Preliminary

C28x Emulation Technical Reference Manual

User's Guide

Literature Number: SPRUF82
August 2008

Contents

List of Figures

List of Tables

C28x Emulation Features

This document describes the software aspects of the debug function for the C28x emulation environment.

1.1 Debug Overview

The on-chip development interface provides:

- Non-intrusive access to internal and external memory
- Minimally-intrusive access to CPU and peripheral registers
- Code-execution control
  - Break on a software breakpoint instruction (instruction replacement)
  - Break after the execution of a single instruction (single-stepping)
  - CPU halt, execute from any instruction location

The C28x on-chip development interface is based on the IEEE Std. 1149.1, 'Standard Test Access Port and Boundary-Scan Architecture", often referred to as the JTAG interface. Please refer to the IEEE standard for an explanation of the JTAG terms used in this chapter and for a description of the TAP controller states.

Emulation Functionality

2.1 Execution Control

Stop-mode halts execution of all code at break events, such as software breakpoint instructions, or host requests. Stop-mode provides complete control of program execution, allowing the disabling of all interrupts (including those which are non-maskable), and reset.

2.1.1 Stop-mode Execution Control

Stop-mode causes break events to suspend program execution at the next interrupt boundary (generally identical to the next instruction boundary). When execution is suspended, all interrupts (including reset and non-maskable interrupts) are ignored until the CPU receives a debug run directive. If an interrupt occurs simultaneously with a debug event, the debug event has priority. However, after the processing of an interrupt has begun, the CPU cannot process a debug event until the first instruction boundary following one of these events.

2.1.2 User Execution Control

The debug user interface 'Step' and 'Run' commands differ in that the 'Run' command grants interrupts and resets higher priority over the execution of the next instruction, whereas 'Step' does not.

To repeat some of the key concepts:

- Performing a 'Run': if there are enabled interrupts outstanding and the global interrupt flag (INTM) is enabled, the highest priority interrupt will be executed, otherwise, the instruction pointed to by the PC will be executed. For an executed interrupt, the interrupt context switch is counted as 1 instruction; hence the PC will be pointing to the initial instruction of the ISR.
- Performing a 'Step'; the instruction pointed to by the PC will be performed.

2.1.3 Execution-Control State Machine

The C28x has an execution-control state machine with the following states:

EXE_CONT— Execute instructions as if the emulator is disconnected. Break events are ignored.

EXE_COND— Execute instructions until a break event occurs.

EXE_STEP— Execute a single instruction, then suspend background execution.

EXE_HALT— Suspend background execution.

One of these four states will always be the desired execution state of C28x. Note that debug actions would not be performed in EXE_CONT, but could be in any of the other states. Debug actions which check status and control the execution state machine can be performed in any state.

2.2 Pipeline Behavior During Debug

Pipeline assumptions:

- No changes to registers or memory occur before the Decode 2 stage of the pipeline. Instructions which have been fetched into the Fetch 1, Fetch 2 or Decode 1 stages can be discarded and later re-fetched without changing the state of the programmer's model.
- Changes to registers or memory which occur within the Decode 2 and later stages (Read 1, Read 2, Execute, Write 1, and Write 2) do not rely on instructions prior to the Decode 2 stage (i.e. the contents of stages prior to Decode 2 could be discarded due to an interrupt or a branch).
- The act of discarding the contents of pipeline stages prior to Decode 2 while allowing the Decode 2 and later pipeline stages to complete is referred to as flushing the pipeline.
- The IC register (as displayed by the debugger) generally holds the address of the instruction in the Decode 1 stage of the pipeline. This is the instruction to be 'performed" next (assuming no instruction jam occurs).

Figure 2-1. Pipeline During Jam

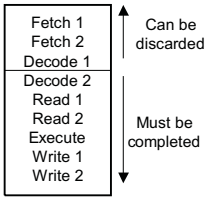


Table 2-1. Halting Due To Software Breakpoint (ESTOP0)[(1)]

Fetch1	ESTOP	D	E	F	..	..	..	..	..
Fetch2	C	ESTOP	D	E	F	..	..	..	..
1-Dec	B	C	ESTOP	D	E	F	..	..	..
2-Dec	A	B	C	null	null	null	null	null	null
Read1		A	B	C	null	null	null	null	null
Read2			A	B	C	null	null	null	null
Exec				A	B	C	null	null	null
Write1					A	B	C	null	null

(1) Instructions A-C are executed, D-F are not.

NULL is jammed into the Decode 2 state of the pipeline when a break event occurs, or a preemptive DT-DMA request is made. NULL is similar to a NOP, except that NULL does not increment the IC/PC (i.e. the state of the CPU remains unchanged) nor change the programmer's model of the CPU state (e.g. status bits are unchanged).

The CPU is considered 'halted" when the NULL instruction has been jammed into all pipeline stages from Dec2 to Write1.

Pipeline behavior as affected by debug events:

- A software breakpoint instruction causes a debug event at the beginning of the Decode 2 stage of the instruction. If the breakpoint instruction was an ESTOP0 (as set by the debugger), the IC points to the address of the ESTOP0 instruction being executed. When an ESTOP1 si encountered (as embedded in code), the IC points to the address after the ESTOP1 instruction. In either case the status bits do not change from their prior value.
- A host break event causes a debug event at the beginning of the Decode 2 stage of the earliest valid instruction after the host break event has been detected. At this point the IC points to the address of the instruction which would have gone into the Decode 2 stage, but did not due to the break event from the host.
- Pipeline behavior that all debug events share include:
  - All pipeline stages prior to the second decode stage are discarded, whereas the second decode stage and all those after it complete.

– The DBGACK signal is asserted.
– Interrupts are blocked

When the emulator is not connected or when debug software has disabled emulation, ESTOPs are treated as NOPs and the program counter is incremented past the ESTOP0 or ESTOP1 instruction.

When single stepping in debug mode, the instruction pointed to by the PC is fetched. This instruction will be in the Decode 2 stage of the pipeline when the single step break event causes all subsequent instructions to be discarded (they may be re-fetched and executed in subsequent single step operations).

Table 2-2. Single-Stepping an Instruction(1)

Fetch1	Z	D	E	F	G	..	..	..	..
Fetch2	..	Z	D	E	F	G	..	..	..
1-Dec	..	..	Z	D	E	F	G	..	..
2-Dec	null	null	null	Z	null	null	null	null	null
Read1	null	null	null	null	Z	null	null	null	null
Read2	null	null	null	null	null	Z	null	null	null
Exec	null	null	null	null	null	null	Z	null	null
Write1	null	null	null	null	null	null	null	Z	null

(1) Instruction Z is single-stepped. D-G are not executed.

Should two or more break events occur simultaneously, all of those which caused the break event, or occur before the CPU enters the debug state, will be indicated in the debug status register (DBGSTAT).

- If single-stepping a breakpoint instruction, an ESTOP0 event occurs at the beginning of the Decode 2 stage and the IC does not increment (this is identical behavior to the normal operation of the ESTOP0 instruction). In essence, the ESTOP0 instruction is never actually executed.
- When single-stepping an instruction that causes a break event, DBGSTAT indicates the break event, but the CPU has actually been stopped by the single-step.

Table 2-3. Resuming Running[(1)]

Fetch1	..	D	E	F	etc					
Fetch2	..	..	D	E	F	etc				
1-Dec	..	..	..	D	E	F	etc			
2-Dec	null	null	null	null	D	E	F	etc		
Read1	null	null	null	null	null	D	E	F	etc	
Read2	null	null	null	null	null	null	D	E	F	etc
Exec	null	null	null	null	null	null	null	D	E	F
Write1	null	null	null	null	null	null	null	null	D	E

(1) Instructions D-F (and onward) are executed.

2.3 Debug State Overview

When a C28x device is in the debug state, it is not executing code. Actions such as decoding a software breakpoint instruction are referred to as debug events (see the upcoming Debug Status section for a complete list). When a debug event causes the device to enter the debug state, it is also referred to as a break event. A software break instruction is a software break event.

The debug mode (DBGM) bit is very similar to INTM:

- It is active low (i.e., a value of 0 indicates that the debug mode is enabled, a value of 1 indicates it is disabled).
- DBGM is set to 1 on an interrupt. It will be restored on returning from the interrupt.
- It is controlled through the SETC/CLRC instructions. It can be set or cleared in parallel with the INTM bit (this is commonly done).

