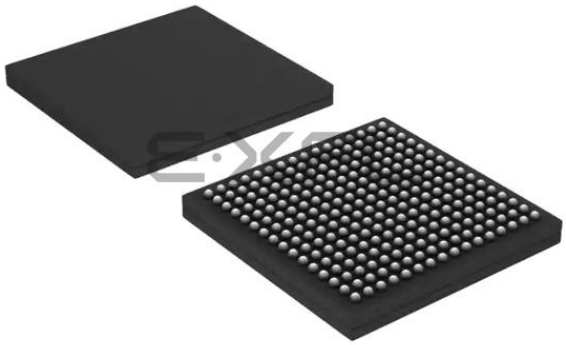


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	Coldfire V2
Core Size	32-Bit Single-Core
Speed	66MHz
Connectivity	CANbus, EBI/EMI, Ethernet, I²C, SPI, UART/USART
Peripherals	DMA, LVD, POR, PWM, WDT
Number of I/O	150
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	256-LBGA
Supplier Device Package	256-MAPBGA (17x17)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mcf5282cvm66j

- User’s manuals — These books provide details about individual ColdFire implementations and are intended to be used in conjunction with the *ColdFire Programmers Reference Manual.*
- Data sheets — Data sheets provide specific data regarding pin-out diagrams, bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Product briefs — Each device has a product brief that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of an device’s reference manual.
- Application notes — These short documents address specific design issues useful to programmers and engineers working with Freescale Semiconductor processors.

Additional literature is published as new processors become available. For a current list of ColdFire documentation, refer to http://www.freescale.com/coldfire.

Conventions

This document uses the following notational conventions:

MNEMONICS	In text, instruction mnemonics are shown in uppercase.
`mnemonics`	In code and tables, instruction mnemonics are shown in lowercase.
italics	Italics indicate variable command parameters. Book titles in text are set in italics.
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
REG[FIELD]	Abbreviations for registers are shown in uppercase. Specific bits, fields, or ranges appear in brackets. For example, RAMBAR[BA] identifies the base address field in the RAM base address register.
nibble	A 4-bit data unit
byte	An 8-bit data unit
word	A 16-bit data unit[1]
longword	A 32-bit data unit
x	In some contexts, such as signal encodings, x indicates a don’t care.
n	Used to express an undefined numerical value
~	NOT logical operator
&	AND logical operator
|	OR logical operator

[1]The only exceptions to this appear in the discussion of serial communication modules that support variable-length data transmission units. To simplify the discussion these units are referred to as words regardless of length.

Although the multiplier array is implemented in a four-stage pipeline, all arithmetic MAC instructions have an effective issue rate of 1 cycle, regardless of input operand size or type.

All arithmetic operations use register-based input operands, and summed values are stored in an accumulator. Therefore, an additional MOVE instruction is needed to store data in a general-purpose register. One new feature in EMAC instructions is the ability to choose the upper or lower word of a register as a 16-bit input operand. This is useful in filtering operations if one data register is loaded with the input data and another is loaded with the coefficient. Two 16-bit multiply accumulates can be performed without fetching additional operands between instructions by alternating word choice during calculations.

The EMAC has four accumulator registers versus the MAC's single accumulator. The additional registers improve the performance of some algorithms by minimizing pipeline stalls needed to store an accumulator value back to general-purpose registers. Many algorithms require multiple calculations on a given data set. By applying different accumulators to these calculations, it is often possible to store one accumulator without any stalls while performing operations involving a different destination accumulator.

The need to move large amounts of data presents an obstacle to obtaining high throughput rates in DSP engines. Existing ColdFire instructions can accommodate these requirements. A MOVEM instruction can efficiently move large data blocks by generating line-sized burst references. The ability to load an operand simultaneously from memory into a register and execute a MAC instruction makes some DSP operations such as filtering and convolution more manageable.

The programming model includes a mask register (MASK), which can optionally be used to generate an operand address during MAC + MOVE instructions. The register application with auto-increment addressing mode supports efficient implementation of circular data queues for memory operands.

3.3.1 Fractional Operation Mode

This section describes behavior when the fractional mode is used (MACSR[F/I] is set).

3.3.1.1 Rounding

When the processor is in fractional mode, there are two operations during which rounding can occur:

1. Execution of a store accumulator instruction (`move.l ACCx,Rx`). The lsbs of the 48-bit accumulator logic are used to round the resulting 16- or 32-bit value. If MACSR[S/U] is cleared, the low-order 8 bits round the resulting 32-bit fraction. If MACSR[S/U] is set, the low-order 24 bits are used to round the resulting 16-bit fraction.
2. Execution of a MAC (or MSAC) instruction with 32-bit operands. If MACSR[R/T] is zero, multiplying two 32-bit numbers creates a 64-bit product truncated to the upper 40 bits; otherwise, it is rounded using round-to-nearest (even) method.

To understand the round-to-nearest-even method, consider the following example involving the rounding of a 32-bit number, R0, to a 16-bit number. Using this method, the 32-bit number is rounded to the closest 16-bit number possible. Let the high-order 16 bits of R0 be named R0.U and the low-order 16 bits be R0.L.

