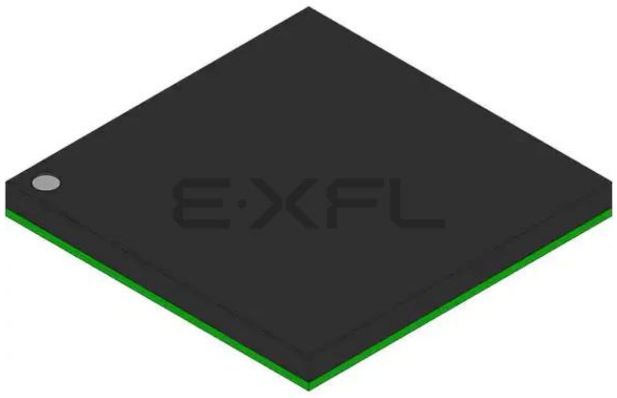
Freescale Semiconductor - MCF5272VF66J Datasheet



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/pro/item?MUrl=&PartUrl=mcf5272vf66j

Table 2-14. Two-Operand Instruction Execution Times (continued)

Opcode	<ea>	Effective Address							
		Rn	(An)	(An)+	–(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
bclr	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
bclr	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
bset	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
bset	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
btst	Dy,<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
btst	#imm,<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
cmp.l	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
cmpi.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
divs.w	<ea>,Dx	20(0/0)	23(1/0)	23(1/0)	23(1/0)	23(1/0)	24(1/0)	23(1/0)	20(0/0)
divu.w	<ea>,Dx	20(0/0)	23(1/0)	23(1/0)	23(1/0)	23(1/0)	24(1/0)	23(1/0)	20(0/0)
divs.l	<ea>,Dx	35(0/0)	38(1/0)	38(1/0)	38(1/0)	38(1/0)	—	—	—
divu.l	<ea>,Dx	35(0/0)	38(1/0)	38(1/0)	38(1/0)	38(1/0)	—	—	—
eor.l	Dy,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
eori.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
lea	<ea>,Ax	—	1(0/0)	—	—	1(0/0)	2(0/0)	1(0/0)	—
lsl.l	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
lsr.l	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
mac.w	Ry,Rx	1(0/0)	—	—	—	—	—	—	—
mac.l	Ry,Rx	3(0/0)	—	—	—	—	—	—	—
msac.w	Ry,Rx	1(0/0)	—	—	—	—	—	—	—
msac.l	Ry,Rx	3(0/0)	—	—	—	—	—	—	—
mac.w	Ry,Rx,ea,Rw	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
mac.l	Ry,Rx,ea,Rw	—	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—
moveq	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
msac.w	Ry,Rx,ea,Rw	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
msac.l	Ry,Rx,ea,Rw	—	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—
muls.w	<ea>,Dx	4(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	7(1/0)	6(1/0)	4(0/0)
mulu.w	<ea>,Dx	4(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	8(1/0)	6(1/0)	4(0/0)
muls.l	<ea>,Dx	6(0/0)	8(1/0)	8(1/0)	8(1/0)	8(1/0)	—	—	—
mulu.l	<ea>,Dx	6(0/0)	8(1/0)	8(1/0)	8(1/0)	8(1/0)	—	—	—
or.l	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
or.l	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
or.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
rems.l	<ea>,Dx	35(0/0)	38(1/0)	38(1/0)	38(1/0)	38(1/0)	—	—	—
remu.l	<ea>,Dx	35(0/0)	38(1/0)	38(1/0)	38(1/0)	38(1/0)	—	—	—
sub.l	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)

Table 2-14. Two-Operand Instruction Execution Times (continued)

Opcode	<ea>	Effective Address							
		Rn	(An)	(An)+	–(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
sub.l	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
subi.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
subq.l	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
subx.l	Dy,Dx	1(0/0)	—	—	—	—	—	—	—

2.7.4 Miscellaneous Instruction Execution Times

Table 2-15 lists timings for miscellaneous instructions.

Table 2-15. Miscellaneous Instruction Execution Times

Opcode	<ea>	Effective Address							
		Rn	(An)	(An)+	–(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
cpushl	(Ax)	—	11(0/1)	—	—	—	—	—	—
link.w	Ay,#imm	2(0/1)	—	—	—	—	—	—	—
move.w	CCR,Dx	1(0/0)	—	—	—	—	—	—	—
move.w	<ea>,CCR	1(0/0)	—	—	—	—	—	—	1(0/0)
move.w	SR,Dx	1(0/0)	—	—	—	—	—	—	—
move.w	<ea>,SR	7(0/0)	—	—	—	—	—	—	7(0/0)
movec	Ry,Rc	9(0/1)	—	—	—	—	—	—	—
movem.l [1]	<ea>,&list	—	1+n(n/0)	—	—	1+n(n/0)	—	—	—
movem.l	&list,<ea>	—	1+n(0/n)	—	—	1+n(0/n)	—	—	—
nop		3(0/0)	—	—	—	—	—	—	—
pea	<ea>	—	2(0/1)	—	—	2(0/1)[2]	3(0/1)[3]	2(0/1)	—
pulse		1(0/0)	—	—	—	—	—	—	—
stop	#imm	—	—	—	—	—	—	—	3(0/0)[4]
trap	#imm	—	—	—	—	—	—	—	15(1/2)
trapf		1(0/0)	—	—	—	—	—	—	—
trapf.w		1(0/0)	—	—	—	—	—	—	—
trapf.l		1(0/0)	—	—	—	—	—	—	—
unlk	Ax	2(1/0)	—	—	—	—	—	—	—
wddata.l	<ea>	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
wdebug.l	<ea>	—	5(2/0)	—	—	5(2/0)	—	—	—

[1] *n* is the number of registers moved by the MOVEM opcode.
[2] PEA execution times are the same for (d16,PC).
[3] PEA execution times are the same for (d8,PC,Xi*SF).
[4] The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.

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.

5.2 Signal Description

Table 5-1 describes debug module signals. All ColdFire debug signals are unidirectional and related to a rising edge of the processor core's clock signal. The standard 26-pin debug connector is shown in Section 5.8, "Freescale-Recommended BDM Pinout."

Table 5-1. Debug Module Signals

Signal	Description
Development Serial Clock (DSCLK)	Internally synchronized input. (The logic level on DSCLK is validated if it has the same value on two consecutive rising CLKIN edges.) Clocks the serial communication port to the debug module during packet transfers. Maximum frequency is 1/5 the processor status clock (PSTCLK) speed. At the synchronized rising edge of DSCLK, the data input on DSI is sampled and DSO changes state.
Development Serial Input (DSI)	Internally synchronized input that provides data input for the serial communication port to the debug module.
Development Serial Output (DSO)	Provides serial output communication for debug module responses. DSO is registered internally.
Breakpoint ($\overline{\text{BKPT}}$)	Input used to request a manual breakpoint. Assertion of $\overline{\text{BKPT}}$ puts the processor into a halted state after the current instruction completes. Halt status is reflected on processor status signals (PST[3:0]) as the value 0xF.
Processor Status Clock (PSTCLK)	Delayed version of the processor clock. Its rising edge appears in the center of valid PST and DDATA output. See Figure 5-2. PSTCLK indicates when the development system should sample PST and DDATA values.
Debug Data (DDATA[3:0])	These output signals display the register breakpoint status as a default, or optionally, captured address and operand values. The capturing of data values is controlled by the setting of the CSR. Additionally, execution of the WDDATA instruction by the processor captures operands which are displayed on DDATA. These signals are updated each processor cycle.
Processor Status (PST[3:0])	These output signals report the processor status. Table 5-2 shows the encoding of these signals. These outputs indicate the current status of the processor pipeline and, as a result, are not related to the current bus transfer. The PST value is updated each processor cycle.

Figure 5-2 shows PSTCLK timing with respect to PST and DDATA.

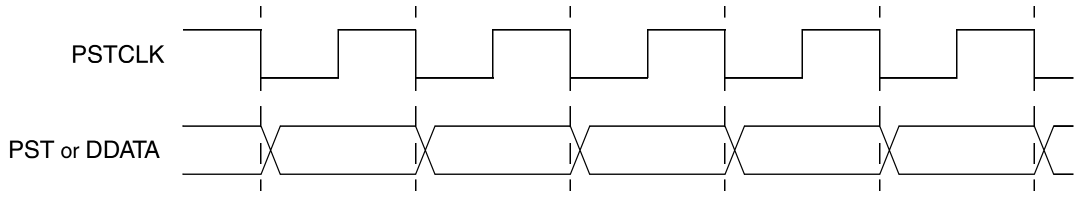


Figure 5-2. PSTCLK Timing

5.5.3.2 Command Sequence Diagrams

The command sequence diagram in Figure 5-16 shows serial bus traffic for commands. Each bubble represents a 17-bit bus transfer. The top half of each bubble indicates the data the development system sends to the debug module; the bottom half indicates the debug module's response to the previous development system commands. Command and result transactions overlap to minimize latency.

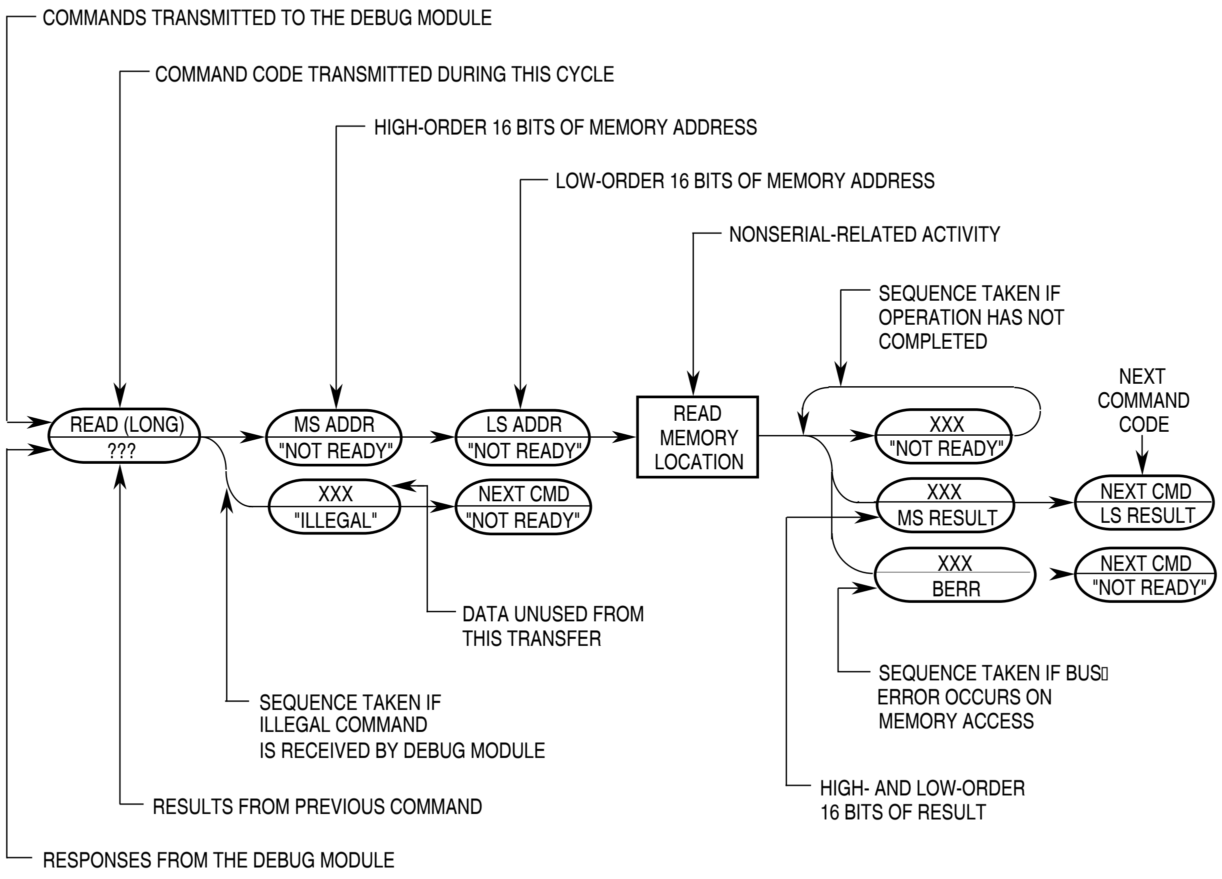

COMMANDS TRANSMITTED TO THE DEBUG MODULE
COMMAND CODE TRANSMITTED DURING THIS CYCLE
HIGH-ORDER 16 BITS OF MEMORY ADDRESS
LOW-ORDER 16 BITS OF MEMORY ADDRESS
NONSERIAL-RELATED ACTIVITY
SEQUENCE TAKEN IF OPERATION HAS NOT COMPLETED
NEXT COMMAND CODE
READ (LONG) ???
MS ADDR "NOT READY"
LS ADDR "NOT READY"
READ MEMORY LOCATION
XXX "NOT READY"
XXX "ILLEGAL"
NEXT CMD "NOT READY"
XXX MS RESULT
NEXT CMD LS RESULT
XXX BERR
NEXT CMD "NOT READY"
DATA UNUSED FROM THIS TRANSFER
SEQUENCE TAKEN IF BUS ERROR OCCURS ON MEMORY ACCESS
SEQUENCE TAKEN IF ILLEGAL COMMAND IS RECEIVED BY DEBUG MODULE
HIGH- AND LOW-ORDER 16 BITS OF RESULT
RESULTS FROM PREVIOUS COMMAND
RESPONSES FROM THE DEBUG MODULE

Figure 5-16. Command Sequence Diagram

The sequence is as follows:

- In cycle 1, the development system command is issued (READ in this example). The debug module responds with either the low-order results of the previous command or a command complete status of the previous command, if no results are required.
- In cycle 2, the development system supplies the high-order 16 address bits. The debug module returns a not-ready response unless the received command is decoded as unimplemented, which is indicated by the illegal command encoding. If this occurs, the development system should retransmit the command.

NOTE

A not-ready response can be ignored except during a memory-referencing cycle. Otherwise, the debug module can accept a new serial transfer after 32 processor clock periods.

Chapter 6
System Integration Module (SIM)

This chapter provides detailed operation information regarding the system integration module (SIM). It describes the SIM programming model, bus arbitration, power management, and system-protection functions for the MCF5272.

6.1 Features

The SIM, shown in Figure 6-1, provides overall control of the bus and serves as the interface between the ColdFire core processor complex and the internal peripheral devices.

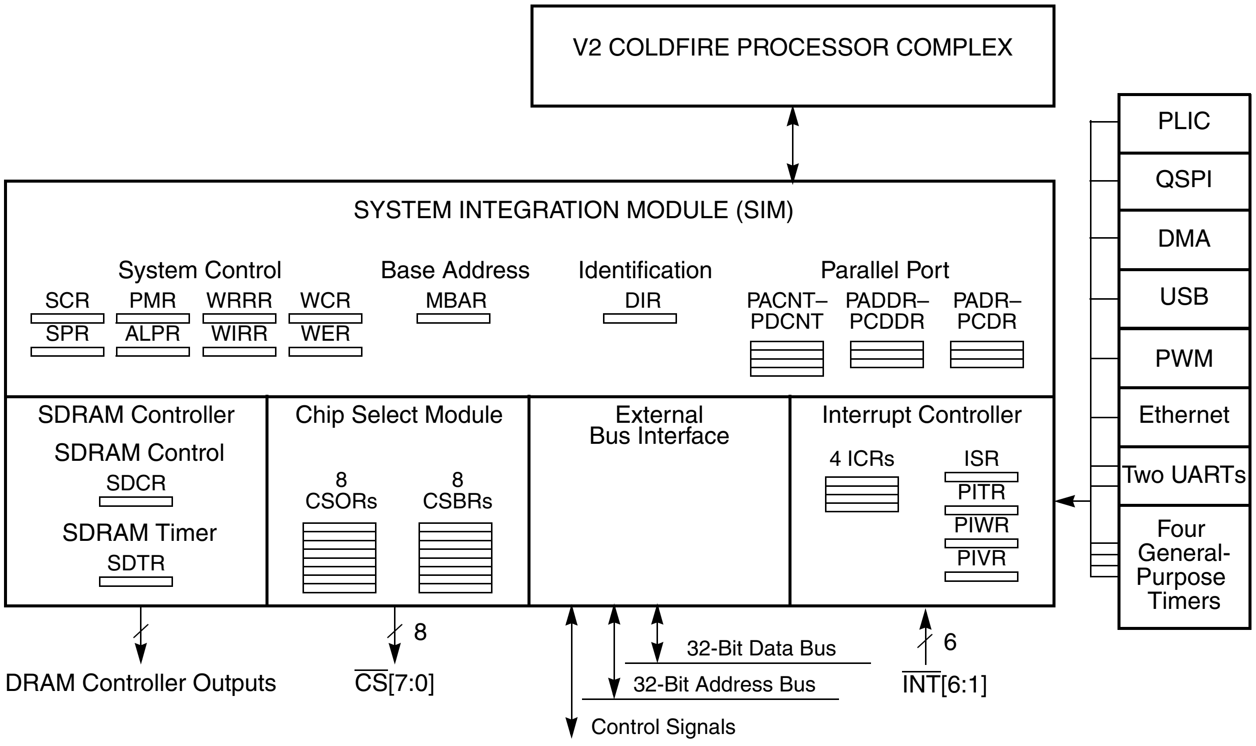


Figure 6-1. SIM Block Diagram

Chapter 2, “ColdFire Core.” Pending interrupts from external sources ($\overline{\text{INT}}$[6:1]) can be cleared using the ICRs.

For an interrupt to be successfully processed, stack RAM must be available. A programmable chip select is often used for the RAM, in which case, the RAM is not immediately available at startup. Thus, no interrupts are recognized until PIVR is initialized. The RAM chip select and system stack should be set up before this initialization.

If more than one interrupt source has the same interrupt priority level (IPL), the interrupt controller daisy chains the interrupts with the priority order following the bit placement in the PIWR, with $\overline{\text{INT1}}$ having the highest priority and SWTO having the lowest priority, as shown in Figure 7-8.

7.2.1 Interrupt Controller Registers

This section describes the registers associated with the interrupt controller. Table 7-2 gives the nomenclature used for the interrupt and power management registers.

Table 7-2. Interrupt and Power Management Register Mnemonics

Mnemonic or Portion Thereof	Description
INT1, INT2, INT3, INT4, INT5, INT6	External interrupt signals 1–6.
TMR0, TMR1, TMR2, TMR3	Timers 3–0 from timer module
USB0, USB1, USB2, USB3, USB4, USB5, USB6, USB7	USB endpoint 0–7
UART1, UART2	UART1, UART2 modules
PLIP	PLIC 2-KHz periodic interrupt, 2B+D data
PLIA	PLIC asynchronous and maintenance channels interrupt
DMA	DMA controller interrupt
ETx	Ethernet module transmit data interrupt
ERx	Ethernet module receive data interrupt
ENTC	Ethernet module non-time-critical interrupt
QSPI	Queued serial peripheral interface
IPL2, IPL1, IPL0	Interrupt priority level bits 2–0
PI	Pending interrupt
PDN	Power down enable
WK	Wakeup enable
SWTO	Software watchdog timer time out

However, to reduce power consumption, USB_VDD may be left unconnected if either of the following is true:

- The USB module is not used.
- An external transceiver is used.

12.2.1.2 Clock Generator

The USB module requires two clock inputs; the system clock and a 48-MHz data clock. The data clock source is selectable between the USB_ExtCLK pin or the system clock. The clock generator automatically uses the external clock as the data clock if it detects the presence of a clock signal on the USB_CLK pin during system reset. This automatic selection can be overridden by setting bit USBEPCTL0[CLK_SEL], . If a clock signal is not present on USB_CLK, the clock generator uses the system clock.

NOTE

In all cases, the system clock must be at least 24-MHz for the USB module to function properly. If the system clock is used for the USB data clock, the system clock frequency must be 48-MHz.

12.2.1.3 USB Control Logic

The control logic performs the following functions:

- For transmitted data:
  - Packet creation
  - Parallel-to-serial conversion
  - CRC generation
  - NRZI encoding
  - Bit stuffing
- For received data:
  - Sync detection
  - Packet identification
  - End-of-packet detection
  - Serial-to-parallel conversion
  - CRC validation
  - NRZI decoding
  - Bit unstuffing
- For error detection:
  - Bad CRC
  - Timeout waiting for end-of-packet
  - Bit stuffing violations

Table 12-1. USB Device Requests (continued)

Device Request	USB Request Processor Action
get_status	Returns the current status of the specified device, endpoint or interface. No user notification is provided, no user action required.
set_address	Loads the specified address into USBFAR. The control logic begins responding to the new address once the status stage of the request completes successfully. No user notification is provided unless debug mode is enabled.
set_configuration	Reads the configuration RAM. If the configuration is cleared, the USB module is placed into the unconfigured state. If a valid configuration is selected, the appropriate endpoint controllers are activated. An invalid configuration number or error in the configuration descriptor causes the USB module to return a STALL response to the host. The user is notified when this request completes successfully and must initialize the active endpoint controllers.
set_descriptor	Not supported. Returns request error.
set_feature	Sets the specified feature. Remote wakeup and endpoint halt are the only features defined in the USB Specification, Revision 1.0. If the specified feature is remote wakeup, the USB enables the remote wakeup functionality. If the specified feature is endpoint halt, the USB request processor halts the selected endpoint. A halted endpoint returns a STALL response to any requests.
set_interface	Allows the host to select an alternate setting for the specified interface. If a valid alternate setting is selected, the appropriate endpoint controllers are activated. The user is notified upon successful completion of this request and must reinitialize the affected endpoint controllers. NOTE: The user must read the descriptor structure to determine which endpoints correspond to a given interface.
sync_frame	Passed to the user as a vendor specific request.

12.3.2.8 USB Specification Number Register (SPECR)

Figure 12-11 shows the USB specification number register.

	15–4	3–0
Field	SPEC	MRN
Reset	0001_0001_0000_0001	
R/W	Read	
Addr	MBAR + 0x1022	

Figure 12-11. USB Specification Number Register (SPECR)

Table 12-9 lists field descriptions for the USB specification number register.

Table 12-9. SPECR Field Descriptions

Bits	Name	Descriptions
15–4	SPEC	USB specification release number. This field identifies the release of the USB Specification that the device complies with. The USB Specification Release Number is in binary-coded decimal (BCD) format.
3–0	MRN	Module revision number.

12.3.2.9 USB Endpoint 0 Status Register (EP0SR)

Figure 12-12 shows the USB endpoint 0 status register. Only written via the standard SET_CONFIGURATION request by the host.

	15–12	11	10–3	2	1	0
Field	CONFIG	WAKE_ST	—	HALT_ST	DIR	—
Reset	0000_0000_0000_0001					
R/W	Read					
Addr	MBAR + 0x1026					

Figure 12-12. USB Endpoint 0 Status Register (EP0SR)

Table 12-10 lists field descriptions the USB endpoint 0 status register.

Table 12-10. EP0SR Field Descriptions

Bits	Name	Descriptions
15–12	CONFIG	Current configuration number. Indicates which configuration is active. Receiving a SET_CONFIGURATION USB standard device request sets the active configuration. A configuration setting of 0 means that the device is unconfigured. Valid only when EPISR0[DEV_CFG] is set
11	WAKE_ST	Remote wakeup status. Indicates whether the USB module has been enabled to generate remote wakeup resume signaling. 1 Enabled. The USB host has issued the SET_FEATURE request with the Remote wakeup feature selector set. 0 Disabled. The USB host has issued the CLEAR_FEATURE request with the remote wakeup feature selector set, or has not set this feature since a USB or system reset has occurred.
10–3	—	Reserved, should be cleared.

Chapter 14
Queued Serial Peripheral Interface (QSPI) Module

This chapter describes the queued serial peripheral interface (QSPI) module. Following a feature-set overview is a description of operation including details of the QSPI's internal RAM organization. The chapter concludes with the programming model and a timing diagram.

14.1 Overview

The queued serial peripheral interface module provides a serial peripheral interface with queued transfer capability. It allows users to enqueue up to 16 transfers at once, eliminating CPU intervention between transfers. Transfer RAMs in the QSPI are indirectly accessible using address and data registers.

Functionality is very similar, but not identical, to the QSPI portion of the QSM (queued serial module) implemented in the MC68332.

14.2 Features

- Programmable queue to support up to 16 transfers without user intervention
- Supports transfer sizes of 8 to 16 bits in 1-bit increments
- Four peripheral chip-select lines for control of up to 15 devices
- Baud rates from 129.4 Kbps to 16.5 Mbps at 66 MHz
- Programmable delays before and after transfers
- Programmable clock phase and polarity
- Supports wraparound mode for continuous transfers

14.3 Module Description

The QSPI module communicates with the integrated ColdFire CPU using internal memory mapped registers located starting at MBAR + 0xA0. See also Section 14.5, "Programming Model." A block diagram of the QSPI module is shown in Figure 14-1.

Table 16-4. USR*n* Field Descriptions (continued)

Bits	Name	Description
1	FFULL	FIFO full. This bit is equivalent to URF[FULL]. 0 The FIFO is not full but may hold unread characters. 1 A character was received and the receiver FIFO is now full. Any characters received when the FIFO is full are lost.
0	RxRDY	Receiver ready 0 The CPU has read the receiver buffer and no characters remain in the FIFO after this read. 1 One or more characters were received and are waiting in the receiver buffer FIFO.

16.3.4 UART Clock-Select Registers (UCSR*n*)

The UART clock-select registers (UCSR*n*) select an external clock on the URT_CLK input (divided by 1 or 16) or a prescaled CLKIN as the clocking source for the transmitter and receiver. See Section 16.5.1, “Transmitter/Receiver Clock Source.” The transmitter and receiver can use different clock sources. To use CLKIN for both, set UCSR*n* to 0xDD.

	7–4	3–0
Field	RCS	TCS
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x104 (UCSR0), 0x144 (UCSR1)	

Figure 16-5. UART Clock-Select Registers (UCSR*n*)

Table 16-5 describes UCSR*n* fields.

Table 16-5. UCSR*n* Field Descriptions

Bits	Name	Description
7–4	RCS	Receiver clock select. Selects the clock source for the receiver channel. 1101 Prescaled CLKIN 1110 URT_CLK divided by 16 1111 URT_CLK
3–0	TCS	Transmitter clock select. Selects the clock source for the transmitter channel. 1101 Prescaled CLKIN 1110 URT_CLK divided by 16 1111 URT_CLK

	7	6			3	2	1	0
Field	COS	ABC	RXFIFO	TXFIFO	RXFTO	DB	FFULL/RxRDY	TxRDY
Reset	0000_0000							
R/W	Read only for status, write only for mask.							
Address	MBAR + 0x114 (UISR0), 0x154 (UISR1); MBAR + 0x114 (UIMR0), 0x154 (UIMR1)							

Figure 16-11. UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

Table 16-9 describes UISR*n* and UIMR*n* fields.

Table 16-9. UISR*n*/UIMR*n* Field Descriptions

Bits	Name	Description
7	COS	Change-of-state. 0 UIPCR*n*[COS] is not selected. 1 Change-of-state occurred on $\overline{\text{CTS}}$ and was programmed in UACR*n*[IEC] to cause an interrupt.
6	ABC	Autobaud calculation. 0 Autobaud is disabled or is waiting for the first receiver character. 1 The baud rate has been calculated and loaded into the clock source divider (UDU*n* and UDL*n*). After being set, this bit is cleared by writing to UCR*n*[ENAB].
5	RxFIFO	Receiver FIFO status. After being set, this bit is cleared by reading URB*n*. 0 FIFO status indication is disabled or the receiver status has not changed. 1 The receiver status has changed as programmed in URF*n*[RXS].
4	TxFIFO	Transmitter FIFO status. After being set, this bit is cleared by writing UTB*n*. 0 FIFO status indication is disabled or the transmitter status has not changed. 1 The transmitter status has changed as programmed in UTF*n*[TXS].
3	RxFTO	Receiver FIFO timeout. 0 No receiver FIFO timeout. This bit is cleared by reading all remaining data in the receiver FIFO, by receiving another character into the FIFO, or if the receiver is disabled. The count to timeout is restarted when RxFTO is cleared. 1 The receiver FIFO has timed out at 64 baud with unread data below the FIFO fullness level.
2	DB	Delta break. 0 No new break-change condition to report. Section 16.3.5, “UART Command Registers (UCRn),” describes the RESET BREAK-CHANGE INTERRUPT command. 1 The receiver detected the beginning or end of a received break.
1	FFULL/RxRDY	RxRDY (receiver ready) if UMR1*n*[FFULL/RxRDY] = 0; FIFO full (FFULL) if UMR1*n*[FFULL/RxRDY] = 1. Duplicate of USR*n*[FFULL/RxRDY].
0	TxRDY	Transmitter ready. This bit is the duplication of USR*n*[TxRDY]. 0 The transmitter holding register was loaded by the CPU or the transmitter is disabled. Characters loaded into the transmitter holding register when TxRDY = 0 are not sent. 1 The transmitter holding register is empty and ready to be loaded with a character.

Table 17-5. PBCNT Field Descriptions

Bits	Name	Description
31–30	PBCNT15	Configure pin P10 00 PB15 01 E_MDC 1*x* Reserved
29–28	PBCNT14	Configure pin L9 00 PB14 01 E_RxER 1*x* Reserved
27–26	PBCNT13	Configure pin M9 00 PB13 01 E_RxD1 1*x* Reserved
25–24	PBCNT12	Configure pin N9 00 PB12 01 E_RxD2 1*x* Reserved
23–22	PBCNT11	Configure pin P9 00 PB11 01 E_RxD3 10 Reserved 11 Reserved
21–20	PBCNT10	Configure pin L8 00 PB10 01 E_TxD1 1*x* Reserved
19–18	PBCNT9	Configure pin M8 00 PB9 01 E_TxD2 1*x* Reserved
17–16	PBCNT8	Configure pin N8 00 PB8 01 E_TxD3 1*x* Reserved
15–14	PBCNT7	Configure pin M6 00 PB7 01 TOUT0 1*x* Reserved
13–12	PBCNT6	Configure pin G4 00 PB6 01 Reserved 1*x* Reserved
11–10	PBCNT5	Configure pin F3 00 PB5 01 $\overline{\text{TA}}$ 1*x* Reserved

17.3 Data Direction Registers

These registers are used to program GPIO port signals as inputs or outputs. The data direction bit for any line is ignored unless that line is configured for general purpose I/O in the appropriate control register. If a GPIO line changes from an input to an output, the initial data on that pin is the last data written to the latch by the corresponding data register.

At system reset, these register bits are all cleared, configuring all port I/O lines as general purpose inputs. Bootstrap software must write an appropriate value into the data direction register to configure GPIO port signals as outputs. When these registers are first written, any internal pullups on the corresponding I/O pins are disabled.

A detailed description is provided only for data direction register A (PADDR). The control bits in all three registers operate in the same manner.

17.3.1 Port A Data Direction Register (PADDR)

The PADDR determines the signal direction of each parallel port pin programmed as a GPIO port in the PACNT.

	15 … 0
Field	PADDR
Reset	0000_0000_0000_0000
R/W	Read/Write
Addr	MBAR + 0x0084

Figure 17-4. Port A Data Direction Register (PADDR)

Table 17-9. PADDR Field Descriptions

Bits	Name	Description
15–0	PADDR	Data direction bits. Each data direction bit selects the direction of the signal as follows: 0 Signal is defined as an input. 1 Signal is defined as an output.

17.3.2 Port B Data Direction Register (PBDDR)

The PBDDR determines the signal direction of each parallel port pin programmed as a GPIO port in the PBCNT.

	15 … 0
Field	PBDDR
Reset	0000_0000_0000_0000
R/W	Read/Write
Addr	MBAR + 0x008C

Figure 17-5. Port B Data Direction Register (PBDDR)

22.2 Package Dimensions

Figure 22-2 shows MCF5272 package dimensions.

Figure 22-2. 196 MAPBGA Package Dimensions (Case No. 1128A-01)

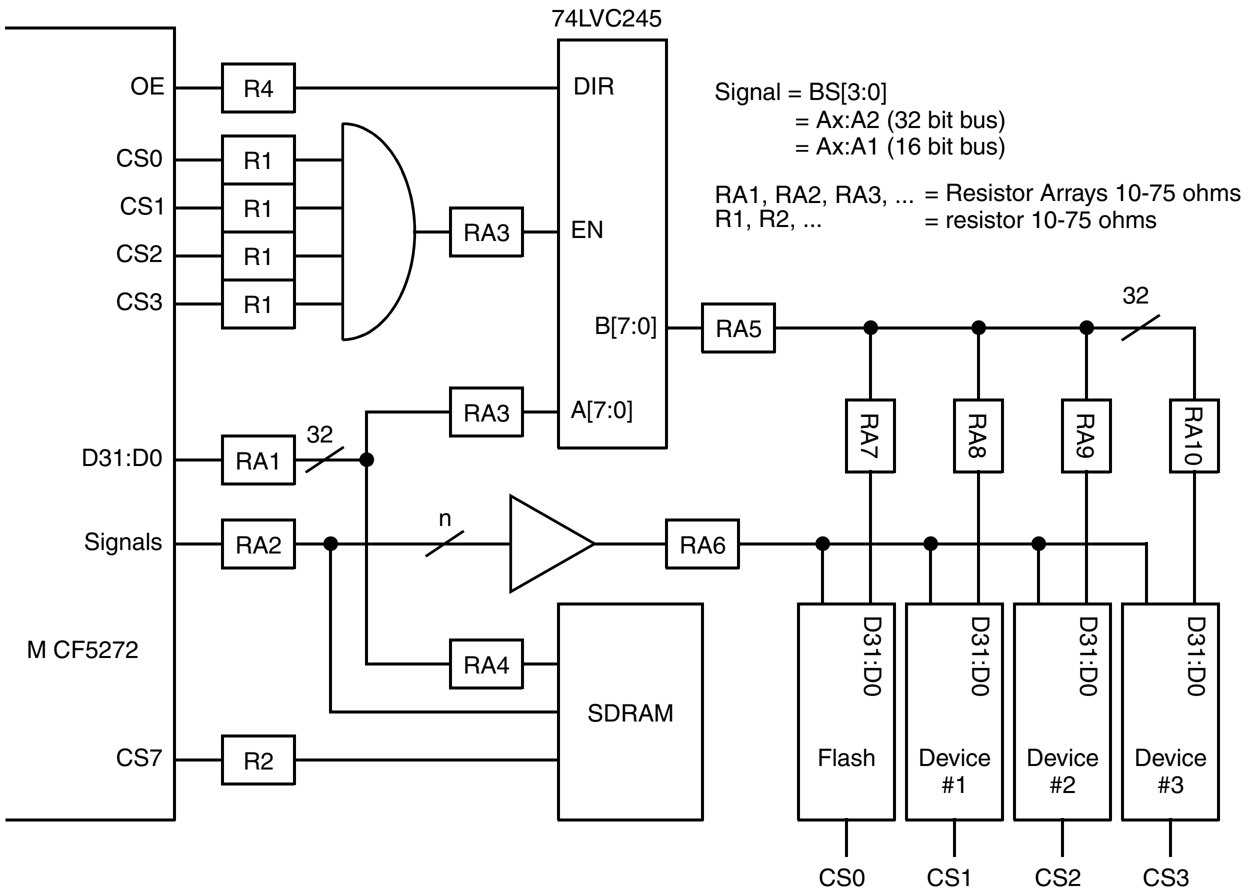


Figure B-1. Buffering and Termination