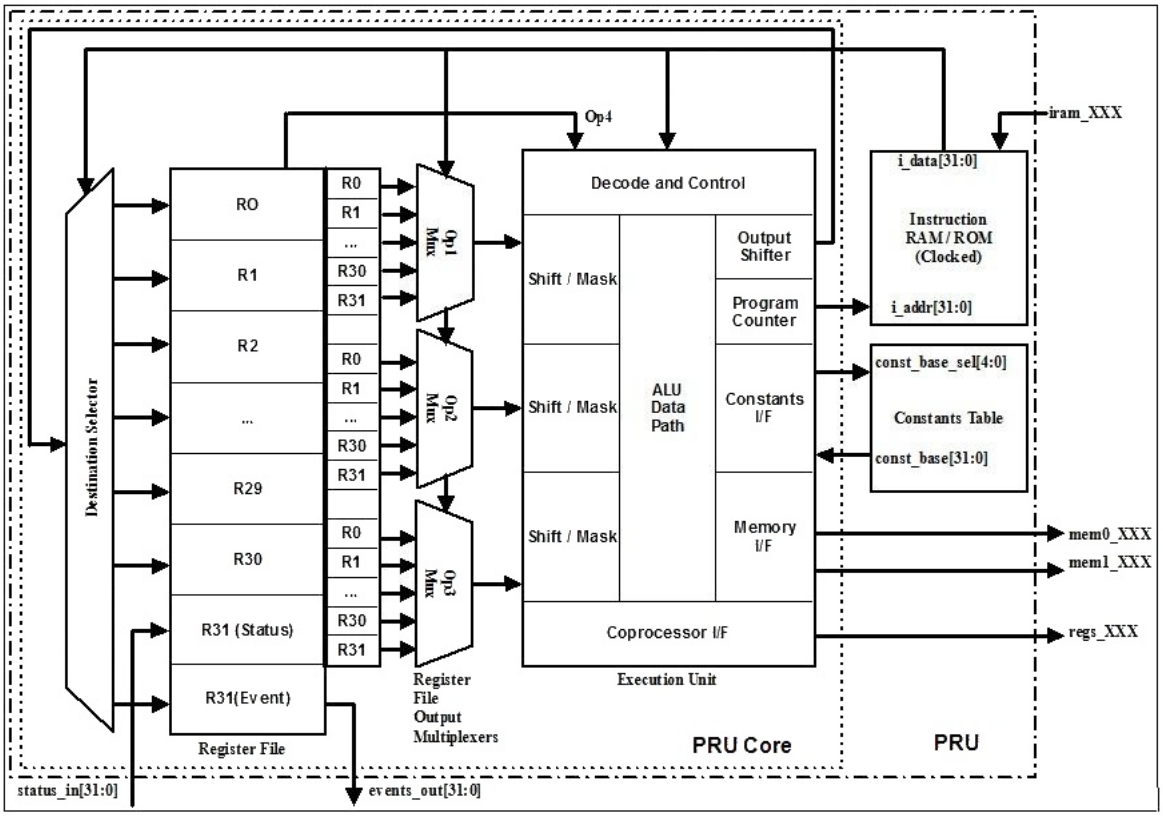
The PRU is a optimized for performing embedded tasks that require manipulation of packed memory mapped data structures, handling of system events that have tight realtime constraints and interfacing with systems external to the DSP/SoC. The PRU is both very small and very efficient at handling such tasks.

The major attributes of the PRU are as follows:

Attribute	Value
IO Architecture	Load / Store
Data Flow Architecture	Register to Register
Core Level Bus Architecture	
Type	4-Bus Harvard (1 Instruction , 3 Data)
Instruction I/F	32-Bit
Memory I/F 0	32-Bit
Memory I/F 1	32-Bit
Execution Model	
Issue Type	Scalar
Pipelining	None
Ordering	In-order
ALU Type	Unsigned Integer
Registers	
General Purpose (GP)	30 (R1 – R30)
External Status	1 (R31)
GP / Indexing	1 (R0)
Addressability	Bit, Byte (8-bit), Halfword (16-bit), Word (32-bit), Pointer
Addressing Modes	
Load Immediate	16-bit Immediate
Load / Store – Memory	Register Base + Register Offset
	Register Base + 8-bit Immediate Offset
	Register Base with auto increment / decrement
	Constant Table Base + Register Offset
	Constant Table Base + 8-bit Immediate Offset
	Constant Table Base with auto increment / decrement
Data Path Width	32-Bits
Instruction Width	32-Bits
Accessibility to Internal PRU Structures	Provides 32-bit Slave with 3 regions: ▪ Instruction RAM ▪ Control / Status registers ▪ Debug access to internal registers (R0-R31) and constant table

Description

The processor is based on a four bus architecture which allows instructions to be fetched and executed concurrently with data transfers. Additionally, an input is provided in order to allow external status information to be reflected in the internal processor status register. The figure below shows a block diagram of the processing element and the associated instruction RAM/ROM that contains the code that is to be executed.



PRU Memory map

PRU Memory Map Overview

The PRU control / status registers region contains control and status registers for the PRU. The control / status registers region memory map is shown in Table 1. Refer to the subsystem memory map for the base absolute address of the PRU register region.

Table 1: PRU Control/Status Register Memory Map

Address Offset	Register Name	Register Description
0x0000	CONTROL	PRU Control Register
0x0004	STATUS	PRU Status Register
0x0008	WAKEUP	PRU Wakeup Enable Register
0x000C	CYCLECNT	PRU Cycle Count
0x0010	STALLCNT	PRU Stall Count
0x0020	CONTABBLKIDX0	PRU Constant Table Block Index Register 0
0x0028	CONTABPROPTR0	PRU Constant Table Programmable Pointer Register 0
0x002C	CONTABPROPTR1	PRU Constant Table Programmable Pointer Register 1
0x0400 – 0x047C	INTGPR0-INTGPR31	PRU Internal General Purpose Registers (for Debug)
0x0480 – 0x04FC	INTCTER0 - INTCTER31	PRU Internal Constants Table Entry Registers (for Debug)

PRU Control Register (0x0000)