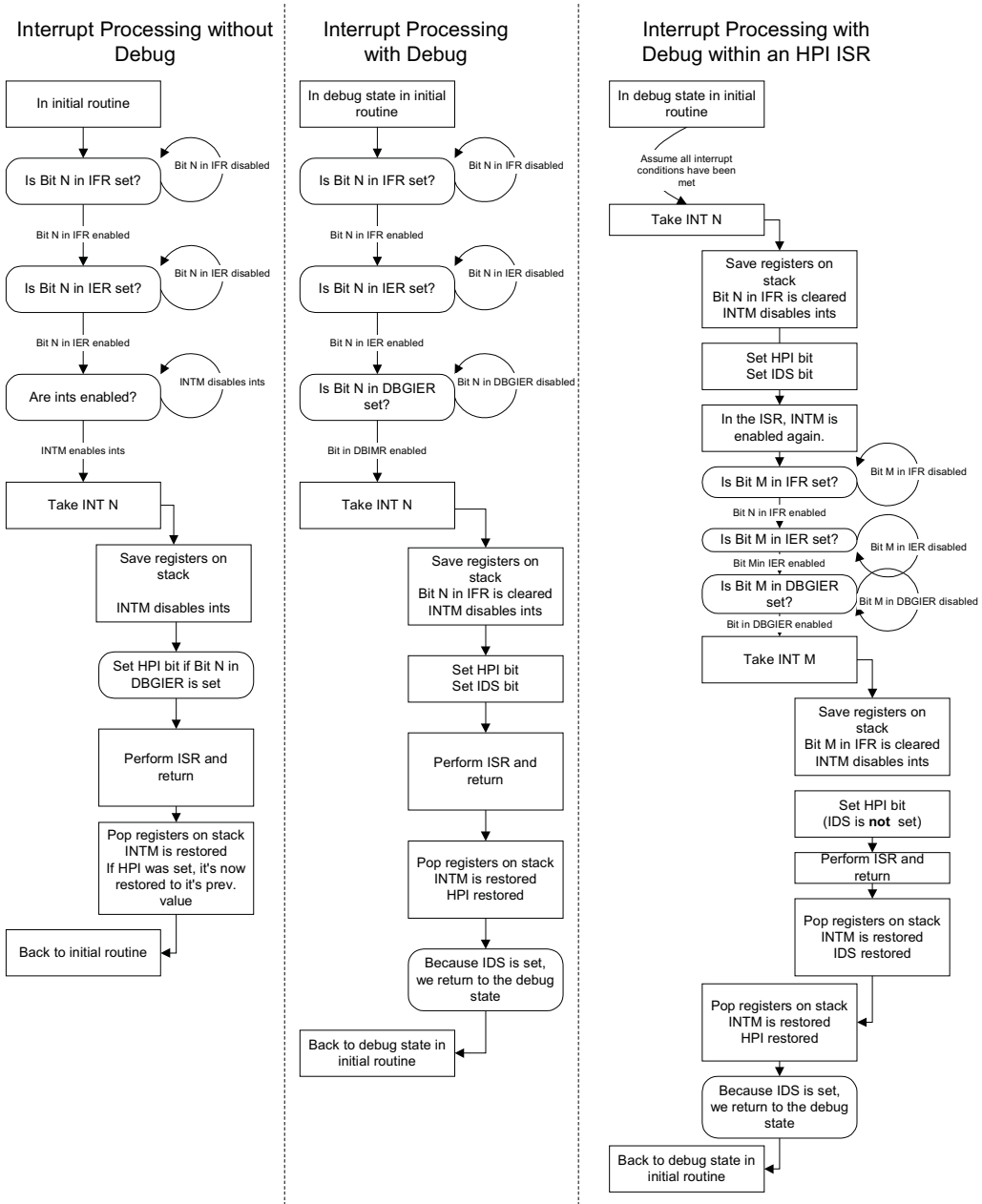
In the debug state, the PC and status bits are held to their values prior to the debug-event. The PC points to the instruction at which the debug-event occurred; that instruction is not considered executed (this is the type of behavior exhibited when taking an interrupt). Note that in some architectural documents,. the register which points to the 'next-to-be executed instruction" is the IC, and the PC is the instruction which is being executed.. When execution resumes, that instruction and those following it need to be re-fetched. The section on Pipeline Behavior During Debug gives more details.

The ESTOP0 and ESTOP1 instructions will cause the device to halt regardless of the setting of the DBGM bit, although it is not recommended that these instructions be used in this manner.

2.3.1 Interrupts During the Debug State

In debug mode, all interrupts including NMI and RESET are blocked (although latched) while the device is halted in the debug state. Note that in debug-mode, when multiple instances of the same interrupts occur without the first instance being serviced, the later interrupts are lost.

Figure 2-2. C28x Interrupt Processing



2.3.2 Single Stepping

Single-stepping overrides the debug state condition for the execution of one instruction of background code by executing from one interrupt boundary to the next. STEP acts as if all interrupts (including NMI and RESET) are disabled, whereas RUN takes any outstanding enabled interrupt. When doing a RUN, an interrupt context switch is counted as one instruction. For example, when doing a RUN 1 with a pending enabled interrupt, only the interrupt context switch of a taken interrupt-service routine (ISR) executes before execution suspends. In this case, the PC points to the first instruction of the ISR when execution suspends.

C28x Emulation Registers

C28x emulation registers are listed in the following sections. Note that in some cases not all register contents are related to emulation use.

3.1 CPU Registers

Registers within the CPU related to emulation.

Name	Size	Description
ST1	16 bits	Status Register 1
IER	16 bits	Interrupt Enable Register
DBGIER	16 bits	Debug Interrupt Enable Register
IFR	16 bits	Interrupt Flag Register
PC	22 bits	Program Counter

3.1.1 Status Registers

Name	Size	Read/Write	Description
ST1	16 bits		Status Register 1:
		R	Bit 7: IDLE Status Bit (IDLESTAT)
		R/W	Bit 6: Emulation Access Enable Bit (EALLOW)
		R/W	Bit 1: Debug Enable Mask Bit (DBGM)
		R/W	Bit 0: Interrupt Global Mask Bit (INTM)

3.1.1.1 Status Register 1 Bit Definitions

DBGM— Debug Enable Mask Bit.

This bit is reserved for emulation purposes. It is used to block debug events from time-critical portions of code execution. A DBGM value of 1 disables debug access, while a value of 0 enables access. DBGM is set to 1 by an interrupt or reset signal.

The DBGM bit can be ignored by the emulation system. This allows the debugger user to have/restore full control of emulation accesses during debug.

EALLOW— Emulation Access Enable Bit

This bit enables application writes to non-CPU emulation registers. When this bit is a 1, application access to emulation registers is allowed, while a 0 value disables such access. When access is disabled, application writes to emulation registers are ignored.

On interrupts or traps, the current state of the EALLOW bit is saved on the stack. EALLOW is set to 0 by an interrupt (after stack push) or reset signal. At the very start of an ISR, access to the non-CPU emulation registers is disabled. The user can re-enable access using the 'EALLOW' instruction. The 'RETURN_INT' instruction restores the previous state of the EALLOW bit saved on the stack.

The EALLOW bit can be overridden via boundary scan through the JTAG port. This allows the debugger user to have full control of emulation register accesses during debug.

An application can verify if the emulation override is in effect by examining this bit after attempting to set it to a 1. If an emulation override is in effect, this bit will remain a 0 even after an application writes a 1 to it.

There are no issues related to EALLOW and pipeline protection. Changing an EALLOW setting directly followed by a peripheral read is supported, because EALLOW does not block reads. Writes are protected because both EALLOW and peripheral writes occur in the EXECUTE pipeline stage. Aa peripheral write does not precede an EALLOW write.

If any emulation register is designed such that it is modified by a read operation (e.g. cleared on read), then the EALLOW function must inhibit the read from occurring while EALLOW is 0.

IDLESTAT— IDLE Status Bit

This read-only bit is set when the IDLE instruction is executed. It is cleared by any of the following events:

- An interrupt is serviced
- An interrupt is not serviced but takes the core out of the IDLE state
- A valid instruction enters the instruction register (the register that holds the instruction currently being decoded)
- A device reset occurs

When the CPU services an interrupt, the current value of IDLESTAT is saved on the stack (when ST1 is saved on the stack), and then IDLESTAT is cleared. Upon return from the interrupt, IDLESTAT is not restored from the stack.

INTM— Interrupt Global Mask Bit

This bit globally enables or disables all maskable interrupts (those that can be blocked by software).

If INTM is 0, maskable interrupts are globally enabled. In order to be acknowledged by the CPU, a maskable interrupt must also be locally enabled by the interrupt enable register (IER).

If INTM is 1, maskable interrupts are globally disabled. Even if a maskable interrupt is locally enabled by the IER, it is not acknowledged by the CPU.

INTM has no effect on non-maskable interrupts, including a hardware reset or the NMI interrupt.

When the CPU services an interrupt, the current value of INTM is saved on the stack (when ST1 is saved on the stack), and then INTM is set. Upon return from the interrupt, INTM is restored from the stack.

3.1.2 Interrupt Enable Register (IER)

The IER is used to individually enable (bit set to 1) or disable (bit set to 0) the maskable interrupts supported on C28x. In stop-mode, maskable interrupts are not serviced while ST1:INTM is set (1) or when the CPU is halted. This register is set to 0 on a reset signal.

Name	Size	Read/Write	Description
IER	16 bits		Interrupt Enable Register:
		R/W	Bit 13: Interrupt enable (INT14)
		R/W	Bit 12: Interrupt enable (INT13)
		R/W	Bit 11: Interrupt enable (INT12)
		R/W	Bit 10: Interrupt enable (INT11)
		R/W	Bit 09: Interrupt enable (INT10)
		R/W	Bit 08: Interrupt enable (INT9)
		R/W	Bit 07: Interrupt enable (INT8)
		R/W	Bit 06: Interrupt enable (INT7)
		R/W	Bit 05: Interrupt enable (INT6)

Name	Size	Read/Write	Description
		R/W	Bit 04: Interrupt enable (INT5)
		R/W	Bit 03: Interrupt enable (INT4)
		R/W	Bit 02: Interrupt enable (INT3)
		R/W	Bit 01: Interrupt enable (INT2)
		R/W	Bit 00: Interrupt enable (INT1)

3.1.3 Interrupt Flag Register (IFR)

The IFR indicates which C28x interrupt sources are pending (IFR bit set to 1). The interrupt inputs are sampled every CPU clock cycle, and if an active interrupt is recognized, then the corresponding bit in the IFR is set and latched. These bits are not set by soft interrupt (INTR) or trap (TRAP) instructions. These registers are set to 0 on a reset signal.

Name	Size	Read/Write	Description
IFR0	16 bits		Interrupt Flag Register:
		R/W	Bit 15: Interrupt flag (RTOSINT)
		R/W	Bit 14: Interrupt flag (DLOGINT)
		R/W	Bit 13: Interrupt flag (INT14)
		R/W	Bit 12: Interrupt flag (INT13)
		R/W	Bit 11: Interrupt flag (INT12)
		R/W	Bit 10: Interrupt flag (INT11)
		R/W	Bit 09: Interrupt flag (INT10)
		R/W	Bit 08: Interrupt flag (INT9)
		R/W	Bit 07: Interrupt flag (INT8)
		R/W	Bit 06: Interrupt flag (INT7)
		R/W	Bit 05: Interrupt flag (INT6)
		R/W	Bit 04: Interrupt flag (INT5)
		R/W	Bit 03: Interrupt flag (INT4)
		R/W	Bit 02: Interrupt flag (INT3)
		R/W	Bit 01: Interrupt flag (INT2)
		R/W	Bit 00: Interrupt flag (INT1)

3.2 Non-CPU Registers

Emulation registers that are not within the CPU are memory mapped registers. Note that all emulation analysis registers exist in data memory space. Locations in gray are reserved.

