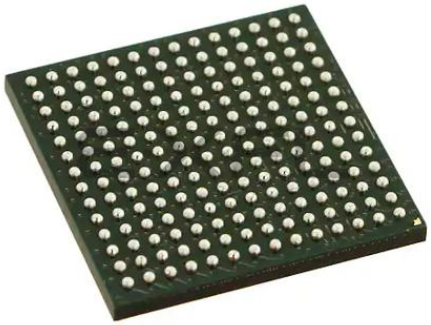

Welcome to E-XFL.COM

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	EBI/EMI, Ethernet, I²C, SPI, UART/USART, USB
Peripherals	DMA, WDT
Number of I/O	32
Program Memory Size	16KB (4K x 32)
Program Memory Type	ROM
EEPROM Size	-
RAM Size	1K x 32
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	196-LBGA
Supplier Device Package	196-LBGA (15x15)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mcf5272vf66r2

List of Figures (Continued)

Table of Contents (Continued)

MCF5272 ColdFire® Integrated Microprocessor

User's Manual

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://www.freescale.com/

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Document Revision History

Rev. No.	Substantive Change(s)
2.1	Updated to meet Freescale identity guidelines.
3	• Formatting, layout, spelling, and grammar corrections. • Corrected the TxFIFO bit description In Table 16-9 (was "Once set, this bit is cleared by reading UTB*n*", is "After being set, this bit is cleared by writing UTB*n*"). • Corrected Figure 20-12 ($\overline{OE}$ signal was asserting on the third SDCLK clock cycle, is asserting on the second SDCLK clock cycle). • Corrected Figure 20-13 (R/$\overline{W}$ and $\overline{BS}$ signals were asserting on the third SDCLK clock cycle, are asserting on the second SDCLK clock cycle). • Corrected Figure 20-16 ($\overline{OE}$ signal was asserting on the third SDCLK clock cycle, is asserting on the second SDCLK clock cycle). • Corrected Figure 20-17 (R/$\overline{W}$ and $\overline{BS}$ signals were asserting on the third SDCLK clock cycle, are asserting on the second SDCLK clock cycle).

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.

© Freescale Semiconductor, Inc., 2005. All rights reserved.

For register-to-memory operations, the stage functions (DS/OC, AG/EX) are effectively performed simultaneously allowing single-cycle execution. For read-modify-write instructions, the pipeline effectively combines a memory-to-register operation with a store operation.

2.1.1.2.1 Illegal Opcode Handling

On Version 2 ColdFire implementations, only some illegal opcodes (0x0000 and 0x4AFC) are decoded and generate an illegal instruction exception. Additionally, attempting to execute an illegal line A or line F opcode generates unique exception types. If any other unsupported opcode is executed, the resulting operation is undefined.

2.1.1.2.2 Hardware Multiply/Accumulate (MAC) Unit

The MAC is an optional unit in Version 2 that provides hardware support for a limited set of digital signal processing (DSP) operations used in embedded code, while supporting the integer multiply instructions in the ColdFire microprocessor family. The MAC features a three-stage execution pipeline, optimized for 16 x 16 multiplies. It is tightly coupled to the OEP, which can issue a 16 x 16 multiply with a 32-bit accumulation plus fetch a 32-bit operand in a single cycle. A 32 x 32 multiply with a 32-bit accumulation requires three cycles before the next instruction can be issued.

Figure 2-2 shows basic functionality of the MAC. A full set of instructions are provided for signed and unsigned integers plus signed, fixed-point fractional input operands.

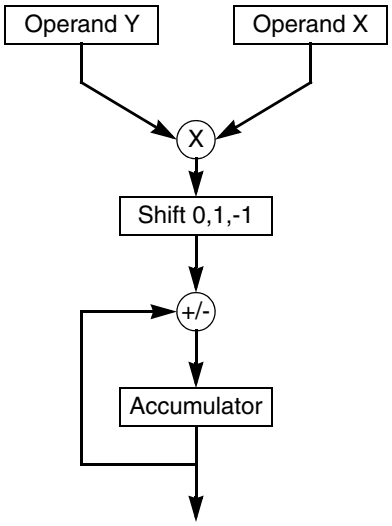

Figure 2-2. ColdFire Multiply-Accumulate Functionality Diagram

The MAC provides functionality in the following three related areas, which are described in detail in Chapter 3, “Hardware Multiply/Accumulate (MAC) Unit.”

- Signed and unsigned integer multiplies
- Multiply-accumulate operations with signed and unsigned fractional operands
- Miscellaneous register operations

2.7 Instruction Timing

The timing data presented in this section assumes the following:

- The OEP is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP spends no time waiting for the IFP to supply opwords and/or extension words.
- The OEP experiences no sequence-related pipeline stalls. For the MCF5272,the most common example of this type of stall involves consecutive store operations, excluding the MOVEM instruction. For all store operations (except MOVEM), certain hardware resources within the processor are marked as busy for two clock cycles after the final DSOC cycle of the store instruction. If a subsequent store instruction is encountered within this two-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive store operations is two cycles.
- The OEP can complete all memory accesses without memory causing any stall conditions. Thus, timing details in this section assume an infinite zero-wait state memory attached to the core.
- All operand data accesses are assumed to be aligned on the same byte boundary as the operand size:
  — 16-bit operands aligned on 0-modulo-2 addresses
  — 32-bit operands aligned on 0-modulo-4 addresses
  Operands that do not meet these guidelines are misaligned. Table 2-9 shows how the core decomposes a misaligned operand reference into a series of aligned accesses.