Bits	Field	Type	Reset	Description
31:16	PCRESETVAL	r/w	0	Program Counter Reset Value: This field controls the address where the PRU will start executing code from after it is taken out of reset*
15	RUNSTATE	r	0	Run State: This bit indicates whether the PRU is currently executing an instruction or is halted. 0 = PRU is halted and host has access to the instruction RAM and debug registers regions. 1 = PRU is currently running and the host is locked out of the instruction RAM and debug registers regions This bit is used by an external debug agent to know when the PRU has actually halted when waiting for a HALT instruction to execute, a single step to finish, or any other time when the pru_enable has been cleared.
14:9	RESERVED	r	0	-
8	SINGLESTEP	r/w	0	Single Step Enable: This bit controls whether or not the PRU will only execute a single instruction when enabled. 0 = PRU will free run when enabled 1 = PRU will execute a single instruction and then the pru_enable bit will be cleared. Note that this bit does not actually enable the PRU, it only sets the policy for how much code will be run after the PRU is enabled. The pru_enable bit must be explicitly asserted. It is legal to initialize both the single_step and pru_enable bits simultaneously. (Two independent writes are not

				required to cause the stated functionality)
7:4	RESERVED	r	0	-
3	COUNTENABLE	r/w	0	PRU Cycle Counter Enable: Enables PRU cycle counters 0 = Counters not enabled 1 = Counters enabled
2	SLEEPING	r/w	0	PRU Sleep Indicator: This bit indicates whether or not the PRU is currently asleep. 0 = PRU is not asleep 1 = PRU is asleep If this bit is written to a 0, the PRU will be forced to power up from sleep mode.
1	ENABLE	r/w	0	Processor Enable: This bit controls whether or not the PRU is allowed to fetch new instructions 0 = PRU is disabled 1 = PRU is enabled If this bit is de-asserted while the PRU is currently running and has completed the initial cycle of a multi-cycle instruction (LBxO,SBxO,SCAN, etc.), the current instruction will be allowed to complete before the PRU pauses execution. Otherwise, the PRU will halt immediately. Because of the unpredictability/timing sensitivity of the instruction execution loop, this bit is not a reliable indication of whether or not the PRU is currently running. The pru_state bit should be consulted for an absolute indication of the run state of the core. When the PRU is halted, it's internal state remains coherent therefore this bit can be reasserted without issuing a software reset and the PRU will resume processing exactly where it left off in the instruction stream.
0	SOFTRESET	r	0 => 1	Soft Reset: When this bit is cleared, the PRU will be reset. This bit is set back to 1 on the next cycle after it has been cleared.

PRU Status Register (0x0004)

Bits	Field	Type	Reset	Description
31:16	RESERVED	r	0	-
15:0	PCOUNTER	r	0	Program Counter: This field is a registered (1 cycle delayed) reflection of the PRU program counter*
* Note that the PC is an instruction address where each instruction is a 32 bit word. This is not a byte address and to compute the byte address just multiply the PC by 4 (PC of 2 = byte address of 0x8, or PC of 8 = byte address of 0x20).				

PRU Wakeup Enable Register (0x0008)

Bits	Field	Type	Reset	Description
31:0	BITWISEENABLES	r/w	0	Wakeup Enables: This field is ANDed with the incoming R31 status inputs (whose bit positions were specified in the stmap parameter) to produce a vector which is unary ORed to produce the status_wakeup source for the core. Setting any bit in this vector will allow the corresponding status input to wake up the core when it is asserted high. The PRU should set this enable vector prior to executing a SLP (sleep) instruction to ensure that the desired sources can wake up the core.

PRU Cycle Count Register (0x000C)

This register counts the number of cycles for which the PRU has been enabled.

Bits	Field	Type	Reset	Description
31:0	CYCLECOUNT	r/wc	0	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits "ENABLE" and "COUNTENABLE" set in the PRU control register). Counting halts while the PRU is disabled or counter is disabled, and resumes when re-eneabled. Counter clears the "COUNTENABLE" bit in the PRU control register when the count reaches 0xFFFFFFFF. (Count does does not wrap). The register can be read at any time. The register can be cleared when the counter or PRU is disabled. Clearing this register also clears the PRU Stall Count Register.

PRU Stall Count Register (0x0010)

This register counts the number of cycles for which the PRU has been enabled, but unable to fetch a new instruction. It is linked to the Cycle Count Register (0x0C) such that this register reflects the stall cycles measured over the same cycles as counted by the cycle count register. Thus the value of this register is always less than or equal to cycle count.

Bits	Field	Type	Reset	Description
31:0	STALLCOUNT	r	0	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits "ENABLE" and "COUNTENABLE" set in the PRU control register), and the PRU was unable to fetch a new instruction for any reason. Counting halts while the PRU is disabled or the counter is disabled, and resumes when re-enabled. The register can be read at any time. The register is cleared when PRU Cycle Count Register is cleared.

PRU Constant Table Block Index Register (0x0020)

This register is used to set the block indices which are used to modify entries 24 and 25 in the PRU Constant Table. This register can be written by the PRU whenever it needs to change to a new base pointer for a block in the State / Scratchpad RAM. This function is useful since the PRU is often processing multiple processing threads which require it to change contexts. The PRU can use this register to avoid requiring excessive amounts of code for repetitive context switching. The format of this register is as follows:

Bits	Field	Type	Reset	Description
31:20	RESERVED	r	0	-
19:16	C25	r/w	0	PRU Constant Entry 25 Block Index: This field sets the value that will appear in bits 11:8 of entry 25 in the PRU Constant Table
15:4	RESERVED	r	0	-
3:0	C24	r/w	0	PRU Constant Entry 24 Block Index: This field sets the value that will appear in bits 11:8 of entry 24 in the PRU Constant Table

PRU Constant Table Programmable Pointer Register 0 (0x0028)

This register allows the PRU to set up the 256-byte page index for entries 28 and 29 in the PRU Constant Table which serve as general purpose pointers which can be configured to point to any locations inside the session router address map. This register is useful when the PRU needs to frequently access certain structures inside the session router address space whose locations are not hard coded such as tables in scratchpad memory. This register is formatted as follows:

Bits	Field	Type	Reset	Description
31:16	C29	r/w	0	PRU Constant Entry 29 Pointer: This field sets the value that will appear in bits 23:8 of entry 29 in the PRU Constant Table
15:0	C28	r/w	0	PRU Constant Entry 28 Pointer: This field sets the value that will appear in bits 23:8 of entry 28 in the PRU Constant Table

PRU Constant Table Programmable Pointer Register 1 (0x002C)

This register functions the same as the PRU Constant Table Programmable Pointer Register 0 but allows the PRU to control entries 30 and 31 in the PRU Constant Table. This register is formatted as follows:

Bits	Field	Type	Reset	Description
31:16	C31	r/w	0	PRU Constant Entry 31 Pointer: This field sets the value that will appear in bits 23:8 of entry 31 in the PRU Constant Table
15:0	C30	r/w	0	PRU Constant Entry 30 Pointer: This field sets the value that will appear in bits 23:8 of entry 30 in the PRU Constant Table

PRU Internal General Purpose Register n (0x0400 + 4*n)

This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written. This register is formatted as follows:

Bits	Field	Type	Reset	Description
31:0	INTGPRn	r/w	x	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile.

PRU Internal Constants Table Entry Register n (0x0480 + 4*n)

This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table. This register is formatted as follows:

Bits	Field	Type	Reset	Description
31:0	INTCTERn	r	x	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.

PRU Instruction RAM Region

The instruction RAM region contains storage for instructions. The instruction RAM region memory map is as follows:

Instruction RAM Memory Region

Address Offset	Register
0x0000	PRU Instruction RAM

PRU Instruction RAM (0x0000)

A total of 1K 32-bit words of instruction storage are provided for the PRU. This memory is to be initialized by an external (to the PRUSS) host processor. This region is only accessible to external masters when the PRU is not-running.

- Note that the instruction RAM is accessed using byte addresses on the bus. The PC is an instruction address where each instruction is a 32 bit word and is not a byte address. To compute the byte address just multiply the PC by 4 (PC of 2 = byte address of 0x8, or PC of 8 = byte address of 0x20), or to compute the PC just divide the word aligned byte address by 4 (byte address of 0x8 = PC of 2, byte address of 0x20 = PC of 8).

Constants Table

The PRU Constants Table is a structure connected to a dedicated interface on the PRU core within the PRU that is used to provide the base address for the Load Burst Constant + Offset (LBCO) and Store Burst Constant + Offset (SBCO) instructions. The PRU Constants Table is provided in order to maximize the usage of the PRU register file for embedded processing applications by moving many of the commonly used constant or deterministically calculated base addresses from the internal register file to an external table. Since this table is accessed using a dedicated interface, no performance difference is realized between the LBCO and LBBO or SBCO and SBBO instructions. The constants in the table are provided in Table 2.

Table 2: Constants Table

Entry #	Region Pointed To	Value [31:0]
0	PRU0/1 Local INTC	0x00004000
1	Timer64P0	0x01C20000
2	I2C0	0x01C22000
3	PRU0/1 Local Data	0x00000000
4	PRU1/0 Local Data	0x00002000
5	MMC/SD	0x01C40000
6	SPI0	0x01C41000
7	UART0	0x01C42000
8	McASP0 DMA	0x01D02000
9*	*RESERVED*	0x01D06000
10*	*RESERVED*	0x01D0A000
11	UART1	0x01D0C000
12	UART2	0x01D0D000
13	USB0	0x01E00000
14	USB1	0x01E25000
15	UHPI Config	0x01E10000
16*	*RESERVED*	0x01E12000
17	I2C1	0x01E28000
18	EPWM0	0x01F00000
19	EPWM1	0x01F02000
20*	*RESERVED*	0x01F04000
21	ECAP0	0x01F06000
22	ECAP1	0x01F07000
23	ECAP2	0x01F08000
24	PRU0/1 Local Data	0x00000n00, n=c24_blk_index[3:0]
25	McASP0 Control	0x01D00n00, n=c25_blk_index[3:0]
26*	*RESERVED*	0x01D04000
27*	*RESERVED*	0x01D08000
28	DSP Megamodule RAM/ROM	0x11nnnn00, nnnn=c28_pointer[15:0]
29	EMIFA SDRAM	0x40nnnn00, nnnn=c29_pointer[15:0]
30	Shared RAM	0x80nnnn00, nnnn=c30_pointer[15:0]
31	mDDR/DDR2 Data	0xC0nnnn00, nnnn=c31_pointer[15:0]

* These constants cannot be used due to memory map restrictions.

Notes:

1. Constants not in this table can be created 'on the fly' by two consecutive LDI #16 instructions. These constants are just ones that are expected to be commonly used, enough so to be hard-coded into the PRU constants table.
2. Constants table entries 24 through 31 are not fully hard-coded, but contain a programmable bitfield (ex. c24_blk_index[3:0]) that is programmable through the PRU control register space (0x01C3_7000 - 0x01C3_73FF for PRU0 and 0x01C3_7800 - 0x01C3_7BFF for PRU1).

PRU Module Interface

Event out Mapping (R31): PRU System Events

PRU Event Interface directly feeds pulsed event information out of the PRU's internal ALU. This provides the interface logic between the PRU R31 event_out[5:0] and the system interrupts for the PRUSS Interrupt controller(INTC).

Bits	Field Name	Description
31:6	RESERVED	

5	pruX_vec_valid	Valid strobe for vector output
4:0	pruX_vec[4:0]	Vector output

Writing a '1' to pruX_vec_valid (R31 bit 5) simultaneously with a channel number from 0 to 31 written to pruX_vec[4:0] (R31 bits 4:0) creates a pulse on the output of the appropriate de-multiplexer output. For example, writing '100000' will generate a pulse on demux channel 0, writing '100001' will generate a pulse on demux channel 1, ... writing '111111' will generate a pulse on demux channel 31, and writing 'oxxxxx' will not generate any pulse on the demux output. The demultiplexed values from both of the PRUs are logically ORed together. The composite demultiplexed output channels 0 through 31 are connected to system interrupts 32 through 63 respectively.