Name	Address	Size	Read/Write	Description
DBGSTAT	0x0800	16	R	Debug Status Register
Reserved	0x0801			
MF_REG_0	0x0802:0803	32	W	Miscellaneous Function Register 0 – only writable if MON_PRIV is enabled. Check the FKWOK bit in DBGSTAT to verify write.
DMSG	0x0804:0805	32	W	Data Message Passing Register. Check the FKWOK bit in DBGSTAT to verify write.
CMSG	0x0806:0807	32	W	Command Message Passing Register. Check the FKWOK bit in DBGSTAT to verify write.
DIE_ID[0]	0x0808	16	R	C2700-E4 and later devices: Word 0 of 64-bit Die ID. Bit 07:00 Wafer 'X" coordinate Bit 15:08 Wafer 'Y" coordinate
DIE_ID[1]	0x0809	16	R	C2700-E4 and later devices: Word 1 of 64-bit Die ID. Bit 21:16 Wafer #

Name	Address	Size	Read/Write	Description
				Bit 45:22 Lot # (23-bits) – continued in next word
DIE_ID[2]	0x080A	16	R	C2700-E4 and later devices: Word 2 of 64-bit Die ID. See previous word for 13 bits Bit 48:46 Fab #
DIE_ID[3]	0x080B	16	R	C2700-E4 and later devices: Word 3 of 64-bit Die ID. Bit 49 Memory row repaired (0=yes, 1=no) Bit 50 Memory col repaired (0=yes, 1=no) Bit 52:51 Test Code Bit 55:53 Reserved Bit 63:56 Hamming Code
Reserved	0x080C: 0x0856			Do not use

3.2.1 Debug Status Register (DBGSTAT)

Name	Size	Read/Write	Description
DBGSTAT(1)	16 bits		Debug Status Register:
		R/W	Bit 15: Device in IDLE state (IDLE_FLAG)
		R/W	Bits 14-12: Reserved
		R/W	Bit 11: Stop due to ESTOP 1 (ESTOP1)
		R/W	Bit 10: Stop due to ESTOP 0 (ESTOP0)
		R/W	Bit 09: High priority interrupt being serviced (HPI)
		R/W	Bit 08: Interrupt during debug state (IDS)
		R/W	Bit 07-06: EXSM current state (EXSM[1:0])
		R/W	Bit 05: Write to FXREG succeeded (FXWOK)
		R	Bit 04-00: Debug Frame Counter value (DFC[4:0])
		R	
		R	

(1) This register is stored on the stack as part of an interrupt context save. Bits that are described as writeable can only can be written via a context restore.

3.2.1.1 Debug Status Register Bit Definitions

IDLE_FLAG— Device in IDLE state flag.

This bit indicates that the CPU has entered the IDLE state. This bit is set to 0 when the CPU leaves the IDLE state or when changing the PC.

ESTOP1— CPU stop due to ESTOP1 instruction.

This bit indicates that the decode of an 'ESTOP1” instruction is the source of the CPU stop. This bit is set to 1 to indicate that an 'ESTOP1” instruction caused the stop. Cleared on STARTING signal or taking an IDS.

ESTOP0— CPU stop due to ESTOP0 instruction

This bit indicates that the decode of an 'ESTOP0” instruction is the source of the CPU stop. This bit is set to 1 to indicate that an 'ESTOP0” instruction caused the stop. Cleared on STARTING or taking an IDS

HPI— High priority interrupt being serviced flag.

This bit indicates that a high priority interrupt (one enabled in DBIMR) is being serviced. This bit is set to 1 to indicate that a high priority interrupt is being serviced. This bit is not set for interrupts triggered by INTR or TRAP instructions. Reset or ABORTI clears this bit.

IDS— CPU interrupted during debug state flag.

This bit indicates that the CPU was interrupted while stopped in the debug state. This bit is set to 1 to indicate the occurrence of an interrupt while in the debug state. Reset or ABORTI clears this bit.

EXSM[1:0]— Current execute state machine state.

These bits indicate the current value of the execute state machine.

EXSM Value	Description
00	EXE state - normal execution
01	DSUSP state - execution stops due to break
10	IDS state - reserved
11	IDBG state - reserved

FXWOK— CPU write to FXREG succeeded flag.

This bit indicates that the CPU write to the FXREG register succeeded. This bit is set to a 1 to indicate that the write succeeded, to 0 if the write was blocked.

- A write is attempted to CMSG/DMSG when EALLOW is 0.
- A write is attempted to MF_REG_0 when EALLOW is 0.
- A write is attempted to CMSG/DMSG and meets the EALLOW requirements, but the internal emulation resources are busy performing a DT-DMA access.
- A write is attempted to MF_REG_0 and meets the EALLOW requirements, but the internal emulation resources are busy with a JTAG scan access to this register.
- A CPU reset occurs.

The FXWOK bit is set when a write to CMSG/DMSG or MF_REG_0 succeeds.

DFC[4:0]— Debug Frame Counter value.

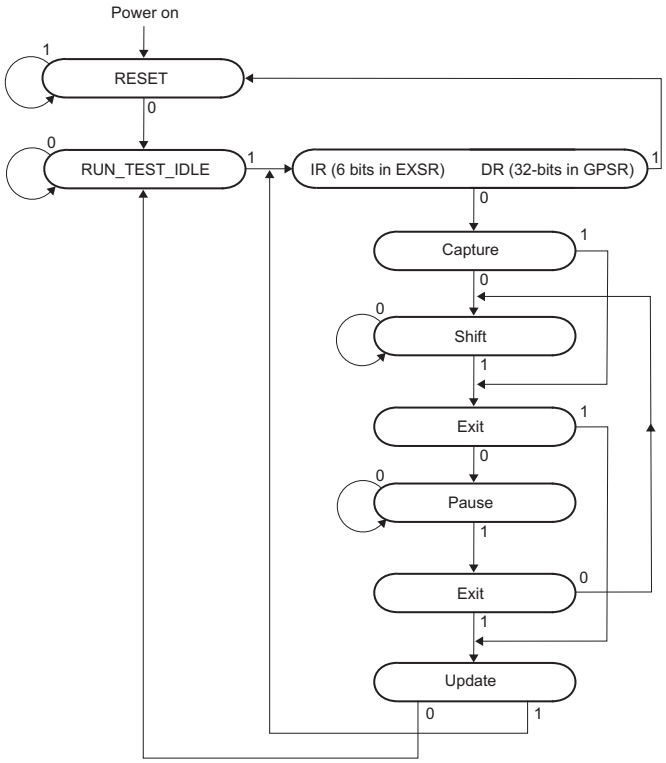
These bits contain the current value of the debug frame counter. The debug frame counter is used to count the number of nested high priority interrupts containing active breakpoints. These bits are set to 11111b by a synchronization loss (MF_REG_0: SYNC_LOSS).

3.2.2 Scan Accessible Registers

Emulation registers which are accessible primarily through JTAG scan (MF_REG_0 can be accessed through the memory map, but the others can not). Table 3-1 lists scan-accessible registers and Figure 3-1 shows a JTAG state diagram.

Table 3-1. Scan Accessible Registers

Name	Size	Description
GPSR	32 bits	General Purpose Shift Register
EXSR	6 bits	Extension Shift Register
IR	6 bits	JTAG Instruction Register
SSR	32 bits	System Status Register
MSR	6 bits	Mini Status Register
MF_REG_0	32 bits	Miscellaneous Function Register 0
MF_REG_1	32 bits	Miscellaneous Function Register 1
DC_STRBS	32 bits	Debug Control Strobes Register
JXREG	32 bits	Temporary JTAG clock domain register
FXREG	32 bits	Temporary functional clock domain register



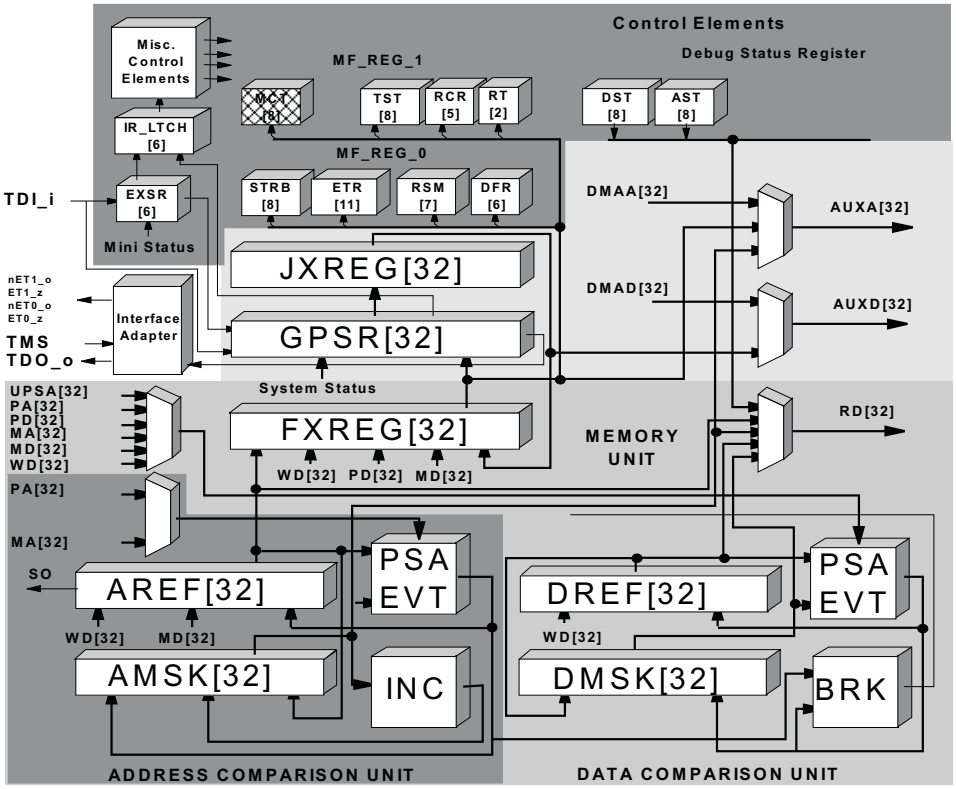
Note: State transitions are controlled by the value of TMS.