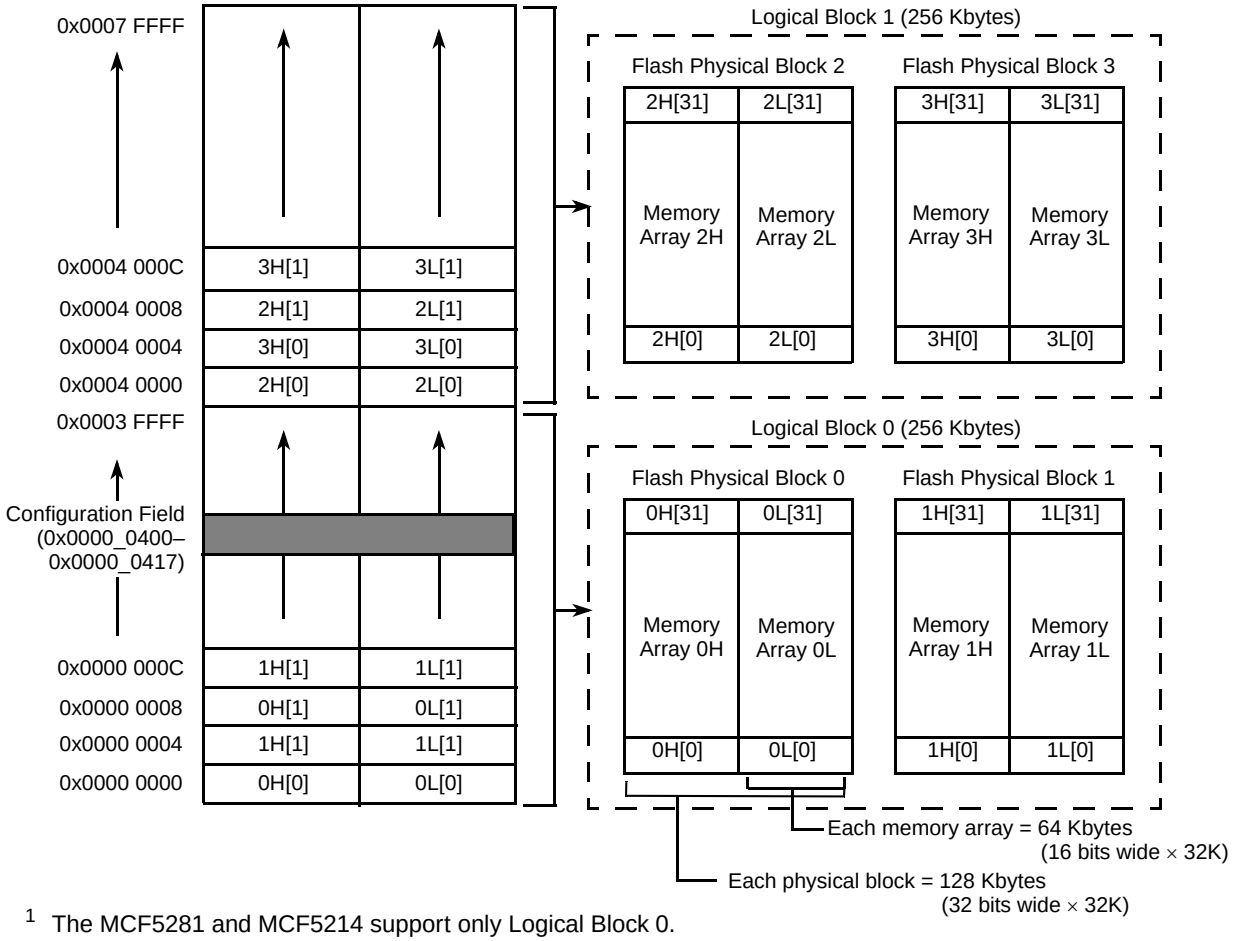
- If R0.L is less than 0x8000, the result is truncated to the value of R0.U.
- If R0.L is greater than 0x8000, the upper word is incremented (rounded up).

6.3 Memory Map

Figure 6-2 shows the memory map for the CFM array. The CFM array can reside anywhere in the memory space of the MCU. The starting address of the array is determined by the CFM array base address which must reside on a natural size boundary; that is, the CFM array base address must be an integer multiple of the array size. The CFM register space must reside on a 64 byte boundary as determined by the CFM register base address. Figure 6-2 shows how multiple 32,768 by 16-bit Flash physical blocks interleave to form a contiguous non-volatile memory space. Each pair of 32-bit blocks (even and odd) interleave every 4 bytes to form a 256-Kbyte section of memory.

NOTE

The CFM on the MCF5281 and MCF5214 is constructed with four banks of 32K x 16-bit Flash arrays to generate 256 Kbytes of 32-bit Flash memory.



[1] The MCF5281 and MCF5214 support only Logical Block 0.

Figure 6-2. CFM Array Memory Map

The CFM module has hardware interlocks to protect data from accidental corruption. The <<BLOCK NAME>> memory array is logically divided into 16-Kbyte sectors for the purpose of data protection and access control. A flexible scheme allows the protection of any combination of logical sectors (see Section 6.3.4.4, “CFM Protection Register (CFMPROT)”). A similar mechanism is available to control supervisor/user and program/data space access to these sectors.

7.3.2.3 Flash

The Flash module is in a low-power state if not being accessed. No recovery time is required after exit from any low-power mode.

The MCF5280 does not have a Flash module.

7.3.2.4 System Control Module (SCM)

The SCM's core Watchdog timer can bring the device out of all low-power modes except stop mode. In stop mode, all clocks stop, and the core Watchdog does not operate.

When enabled, the core Watchdog can bring the device out of low-power mode in one of two ways. If the core Watchdog reset/interrupt select (CSRI) bit is set, then a core Watchdog timeout will cause a reset of the device. If the CSRI bit is cleared, then a core Watchdog interrupt may be enabled and upon watchdog timeout, can bring the device out of low-power mode. This system setup must meet the conditions specified in Section 7.3.1, "Low-Power Modes" for the core Watchdog interrupt to bring the part out of low-power mode.

7.3.2.5 SDRAM Controller (SDRAMC)

SDRAMC operation is unaffected by either the wait or doze modes; however, the SDRAMC is disabled by stop mode. Since all clocks to the SDRAMC are disabled by stop mode, the SDRAMC will not generate refresh cycles.

To prevent loss of data the SDRAM should be placed in self-refresh mode by setting DCR[IS] before entering stop mode. The SDRAM self-refresh mode allows the SDRAM to enter a low-power state where internal refresh operations are used to maintain the integrity of the data stored in the SDRAM.

When stop mode is exited clearing the DCR[IS] bit will cause the SDRAM to exit the self-refresh mode and allow bus cycles to the SDRAM to resume.

NOTE

The SDRAM is inaccessible while in the self-refresh mode. Therefore, if stop mode is used the vector table and any interrupt handlers that could wake the processor should not be stored in or attempt to access SDRAM.

7.3.2.6 Chip Select Module

In wait and doze modes, the chip select module continues operation but does not generate interrupts; therefore it cannot bring a device out of a low-power mode. This module is stopped in stop mode.

7.3.2.7 DMA Controller (DMAC0–DMA3)

In wait and doze modes, the DMA controller is capable of bringing the device out of a low-power mode by generating an interrupt either upon completion of a transfer or upon an error condition. The completion of transfer interrupt is generated when DMA interrupts are enabled by the setting of the DCR[INT] bit, and an interrupt is generated when the DSR[DONE] bit is set. The interrupt upon error condition is generated when the DCR[INT] bit is set, and an interrupt is generated when either the CE, BES or BED bit in the DSR becomes set.

The DMA controller is stopped in stop mode and thus cannot cause an exit from this low-power mode.

7.3.2.8 UART Modules (UART0, UART1, and UART2)

In wait and doze modes, the UART may generate an interrupt to exit the low-power modes.

- Clearing the transmit enable bit (TE) or the receiver enable bit (RE) disables UART functions.
- The UARTs are unaffected by wait mode and may generate an interrupt to exit this mode.

In stop mode, the UARTs stop immediately and freeze their operation, register values, state machines, and external pins. During this mode, the UART clocks are shut down. Coming out of stop mode returns the UARTs to operation from the state prior to the low-power mode entry.