This allows the PRU to assert one of the systems interrupts 32-63 by writing to its own R31 register. The system interrupt is used to either post a completion event to one of the host CPUs (ARM, DSP) or to signal the other PRU. The host to be signaled is determined by the system interrupt to interrupt channel mapping (programmable). Refer to the section on the PRUSS Interrupt Controller (INTC) for more details.

Status Mapping (R31): Interrupt Events Input

The PRU Real Time Status Interface directly feeds information into register 31(R31) of the PRU's internal register file. The firmware on the PRU uses the status information to make decisions during execution. The status interface is comprised of signals from different modules inside of the PRUSS which require some level of interaction with the PRU. More details on the Host interrupts imported into bit 30 and 31 of register R31 of both the PRUs is provided in the chapter 3, PRUSS Interrupt Controller (INTC).

Bits	Name	Description
31	pru_intr_in[1]	PRU Interrupt 1 from INTC
30	pru_intr_in[0]	PRU Interrupt 0 from INTC
29:0	pruX_r31_status[29:0]	Status inputs from primary input

General Purpose Inputs (R31)

The pruX_r31_status[29:0] are mapped out of the PRUSS and are brought out as general purpose input pins. The values input to the pins "pruX_R31[29:0]" are reflected in the R31 register on bits [29:0] to be used by the program running on the PRU. Each PRU of the PRUSS has a separate mapping to pins, so that there are 60 total general purpose inputs to the PRUSS.

General Purpose Outputs (R30)

The pruX_r30[31:0] bits are exported out of the PRUSS and are brought out as general purpose output pins. The values written to register R30 will be reflected on the general purpose output pins "pruX_R30[31:0]" to be used by the program running on the PRU. Each PRU of the PRUSS has a separate mapping to pins, so that there are 64 total general purpose outputs from the PRUSS.

Instruction Set

The instruction set is divided into four major categories:

1. Instructions which move data in or out of the processors internal registers
2. Instructions which perform an arithmetic operation
3. Instructions which perform a logical operation
4. Instructions which control program flow

The following sections give a complete list and short description of all of the supported instructions. In these descriptions, the following abbreviations are used:

Table 3: Abbreviations for Instruction Descriptions

Abbreviation	Description
Rs1	Source register 1 from the instruction
Op2	Operand 2 from the instruction – can be either a register (Rs2) or an 8-bit immediate value
Rd	Destination register from the instruction
BrOff	Branch offset from the instruction – a 10-bit 2's complement relative offset
WdCnt	Word count – the # of 32-bit data phases that occur in the burst on an external memory interface
CPI	Clock Cycles Per Instruction

Load / Store Instructions

Table 4: Load/Store Instructions

Mnemonic	Instruction	Description	CPI
LDI	Load Immediate	Load 16-bit immediate value into internal register	1
LBBO	Load Burst,Base + Offset	Load variable length burst of bytes through one of the memory interfaces into internal register(s) using a register as the base address and a register or an 8-bit immediate as the offset	1 + WdCnt (VBUS) 2 + WdCnt (VBUSP)
SBBO	Store Burst, Base + Offset	Store variable length burst of bytes through one of the memory interfaces from internal register(s) using a register as the base address and a register or 8-bit immediate as the offset	1 + WdCnt

LBCO	Load Burst,Constant + Offset	Load variable length burst of bytes through one of the memory interfaces into internal register(s) using an indexed constant as the base address and a register or an 8-bit immediate as the offset	1 + WdCnt (VBUS) 2 + WdCnt (VBUSP)
SBCO	Store Burst, Constant + Offset	Store variable length burst of bytes through one of the memory interfaces from internal register(s) using an indexed constant as the base address and a register or 8-bit immediate as the offset	1 + WdCnt

Arithmetic Instructions

Table 5: Arithmetic Instructions

Mnemonic	Instruction	Description	CPI
ADD	Integer Add	Adds Rs1 and Op2, writes result to Rd, and saves carry.	1
ADC	Integer Add With Carry	Adds Rs1, Op2 and the saved carry, writes result to Rd, and saves carry.	1
SUB	Integer Subtract	Subtracts Op2 from Rs1 and writes result to Rd and saves carry (borrow).	1
SUC	Integer Subtract With Carry	Subtracts Op2 from Rs1 then subtracts the saved carry (borrow) and writes result to Rd and saves carry (borrow).	1
RSB	Integer Reverse Subtract	Subtracts Rs1 from Op2 and writes result to Rd and saves carry (borrow).	1
RSC	Integer Reverse Subtract With Carry	Subtracts Rs1 from Op2 then subtracts the saved carry (borrow) and writes result to Rd and saves carry (borrow).	1

Logical Instructions

Table 6: Logical Instructions

Mnemonic	Instruction	Description	CPI
AND	Bitwise And	Bitwise ANDs Rs1 with Op2 and stores to Rd.	1
OR	Bitwise Or	Bitwise ORs Rs1 with Op2 and stores to Rd.	1
XOR	Bitwise Exclusive Or	Bitwise exclusive ORs Rs1 with Op2 and stores to Rd.	1
NOT	Bitwise Invert	Bitwise inverts Rs1 and stores to Rd.	1
LSR	Logical Shift Right	Shifts Rs1 right (with zero fill) by the value given in the 5 LSBs of Op2 and stores to Rd.	1
LSL	Logical Shift Left	Shifts Rs1 left (with zero fill) by the value given in the 5 LSBs of Op2 and stores to Rd.	1
MIN	Minimum	Compares Rs1 and Op2 and the smaller value is copied to Rd.	1
MAX	Maximum	Compares Rs1 and Op2 and the larger value is copied to Rd.	1
CLR	Clear Bit	Copies Rs1 to Rd but with a bit specied by the 5 LSBs of Op2 cleared during the copy.	1
SET	Set Bit	Copies Rs1 to Rd but with a bit specied by the 5 LSBs of Op2 set during the copy.	1
LMBD	Left-most Bit Detect	Scans Rs1 from the leftmost bit for a bit equal to bit 0 of Rs2. When found, the bit number (0 to 31) is written to Rd. If not found, the value 32 is written to Rd.	1
SCAN	Scan Register File	Scans the register file for a byte pattern of a programmable length (up to 4 bytes) with a programmable field count and field stride. The Op1 register contains 4 fields: Rn.b0 = offset in the register file from R0.b0 to start scanning Rn.b1 = fc (field count), the number of fields to scan Rn.b2 = fw (field width), the size in bytes of the field to scan for (1, 2, or 4 bytes) Rn.b3 = fs (field stride), the number of bytes to advance to the next field in the register file (1 to 4 bytes) Op1 is updated after the scan with the offset of the match (or 0xFF if no match) in Rn.b0, and the fields remaining in the scan (including the matching field) in Rn.b1. The Op2 is the field to scan for.	IF fw = fs 2+((fc*fw+3)/4) ELSE 2+fc This is a worst case cycle count. Matching scans could take fewer cycles.