Figure 3-1. JTAG States

3.2.3 General Purpose and Extension Shift Registers (GPSR and EXSR)

All scan-accessible registers are accessed through the GPSR and EXSR register pair. The contents of the GPSR and EXSR register pair are parallel-loaded into destination registers, or are loaded from source registers, based on the JTAG IR opcode and JTAG state. All register bits in both the EXSR and GPSR registers shift in the JTAG SHIFT_IR and SHIFT_DR states. Both registers load during the CAPTURE_IR and CAPTURE_DR states.

Figure 3-2. Icemaker Block Diagram



3.2.3.1 JTAG Instruction Scan

Instruction scans use the combined 38-bit instruction path formed by the EXSR and GPSR. The EXSR is nearest the TDI, hence is in the most significant position. The EXSR holds the JTAG IR, while the GPSR contents depend on the class of the IR code. In scan-controller memory, the least significant bit is closest to TDO, which is GPSR[0]. Because the GPSR is 32-bits, and scanned out first in bit order, there is no need to rearrange the bits.

3.2.3.2 JTAG Data Scan

The GPSR provides the data scan path for all JTAG IR classes except BYPASS and ATPG modes. The format of the GPSR data depends on the JTAG IR. Both registers are loaded in the CAPTURE_IR (GPSR contents depend on the JTAG IR, EXSR with mini status) and CAPTURE_DR (GPSR contains system status, EXSR counter initialized) states.

3.2.4 JTAG Instruction Register (JTAGIR)

Loaded from the EXSR in the UPDATE_IR state.

Mnemonic	Class	Clock	Use JXSM	Data Path	Description
BYPASS 0 ATPG_CPU ATPG_EXT	00xxxx 10000 10001	FUNC	N	Bypass	Bypass if not redefined by external logic. The ATPG opcodes are reserved for test.
Reserved	10xxxx	FUNC	N	GPSR	Reserved
SYS_ACC_W	1100zz	FUNC	Y	GPSR	Single MU write - the zz setting corresponds to: 00 — use GPSR as address of byte data (not valid for C28x/C28x) 01 — use GPSR as address of 16-bit word data 10 — use GPSR as address of 32-bit long data 11 — use GPSR as data
SYS_ACC_R	1101xx	FUNC	Y	GPSR	Single MU read - the xx setting corresponds to: 00 — use GPSR as address of byte data (not valid for C28x/C28x) 01 — use GPSR as address of 16-bit word 1X — use GPSR as address of 32-bit long word
TEST_ACC_W	111000	TEST	N	TEST	Reserved for accelerated tester download
TEST_ACC_R	111001	TEST	N	TEST	Reserved for accelerated tester upload
Reserved	111010				
LD_MF_REG_0	111011	FUNC	Y	GPSR	Load GPSR into MF_REG_0. Apply on Run-Test/ Idle state.
LD_DMA_WD	111100	FUNC	Y	GPSR	Use GPSR contents as DMA write data.
LD_MF_REG_1	111101	FUNC	Y	GPSR	Load GPSR into MF_REG_1. Apply on Run-Test/ Idle state.
DC_STRBS	111110	FUNC	N	GPSR	Load GPSR into DC_STRBS. Apply on Run-Test/ Idle state.
BYPASS	111111	FUNC	N	Bypass	JTAG Bypass

3.2.4.1 SYS_ACC MU Address

The address portion of a SYS_ACC_R or SYS_ACC_W is defined below. The MU access method is designed to support all native access modes of the C28x. The DT-DMA access method of the ACU is meant as a high bandwidth access method, and as such, only supports the most efficient transfer methods.

Note that for MU writes, the data word must be written to the GPSR before the address. So, a LD_DMA_WD opcode with the write data in the GPSR would be scanned in and the Run-Test/Idle state entered. Then a SYS_ACC_W opcode with the write address in the GPSR would be scanned in and the Run-Test/Idle state entered.

Depending on the JTAG IR command used for MU access, and the MUCYC code, alignment restrictions imposed by the CPU are expected to be valid for MU accesses as well. The following table illustrates the expected behavior.

In the event of a reserved code being presented to the hardware, the ‘natural’ access type of the address space should be assumed (e.g. a long read to an I/O address should perform a word access).

MU Access Size	Address	Data	Note
Byte (8 bit)	----	----	Not supported on C28x/C28x
16-bit Word	Program Memory – 22-bit word address.	16 LSBs of GPSR.	Only supported on C28x. On Rev 0, do not perform writes to program space while running.
16-bit Word	Data Memory – 32/22-bit word address.	16 LSBs of GPSR.	
16-bit Word	I/O - 16 bit word address	16 LSBs of GPSR.	All I/O accesses performed with this type in emulation software.
32-bit Long	Program Memory – 22-bit word address, must be even-word aligned.	32 bits of GPSR. Upper half of GPSR contains upper addressed word.	Only supported on C28x. On Rev 0, do not perform writes to program space while running.
32-bit Long	Data Memory – 32/22-bit word address, must be even-word aligned.	32 bits of GPSR. Upper half of GPSR contains upper addressed word.	
32-bit Long	I/O - Reserved		

Name	Size	Read/Write	Description
MU Address	32 bits		MU Address:
		W	Bit 31: RESERVED
		W	Bit 30: Ignore DFC ≠ DFR (NO_ID)
		W	Bit 29: Preemptive access (MUHP)
		W	Bit 28: HPI access (MUHPI)
		W	Bit 27: Ignore DBGM (MUDBG)
		W	Bit 26-24: MU access type (MUCYC[2:0])
		W	Bit 23-00: 24 bit address (ADDR[23:0])

3.2.4.2 SYS_ACC MU Address Register Bit Definitions

NO_ID— Qualify access with DFC == DFR.

When this bit is set to a 1, DFC == DFR is not required for access. When set to a 0, the access is not performed unless DFC == DFR.

MUHP — High priority access enable.

When this bit is set to a 1, accesses are performed preemptively (high priority). When set to 0, the access is low priority.

MUHPI— High priority interrupt access enable.

When this bit is set to a 1, accesses are performed even during high priority (debug) interrupt service routines. When set to 0, accesses are performed only while HPI is inactive (0).

MUDBG— Debug mode access enable.

When this bit is set to a 1, accesses are performed regardless of the value of the debug enable bit (ST1:DBGM). When set to 0, DBGM must be active to perform the access.

MUCYC— MU access type.

MUCYC Value	Description
000	Program Memory
001	Register
011	Memory Map configuration registers (-E devices)
100	**Rev 0 & 1:** Reserved. **Rev 2:** IO Memory.

101 Maximum data memory access. Will seize system/analysis resources owned by the application. All analysis resources must be in data memory to allow use of this function.

111 **Rev 0 & 1:** Instruction jam. Acts like a memory write, with the data portion containing the instruction to execute.
Rev 2: Reserved

ADDR— 24 bit address

24-bit byte address defined for the memory access. On the C28x, only the lower 22-bits are valid, and the upper 2 bits are served. On the C28x, the upper 8-bits of an address come from MF_REG_1:UADDR[7:0].

3.2.5 System Status Register (SSR)

Loaded into the GPSR on CAPTURE_DR or CAPTURE_IR states. The BYPASS opcode insures that the next IR scan will return status.

Name	Size	Read/Write	Description
SSR	32 bits		System Status Register:
		R	Bit 31: Rev 1+:Device in 'Wait-In-Reset" mode (WAIT_RST)
		R	Bit 30: Data available in FXREG (TDAVAIL)
		R	Bit 29: Rev 1+: Transfer Data Request (TDAREQ)
		R	Bit 28: Rev 1+: Transfer Data End (TDAEND)
		R	Bit 27: Execute state machine state is IDS (IDS_STATE)
		R	Bit 26: Execution has started (STARTED)
		R	Bit 25-24: Current run state machine state (RSM)
		R	Bit 23: Rev 1+: Program Ready status (PRDY_STAT)
		R	Bit 22: Interrupt mode bit from ST1 (INTM)
		R	Bit 21: High priority interrupt from DBGSTAT (HPI)
		R	Bit 20: Debug enable bit from ST1 (DBG)
		R	Bit 19: Synchronization loss (SYNC_LOSS)
		R	Bit 18: FXSM register is ready (FX_RDY)
		R	Bit 17: JXSM register is ready (JX_RDY)
		R	Bit 16: DFC and DFR are equal (ID_MATCH)
		R	Bit 15: Monitor privilege granted to app (MON_PRIV)
		R	Bit 14: NMI active (NMI_ACT)
		R	Bit 13: CPU Ready active (RDY_ACT)
		R	Bit 12: Reset output latch active (RES_ACT)
		R	Bit 11: Set by rising edge of system clock (CLK_ACT)
		R	Bit 10: JTAG Test-Logic-Reset state (TLR_ACT)
		R	Bit 09: Debugger connected (DCON)
		R	Bit 08: Current nET0 state (ET0_VAL)
		R	Bit 07: Current nET1 state (ET1_VAL)
		R	Bit 0602: Mega-module release ID (REL_ID[4:0])
		R	Bit 01: Zero (ZERO)
		R	Bit 00: One (ONE)

3.2.5.1 System Status Register Bit Definitions

WAIT_RST— Rev 0: Reserved. Rev 1+: Device is in 'Wait-In-Reset' mode.

This bit, when set to a 1, indicates that the device is waiting in the reset state. This mode is often used as a feature to aid in the programming of devices during end-equipment manufacturing.

Wait-In-Reset mode is entered by connecting the JTAG nTRST, nET1, and nET0 pins to a fixed value and applying power to the device.

TDAVAIL— Data available in FXREG.

1= data is available in the FXREG. This, along with the state of FX_RDY and JX_RDY, can determine the state of the FXSM/JXSM state machines. If FX_RDY & JX_RDY are active, then the state machines are waiting on data. If JX_RDY & TDAVIL are active, then the FX_REG is loaded with data. If neither of these are true, then the state machines are in an intermediate state.