7.3.2.9 I^2C Module

When the I^2C Module is enabled by the setting of the I2CR[IEN] bit and when the device is not in stop mode, the I^2C module is operable and may generate an interrupt to bring the device out of a low-power mode. For an interrupt to occur, the I2CR[IIE] bit must be set to enable interrupts, and the setting of the I2SR[IIF] generates the interrupt signal to the CPU and interrupt controller. The setting of I2SR[IIF] signifies either the completion of one byte transfer or the reception of a calling address matching its own specified address when in slave receive mode.

In stop mode, the I^2C Module stops immediately and freezes operation, register values, and external pins. Upon exiting stop mode, the I^2C resumes operation unless stop mode was exited by reset.

7.3.2.10 Queued Serial Peripheral Interface (QSPI)

In wait and doze modes, the queued serial peripheral interface (QSPI) may generate an interrupt to exit the low-power modes.

- Clearing the QSPI enable bit (SPE) disables the QSPI function.
- The QSPI is unaffected by wait mode and may generate an interrupt to exit this mode.

In stop mode, the QSPI stops immediately and freezes operation, register values, state machines, and external pins. During this mode, the QSPI clocks are shut down. Coming out of stop mode returns the QSPI to operation from the state prior to the low-power mode entry.

7.3.2.11 DMA Timers (DMAT0–DMAT3)

In wait and doze modes, the DMA timers may generate an interrupt to exit a low-power mode. This interrupt can be generated when the DMA Timer is in either input capture mode or reference compare mode.

In input capture mode, where the capture enable (CE) field of the timer mode register (DTMR) has a non-zero value and the DMA enable (DMAEN) bit of the DMA timer extended mode register (DTXMR) is cleared, an interrupt is issued upon a captured input. In reference compare mode, where the output reference request interrupt enable (ORRI) bit of DTMR is set and the DTXMR[DMAEN] bit is cleared, an interrupt is issued when the timer counter reaches the reference value.

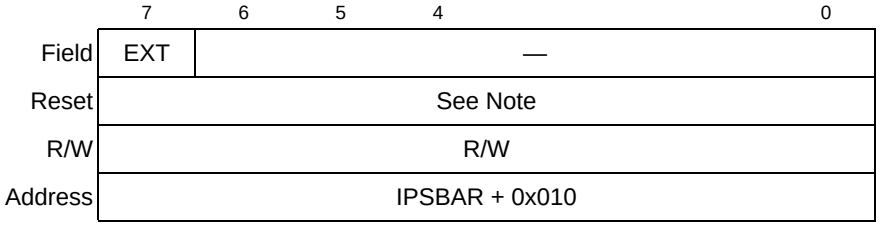
DMA timer operation is disabled in stop mode, but the DMA timer is unaffected by either the wait or doze modes and may generate an interrupt to exit these modes. Upon exiting stop mode, the timer will resume operation unless stop mode was exited by reset.

8.4.3 Core Reset Status Register (CRSR)

The CRSR contains a bit for two of the reset sources to the CPU. A bit set to 1 indicates the last type of reset that occurred. The CRSR is updated by the control logic when the reset is complete. Only one bit is set at any one time in the CRSR. The register reflects the cause of the most recent reset. To clear a bit, a logic 1 must be written to the bit location; writing a zero has no effect.

NOTE

The reset status register (RSR) in the reset controller module (see Chapter 29, “Reset Controller Module”) provides indication of all reset sources except the core watchdog timer.



Note: The reset value of EXT and CWDR depend on the last reset source. All other bits are initialized to zero.

Figure 8-3. Core Reset Status Register (CRSR)

Table 8-4. CRSR Field Descriptions

Bits	Name	Description
7	EXT	External reset. 1 An external device driving $\overline{\text{RSTI}}$ caused the last reset. Assertion of reset by an external device causes the processor core to initiate reset exception processing. All registers are forced to their initial state.
6-0	—	Reserved.

8.4.4 Core Watchdog Control Register (CWCR)

The core watchdog timer prevents system lockup if the software becomes trapped in a loop with no controlled exit. The core watchdog timer can be enabled or disabled through CWCR[CWE]. By default it is disabled. If enabled, the watchdog timer requires the periodic execution of a core watchdog servicing sequence. If this periodic servicing action does not occur, the timer times out, resulting in a watchdog timer interrupt. If the timer times out and the core watchdog transfer acknowledge enable bit (CWCR[CWTA]) is set, a watchdog timer interrupt is asserted. If a core watchdog timer interrupt acknowledge cycle has not occurred after another timeout, CWT TA is asserted in an attempt to allow the interrupt acknowledge cycle to proceed by terminating the bus cycle. The setting of CWCR[CWTAVAL] indicates that the watchdog timer TA was asserted.

NOTE

The core watchdog timer is available to provide compatibility with the watchdog timer implemented on previous ColdFire devices. However, there is a second watchdog timer available that has new features. See Chapter 18, “Watchdog Timer Module” for more information.

9.7.4.7 PLL Loss of Lock Reset

If the LOLRE bit in the SYNCR is set, a loss of lock condition asserts reset. Reset reinitializes the LOCK and LOCKS flags. Therefore, software must read the LOL bit in the reset status register (RSR) to determine if a loss of lock caused the reset. See Section 29.4.2, “Reset Status Register (RSR).”

To exit reset in PLL mode, the reference must be present, and the PLL must achieve lock.

In external clock mode, the PLL cannot lock. Therefore, a loss of lock condition cannot occur, and the LOLRE bit has no effect.

9.7.4.8 Loss of Clock Detection

The LOCEN bit in the SYNCR enables the loss of clock detection circuit to monitor the input clocks to the phase and frequency detector (PFD). When either the reference or feedback clock frequency falls below the minimum frequency, the loss of clock circuit sets the sticky LOCS flag in the SYNSR.

NOTE

In external clock mode, the loss of clock circuit is disabled.

9.7.4.9 Loss of Clock Reset

The clock module can assert a reset when a loss of clock or loss of lock occurs. When a loss-of-clock condition is recognized, reset is asserted if the LOCRE bit in SYNCR is set. The LOCS bit in SYNSR is cleared after reset. Therefore, the LOC bit must be read in RSR to determine that a loss of clock condition occurred. LOCRE has no effect in external clock mode.

To exit reset in PLL mode, the reference must be present, and the PLL must acquire lock.

Reset initializes the clock module registers to a known startup state as described in Section 9.6, “Memory Map and Registers.”

9.7.4.10 Alternate Clock Selection

Depending on which clock source fails, the loss-of-clock circuit switches the system clocks source to the remaining operational clock. The alternate clock source generates the system clocks until reset is asserted. As Table 9-9 shows, if the reference fails, the PLL goes out of lock and into self-clocked mode (SCM). The PLL remains in SCM until the next reset. When the PLL is operating in SCM, the system frequency depends on the value in the RFD field. The SCM system frequency stated in electrical specifications assumes that the RFD has been programmed to binary 000. If the loss-of-clock condition is due to PLL failure, the PLL reference becomes the system clocks source until the next reset, even if the PLL regains and relocks.

Table 9-9. Loss of Clock Summary

Clock Mode	System Clock Source Before Failure	Reference Failure Alternate Clock Selected by LOC Circuit[1] Until Reset	PLL Failure Alternate Clock Selected by LOC Circuit Until Reset
PLL	PLL	PLL self-clocked mode	PLL reference
External	External clock	None	NA

1 The LOC circuit monitors the reference and feedback inputs to the PFD. See Figure 9-5.

Table 14-4. Pin Reset States at Reset (Single-Chip Mode) (continued)

Signal	Reset	I/O
Debug Support Signals		
JTAG_EN	—	I
DSCLK/$\overline{\text{TRST}}$	—	I
$\overline{\text{BKPT}}$/TMS	—	I
DSI/TDI	—	I
DSO/TDO	High	O
TCLK	—	I
DDATA[3:0]	DDATA{3:0]	O
PST[3:0]	PST[3:0]	O

14.1.2 External Boot Mode

When booting from external memory, the address bus, data bus, and bus control signals will default to their bus functionalities as shown in Table 14-5. As in single-chip mode, the signals listed in Table 14-4 will operate as described above. All other signals will default to GPIO inputs.

Table 14-5. Default Signal Functions After System Reset (External Boot Mode)

Signal	Reset	I/O
A[23:0]	A[23:0]	O
D[31:0]	—	I/O
$\overline{\text{BS}}$[3:0]	High	O
$\overline{\text{OE}}$	High	O
$\overline{\text{TA}}$	—	I
$\overline{\text{TEA}}$	—	I
R/$\overline{\text{W}}$	High	O
SIZ[1:0]	High	O
$\overline{\text{TS}}$	High	O
$\overline{\text{TIP}}$	High	O
$\overline{\text{CS}}$[6:0]	High	O

14.2 External Signals

The following sections describe the external signals on the device.

14.2.1 External Interface Module (EIM) Signals

These signals are used for doing transactions on the external bus.

14.2.10.4 Request-to-Send ($\overline{\text{URTS}}$[1:0])

The $\overline{\text{URTS}}$[1:0] signals are automatic request to send outputs from the UART modules. $\overline{\text{URTS}}$[1:0] can also be configured to be asserted and negated as a function of the Rx FIFO level.

The $\overline{\text{URTS}}$[1:0] outputs are each offered as secondary functions on four pins: DTIN3, DTOUT3, DTIN1 and DTOUT1.

14.2.11 General Purpose Timer Signals

These pins provide the external interface to the general purpose timer functions.

14.2.11.1 GPTA[3:0]

These pins provide the external interface to the timer A functions.

These pins can also be configured as GPIO PTA[3:0].

14.2.11.2 GPTB[3:0]

These pins provide the external interface to the timer B functions.

These pins can also be configured as GPIO PTB[3:0].

14.2.11.3 External Clock Input (SYNCA/SYNCB)

These pins are used to clear the clock for each of the two timers, and are provided as a means of synchronization to externally clocked or timed events.

14.2.12 DMA Timer Signals

This section describes the signals of the four DMA timer modules.

14.2.12.1 DMA Timer 0 Input (DTIN0)

The DMA timer 0 input (DTIN0) can be programmed to cause events to occur in DMA timer 0. It can either clock the event counter or provide a trigger to the timer value capture logic.

This pin can also be configured as GPIO PTD1, secondary function $\overline{\text{UCTS1}}$, or secondary function $\overline{\text{UCTS0}}$.

14.2.12.2 DMA Timer 0 Output (DTOUT0)

The programmable DMA timer output (DTOUT0) pulse or toggle on various timer events.

This pin can also be configured as GPIO PTD0, secondary function $\overline{\text{UCTS1}}$, or secondary function $\overline{\text{UCTS0}}$.

14.2.12.3 DMA Timer 1 Input (DTIN1)

The DMA timer 1 input (DTIN1) can be programmed to cause events to occur in DMA timer 1. This can either clock the event counter or provide a trigger to the timer value capture logic.

22.3.3 QSPI Wrap Register (QWR)

The QSPI wrap register provides halt transfer control, wraparound settings, and queue pointer locations.

IPSBAR Offset: 0x00_0348 (QWR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HALT	WREN	WRTO	CSIV	ENDQP				CPTQP				NEWQP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-5. QSPI Wrap Register (QWR)

Table 22-5. QWR Field Descriptions

Field	Description
15 HALT	Halt transfers. Assertion of this bit causes the QSPI to stop execution of commands after it has completed execution of the current command.
14 WREN	Wraparound enable. Enables wraparound mode. 0 Execution stops after executing the command pointed to by QWR[ENDQP]. 1 After executing command pointed to by QWR[ENDQP], wrap back to entry zero, or the entry pointed to by QWR[NEWQP] and continue execution.
13 WRTO	Wraparound location. Determines where the QSPI wraps to in wraparound mode. 0 Wrap to RAM entry zero. 1 Wrap to RAM entry pointed to by QWR[NEWQP].
12 CSIV	QSPI_CS inactive level. 0 QSPI chip select outputs return to zero when not driven from the value in the current command RAM entry during a transfer (that is, inactive state is 0, chip selects are active high). 1 QSPI chip select outputs return to one when not driven from the value in the current command RAM entry during a transfer (that is, inactive state is 1, chip selects are active low).
11–8 ENDQP	End of queue pointer. Points to the RAM entry that contains the last transfer description in the queue.
7–4 CPTQP	Completed queue entry pointer. Points to the RAM entry that contains the last command to have been completed. This field is read only.
3–0 NEWQP	Start of queue pointer. This 4-bit field points to the first entry in the RAM to be executed on initiating a transfer.

22.3.4 QSPI Interrupt Register (QIR)

The QIR contains QSPI interrupt enables and status flags.

IPSBAR Offset: 0x00_034C (QIR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WCEFB	ABRTB	0	ABRTL	WCEFE	ABRTE	0	SPIFE	0	0	0	0	WCEF	ABRT	0	SPIF
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-6. QSPI Interrupt Register (QIR)

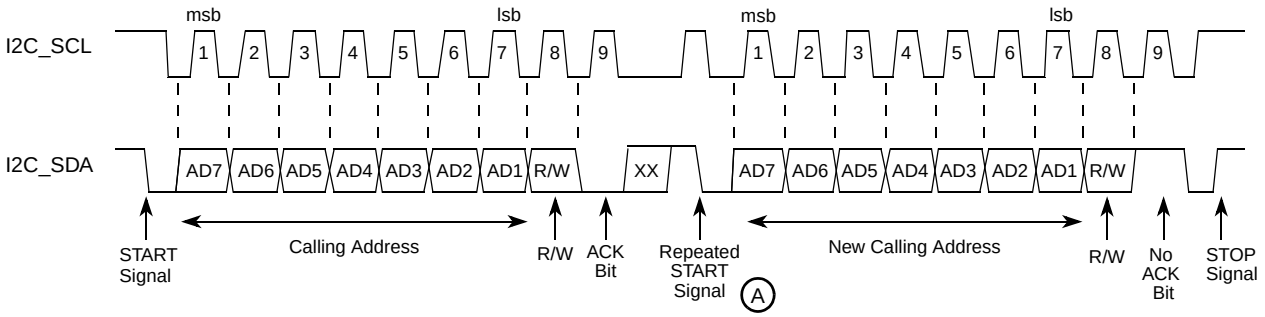


Figure 24-10. Repeated START

Various combinations of read/write formats are then possible:

- The first example in Figure 24-11 is the case of master-transmitter transmitting to slave-receiver. The transfer direction is not changed.
- The second example in Figure 24-11 is the master reading the slave immediately after the first byte. At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes slave-transmitter.
- In the third example in Figure 24-11, START condition and slave address are repeated using the repeated START signal. This is to communicate with same slave in a different mode without releasing the bus. The master transmits data to the slave first, and then the master reads data from slave by reversing the R/$\overline{W}$ bit.

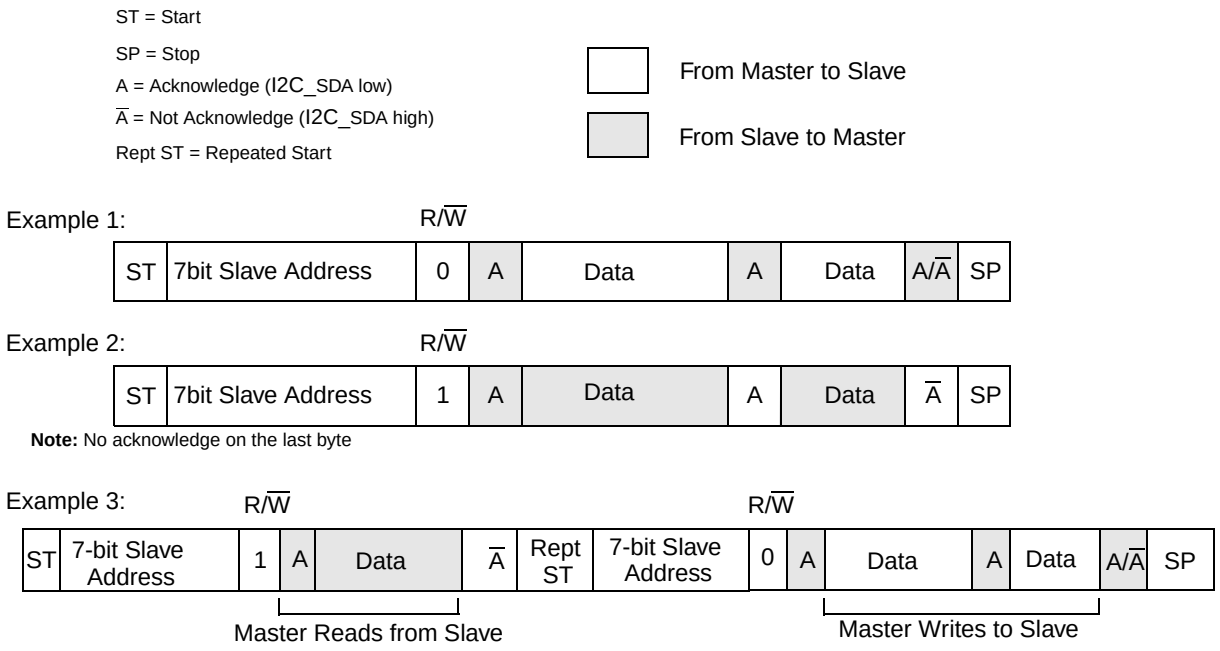


Figure 24-11. Data Transfer, Combined Format

24.3.7 Clock Synchronization and Arbitration

I²C is a true multi-master bus that allows more than one master connected to it. If two or more master devices simultaneously request control of the bus, a clock synchronization procedure determines the bus clock. Because wire-AND logic is performed on the I2C_SCL line, a high-to-low transition on the I2C_SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the I2C_SCL line low until the clock high state is reached. However, change of low to high in this device's clock may not change the state of the I2C_SCL line if another device clock remains within its low period. Therefore, synchronized clock I2C_SCL is held low by the device with the longest low period.

Devices with shorter low periods enter a high wait state during this time (see Figure 24-12). When all devices concerned have counted off their low period, the synchronized clock (I2C_SCL) line is released and pulled high. At this point, the device clocks and the I2C_SCL line are synchronized, and the devices start counting their high periods. The first device to complete its high period pulls the I2C_SCL line low again.

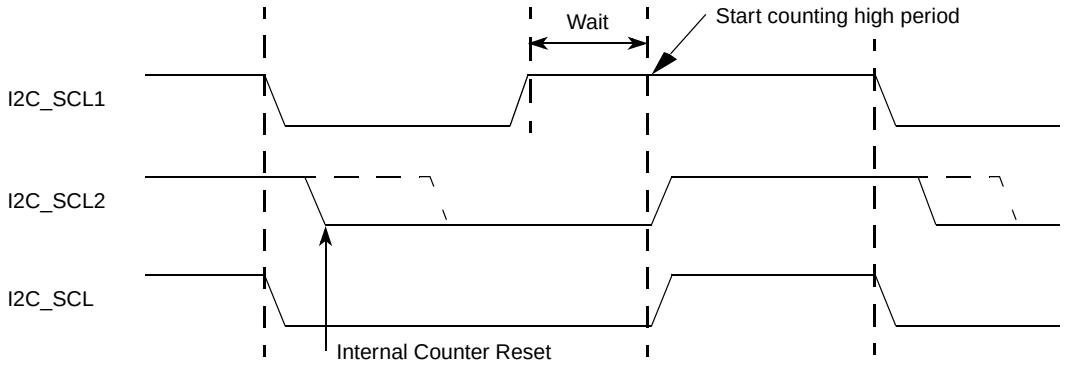


Figure 24-12. Clock Synchronization

A data arbitration procedure determines the relative priority of the contending masters. A bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving I2C_SDA output (see Figure 24-13). In this case, transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets I2SR[IAL] to indicate loss of arbitration.

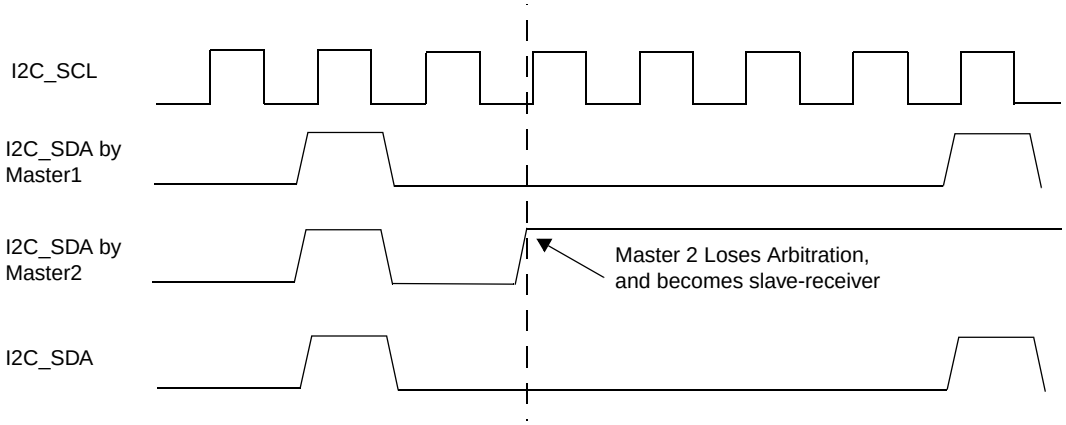


Figure 24-13. Arbitration Procedure

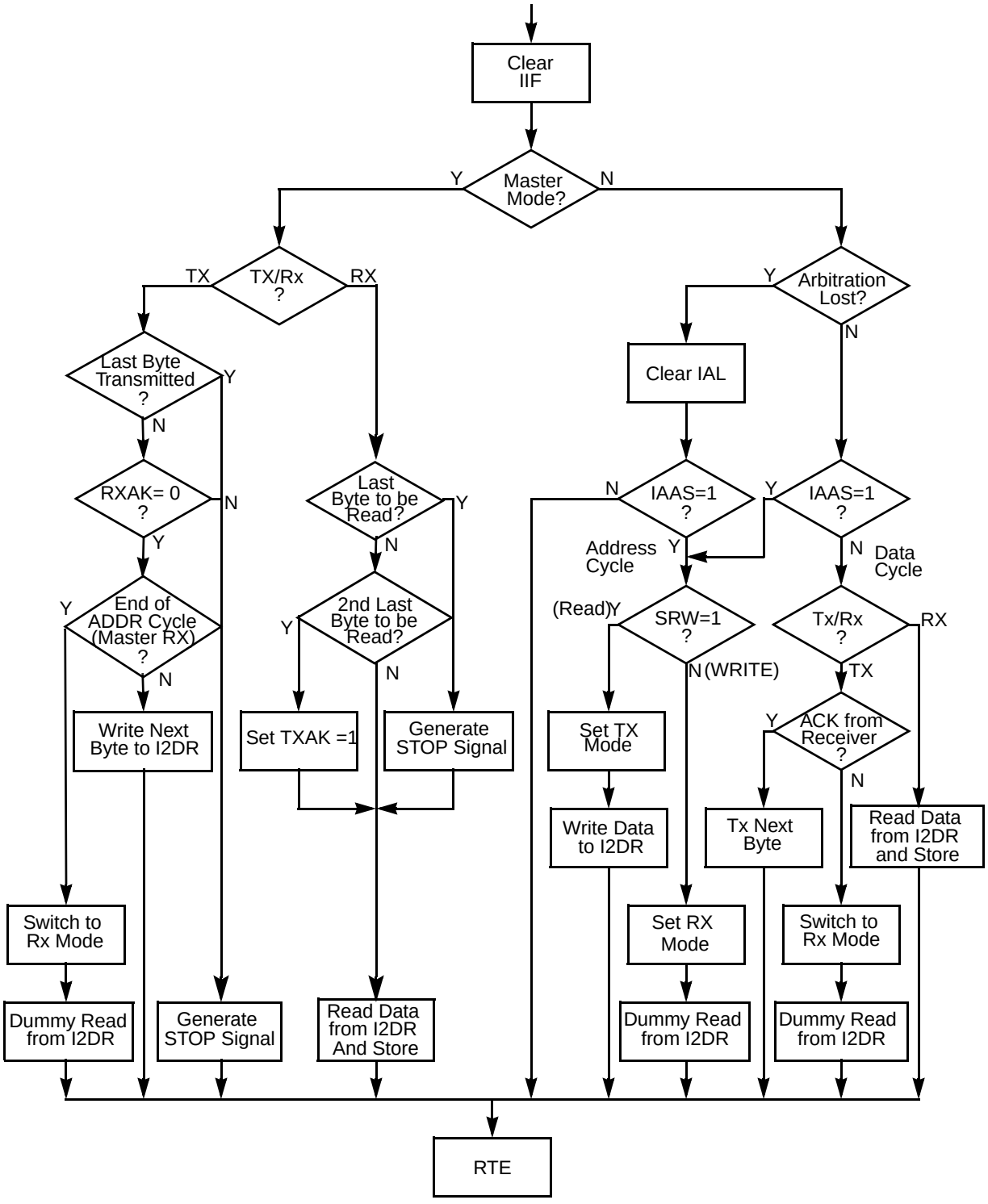


Figure 24-14. Flow-Chart of Typical I²C Interrupt Routine

Table 25-8. CANMCR Field Descriptions (continued)

Bits	Name	Description
9	SOFTRST	Soft reset. When this bit is asserted, the FlexCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal FlexCAN circuitry to completely reset before executing another access to CANMCR. The FlexCAN clears this bit once the internal reset cycle is completed. 0 Soft reset cycle completed 1 Soft reset cycle initiated
8	FRZACK	FlexCAN disable. When the FlexCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the FlexCAN has entered debug mode. When debug mode is exited, this bit is negated once the FlexCAN prescaler is enabled. This is a read-only bit. 0 The FlexCAN has exited debug mode and the prescaler is enabled. 1 The FlexCAN has entered debug mode, and the prescaler is disabled.
7	SUPV	Supervisor/user data space. The SUPV bit places the FlexCAN registers in either supervisor or user data space. 0 Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode. 1 Registers with access controlled by the SUPV bit are restricted to supervisor mode.
6	SELF-WAKE	Self wake enable. This bit allows the FlexCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the FlexCAN enters low-power stop mode, the FlexCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the FlexCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to Section 25.4.11.2, “Low-Power Stop Mode for Power Saving” for more information on entry into and exit from low-power stop mode. 0 Self wake disabled. 1 Self wake enabled.
5	APS	Auto-power save. The APS bit allows the FlexCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 Auto-power save mode disabled; clocks run normally. 1 Auto-power save mode enabled; clocks stop and restart as needed.
4	STOPACK	Stop acknowledge. When the FlexCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the FlexCAN has entered low-power stop mode. When the FlexCAN exits low-power stop mode, the STOPACK bit is cleared once the FlexCAN’s clocks are running. 0 The FlexCAN is not in low-power stop mode and its clocks are running. 1 The FlexCAN has entered low-power stop mode and its clocks are stopped
3–0	—	Reserved, should be cleared.

	7	6	5	4	3	2	1	0
Field	—				DDQB3	DDQB2	DDQB1	DDQB0
Reset	0000_0000							
R/W	R							
Address	IPSBAR + 0x19_0009							

Figure 28-7. Port QB Data Direction Register (DDRQB)

28.6.5 Control Registers

This subsection describes the QADC control registers.

28.6.5.1 QADC Control Register 0 (QACR0)

QACR0 establishes the QADC sampling clock (QCLK) with prescaler parameter fields and defines whether external multiplexing is enabled. Typically, these bits are written once when the QADC is initialized and not changed thereafter. The bits in this register are read anytime, write anytime (except during stop mode).

	15	14	13	12	11	8
Field	MUX	—		TRG	—	
Reset	0000_0000					
R/W:	R/W	R		R/W	R	

	7	6	5	4	3	2	1	0
Field	—	QPR6	QPR5	QPR4	QPR3	QPR2	QPR1	QPR0
Reset	0001_0011							
R/W:	R	R/W						
Address	IPSBAR + 0x19_000a, 0x19_000b							

Figure 28-8. QADC Control Register 0 (QACR0)

Table 28-4. QACR0 Field Descriptions

Bit(s)	Name	Description
15	MUX	Externally multiplexed mode. Configures the QADC for operation in externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[1:0] signals to be outputs. 1 Externally multiplexed, up to 18 possible channels 0 Internally multiplexed, up to 8 possible channels
14–13	—	Reserved, should be cleared.
12	TRG	Trigger assignment. Determines the queue assignment of the ETRIG[2:1] signals. 1 ETRIG1 triggers queue 2; ETRIG2 triggers queue 1. 0 ETRIG1 triggers queue 1; ETRIG2 triggers queue 2.

[1] All channels not listed are reserved or unimplemented and return undefined results.

28.6.8 Result Registers

The result word table is a 64 half-word (128 byte) long by 10-bit wide RAM. An entry is written by the QADC after completing an analog conversion specified by the corresponding CCW table entry.

28.6.8.1 Right-Justified Unsigned Result Register (RJURR)

	15 … 10	9 … 8
Field	—	RESULT
Reset	0000_00	Undefined
R/W:	R	R/W

	7 … 0
Field	RESULT
Reset	Undefined
R/W:	R/W
Address	IPSBAR + 0x19_0280, 0x19_02fe

Figure 28-15. Right-Justified Unsigned Result Register (RJURR)

Table 28-18. RJURR Field Descriptions

Bit(s)	Name	Description
15–10	—	Reserved, should be cleared.
9–0	RESULT	The conversion result is unsigned, right-justified data.

28.6.8.2 Left-Justified Signed Result Register (LJSRR)

	15	14 … 8
Field	S	RESULT
Reset	Undefined	
R/W:	R/W	

	7 … 0
Field	RESULT
Reset	Undefined
R/W:	R/W
Address	IPSBAR + 0x19_0300, 0x19_037e

Figure 28-16. Left-Justified Signed Result Register (LJSRR)

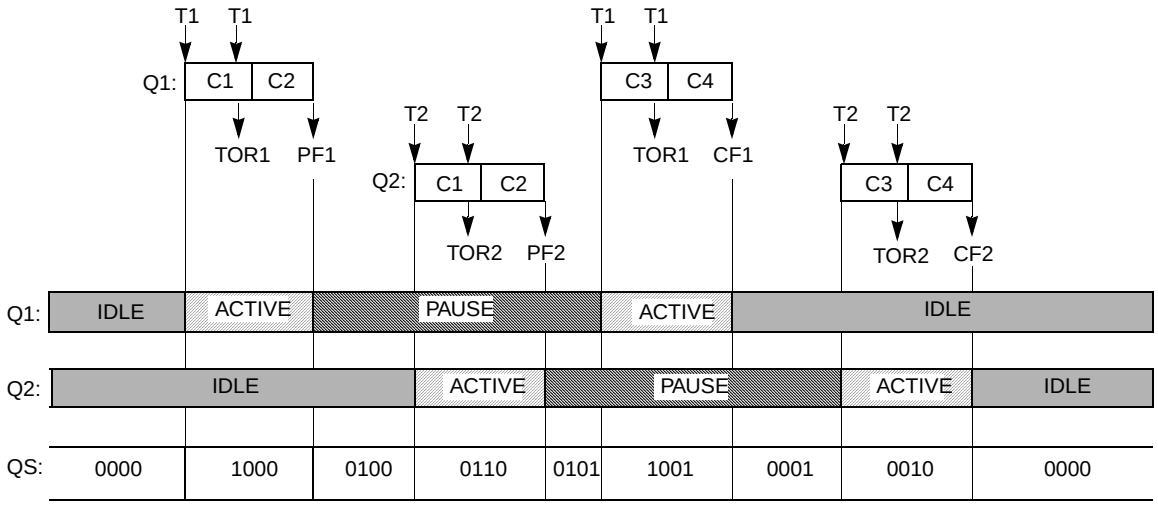


Figure 28-25. CCW Priority Situation 3

The next two situations consider trigger events that occur for the lower priority queue 2, while queue 1 is actively being serviced.

Situation S4 (Figure 28-26) shows that a queue 2 trigger event is recognized while queue 1 is active is saved, and as soon as queue 1 is finished, queue 2 servicing begins.

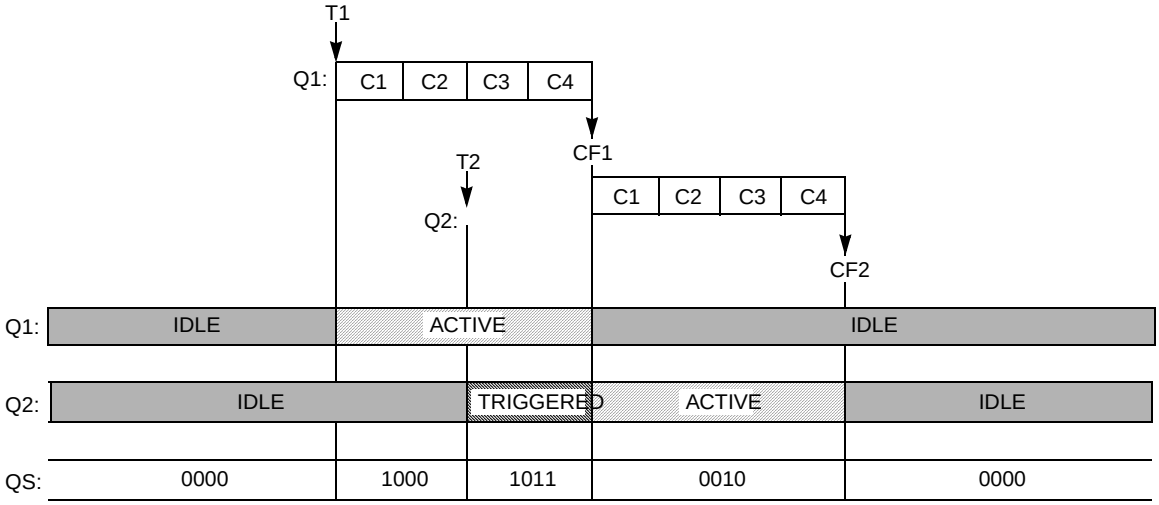


Figure 28-26. CCW Priority Situation 4

Situation S5 (Figure 28-27) shows that when multiple queue 2 trigger events are detected while queue 1 is busy, the trigger overrun error bit is set, but queue 1 execution is not disturbed. Situation S5 also shows that the effect of queue 2 trigger events during queue 1 execution is the same when the pause feature is used for either queue.

Table 30-6 describes ABLR fields.

Table 30-6. ABLR Field Description

Bits	Name	Description
31–0	Address	Low address. Holds the 32-bit address marking the lower bound of the address breakpoint range. Breakpoints for specific addresses are programmed into ABLR.

Table 30-7 describes ABHR fields.

Table 30-7. ABHR Field Description

Bits	Name	Description
31–0	Address	High address. Holds the 32-bit address marking the upper bound of the address breakpoint range.

30.4.4 Configuration/Status Register (CSR)

The CSR defines the debug configuration for the processor and memory subsystem and contains status information from the breakpoint logic. CSR is write-only from the programming model. It can be read from and written to through the BDM port. CSR is accessible in supervisor mode as debug control register 0x00 using the WDEBUG instruction and through the BDM port using the RDMREG and WDMREG commands.

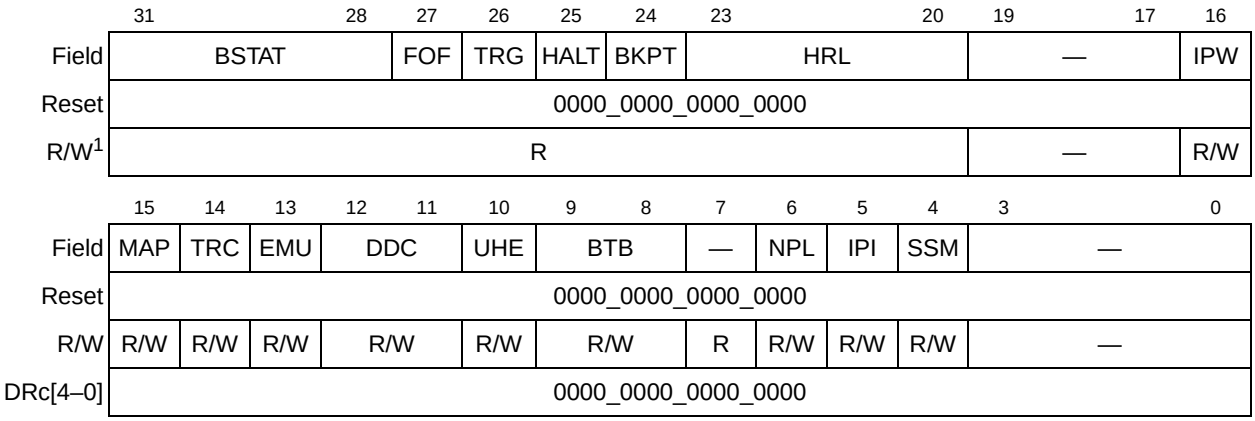


Figure 30-7. Configuration/Status Register (CSR)

30.5.3.3.4 Write Memory Location (WRITE)

Write data to the memory location specified by the longword address. The address space is defined by BAAR[TT,TM]. Hardware forces low-order address bits to zeros for word and longword accesses to ensure that word addresses are word-aligned and longword addresses are longword-aligned.

Command Formats:

	15–12	11–8	7–4	3–1
Byte	0x1	0x8	0x0	0x0
	A[31:16]			
	A[15:0]			
	X X X X X X X X	D[7:0]		
Word	0x1	0x8	0x4	0x0
	A[31:16]			
	A[15:0]			
	D[15:0]			
Longword	0x1	0x8	0x8	0x0
	A[31:16]			
	A[15:0]			
	D[31:16]			
	D[15:0]			

Figure 30-23. WRITE Command Format

Table 32-1. Signal Description by Pin Number (continued)

MAPBGA Pin	Pin Functions			MAPBGA Pin	Pin Functions		
	Primary	Secondary	Tertiary		Primary	Secondary	Tertiary
G12	VDD	—	—	R12	GPTB2	PTB2	—
G13	QSPI_CS2	PQS5	—	R13	GPTA2	PTA2	—
G14	QSPI_CS3	PQS6	—	R14	CLKMOD0	—	—
G15	$\overline{\text{DRAMW}}$	PSD3	—	R15	$\overline{\text{BS1}}$	PJ5	—
G16	$\overline{\text{SDRAM\_CS0}}$	PSD1	—	R16	$\overline{\text{BS0}}$	PJ4	—
H1	D26	PA2	—	T1	VSSA	—	—
H2	D25	PA1	—	T2	AN2	PQB2	ANY
H3	D24	PA0	—	T3	AN0	PQB0	ANW
H4	D23	PB7	—	T4	AN53	PQA1	MA1
H5	VDD	—	—	T5	VRL	—	—
H6	VDD	—	—	T6	D2	PD2	—
H7	VSS	—	—	T7	UTXD0	PUA0	—
H8	VSS	—	—	T8	EXTAL	—	—
H9	VSS	—	—	T9	TCLK	—	—
H10	VSS	—	—	T10	DSO	TDO	—
H11	VDD	—	—	T11	$\overline{\text{RCON}}$	—	—
H12	VDD	—	—	T12	GPTB3	PTB3	—
H13	$\overline{\text{SDRAM\_CS1}}$	PSD2	—	T13	GPTA3	PTA3	—
H14	SCKE	PSD0	—	T14	CLKMOD1	—	—
H15	$\overline{\text{SRAS}}$	PSD5	—	T15	$\overline{\text{BS2}}$	PJ6	—
H16	$\overline{\text{SCAS}}$	PSD4	—	T16	VSS	—	—

[1] NC = no connect