Program Flow Control Instructions

Table 7: Program Flow Control Instructions

Mnemonic	Instruction	Description	CPI
QBBS	Quick Branch – Bit Set	Adds BrOff to the program counter if the bit in Rs1 specified by the 5 LSBs of Op2 is a 1	1
QBBC	Quick Branch – Bit Clear	Adds BrOff to the program counter if the bit in Rs1 specified by the 5 LSBs of Op2 is a 0	1
QBGT	Quick Branch – Greater Than	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is greater than Rs1.	1
QBGE	Quick Branch – Greater Than or Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is greater than or equal to Rs1.	1
QBLT	Quick Branch – Less Than	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is less than Rs1.	1
QBLE	Quick Branch – Less Than or Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is less than or equal to Rs1.	1

QBEQ	Quick Branch – Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is equal to Rs1.	1
QBNE	Quick Branch – Not Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is not equal to Rs1.	1
JMP	Unconditional Jump	Sets the program counter equal to either the contents of a register or to a 16-bit immediate value	1
JAL	Unconditional Jump and Link	Saves the current program counter into Rd and sets the program counter equal to either the contents of a register or to a 16-bit immediate value.	1
HALT	Halt Processor	Disables the PRU and does not increment the program counter. When the PRU is re-enabled, it will continue processing at this instruction.	1
SLP	Sleep	Pauses execution of the current program and disables the clock for the majority of the core until a specified external event (un-masked status bit) is asserted.	1 to infinity

Instruction Formats

A total of 7 different instruction formats are supported for the various operations. The following sections describe the position, size, and function of each of the fields within the various formats

Format 1a: (All Arithmetic and Logical Functions – Register Op2)

Bits	Name	Value	Meaning
31:29	OP	0b000	Specifies Format 1
28:25	ALUOP	0	ADD
		1	ADC
		2	SUB
		3	SUC
		4	LSL
		5	LSR
		6	RSB
		7	RSC
		8	AND
		9	OR
		10	XOR
		11	NOT
		12	MIN
		13	MAX
		14	CLR
		15	SET
24	IO	0	Op2 is a register
23:21	Rs2Sel	0	Select bits 7:0 from the source register 2
		1	Select bits 15:8 from the source register 2
		2	Select bits 23:16 from the source register 2
		3	Select bits 31:24 from the source register 2
		4	Select bits 15:0 from the source register 2
		5	Select bits 23:8 from the source register 2
		6	Select bits 31:16 from the source register 2
		7	Select bits 31:0 from the source register 2
20:16	Rs2	0 – 31	This field selects the register number which contains the second source operand
15:13	Rs1Sel	0	Select bits 7:0 from the source register 1
		1	Select bits 15:8 from the source register 1
		2	Select bits 23:16 from the source register 1
		3	Select bits 31:24 from the source register 1
		4	Select bits 15:0 from the source register 1
		5	Select bits 23:8 from the source register 1
		6	Select bits 31:16 from the source register 1
		7	Select bits 31:0 from the source register 1
12:8	Rs1	0 – 31	This field selects the register number which contains the first source operand
7:5	RdSel	0	Select bits 7:0 of the destination register
		1	Select bits 15:8 of the destination register
		2	Select bits 23:16 of the destination register
		3	Select bits 31:24 of the destination register
		4	Select bits 15:0 of the destination register
		5	Select bits 23:8 of the destination register

		6	Select bits 31:16 of the destination register
		7	Select bits 31:0 of the destination register
4:0	Rd	0 – 31	This field selects the destination register number to which the result should be written.

Format 1b: (All Arithmetic and Logical Functions – Immediate Op2)

Bits	Name	Value	Meaning
31:29	OP	0b000	Specifies Format 1
28:25	ALUOP	0	ADD
		1	ADC
		2	SUB
		3	SUC
		4	LSL
		5	LSR
		6	RSB
		7	RSC
		8	AND
		9	OR
		10	XOR
		11	NOT
		12	MIN
		13	MAX
		14	CLR
		15	SET
24	IO	1	Op2 is an 8-bit immediate
23:16	Imm2	0 – 255	This field is the 8-bit immediate value to be used as the source operand 2.
15:13	Rs1Sel	0	Select bits 7:0 from the source register 1
		1	Select bits 15:8 from the source register 1
		2	Select bits 23:16 from the source register 1
		3	Select bits 31:24 from the source register 1
		4	Select bits 15:0 from the source register 1
		5	Select bits 23:8 from the source register 1
		6	Select bits 31:16 from the source register 1
		7	Select bits 31:0 from the source register 1
12:8	Rs1	0 – 31	This field selects the register number which contains the first source operand
7:5	RdSel	0	Select bits 7:0 of the destination register
		1	Select bits 15:8 of the destination register
		2	Select bits 23:16 of the destination register
		3	Select bits 31:24 of the destination register
		4	Select bits 15:0 of the destination register
		5	Select bits 23:8 of the destination register
		6	Select bits 31:16 of the destination register
		7	Select bits 31:0 of the destination register
4:0	Rd	0 – 31	This field selects the destination register number to which the result should be written.