TDREQ— Rev 0: Reserved. Rev 1+: Transfer Data Request

This bit is set to 1 when the ACU's DMA control register is used to start a data download (transfer from the host to the application, under the control of the application). TDREQ goes inactive after the first word is transferred.

This is intended for use with some future version of RTDX, where host-to-target transfers can be initiated and controlled more efficiently via the target. In this case, the target would not need to send the host information about where the data is stored in memory – only that it's ready to receive data.

IDS_STATE— Rev 0: IDS State. **Rev 1+:** IDS State (Delayed)

This bit is set to 1 when the IDS is in the EXSM state. On rev 1+, it's timed such that it changes in-sync with the ID_MATCH bit below (which reflects if DFC==DFR). It is important that these two bits of information be timed together, because they can be used to determine if the DFR needs to be updated:

- In IDS, DFR!=DFC is expected.
- If not in IDS, then DFC=DFR is expected.
- If EXSM==IDS, then a RT interrupt is being serviced, so DFC+1 = DFR (IDMATCH=0) is expected; therefore, if the DFC==DFR while in IDS, then the DFR must be changed.

There are two cases where the DFR must be updated:

- If EXSM==DSUSP==>halt, expect DFC = DFR (IDMATCH=1). If !IDS and !IDMATCH, then the DFR must be changed.
- If EXSM==IDBG or EXE==>running, expect DFC = DFR (IDMATCH=1). If !IDS and !IDMATCH, then the DFR must be changed

STARTED— Execution started flag.

This bit, when set to a 1, indicates that device execution has started. This flag is cleared by writing 0 to the MF_REG_0:CSTARTED bit.

RSM— Current value of the Run State Machine (RSM).

These bits reflect the current value commanded to the RSM.

RSM Value	Description
00	EXE_HALT - do not exe
01	EXE_STEP - execute a single instruction (or interrupt)
10	EXE_COND - execute until a stop request
11	EXE_CONT - execute, ignoring all stop requests (including ESTOPs)

PRDY_STAT— Rev 0: Reserved. **Rev 1+:** Program Memory Ready Status

RDY_STAT goes active low when the RDY signal is active for an access to program memory – IFSTAT will always clear PRDY_STAT. Setting DC_STRBS:CLR_RDY_ACT will set PRDY_STAT to 0, as well as clearing RDY_ACT.

This bit was added to ensure that it is always possible to determine when the CPU is 'hung' due to a non-completing access to program memory. Because a non-completing program fetch causes the pipeline to stall until RDY goes active, it is vital that this error be detectable.

INTM— Current interrupt enable mask bit from status register ST1.

This bit reflects the current value of the interrupt enable mask. When set to a 1, it indicates that all maskable interrupts are disabled. INTM is set to 1 during a reset.

HPI— Current high priority interrupt bit from status register DBGSTAT.

This bit reflects the current value of the high priority interrupt bit. When set to a 1, indicates that a high priority interrupt is being serviced. HPI is set to 0 by reset or ABORTI.

DBGM— Current debug enable mask bit from status register ST1.

This bit reflects the current value of the debug enable mask (DBGM) bit. When set to a 1, it indicates that debug events are to be blocked from occurring. Set to 0 by reset.

SYNC_LOSS— Synchronization loss detected.

This bit, when set to a 1, indicates that a synchronization loss has occurred (reset, ABORTI, or EXE_CONT commanded). Cleared to 0 by writing to MF_REG_0:SYNC_LOSS bit.

FX_RDY— FXSM is in ready state.

The FXREG State Machine (FXSM) controls access to the FXREG for debug core operations (DMA, message passing, execution control register loads). This bit, when set to a 1, indicates that the FXREG and state machine are available for an operation.

JX_RDY— JXSM is in ready state.

The JXREG State Machine (JXSM) controls access to the JXREG for debug core operations. This bit, when set to a 1, indicates that the JXREG and state machine are available for an operation.

ID_MATCH— Debug Frame Counter (DFC) matches Debug Reference Counter (DRC) flag.

This bit, when set to a 1, indicates that the DFC and DRC contain the same value. See IDS_STATE above for information on how this can be used

MON_PRIV— Monitor privilege granted to application flag.

This bit, when set to a 1, indicates that the application has acquired monitor privilege. Monitor privilege indicates that the application (or monitor program) owns the execution control of the device through the emulation interrupt EMUINT. When MON_PRIV is set, all emulation events that would normally generate the ‘stopping’ signal force the generation of EMUINT. This signal is set to 0 at reset.

NMI_ACT— Non-maskable interrupt active flag.

This bit, when set to a 1, indicates that a NMI is currently active. This flag does not need to be cleared like RDY_ACT, RES_ACT, CLK_ACT, or TLR_ACT.

RDY_ACT— CPU RDY active flag.

This bit, when set to a 1, indicates a CPU RDY condition. This flag is cleared by writing to the DC_STRB:CRDY_ACT bit. An asserted ready-active flag indicates that all previous bus operations have completed. Any cycle in which a bus request was active, but no READY signal was received prevents RDY_ACT from being set.

RES_ACT— CPU reset active flag.

This bit, when set to a 1, indicates that the reset latch output is active. This flag is cleared by writing to the DC_STRB:CRES_ACT bit.

CLK_ACT— CPU clock detection flag.

This bit is set by the rising edge of the system clock. This flag is cleared by writing to the DC_STRB:CCLK_ACT bit.

TLR_ACT— JTAG Test-Logic-Reset state entered flag.

This bit, when set to a 1, indicates that the JTAG Test-Logic-Reset state has been entered. This flag is cleared by writing to the DC_STRB:CTLR_ACT bit.

DCON— Debugger connected flag.

This bit, when set to a 1, indicates that the debugger is 'connected' to the device. This flag can be cleared or set by writing to the DC_STRB:DCONID field. This bit is cleared on entering the JTAG Test-Logic-Reset state.

ET0_VAL— Current nET0 value.

This bit indicates the current input value of the nET1 signal.

ET1_VAL— Current nET1 value.

This bit indicates the current value of the nET0 signal.

REL_ID[4:0]— Mega-module release ID.

Defines the revision level of the CPU Mega Module / ICEMaker.
0 indicates a rev 0 ICEMaker (C28x-C1)
1 indicates a rev 1 ICEMaker (C28x-C3 and C4)
2 indicates a rev 2 ICEMaker (C28x)

ZERO— The constant value '0', required by JTAG standard.

ONE— The constant value '1', required by JTAG standard.

3.2.6 Mini Status Register (MSR)

Loaded into the EXSR on CAPTURE_IR states.

Name	Size	Read/Write	Description
MSR	6 bits		Mini Status Register:
		R	Bit 05-03: GPSR content type (DTYPE)
		R	Bit 2: Rev 2: CPU is in real-time mode (RTMODE)
			Rev 0 & 1: Reserved
		R	Bit 1: CPU has stopped (STOPPED)
		R	Bit 0: Command failure (CFAIL)

DTYPE — Type of data held in GPSR register.

These bits show the contents of the GPSR register scanned out at the same time the EXSR is scanned.

DTYPE Value	Description
000	Status (System Status Register contents)
001	Memory unit word (result of SYS_ACC_R IR class)
010	Data message (DMSG register contents)
110	Command message (CMSG register contents)
100	Packet stream middle word
101	Packet stream first word
011	Packet stream last word
111	Single word packet stream

RTMODE— Rev 0 & 1: Reserved . **Rev 2:** Device is in real-time or stop mode emulation.

This bit, when 1, indicates that the CPU is operating in real-time mode. When 0, the CPU is operating in stop-mode.

STOPPED— CPU stopped flag.

This bit, when set to a 1, indicates that the CPU has stopped or returned from the IDBG state (serviced high priority interrupt). This flag is cleared by writing to the MF_REG_0:CSTOPPED bit.

CFAIL— Command failure flag.

This bit, when set to a 1, indicates that the command that was scanned in while this status data was scanned out will fail because the JXSM is not in the ready state.

3.2.7 Miscellaneous Function Register 0 (MF_REG_0)

This register is writeable via the memory map when the ST1:EALLOW bit is set, otherwise it is accessible via JTAG scan. Reading from this register will generate unpredictable results.

Name	Size	Read/Write	Description
MF_REG_0	32 bits		Miscellaneous Function Register 0:
		W	Bit 31: Enable load of SYNC_LOSS (SYNC_LD)
		W	Bit 30: Sync loss control (SYNC_LOSS)
		W	Bit 29: Enable load of EMUINT (EMUINT_LD)
		W	Bit 28: EMUINT enable (EMUINT)
			Bit 27: Reserved
		W	Bit 26: Enable load of ETR (ETR_LD)
		W	Bit 25-16: Emulation And Test Pin Control (ETR)
		W	Bit 15: Enable load of bits 12-08 (EXE_LD)
		W	Bit 14: Reserved
		W	Bit 13: Reserved
		W	Bit 12: Acknowledge NF_SUSP (CSTOPPED)
		W	Bit 11: Acknowledge STARTING (CSTARTED)
		W	Bit 10: Interrupt enable while executing (DINT)
		W	Bit 09-08: Run State Machine Control (EXE_DIR[1:0])
		W	Bit 07: Enable load of bit 06-00 (QUAL_LD)
		W	Bit 06-05: Rude/Polite Access Control (CND_MSK[1:0])
		W	Bit 04-00: Debug Frame Reference Control (DFR[4:0])

3.2.7.1 Miscellaneous Function Register 0 Bit Definitions

SYNC_LD— Enable load of SYNC_LOSS control bit.

This bit enables the loading of the SYNC_LOSS control bit. When set to a 1, the SYNC_LOSS control bit may be written.

SYNC_LOSS— Synchronization loss control bit.

This bit controls the setting and clearing of a synchronization error by software. When set to a 1, this bit forces a SYNC_LOSS condition. When set to a 0, any outstanding SYNC_LOSS (including those set by hardware) is cleared. SYNC_LD must be enabled for this bit to have any effect. See SSR:SYNC_LOSS for a readable indication of a synchronization error.

EMUINT_LD— Reserved for EMUINTM load enable.