Table 2-9. Misaligned Operand References

A[1:0]	Size	Bus Operations	Additional C(R/W) [1]
x1	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
x1	Long	Byte, Word, Byte	3(2/0) if read 2(0/2) if write
10	Long	Word, Word	2(1/0) if read 1(0/1) if write

[1] Each timing entry is presented as C(r/w), described as follows:
C is the number of processor clock cycles, including all applicable operand fetches and writes, as well as all internal core cycles required to complete the instruction execution.
r/w is the number of operand reads (r) and writes (w) required by the instruction. An operation performing a read-modify write function is denoted as (1/1).

4.2 Local Memory Registers

Table 4-1 lists the local memory registers. Note the following:

- Addresses not assigned to the register and undefined register bits are reserved. Write accesses to these bits have no effect; read accesses return zeros.
- The reset value column indicates the register initial value at reset. Uninitialized fields may contain random values after reset.

Table 4-1. Memory Map of Instruction Cache Registers

Address (using MOVEC)	Name	Width	Description	Reset Value
0x002	CACR	32	Cache control register	0x0000
0x004	ACR0	32	Access control register 0	0x0000
0x005	ACR1	32	Access control register 1	0x0000
0xC00	ROMBAR	32	ROM base address register	Uninitialized (except V = 0)
0xC04	RAMBAR	32	SRAM base address register	Uninitialized (except V = 0)

4.3 SRAM Overview

The SRAM module has the following features:

- 4-Kbyte SRAM, organized as 1K x 32 bits
- Single-cycle access
- Physically located on the ColdFire core's high-speed local bus
- Byte, word, longword address capabilities
- Programmable memory mapping

4.3.1 SRAM Operation

The SRAM module provides a general-purpose memory block the ColdFire core can access in a single cycle. The location of the memory block can be set to any 4-Kbyte address boundary within the 4-Gbyte address space. The memory is ideal for storing critical code or data structures or for use as the system stack. Because the SRAM module is physically connected to the processor's high-speed local bus, it can quickly service core-initiated accesses or memory-referencing commands from the debug module.

Section 4.1, “Interactions Between Local Memory Modules,” describes priorities when an access address hits multiple local memory resources.

4.3.2 SRAM Programming Model

The MCF5272 implements the SRAM base address register (RAMBAR), shown in Figure 4-1 and described in the following section.

The mapping of a given access into the SRAM uses the following algorithm to determine if the access hits in the memory:

```
if (RAMBAR[0] = 1)
        if (requested address[31:12] = RAMBAR[31:12])
                if (address space mask of the requested type = 0)
                        Access is mapped to the SRAM module
                        if (access = read)
                                Read the SRAM and return the data
                        if (access = write)
                                if (RAMBAR[8] = 0)
                                        Write the data into the SRAM
                                else Signal a write-protect access error
```

4.3.2.2 SRAM Initialization

After a hardware reset, the contents of the SRAM module are undefined. The valid bit of RAMBAR is cleared, disabling the module. If the SRAM needs to be initialized with instructions or data, the following steps should be performed:

1. Load RAMBAR, mapping the SRAM module to the desired location.
2. Read the source data and write it to the SRAM. Various instructions support this function, including memory-to-memory MOVE instructions and the MOVEM opcode. The MOVEM instruction is optimized to generate line-sized burst fetches on 0-modulo-16 addresses, so this opcode generally provides the best performance.
3. After data is loaded into the SRAM, it may be appropriate to load a revised value into RAMBAR with new write-protect and address space mask attributes. These attributes consist of the write-protect and address-space mask fields.

The ColdFire processor or an external BDM emulator using the debug module can perform this initialization.

4.3.2.3 Programming RAMBAR for Power Management

Depending on the configuration defined by RAMBAR, instruction fetch accesses can be sent to the SRAM module, ROM module, and instruction cache simultaneously. If the access is mapped to the SRAM module, it sources the read data, discarding the instruction cache access. If the SRAM is used only for data operands, setting RAMBAR[SC,UC] lowers power dissipation by disabling the SRAM during all instruction fetches. Additionally, if the SRAM holds only instructions, setting RAMBAR[SD,UD] reduces power dissipation.

Consider the examples on Table 4-3 of typical RAMBAR settings:

Table 4-3. Examples of Typical RAMBAR Settings

Data Contained in SRAM	RAMBAR[7–0]
Instructions only	0x2B
Data only	0x35
Both instructions and data	0x21

5.4.7 Trigger Definition Register (TDR)

The TDR, shown in Table 5-11, configures the operation of the hardware breakpoint logic that corresponds with the ABHR/ABLR/AATR, PBR/PBMR, and DBR/DBMR registers within the debug module. The TDR controls the actions taken under the defined conditions. Breakpoint logic may be configured as a one- or two-level trigger. TDR[31–16] define the second-level trigger and bits 15–0 define the first-level trigger.

NOTE

The debug module has no hardware interlocks, so to prevent spurious breakpoint triggers while the breakpoint registers are being loaded, disable TDR (by clearing TDR[29,13])before defining triggers.