Format 2

The following represents the 7 most significant bits of all format 2 instructions:

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOB	0	JMP
		1	JAL
		2	LDI
		3	LMBD
		4	SCAN
		5	HALT
		6	*currently reserved for MVIx*

		7 - 13	RESERVED
		14	*currently reserved for RFI – Return From Interrupt*
		15	SLP – Sleep

Format 2a: (JMP,JAL – Register Op2)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOP	0	JMP
		1	JAL
24	IO	0	Op2 is a register
23:21	Rs2Sel	0	Select bits 7:0 from the source register 2
		1	Select bits 15:8 from the source register 2
		2	Select bits 23:16 from the source register 2
		3	Select bits 31:24 from the source register 2
		4	Select bits 15:0 from the source register 2
		5	Select bits 23:8 from the source register 2
		6	Select bits 31:16 from the source register 2
		7	Select bits 31:0 from the source register 2
20:16	Rs2	0 – 31	This field selects the register number which contains the value to be copied to the program counter.
15:8	RESERVED	0	
7:5	RdSel	0	Select bits 7:0 of the destination register
		1	Select bits 15:8 of the destination register
		2	Select bits 23:16 of the destination register
		3	Select bits 31:24 of the destination register
		4	Select bits 15:0 of the destination register
		5	Select bits 23:8 of the destination register
		6	Select bits 31:16 of the destination register
		7	Select bits 31:0 of the destination register
4:0	Rd	0 – 30	This field selects the destination register number to which the pre-incremented program counter should be written. Only written for JAL.

Format 2b: (JMP,JAL – Immediate Op2)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOP	0	JMP - Jump
		1	JAL – Jump and Link
24	IO	1	Jump operand is a 16-bit immediate value
23:8	Imm	0x0– 0xFFFF	This field is the 16-bit immediate value to be copied to the program counter.
7:5	RdSel	0	Select bits 7:0 of the destination register
		1	Select bits 15:8 of the destination register
		2	Select bits 23:16 of the destination register
		3	Select bits 31:24 of the destination register
		4	Select bits 15:0 of the destination register
		5	Select bits 23:8 of the destination register
		6	Select bits 31:16 of the destination register
		7	Select bits 31:0 of the destination register
4:0	Rd	0 – 30	This field selects the destination register number to which the pre-incremented program counter should be written. Only written for JAL.

Format 2c: (LDI)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOP	2	Specifies LDI
24:24	reserved	0	

23:8	Imm	0x0–0xFFFF	This field is the 16-bit immediate value to be used as the source operand 2.
7:5	RdSel	0	Select bits 7:0 of the destination register
		1	Select bits 15:8 of the destination register
		2	Select bits 23:16 of the destination register
		3	Select bits 31:24 of the destination register
		4	Select bits 15:0 of the destination register
		5	Select bits 23:8 of the destination register
		6	Select bits 31:16 of the destination register
		7	Select bits 31:0 of the destination register
4:0	Rd	0 – 31	This field selects the destination register number to which the result should be written.

Format 2d: (LMBD - Leftmost Bit Detect - Register Op2)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOB	3	Specifies LMBD
24	IO	0	Op2 is a register
23:21	Rs2Sel	0	Select bits 7:0 from Rs2
		1	Select bits 15:8 from Rs2
		2	Select bits 23:16 from Rs2
		3	Select bits 31:24 from Rs2
		4	Select bits 15:0 from Rs2
		5	Select bits 23:8 from Rs2
		6	Select bits 31:16 from Rs2
		7	Select bits 31:0 from Rs2
20:16	Rs2	0-31	Rs2 register number 0-31
15:13	Rs1Sel	0	Select bits 7:0 from Rs1
		1	Select bits 15:8 from Rs1
		2	Select bits 23:16 from Rs1
		3	Select bits 31:24 from Rs1
		4	Select bits 15:0 from Rs1
		5	Select bits 23:8 from Rs1
		6	Select bits 31:16 from Rs1
		7	Select bits 31:0 from Rs1
12:8	Rs1	0-31	Rs1 register number 0-31
7:5	RdSel	0	Select bits 7:0 from Rd
		1	Select bits 15:8 from Rd
		2	Select bits 23:16 from Rd
		3	Select bits 31:24 from Rd
		4	Select bits 15:0 from Rd
		5	Select bits 23:8 from Rd
		6	Select bits 31:16 from Rd
		7	Select bits 31:0 from Rd
4:0	Rd	0-31	Rd register number 0-31

Format 2e: (LMBD - Immediate Op2)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOB	3	Specifies LMBD
24	IO	1	Op2 is an 8 bit immediate
23:16	Imm2	0-255	Immediate for src2
15:13	Rs1Sel	0	Select bits 7:0 from Rs1
		1	Select bits 15:8 from Rs1
		2	Select bits 23:16 from Rs1
		3	Select bits 31:24 from Rs1
		4	Select bits 15:0 from Rs1
		5	Select bits 23:8 from Rs1
		6	Select bits 31:16 from Rs1
		7	Select bits 31:0 from Rs1

12:8	Rs1	0-31	Rs1 register number 0-31
7:5	RdSel	0	Select bits 7:0 from Rd
		1	Select bits 15:8 from Rd
		2	Select bits 23:16 from Rd
		3	Select bits 31:24 from Rd
		4	Select bits 15:0 from Rd
		5	Select bits 23:8 from Rd
		6	Select bits 31:16 from Rd
		7	Select bits 31:0 from Rd
4:0	Rd	0-31	Rd register number 0-31

Format 2f: (SCAN - Register Op2)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOB	4	Specifies SCAN
24	IO	0	Op2 is a register
23:21	Rs2Sel	0	Select bits 7:0 from Rs2
		1	Select bits 15:8 from Rs2
		2	Select bits 23:16 from Rs2
		3	Select bits 31:24 from Rs2
		4	Select bits 15:0 from Rs2
		5	Select bits 23:8 from Rs2
		6	Select bits 31:16 from Rs2
		7	Select bits 31:0 from Rs2
20:16	Rs2	0-31	Rs2 register number 0-31
15:13	Rs1Sel	7	Select bits 31:0 from Rs1
12:8	Rs1	0-31	*Must be identical to Rd*
7:5	RdSel	7	Select bits 31:0 from Rd
4:0	Rd	0-31	Rd register number 0-31