This bit is reserved for enabling the loading of the EMUINT control bit. When set to a 1, the EMUINT control bit may be written.

EMUINT— Emulation interrupt (EMUINTM) enable.

Setting EMUINT will enable the generation of EMUINT on the target when the debug state is entered. This is only used to implement a non-JTAG based debug monitor.

ETR_LOAD— Enable load of ETR control bits.

This bit enables the loading of the ETR control bits. When set to a 1, the control bits may be written.

ETR[9:0]— Emulation and test pin control.

These bits control the configuration of the nET0 and nET1 signal extensions to the JTAG interface.

9	**IPM0**	Enable nET0 pulse catcher. — A value of 1 enables the catcher, while a value of 0 disables it. To clear the pulse catcher, first disable it, then enable it. Driving the nET0 output high to low triggers the pulse catcher. The pulse catcher forces nET0 to stay low once triggered. Note that the pulse catcher is only defined for nET0 output.
8-5	**ETSEL0[3:0]**	nET0 control bits

Value	nET0_o	nET0_z	Type	Pulse Stretcher	Description
0	Low	High	OC	No	nET0 not driven, can be used as input
1	Low	nSTOPPING	OC	Yes	nET0 low when stopping, can be used as input.
2	Low	Low	TP	No	Force nET0 low
3	High	Low	TP	No	Force nET0 high
4					***TBD***
5	nAEVENT	Low	TP	Yes	Drive nET0 low on ACU trigger
6	APSA2[0]	Low	TP	Yes	nET0 reflects ACU PSA's MSB (33 of 40)
8	TEST0[0]	Low	TP	No	Reserved for test
9	TEST0[1]	Low	TP	No	Reserved for test
10	TEST0[2]	Low	TP	No	Reserved for test
11	TEST0[3]	Low	TP	No	Reserved for test
12	USER0[0]	Low	TP	No	Reserved for user
13	USER0[1]	Low	TP	No	Reserved for user
14	USER0[2]	Low	TP	No	Reserved for user
15	USER0[3]	Low	TP	No	Reserved for user

Pulse Stretcher — As core speeds increase, the duration of signals such as nSTOPPING from the core decrease; thus, the EMU pulse controlled by these events also becomes proportionally shorter. In order to avoid a situation where the low-going EMU pulse is too short for external logic to detect, pulse stretching logic must be built into the EMU signal-generation logic.

4	**IPM1**	Enable nET1 pulse catcher — A value of 1 enables the catcher, while a value of 0 disables it. To clear the pulse catcher, first disable it, then enable it. Driving the nET1 output high to low triggers the pulse catcher. The pulse catcher forces nET1 to stay low once triggered. Note that the pulse catcher is only defined for nET1 output.
3-0	**ETSEL1[3:0]**	nET1 control bits

Value	nET1_o	nET1_z	Type	Pulse Stretcher	Description
0	Low	High	OC	No	nET1 not driven, can be used as input
1	Low	nSTOPPING	OC	Yes	nET1 low when stopping, can be used as input.
2	Low	Low	TP	No	Force nET1 low
3	High	Low	TP	No	Force nET1 high
11-Apr		Low	TP	Yes	Reserved
12	USER1[0]	Low	TP	No	Reserved for user
13	USER1[1]	Low	TP	No	Reserved for user
14	USER1[2]	Low	TP	No	Reserved for user
15	USER1[3]	Low	TP	No	Reserved for user

Pulse Stretcher — The triggers designated with the Pulse Stretcher active are generated by supplying an input pulse that is 2 functional clocks wide to the CPU. The trigger logic then generates a minimum 50-ns active-low trigger and a 50-ns recovery time. When trigger inputs occur during the recovery time they are ignored. The 50-ns period is specified at the maximum frequency of the core at initial introduction. It is of sufficient width to allow a 2.5× increase in clock speed as the core is cost-reduced in new processes.

EXE_LD— Enable load of CSTOPPED, CSTARTED, DINT, and EXE_DIR.

This bit enables the loading of the CSTOPPED, CSTARTED, DINT, and EXE_DIR control bits. When set to a 1, the control bits may be written.

CSTOPPED— Acknowledge NF_SUSP condition.

This bit, when set to a 1, indicates that the CPU has stopped. Setting this bit to a 0 clears the indication. This bit is useful for error-detection purposes; clear the bit, halt the processor, and verify that CSTOPPED is set. If it is not, then the processor has not stopped yet.

CSTARTED— Acknowledge STARTING condition.

This bit, when set to a 1 indicates that the CPU has started execution. Setting this bit to a 0 clears the indication. This bit is useful for error detection purposes; clear the bit, start the processor, and verify that CSTARTED is set. If it is not, then the processor has not started yet.

DINT— Global interrupt block.

This bit can block all interrupts (including reset and NMI) while the CPU is executing. Setting this bit to a 0 blocks all interrupts; setting it to a 1 enables all otherwise enabled interrupts. Setting this bit differentiates between a debugger run and step command.

EXE_DIR— Run State Machine Control

These bits command a change of state for the run state machine (RSM). Note that stopping or performing a EXE_STEP changes EXE_DIR to the EXE_HALT value.

EXE_DIR Value	Description
00	EXE_HALT - do not execute instructions.
01	EXE_STEP - execute a single instruction cycle (or interrupt cycle).
10	EXE_COND - execute until a stop request
11	EXE_CONT - execute, ignoring all stop requests (including ESTOPs).

QUAL_LD— Enable load of CND_MASK and DFR.

This bit enables the loading of the CND_MASK and DFR control bits. When set to a 1, the control bits may be written.

CND_MSK[1:0]— Rude/polite access control.

These bits control the access policy used by emulation. Certain emulation accesses are inhibited by high priority interrupts and DBGM. These bits act as a mask to enable or disable access control as follows:

- CND_MASK[1] - HPI_MSK, when 1, ignore DBGSTAT:HPI while performing accesses.
- CND_MASK[0] - DBG_MSK, when 1, ignore ST1:DGBM while performing accesses.

DFR[4:0]— Debug Frame Reference

These bits set the current value of the debug frame counter. The debug frame counter is used to count the number of nested high priority interrupts (interrupts with DBIER bits set) containing active breakpoints. These bits are set to all 1s by a synchronization loss (MF_REG_0: SYNC_LOSS). These bits are meant to allow emulation software the ability to change or initialize the current frame of interest.

3.2.7.2 MF_REG_0 Access Control

MF_REG_0 is shared between the debugger (via scan) and the application (via memory map registers). The application can request a memory access through the EALLOW mechanism. Note that even when EALLOW is set, the debugger still has scan access to MF_REG_0 and also has priority. All application writes to MF_REG_0 must still check FXWOK to ensure completion of the write. The following table defines access control for MF_REG_0.

EALLOW	Application Access	JTAG Access	FXWOK upon app access	JTAG error
0	Blocked	Allowed	0	None
1	Allowed, but JTAG scan access has priority	Allowed	1 if application write was successful. 0 if debugger has not completed previous scan access	None

3.2.8 Miscellaneous Function Register 1 (MF_REG_1)

This register is loaded from the GPSR using the JTAG IR class LD_MF_REG_1.

Name	Size	Read/Write	Description
MF_REG_1	32 bits		Miscellaneous Function Register 1:
		W	Bit 31: Rev 2: Enable load of UADDR (UADDR_LD)
		W	Bit 30-24: Rev 2: Upper Address (UADDR[7:1])
		W	Bit 23: Rev 2: Upper Address (UADDR[0])
			Rev 1: Override VMAP (VMAPO)

Name	Size	Read/Write	Description
		W	Bit 22: Pulses NMI (DO_NMI)
		W	Bit 21: Enable load of testing bits (ICETST_LD)
		W	Bit 20: Set MON_PRIV for testing (MON_PRIV)
		W	Bit 19: Isolate ICEMaker from CPU inputs (HERMIT)
		W	Bit 18: Enable load of EO (EO_LD)
		W	Bit 17: EALLOW override (EO)
		W	Bit 16: Enable load of ABORT (ABORT_LD)
		W	Bit 15: Abort memory operation (ABORT)
		W	Bit 14: Enable load of TST (TST_LD)
		W	Bit 13-07: Test mode/clock source control (TST)
		W	Bit 06: Enable load of RCR (RCR_LD)
		W	Bit 05-02: Reset control (RCR)
		W	Bit 01: Enable load of RT (RT_LD)
		W	Bit 00: Real-time control (RT)

3.2.8.1 Miscellaneous Function Register 1 Bit Definitions

UADDR_LD— Rev 0 & 1: Enable load of UADDR[7:0]. **Rev 2:** Enable load of UADDR[7:0].

When set to 1, this bit enables the loading of the UADDR bits.

UADDR[7:1]— Rev 0 & 1: Reserved. **Rev 2:** Upper address

Holds bits 31:25 of the 32-bit address for JTAG-initiated memory writes. Bits 23:0 are supplied by either GPSR (in the case of a SYS_ACC_R or SYS_ACC_W), or the ACU if doing a DMA download. Note that these address bits must be right-shifted by 1 (i.e. bit 31 of the address is bit 30 of MF_REG_1).

UADDR[0]— Rev 0: Reserved. **Rev 1:** Used as VMAPO. **Rev 2:** Upper address

Holds bit 24 of the 32-bit address for JTAG-initiated memory writes. Bits 23:0 will be supplied by either GPSR (in the case of a SYS_ACC_R or SYS_ACC_W) or the ACU if doing a DMA download. Note that these address bits must be right-shifted by 1 (i.e. bit 31 of the address is bit 30 of MF_REG_1).

VAMPO— Rev 0: Reserved. **Rev 1:** Override VMAP. **Rev 2:** Moved to DC_STRBS register

When VMAPO is set, it acts as an override for the VMAP signal to the core, forcing it to 0. This causes vectors to be mapped to low memory (desirable for test purposes). The VMAPO bit is loaded when EO_LD is set.

DO_NMI— Pulse NMI.

This bit, when set to a 1, causes an activation of the non-maskable interrupt (NMI). It is not necessary to clear this bit after setting it. There is no load bit for this.

ICETST_LD— Enable load of ICEMaker test control bits.

This bit enables the loading of the MON_PRIV and HERMIT control bits. When set to a 1, these control bits may be written.

MON_PRIV— Set MON_PRIV

The 'embedded debug monitor' can be controlled by a signal into the core (cpuMON_PRIV). To allow for testing of this mode on devices where this signal is disabled, it can be set by setting this bit via JTAG scan. Note that if the cpuMON_PRIV signal is active, this bit will not be able to clear/override it.

HERMIT— Isolate ICEMaker from CPU inputs.

This bit, when set to a 1, isolates the ICEMaker module from CPU inputs. When 0, the ICEMaker is no longer isolated from CPU signals. The purpose of this control is to inhibit the propagation of unknown signal values in a hardware simulation environment – in actual silicon devices, it would rarely, if ever, be used. This bit is set to a 0 in the JTAG Test-Logic-Reset (TLR) state, so the ICEMaker is NOT isolated by default.

EO_LD— Enable load of EO control bit.

This bit enables the loading of the EO control bit. When set to a 1, the EO control bit may be written.

EO— EALLOW instruction override.

This bit controls the override of the EALLOW instruction. When set to a 1, EALLOW is overridden, inhibiting all CPU writes to emulation registers.

ABORT_LD— Enable load of ABORT control bit.

This bit enables the loading of the ABORT control bit. When set to a 1, the ABORT control bit may be written.

ABORT —Abort a memory access.

This bit, when set to a 1, aborts the ongoing memory operations by asserting the CPU RDY signals. All currently in-progress transactions simultaneously see the asserted CPU RDY signal. The proper operation before asserting ABORT is to HALT CPU execution. ABORT should be considered a catastrophic event (however, if the debugger is used to access invalid memory regions, it will commonly occur). ABORT will remain asserted until de-asserted by emulation software. Note that the debugger software will wait a few milliseconds after initiating a memory access before determining that it had failed, and issuing an abort command.

TST_LD— Enable load of TST control bits.

This bit enables the loading of the TST control bits. When set to a 1, the TST control bits may be written.

TST— Test mode and clock control.

These bits control the CPU clock selection and test mode control functions.

TST[6:5] CLKSRC	Description
00	Functional clock 0
01	ATPG mode
10	Test clock
11	Test clock / 2

TST[4] TSTMAP

Core functional test — When set to 1, the memory map is set to test configuration (B0/B1 only); all other memories will have their READY inputs to the core tied high. ISOLATE should also be used. This is used for functional core verification – for tests desired to run on any cDSP regardless of memory or peripherals.

TST[3] ISO

Core system test — When set to 1, it disconnects the CPU input signals (but not READY inputs – see TSTMAP). Signals which are isolated include RSn, NMIn, INTn[14:0], VMAP, BREAK/TRACE. Used for functional memory verification.

TST[2] RUN N

Run for N-1 clocks — When set to 1,the ACU counter is used to indicate the number of clock cycles to allow the CPU to run before stopping – note that when the counter underflows, the CPU will stop regardless of whether the pipeline is stalled, etc. This can be used for ATPG-at-speed testing, not normal emulation. Using it for normal emulation would result in the CPU being halted in indeterminate states.

TST[1] PWR_ABT

Abort power-down mode — When set to 1, the core re-enters normal operation.This is used for emulation to wake up devices which have entered low-power modes.

TST[0]

Reserved

RCR_LD— Enable load of RCR control bits.

This bit enables the loading of the RCR control bits. When set to a 1, the RCR control bits may be written.

RCR— Reset control settings.

These bits control the reset functions of the device.

RCR[3] RES_SET	When 1, RES_LTCH is set. Not effected by RES_DIS.
RCR[2] RES_CLR	When 1, attempt to clear RES_LTCH.
RCR[1] RES_DIS	When 1, disconnects system resets from RES_LTCH.
RCS[0] RES_STAY	When 1, the incoming reset is held indefinitely until this bit is cleared or RES_CLR is asserted.

RT_LD —Enable load of the RT control bit.

This bit enables the loading of the RT control bit. When set to a 1, the RT control bit may be written.

RT— Real-time mode flag.

This bit, when set to a 1, commands the device to enter real-time mode (high priority interrupts are enabled). When set to a 0, the device enters stop-mode. A JTAG Test-Logic/Reset sets this bit to 1. In rev 0 & 1, there is no method of determining the current value of this bit, but in rev 2, there is a status bit in the EXSR register.

3.2.9 Debug Control Strobes (DC_STRBS)

This register is loaded from the GPSR using the JTAG IR class DC_STRBS.

Name	Size	Read/Write	Description
DC_STRBS	32 bits		DCSTRBS:
		W	Bit 31-21: Reserved.
		W	Bit 20: Rev 2: Enable B0B1MAP override (B0B1MAPO)
			Rev 0 & 1: Reserved
		W	Bit 19: Rev 2: Value for B0B1MAP override (B0B1MAPV)
			Rev 0 & 1: Reserved
		W	Bit 18: Rev 2: : Enable VMAP override (VMAPO)
			Rev 0 & 1: Reserved
		W	Bit 17: Rev 2: : Value for VMAP override (VMAPV)
			Rev 0 & 1: Reserved
		W	Bit 16: Rev 2: Enable load of bits 20:17 (O_LD)
			Rev 0 & 1: Reserved
		W	Bit 15: Force JXSM to JX_READY state (CJXSM)
		W	Bit 14: Force FXSM to FX_RDY state (CFXSM)
		W	Bit 13: Clear RDY_ACT strobe (CRDY_ACT)
		W	Bit 12: Clear RES_ACT strobe (CRES_ACT)
		W	Bit 11: Clear CLK_ACT strobe (CCLK_ACT)
		W	Bit 10: Clear TLR_ACT (CTLK_ACT)

Name	Size	Read/Write	Description
		W	Bit 09: Reserved
		W	Bit 08: Reserved
		W	Bit 07-00: Debugger connected control (DCONID)

3.2.9.1 DC_STRBS Bit Defin

B0B1MAPO— Rev 0 & 1: Reserved. **Rev 2:** Enable B0B1MAP Override

This bit , when set to a 1, overrides the B0B1MAP signal to the core with the value of the B0B1MAPV bit below. When 0, the B0B1MAP signal to the core is used.

B0B1MAPV— Rev 0 & 1: Reserved. **Rev 2:** Value for B0B1MAP Override

When the B0B1MAP bit above is 1, then this bit determines the state of B0B1MAP as seen by the core. A value of 1 sets the B0B1MAP signal high, a value of 0 sets it low.

VMAPO— Rev 0 & 1: Reserved. **Rev 2:** Enable VMAP Override

This bit , when set to a 1, overrides the VMAP signal to the core with the value of the VMAPV bit below. When 0, the VMAP signal to the core is used.

VMAPV— Rev 0 & 1: Reserved. **Rev 2:** Value for VMAP Override

When the VMAP bit above is 1, then this bit determines the state of VMAP as seen by the core. A value of 1 sets the VMAP signal high, a value of 0 sets it low.

O_LD —Rev 0 & 1: Reserved. **Rev 2:** Enable load of the B0B1MAPO, B0B1MAPV, VMAPO, and VMAPV bits.

This bit enables the loading of the B0B1MAP and VMAP override bits. When set to a 1, the B0B1MAPO, B0B1MAPV, VMAPO, and VMAPV control bits may be written.

CJXSM— Force JXSM to JX_RDY state.

This bit , when set to a 1, forces the JXSM (JXREG state machine) to the JX_RDY state. It is not necessary to clear this bit after setting it.

CFXSM —Force FXSM to FX_RDY state.

This bit, when set to a 1, forces the FXSM (FXREG state machine) to the FX_RDY state. Unlike CJXSM, this bit must be written with a 0, after it has been set, to clear the FX_RDY hold.

CRDY_ACT— Clear RDY_ACT signal

This bit, when set to a 1, forces the RDY_ACT signal to a 0. This bit is used to acknowledge and clear a CPU ready indication. It is not necessary to clear this bit after setting it.

CRES_ACT— Clear RES_ACT signal.

This bit, when set to a 1, forces the RES_ACT signal to a 0. This bit is used to acknowledge and clear a CPU reset indication. It is not necessary to clear this bit after setting it.

CCLK_ACT— Clear CLK_ACT signal.

This bit, when set to a 1, forces the CLK_ACT signal to a 0. This bit is used to acknowledge and clear a CPU clock tick indication. It is not necessary to clear this bit after setting it.

CTLR_ACT— Clear TLR_ACT signal.

This bit, when set to a 1, forces the TLR_ACT signal to a 0. This bit is used to acknowledge and clear a JTAG Test Logic/Reset indication. It is not necessary to clear this bit after setting it.

DCONID— Debugger connected control.

These bits control the debugger connected condition. To maintain a connection, these bits must be set to the CONNECT value (0xAAh) each time this register is updated. Any other value will ‘disconnect’ the debugger, ending the debugger session.

Emulation Instructions

Several instructions are provided in the instruction set for use by emulation.

4.1 ESTOP0

This instruction is used to implement a debugger software breakpoint. When an emulator is connected, the ESTOP 0 instruction stops processor execution with the PC pointing to the ESTOP 0 instruction. If an emulator is not connected (run state machine (RSM) is in the EXE_CONT state), this instruction is effectively a NOP, with the PC incrementing past the ESTOP 0 instruction.

This instruction is effective at the DECODE1 pipe stage. This instruction is neither affected by nor affects the ST0 or ST1 registers. It is not repeatable.

The opcode is documented in the CPU User's Guide. It is 0x7625. The assembly mnemonic is 'ESTOP0".

4.2 ESTOP1

An The ESTOP1 and ESTOP0 instructions are similar, except that the PC advances past the ESTOP1 instruction. This instruction is used to embed a breakpoint into an application. The embedded breakpoint may be used while connected to an emulator or not. In the former case, the embedded breakpoint may act like a software breakpoint.

This instruction is effective at the DECODE2 pipe stage. This instruction is neither affected by nor affects the ST0 or ST1 registers. It is not repeatable.

The opcode is documented in the CPU User's Guide. It is 0x7624. The assembly mnemonic is 'ESTOP1".

4.3 ABORTI

The ABORTI instruction indicates that an interrupt service routine (ISR) is not to be returned from. This clears the IDS (interrupt during debug) and HPI (high priority interrupt) bits in the DBGSTAT register, disables (clears) DBGM, and resets the Debug Frame Counter. This causes the emulation software to disregard any and all outstanding debug states entered from debug interrupts since the processor was stopped by an emulation event.

ABORTI does not modify DBGIER, affect non-emulation registers like IER or the INTM bit, or clear or change any analysis registers for breakpoints, watchpoints, or PSAs. After executing ABORTI, the application should not issue a return from interrupt instruction, otherwise the previous debug state will be restored and the ABORTI effect will be nullified. The stack pointer will need to be reset manually to reclaim the stack space used by interrupt service routines. Use of ABORTI should be considered catastrophic to normal system operation.

This instruction is effective at the EXECUTE pipe stage It does set the DBGM bit of ST1. It is not repeatable.

The opcode is documented in the CPU User's Guide. It is 0x001. The assembly mnemonic is 'ABORTI'.

4.4 EALLOW/EDIS

The EALLOW instruction enables application writes to non-CPU emulation registers. For any application write to non-CPU emulation registers to succeed, this instruction must be executed and be successful. Success can be determined by examining the EALLOW status register bit in ST1. Note that EALLOW status may be overridden via JTAG scan so it is necessary to check for the success of each EALLOW instruction.

When the application no longer requires access to non-CPU emulation registers, the EDIS instruction must be executed to again inhibit application access. When access is disabled, application writes to emulation registers will have no effect.

This instruction is effective at the EXECUTE pipe stage, and sets the DBGM bit of ST1. It is not repeatable.

The opcodes are documented in the *C28x CPU User's Guide*. The assembly mnemonics are 'EALLOW" and 'EDIS", and the respective opcode values are 0x7622 and 0x761A.

4.5 CLRC DBGM/ SETC DBGM

These instructions are used to enable (CLRC DBGM) or block (SETC DBGM) debug events from time-critical portions of application execution. The use of these instructions provides hints to the emulator system to aid it in setting the level of intrusiveness to use when performing debug actions. The emulator system honors the setting of this condition under most circumstances, but may choose to ignore it, for example as part of an error-recovery routine.

These instructions can also set/clear other ST1 register bits in parallel. Of particular value is the 'SETC DBGM,INTM' and 'CLRC DBGM, INTM' instructions – which can disable/enable both debug events and interrupts at the same time.

These instructions are effective at the DECODE2 pipe stage It does set the DBGM bit of ST1. It is not repeatable.

The opcodes are documented in the CPU User's Guide. The assembly mnemonics are 'SETC' and 'CLRC', and the respective opcode values are 0x3BXX and 0x29XX.

Glossary

A.1 Glossary of Emulation and Architectural Terms and Abbreviations

BASE DEVICE LOGIC— C28x core and peripherals which are not part of the C28x-E's special debug capabilities.

BREAK EVENT— A debug event which has been selected to halt the device. Typically, actions such as the decoding of an ESTOP or request of the host, will halt the device.

BREAKPOINT, SOFTWARE — See ESTOP.

DBGACK— signal which indicates that the device is in debug mode. This has historically been called SUSPEND, and it may be changed to this name to prevent confusion.

DBGM— Debug Mode bit within ST1 which must be enabled to allow polite DT-DMA accesses.

DFC, DFR— See DEBUG FRAME COUNTER and DEBUG FRAME REGISTER.

DT-DMA— Debug and Test Direct Memory Access. A mechanism to access memory and registers without fetching CPU instructions from memory or requiring the CPU to stop.

DEBUG FRAME COUNTER— Counter which keeps track of nested interrupts when the device is stopped in debug mode.

DEBUG FRAME REGISTER— Register which is compared against the Debug Frame Counter to ensure that the device state being accessed is what the debug software expects.

DEBUG EVENT— An action such as the decoding of an ESTOP, occurrence of an analysis watchpoint, or request of the host, which may result in special debug behavior, such as halting the device or pulsing one of the EMU0/EMU1 signals. Often times when 'debug event" is mentioned instead of a 'break event", it means the the device will not enter the debug state.

DEBUG STATE— State in which the device does not execute application code but may perform debug tasks.

DEBUG AND TEST MODE— Used to indicate how the debug resources are to be used.

DEVICE OPERATING MODE— Device power-up mode, typically used for test purposes.

DBGSTAT— Status register. This is saved/restored as part of the interrupt context save/restore.

EALLOW— Emulation instruction which enables access to memory-mapped emulation registers.

EDIS— Emulation instruction which disables access to memory-mapped emulation registers.

ESTOP— Emulation instruction whose decoding produces an event which causes the device to enter the debug state if an emulator is connected. Used to implement software breakpoints.

EXE_COND— Execution state in which instructions are performed until a break event occurs.

EXE_CONT— Execution state in which instructions are performed as if the emulator is disconnect.

EXE_HALT— Execution state in which execution is suspend. Instructions are not fetched.

EXE_STEP— state in which a single instruction is performed, and then execution is suspended.

FLUSHING— The act of discarding the contents of pipeline stages prior to the second decode while allowing the second decode and later pipeline stages to complete.

HPI— High Priority Interrupt bit in ST1 indicates whether the current interrupt is a high priority interrupt.

DBGIER— Debug-enabled Interrupt Register. Indicates which interrupts are time critical.

HOLES— Unused cycles which can be utilized for DT-DMA. Holes can occur naturally due to branches, or be inserted by high priority DT-DMA requests.

IC— Instruction Counter register – Points to the next-to-be-executed instruction.

IDS— Interrupt During Device State bit within ST1 indicates whether device was interrupted by a high priority interrupt while it was in the debug state. If this bit is set when a RETI instruction is executed, the device will enter the debug state upon returning.

IER— Interrupt Enable Register. Part of the standard CPU architecture.

IFR— Interrupt Flag Register. Part of the standard CPU architecture.

IMR— Interrupt Mask Register - this is an older term - IER is now more commonly used. Part of the standard CPU architecture.

INTERRUPT BOUNDARY— Indicates a point at which an interrupt can be taken.

ISR— Interrupt Service Routine. Part of the standard CPU architecture.

JTAG— Abbreviation for the IEEE Std. 1149.1, 'Standard Test Access Port and Boundary-Scan Architecture" which defines a protocol based on a five-pin interface (TCK, TMS, TDI, TDO, and TRST-) used for test and emulation.

NMI— Non-maskable interrupt. Part of the standard CPU architecture.

NULL— Microinstruction which is similar to a NOP but does not increment the PC.

PC— Program Counter register – Points to the instruction being executed. Part of the standard CPU architecture.

PRODUCTION DEVICE— A device which is intended for volume production and sale. Cost (as determined by silicon size, package type, test time, and other factors) is of key importance.

PULSE EVENT— An action such as the decoding of an ESTOP, occurrence of an analysis watchpoint, or request of the host, which causes a pulse on one of the EMU0/EMU1 signals.

RESET— Device reset. Part of the standard CPU architecture.

STEP— Execution of a single application code instruction before returning to the debug mode.

STOPMODE— Debug mode in which no interrupts are serviced and application code is not executed.

SUSPEND— See DBGACK.

TARGET— The device (an C28x, for the purposes of this document) being emulated and/or tested. The term 'target" comes a military phrase meaning 'something likely to be destroyed".

TIME-CRITICAL INTERRUPT— An interrupt which must be serviced within a certain time limit, even while debugging the application. To indicate that an interrupt is time-critical, the appropriate EIMR bit is set.

TLR— JTAG Test-logic-reset state. Test/emulation logic is reset/disabled while this signal is active. This is will be the case when an emulator is not connected or the emulator is reset.

VISIBILITY— The ability to determine what a device is doing - what it's registers and memory contents are, what instructions it is about to execute, etc.

Basic Debug Action Quick Reference

These actions are carried out using the JTAG interface. The data and instructions written to the various registers and memory locations below all pass through the GPSR and EXSR. From there, the contents are directed to a scan-accessible register or memory location. The read path is similar in the opposite direction. (Intermediate steps may be required as shown in Figure 3-1.)

Task	Steps
Run	Write 1 to Miscellaneous Function Register 0 (MF_REG_0) bit 15 (EXE_LD)
	Write 10b (EXE_COND) to bits 9:8 (EXE_DIR field) of MF_REG_0
Halt	Write 1 to Miscellaneous Function Register 0 (MF_REG_0) bit 15 (EXE_LD)
	Write 00b (EXE_HALT) to bits 9:8 (EXE_DIR field) of MF_REG_0
Write Memory	Scan LD_DMA_WD command into JTAG IR (111100b)
	Scan data to be written into GPSR, advance to Run-test/Idle state.
	Scan SYS_ACC_W command into JTAG IR (1100zzb)
	Scan desired register address into GPSR, set zz field to 10b, advance to Run-test/Idle state.
	Write 11b into zz field, shift out register contents from GPSR
Read Memory	Use SYS_ACC_R command in JTAG IR (1101zzb)
	Shift desired register address into GPSR, set zz field to 10b, advance to Run-test/Idle state.
	Scan out register contents from GPSR. DTYPE vslur from EXSR will be 001
Write Register	Scan appropriate register write command (Table 3-1) into JTAG IR
	Scan data to be written into GPSR, advance to Run-test/Idle state.
	Scan appropriate register command (Table 3-1) command into JTAG IR
	Scan out register contents from GPSR
Read Register	Scan appropriate register read command (Table 3-1) into JTAG IR
	Scan data to be written into GPSR, advance to Run-test/Idle state.
	Scan appropriate register command (Table 3-1) command into JTAG IR
	Scan out register contents from GPSR
Single-Step	Write 1 to Miscellaneous Function Register 0 (MF_REG_0) bit 15 (EXE_LD)
	Write 01b (EXE_) to bits 9:8 (EXE_DIR field) of MF_REG_0

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Digital Control	www.ti.com/digitalcontrol
DSP	dsp.ti.com	Energy	www.ti.com/energyapps
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265