A write to TDR clears the CSR trigger status bits, CSR[BSTAT].

Second-Level Trigger

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	TRC		EBL	EDLW	EDWL	EDWU	EDLL	EDLM	EDUM	EDUU	DI	EAI	EAR	EAL	EPC	PCI
Reset	0000_0000_0000_0000															

First-Level Trigger

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		EBL	EDLW	EDWL	EDWU	EDLL	EDLM	EDUM	EDUU	DI	EAI	EAR	EAL	EPC	PCI
Reset	0000_0000_0000_0000															
R/W	Write only. Accessible in supervisor mode as debug control register 0x07 using the WDEBUG instruction and through the BDM port using the WDMREG command.															
DRc[4–0]	0x07															

Figure 5-11. Trigger Definition Register (TDR)

Table 5-14 describes TDR fields.

Table 5-14. TDR Field Descriptions

Bits	Name	Description
31–30	TRC	Trigger response control. Determines how the processor responds to a completed trigger condition. The trigger response is always displayed on DDATA. 00 Display on DDATA only 01 Processor halt 10 Debug interrupt 11 Reserved
15–14	—	Reserved, should be cleared.
29/13	EBL	Enable breakpoint. Global enable for the breakpoint trigger. Setting TDR[EBL] enables a breakpoint trigger. Clearing it disables all breakpoints at that level.

5.5.3.3.3 Read Memory Location (READ)

Read data at the longword address. 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/Result Formats:

		15 12	11 8	7 4	3 0
Byte	Command	0x1	0x9	0x0	0x0
		A[31:16]			
		A[15:0]			
	Result	X X X X X X X X		D[7:0]	
Word	Command	0x1	0x9	0x4	0x0
		A[31:16]			
		A[15:0]			
	Result	D[15:0]			
Longword	Command	0x1	0x9	0x8	0x0
		A[31:16]			
		A[15:0]			
	Result	D[31:16]			
		D[15:0]			

Figure 5-21. READ Command/Result Formats

Command Sequence:

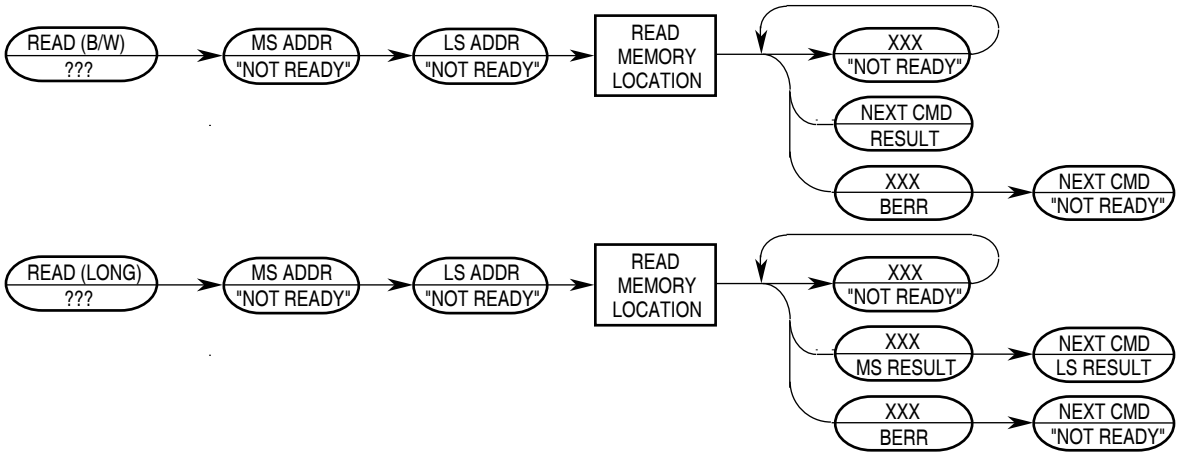

Figure 5-22. READ Command Sequence

Operand Data — The only operand is the longword address of the requested location.

Result Data — Word results return 16 bits of data; longword results return 32. Bytes are returned in the LSB of a word result, the upper byte is undefined. 0x0001 (S = 1) is returned if a bus error occurs.

5.5.3.3.5 Dump Memory Block (DUMP)

DUMP is used with the READ command to access large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. If an initial READ is not executed before the first DUMP, an illegal command response is returned. The DUMP command retrieves subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP commands use this address, perform the memory read, increment it by the current operand size, and store the updated address in the temporary register.

NOTE

DUMP does not check for a valid address; it is a valid command only when preceded by NOP, READ, or another DUMP command. Otherwise, an illegal command response is returned. NOP can be used for intercommand padding without corrupting the address pointer.

The size field is examined each time a DUMP command is processed, allowing the operand size to be dynamically altered.

Command/Result Formats:

		15 12	11 8	7 4	3 0
Byte	Command	0x1	0xD	0x0	0x0
	Result	X X X X	X X X X	D[7:0]	
Word	Command	0x1	0xD	0x4	0x0
	Result	D[15:0]			
Longword	Command	0x1	0xD	0x8	0x0
	Result	D[31:16]			
		D[15:0]			

Figure 5-25. DUMP Command/Result Formats

Command Sequence:

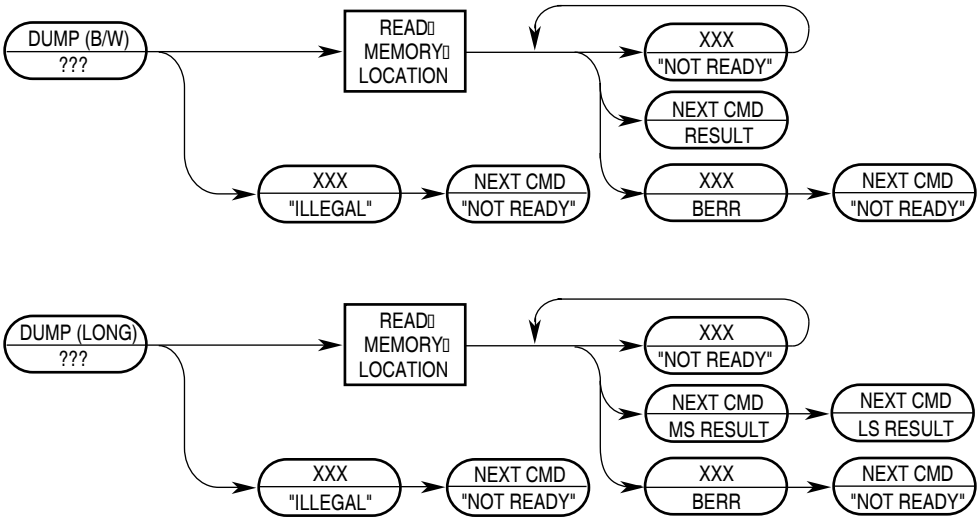

Figure 5-26. DUMP Command Sequence

Command Sequence:

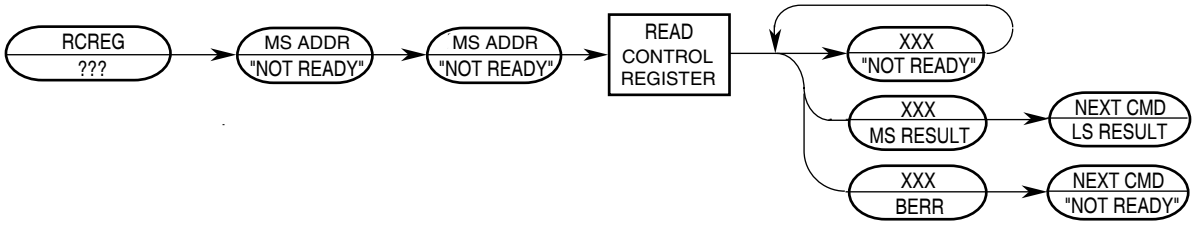

Figure 5-34. RCREG Command Sequence

Operand Data: The only operand is the 32-bit Rc control register select field.

Result Data: Control register contents are returned as a longword, most-significant word first. The implemented portion of registers smaller than 32 bits is guaranteed correct; other bits are undefined.

5.5.3.3.10 Write Control Register (WCREG)

The operand (longword) data is written to the specified control register. The write alters all 32 register bits.

Command/Result Formats:

	15–12	11–8	7–4	3–0
Command	0x2	0x8	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 5-35. WCREG Command/Result Formats

Command Sequence:

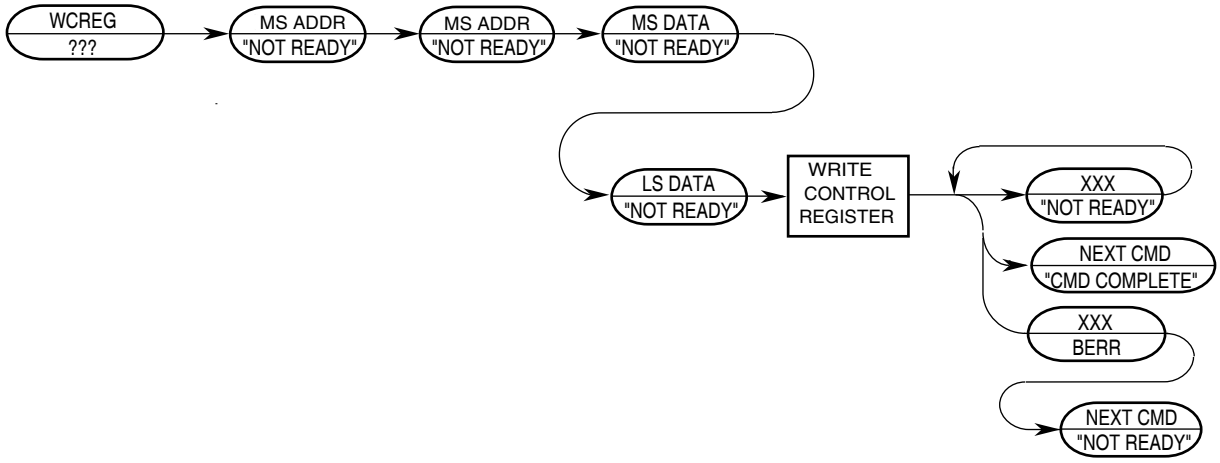

Figure 5-36. WCREG Command Sequence

Table 15-1. TMR*n* Field Descriptions (continued)

Bits	Name	Description
0	RST	Reset timer. 0 A transition from 1 to 0 resets the timer. Other register values can be written. The counter/timer/prescaler are not clocked unless the timer is enabled. 1 Enable timer

15.3.2 Timer Reference Registers (TRR0–TRR3)

Each TRR holds a 16-bit reference value that is compared with the free-running TCN as part of the output compare function. A match occurs when TCN increments to equal TRR.

	15 … 0
Field	REF (16-bit reference value)
Reset	1111_1111_1111_1111
R/W	Read/Write
Addr	MBAR + 0x204 (TRR0); 0x224 (TRR1); 0x244 (TRR2); 0x264 (TRR3)

Figure 15-3. Timer Reference Registers (TRR0–TRR3)

15.3.3 Timer Capture Registers (TCAP0–TCAP3)

Each TCAP is used to latch the TCN value during a capture operation when an edge occurs on the respective TIN0, TIN1, UART0_RxD, or UART1_RxD, as programmed in TMR*n*.

	15 … 0
Field	CAP (16-bit capture counter value)
Reset	0000_0000_0000_0000
R/W	Read Only
Addr	MBAR + 0x208 (TCAP0); 0x228 (TCAP1); 0x248 (TCAP2); 0x268 (TCAP3)

Figure 15-4. Timer Capture Registers (TCAP0–TCAP3)

15.3.4 Timer Counters (TCN0–TCN3)

TCN registers are 16-bit up counters. Reading a TCN*n* gives the current counter value without affecting counting. A write cycle to a TCN register causes it to be cleared.

	15 … 0
Field	COUNT (16-bit timer counter value)
Reset	0000_0000_0000_0000
R/W	Read/Write
Addr	MBAR + 0x20C (TCN0); 0x22C (TCN1); 0x24C (TCN2); 0x26C (TCN3)

Figure 15-5. Timer Counter (TCN0–TCN3)

Table 16-6. UCR*n* Field Descriptions (continued)

Bits	Value	Command	Description
3–2	**TC Field** (This field selects a single command)		
	00	no action taken	Causes the transmitter to stay in its current mode: if the transmitter is enabled, it remains enabled; if the transmitter is disabled, it remains disabled.
	01	transmitter enable	Enables operation of the channel's transmitter. USR*n*[TxEMP,TxRDY] are set. If the transmitter is already enabled, this command has no effect.
	10	transmitter disable	Terminates transmitter operation and clears USR*n*[TxEMP,TxRDY]. If a character is being sent when the transmitter is disabled, transmission completes before the transmitter becomes inactive. If the transmitter is already disabled, the command has no effect.
	11	—	Reserved, do not use.
1–0	**RC** (This field selects a single command)		
	00	no action taken	Causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.
	01	receiver enable	If the UART module is not in multidrop mode (UMR1*n*[PM] ¦ 11), RECEIVER ENABLE enables the channel's receiver and forces it into search-for-start-bit state. If the receiver is already enabled, this command has no effect.
	10	receiver disable	Disables the receiver immediately. Any character being received is lost. The command does not affect receiver status bits or other control registers. If the UART module is programmed for local loop-back or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, the command has no effect.
	11	—	Reserved, do not use.

16.3.6 UART Receiver Buffers (URB*n*)

The receiver buffers contain one serial shift register and a 24-byte FIFO. RxD is connected to the serial shift register. The CPU reads from the top of the stack while the receiver shifts and updates from the bottom when the shift register is full (see Figure 16-24). RB contains the character in the receiver.

	7 … 0
Field	RB
Reset	0000_0000
R/W	Read only
Address	MBAR + 0x10C,0x14C

Figure 16-7. UART Receiver Buffer (URB*n*)

16.3.11 UART Divider Upper/Lower Registers (UDU*n*/UDL*n*)

The UDU*n* registers (formerly called UBG1*n*) hold the MSB, and the UDL*n* registers (formerly UBG2*n*) hold the LSB of the preload value. UDU*n* and UDL*n* concatenate to provide a divider to CLKIN for transmitter/receiver operation, as described in Section 16.5.1.2.1, “CLKIN Baud Rates.”

	7	0
Field	Divider MSB	
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x118 (UDU0), 0x158 (UDU1)	

Figure 16-12. UART Divider Upper Registers (UDU*n*)

	7	0
Field	Divider LSB	
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x11C (UDL0), 0x15C (UDL1)	

Figure 16-13. UART Divider Lower Registers (UDL*n*)

NOTE

The minimum value that can be loaded on the concatenation of UDU*n* with UDL*n* is 0x0002. Both UDU*n* and UDL*n* are write-only and cannot be read by the CPU.

16.3.12 UART Autobaud Registers (UABU*n*/UABL*n*)

The UABU*n* registers hold the MSB, and the UABL*n* registers hold the LSB of the calculated baud rate. If UCR*n*[ENAB] is set, the value in these registers is automatically loaded into UDU*n* and UDL*n*.

	7	0
Field	Autobaud MSB	
Reset	0000_0000	
R/W	Read only	
Address	MBAR + 0x120 (UABU0), 0x160 (UABU1)	

Figure 16-14. UART Autobaud Upper Registers (UABU*n*)

	7	0
Field	Autobaud LSB	
Reset	0000_0000	
R/W	Read only	
Address	MBAR + 0x124 (UABL0), 0x164 (UABL1)	

Figure 16-15. UART Autobaud Lower Registers (UABL*n*)

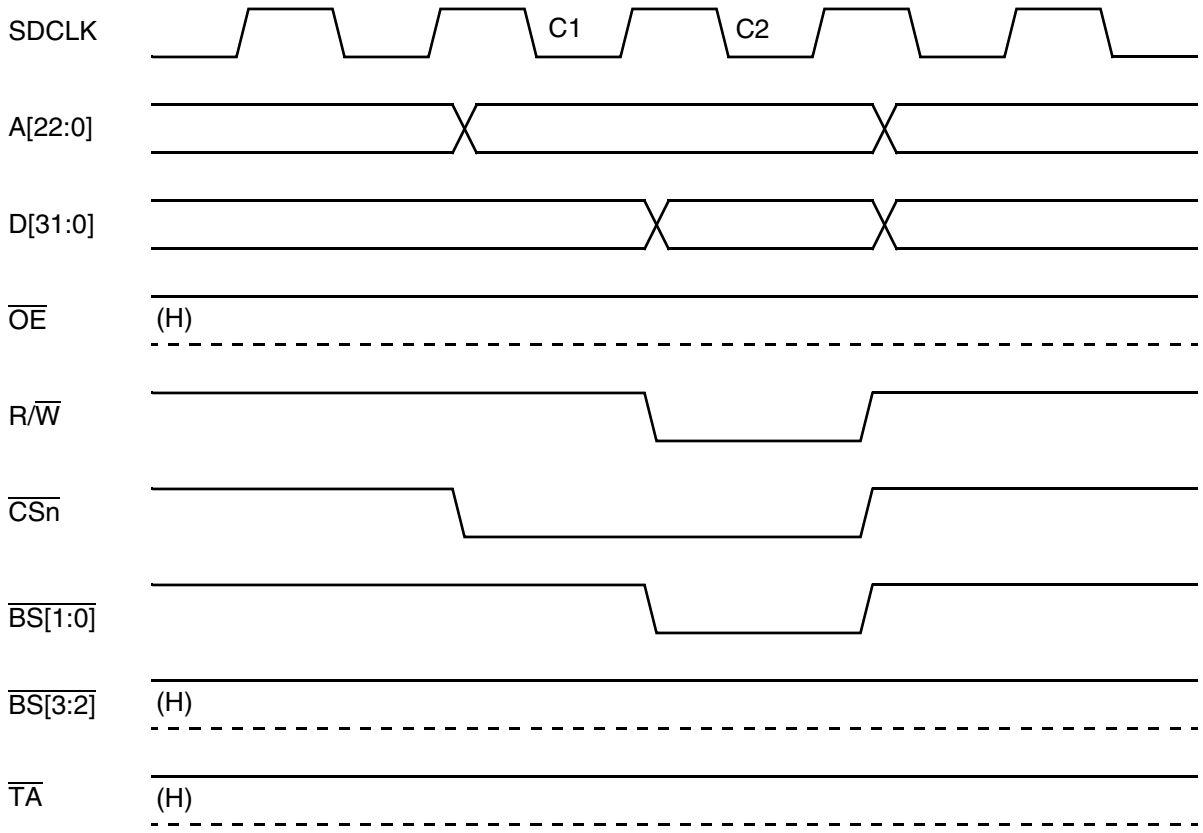

Figure 20-4. Word Write; EBI = 00; 16-/32-Bit Port; Internal Termination

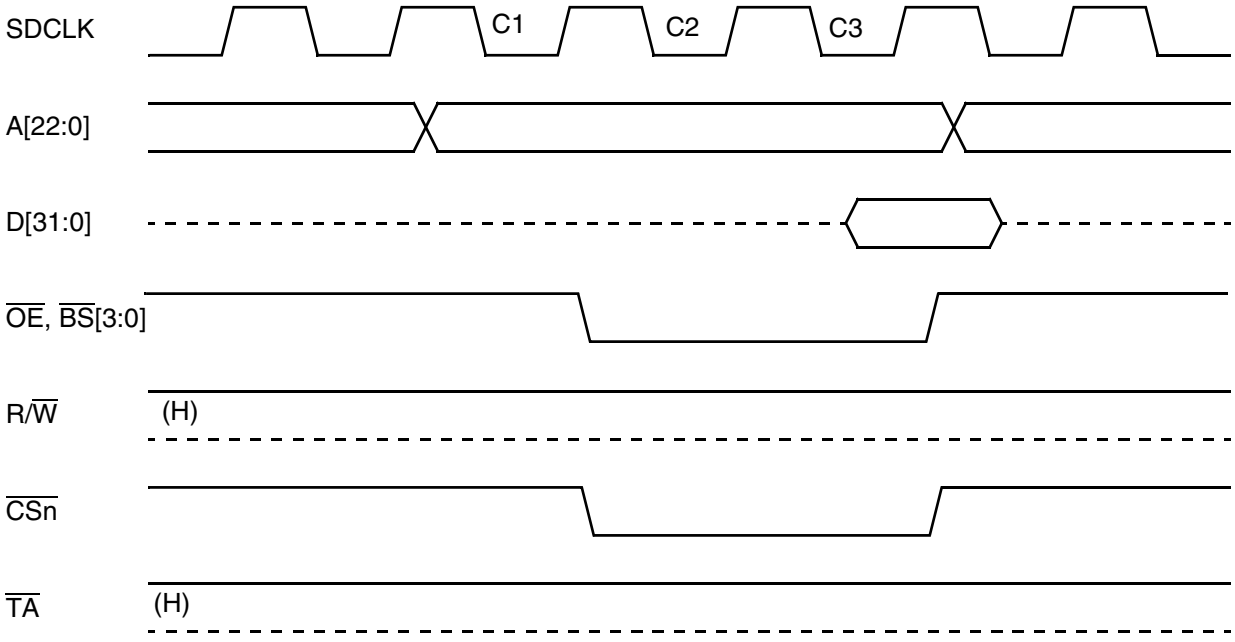

Figure 20-5. Longword Read with Address Setup; EBI = 00; 32-Bit Port; Internal Termination

20.9 Interrupt Cycles

All interrupt vectors are internally generated. The MCF5272 does not support external interrupt acknowledge cycles. The System Integration Module prioritizes all interrupt requests and issues the appropriate vector number in response to an interrupt acknowledge cycle. Refer to the System Integration chapter for details on the interrupt vectors and their priorities.

When an external peripheral device requires the services of the CPU, it can signal the ColdFire core to take an interrupt exception. The external peripheral devices use the interrupt request signals ($\overline{\text{INTx}}$) to signal an interrupt condition to the MCF5272. The interrupt exception transfers control to a routine that responds appropriately.

There are a total of six external interrupt inputs, $\overline{\text{INT}}$[6:1]. Depending on the pin configuration between three and six of these pins are available. Each interrupt input pin is dedicated to an external peripheral. It is possible to have multiple external peripherals share an $\overline{\text{INT}}x$ pin but software must then determine which peripheral caused the interrupt. The interrupt priority level and the signal level of each interrupt pin are individually programmable.

The MCF5272 continuously samples the external interrupt input signals and synchronizes and debounces these signals. An interrupt request must be held constant for at least two consecutive CLK periods to be considered a valid input. MCF5272 latches the interrupt and the interrupt controller responds as programmed. The interrupt service routine must clear the latch in the ICR registers.

NOTE

All internal interrupts are level sensitive only. External interrupts are edge-sensitive as programmed in the PITR. Interrupts must remain stable and held valid for two clock cycles while they are internally synchronized and latched.

The MCF5272 takes an interrupt exception for a pending interrupt within one instruction boundary after processing any other pending exception with a higher priority. Thus, the MCF5272 executes at least one instruction in an interrupt exception handler before recognizing another interrupt request.

20.10 Bus Errors

The system hardware can use the transfer error acknowledge ($\overline{\text{TEA}}$) signal to abort the current bus cycle when a fault is detected. A bus error is recognized during a bus cycle when $\overline{\text{TEA}}$ is asserted.

NOTE

The signal $\overline{\text{TEA}}$ is not intended for use in normal operation since each chip select can be programmed to automatically terminate a bus cycle at a time defined by the bits programmed into the wait state field of the Chip Select Option Register. There is an on chip bus monitor which can be configured to generate an internal $\overline{\text{TEA}}$ signal.

When the MCF5272 recognizes a bus error condition for an access, the access is terminated immediately. An access that requires more than one transfer aborts without completing the remaining transfers if $\overline{\text{TEA}}$ is asserted, regardless of whether the access uses burst or non-burst transfers.

23.4 Debug AC Timing Specifications

Table 23-9 lists specifications for the debug AC timing parameters shown in Figure 23-8.

Table 23-9. Debug AC Timing Specification

Num	Characteristic	0-66 MHz		Units
		Min	Max	
D1	PST[3:0], DDATA[3:0] to PSTCLK valid	—	8.5	nS
D2	PSTCLK to PST[3:0], DDATA[3:0] hold	1	—	nS
D3	DSI-to-DSCLK setup	1	—	PSTCLKs
D4 [1]	DSCLK-to-DSO hold	4	—	PSTCLKs
D5	DSCLK cycle time	5	—	PSTCLKs

[1] DSCLK and DSI are synchronized internally. D4 is measured from the synchronized DSCLK input relative to the rising edge of PSTCLK.

Figure 23-7 shows real-time trace timing for the values in Table 23-9.

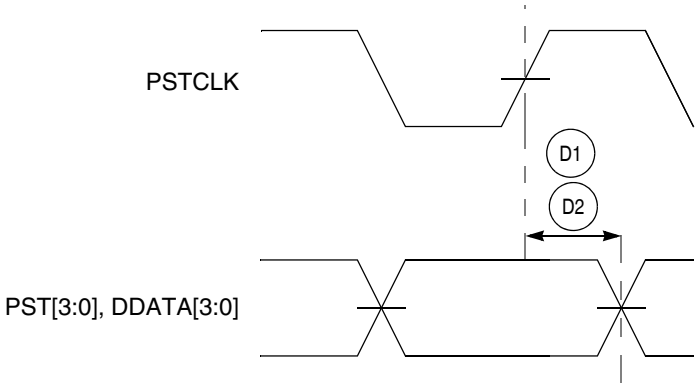

Figure 23-7. Real-Time Trace AC Timing

Figure 23-8 shows BDM serial port AC timing for the values in Table 23-9.

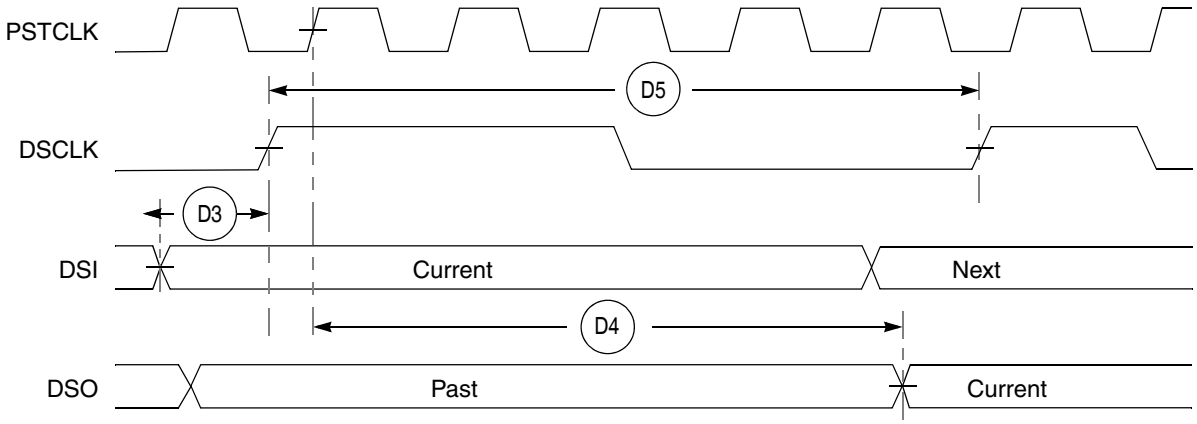

Figure 23-8. BDM Serial Port AC Timing

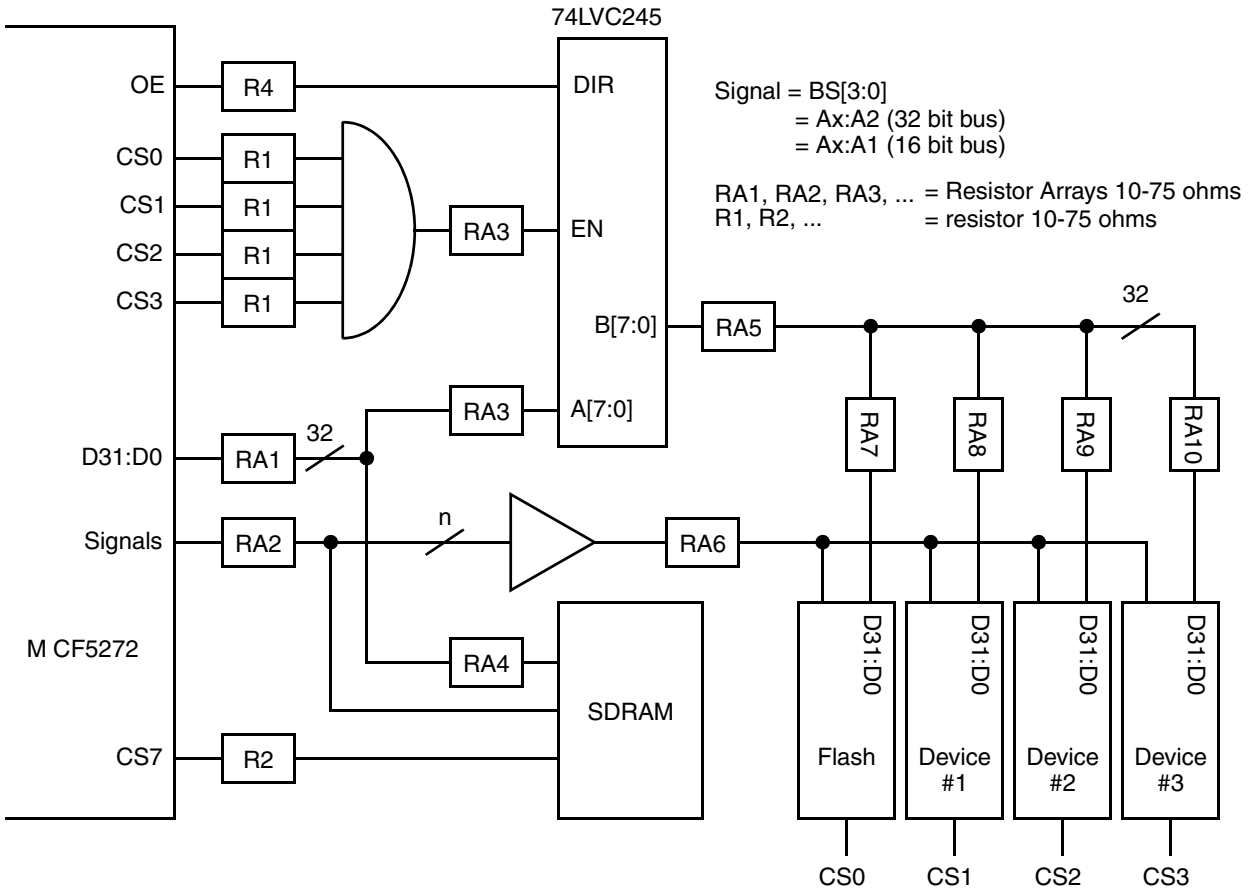

Figure B-1. Buffering and Termination