Format 2g: (SCAN - Immediate Op2)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOB	4	Specifies SCAN
24	IO	1	Op2 is an 8 bit immediate
23:16	Imm2	0-255	Immediate for src2
15:13	Rs1Sel	7	Select bits 31:0 from Rs1
12:8	Rs1	0-31	*Must be identical to Rd*
7:5	RdSel	7	Select bits 31:0 from Rd
4:0	Rd	0-31	Rd register number 0-31

Format 2h: (HALT)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOB	5	Specifies HALT
24:0	reserved	0	

Format 2i: (SLP)

Bits	Name	Value	Meaning
31:29	OP	0b001	Specifies Format 2
28:25	SUBOP	15	SLP – Sleep
24	RESERVED		
23	WakeOnStatus	0	Do not wake when non-masked status is asserted
		1	Wake when non-masked status is asserted

22:0	RESERVED		

Format 4a: (Quick Arithmetic Test and Branch – Register Op2)

Bits	Name	Value	Meaning
31:30	OP	0b01	Specifies Format 4
29	GT	0 – 1	Greater Than: If set specifies that branch should be taken if Op2 > Rs1*
28	EQ	0 – 1	Equal: If set specifies that branch should be taken if Op2 == Rs1*
27	LT	0 – 1	Less Than: If set specifies that branch should be taken if Op2 < Rs1*
26:25	BrOff[9:8]	0 - 3	This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	0	Op2 is a register
23:21	Rs2Sel	0	Select bits 7:0 from the source register 2
		1	Select bits 15:8 from the source register 2
		2	Select bits 23:16 from the source register 2
		3	Select bits 31:24 from the source register 2
		4	Select bits 15:0 from the source register 2
		5	Select bits 23:8 from the source register 2
		6	Select bits 31:16 from the source register 2
		7	Select bits 31:0 from the source register 2
20:16	Rs2	0 – 31	This field selects the register number which contains the second source operand that is to be compared.
15:13	Rs1Sel	0	Select bits 7:0 from the source register 1
		1	Select bits 15:8 from the source register 1
		2	Select bits 23:16 from the source register 1
		3	Select bits 31:24 from the source register 1
		4	Select bits 15:0 from the source register 1
		5	Select bits 23:8 from the source register 1
		6	Select bits 31:16 from the source register 1
		7	Select bits 31:0 from the source register 1
12:8	Rs1	0 – 31	This field selects the register number which contains the first source operand that is to be compared.
7:0	BrOff[7:0]	0 - 255	This field contains the 8-LSBs of the 2-s complement signed offset for the branch
*Branch is taken if any of the 3 conditions are satisfied: GT, EQ or LT.			

Format 4b: (Quick Arithmetic Test and Branch – Immediate Op2)

Bits	Name	Value	Meaning
31:30	OP	0b01	Specifies Format 4
29	GT	0 – 1	Greater Than: If set specifies that branch should be taken if Op2 > Rs1*
28	EQ	0 – 1	Equal: If set specifies that branch should be taken if Op2 == Rs1*
27	LT	0 – 1	Less Than: If set specifies that branch should be taken if Op2 < Rs1*
26:25	BrOff[9:8]	0 - 3	This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	1	Op2 is an 8-bit immediate value
23:16	Imm	0 - 255	8-bit immediate value to be used as second operand to be compared.
15:13	Rs1Sel	0	Select bits 7:0 from the source register 1
		1	Select bits 15:8 from the source register 1
		2	Select bits 23:16 from the source register 1
		3	Select bits 31:24 from the source register 1
		4	Select bits 15:0 from the source register 1
		5	Select bits 23:8 from the source register 1
		6	Select bits 31:16 from the source register 1
		7	Select bits 31:0 from the source register 1
12:8	Rs1	0 – 31	This field selects the register number which contains the first source operand that is to be compared.
7:0	BrOff[7:0]	0 - 255	This field contains the 8-LSBs of the 2-s complement signed offset for the branch
*Branch is taken if any of the 3 conditions are satisfied: GT, EQ or LT.			

Format 5a: (Quick Bit Test and Branch – Register Op2)

Bits	Name	Value	Meaning
31:29	OP	0b110	Specifies Format 5
28	BS	0 – 1	Bit Set: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 1*

27	BC	0 – 1	Bit Clear: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 0*
26:25	BrOff[9:8]	0 - 3	This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	0	Op2 is a register
23:21	Rs2Sel	0	Select bits 7:0 from the source register 2
		1	Select bits 15:8 from the source register 2
		2	Select bits 23:16 from the source register 2
		3	Select bits 31:24 from the source register 2
		4	Select bits 15:0 from the source register 2
		5	Select bits 23:8 from the source register 2
		6	Select bits 31:16 from the source register 2
		7	Select bits 31:0 from the source register 2
20:16	Rs2	0 – 31	This field selects the register number which contains the bit number of the first operand which is to be compared.
15:13	Rs1Sel	0	Select bits 7:0 from the source register 1
		1	Select bits 15:8 from the source register 1
		2	Select bits 23:16 from the source register 1
		3	Select bits 31:24 from the source register 1
		4	Select bits 15:0 from the source register 1
		5	Select bits 23:8 from the source register 1
		6	Select bits 31:16 from the source register 1
		7	Select bits 31:0 from the source register 1
12:8	Rs1	0 – 31	This field selects the register number which contains the first source operand that is to be compared.
7:0	BrOff[7:0]	0 - 255	This field contains the 8-LSBs of the 2-s complement signed offset for the branch
*Branch is taken if any of the 2 conditions are satisfied: BS or BC.			

Format 5b: (Quick Bit Test and Branch – Immediate Op2)

Bits	Name	Value	Meaning
31:29	OP	0b110	Specifies Format 5
28	BS	0 – 1	Bit Set: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 1*
27	BC	0 – 1	Bit Clear: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 0*
26:25	BrOff[9:8]	0 - 3	This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	1	Op2 is a 5-bit immediate value
23:21	RESERVED	0	
20:16	Imm	0 – 31	This field selects the bit number of the first operand which is to be compared.
15:13	Rs1Sel	0	Select bits 7:0 from the source register 1
		1	Select bits 15:8 from the source register 1
		2	Select bits 23:16 from the source register 1
		3	Select bits 31:24 from the source register 1
		4	Select bits 15:0 from the source register 1
		5	Select bits 23:8 from the source register 1
		6	Select bits 31:16 from the source register 1
		7	Select bits 31:0 from the source register 1
12:8	Rs1	0 – 31	This field selects the register number which contains the first source operand that is to be compared.
7:0	BrOff[7:0]	0 - 255	This field contains the 8-LSBs of the 2-s complement signed offset for the branch
*Branch is taken if any of the 2 conditions are satisfied: BS or BC.			

Format 6a: (LBBO/SBBO - Register Offset)

Bits	Name	Value	Meaning
31:29	OP	0b111	Specifies Format 6a / 6b
28	LoadStore	0	SBBO
		1	LBBO
27:25	BurstLen[6:4]		The following 3 fields specify the burst length (in bytes) for the transfer.
15:13	BurstLen[3:1]		
7	BurstLen[0]	0-123	byte count = BurstLen + 1 (1 – 124 Bytes)
		124	byte count = R0 bits 7:0
		125	byte count = R0 bits 15:8
		126	byte count = R0 bits 23:16
		127	byte count = R0 bits 31:24
24	IO	0	The offset is to be taken from a register

23:21	RoSel	0	Select bits 7:0 from the offset register
		1	Select bits 15:8 from the offset register
		2	Select bits 23:16 from the offset register
		3	Select bits 31:24 from the offset register
		4	Select bits 15:0 from the offset register
		5	Select bits 23:8 from the offset register
		6	Select bits 31:16 from the offset register
		7	Select bits 31:0 from the offset register
20:16	Ro	0 – 31	This field selects the register number which contains the beginning offset for the transfer.
12:8	Rb	0 – 31	This field selects the register number which contains the base address for the transfer
6:5	RxByteAddr	0 - 3	This field selects the beginning byte number in the source / destination register for the data transfer.
4:0	Rx	0 – 30	This field selects the beginning source / destination register number for the data transfer.

Format 6b: (LBBO/SBBO - Immediate Offset)

Bits	Name	Value	Meaning
31:29	OP	0b111	Specifies Format 6a / 6b
28	LoadStore	0	SBBO
		1	LBBO
27:25	BurstLen[6:4]		The following 3 fields specify the burst length (in bytes) for the transfer.
15:13	BurstLen[3:1]		
7	BurstLen[0]	0-123	byte count = BurstLen +1 (1 – 124 Bytes)
		124	byte count = R0 bits 7:0
		125	byte count = R0 bits 15:8
		126	byte count = R0 bits 23:16
		127	byte count = R0 bits 31:24
24	IO	1	The offset is an immediate 8-bit value
23:16	Imm	0 - 255	Immediate 8-bit offset value
12:8	Rb	0 – 31	This field selects the register number which contains the base address for the transfer
6:5	RxByteAddr	0 - 3	This field selects the beginning byte number in the source / destination register for the data transfer.
4:0	Rx	0 – 30	This field selects the beginning source / destination register number for the data transfer.

Format 6c: (LBCO/SBCO - Register Offset)

Bits	Name	Value	Meaning
31:29	OP	0b100	Specifies Format 6c / 6d
28	LoadStore	0	SBCO
		1	LBCO
27:25	BurstLen[6:4]		The following 3 fields specify the burst length (in bytes) for the transfer.
15:13	BurstLen[3:1]		
7	BurstLen[0]	0-123	byte count = BurstLen + 1 (1 – 124 Bytes)
		124	byte count = R0 bits 7:0
		125	byte count = R0 bits 15:8
		126	byte count = R0 bits 23:16
		127	byte count = R0 bits 31:24
24	IO	0	The offset is to be taken from a register
23:21	RoSel	0	Select bits 7:0 from the offset register
		1	Select bits 15:8 from the offset register
		2	Select bits 23:16 from the offset register
		3	Select bits 31:24 from the offset register
		4	Select bits 15:0 from the offset register
		5	Select bits 23:8 from the offset register
		6	Select bits 31:16 from the offset register
		7	Select bits 31:0 from the offset register
20:16	Ro	0 – 31	This field selects the register number which contains the beginning offset for the transfer.
12:8	Cb	0 – 31	This field selects the constant table entry number which contains the base address for the transfer
6:5	RxByteAddr	0 - 3	This field selects the beginning byte number in the source / destination register for the data transfer.
4:0	Rx	0 – 30	This field selects the beginning source / destination register number for the data transfer.

Format 6d: (LBCO/SBCO - Immediate Offset)

Bits	Name	Value	Meaning
31:29	OP	0b100	Specifies Format 6c / 6d
28	LoadStore	0	SBCO
		1	LBCO
27:25	BurstLen[6:4]		The following 3 fields specify the burst length (in bytes) for the transfer.
15:13	BurstLen[3:1]		
7	BurstLen[0]	0-123	byte count = BurstLen + 1 (1 – 124 Bytes)
		124	byte count = R0 bits 7:0
		125	byte count = R0 bits 15:8
		126	byte count = R0 bits 23:16
		127	byte count = R0 bits 31:24
24	IO	1	The offset is an immediate 8-bit value
23:16	Imm	0 - 255	Immediate 8-bit offset value
12:8	Cb	0 – 31	This field selects the constant table entry number which contains the base address for the transfer
6:5	RxByteAddr	0 - 3	This field selects the beginning byte number in the source / destination register for the data transfer.
4:0	Rx	0 – 30	This field selects the beginning source / destination register number for the data transfer.

Return to Subsystem Documentation

Click here.

{{

1. switchcategory:MultiCore=
   - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
   - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **Programmable Realtime Unit** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **Programmable Realtime Unit** here.

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***Programmable Realtime Unit*** *here.*

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***Programmable Realtime Unit*** *here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***Programmable Realtime Unit*** *here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***Programmable Realtime Unit*** *here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***Programmable Realtime Unit*** *here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article* ***Programmable Realtime Unit*** *here.*

Fo pl qu ht Pl co ar ***R***

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity