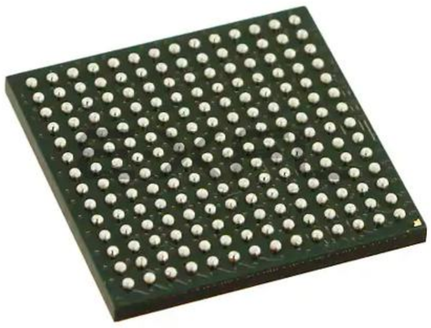
NXP USA Inc. - MCF5272VF66R2J Datasheet



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/product-detail/nxp-semiconductors/mcf5272vf66r2j

Table of Contents (Continued)

Paragraph Number | Title | Page Number

19.16.2.6 D-Channel Grant (DGNT1_INT6/PA15_INT6) 19-33
19.16.3 GCI/IDL TDM Ports 2 and 3 19-34
19.16.3.1 GCI/IDL Delayed Frame Sync 2 (DFSC2/PA12) 19-34
19.16.3.2 GCI/IDL Delayed Frame Sync 3 (DFSC3/PA13) 19-34
19.16.3.3 QSPI_CS3, Port 3 GCI/IDL Data Out 3, PA7 (PA7/DOUT3/QSPI_CS3) ... 19-34
19.16.3.4 INT4 and Port 3 GCI/IDL Data In (INT4/DIN3) 19-35
19.17 JTAG Test Access Port and BDM Debug Port 19-35
19.17.1 Test Clock (TCK/PSTCLK) 19-35
19.17.2 Test Mode Select and Force Breakpoint (TMS/BKPT) 19-35
19.17.3 Test and Debug Data Out (TDO/DSO) 19-36
19.17.4 Test and Debug Data In (TDI/DSI) 19-36
19.17.5 JTAG TRST and BDM Data Clock (TRST/DSCLK) 19-36
19.17.6 Freescale Test Mode Select (MTMOD) 19-36
19.17.7 Debug Transfer Error Acknowledge (TEA) 19-36
19.17.8 Processor Status Outputs (PST[3:0]) 19-36
19.17.9 Debug Data (DDATA[3:0]) 19-37
19.17.10 Device Test Enable (TEST) 19-37
19.18 Operating Mode Configuration Pins 19-37
19.19 Power Supply Pins 19-38

Chapter 20
Bus Operation

20.1 Features 20-1
20.2 Bus and Control Signals 20-1
20.2.1 Address Bus (A[22:0]) 20-2
20.2.2 Data Bus (D[31:0]) 20-2
20.2.3 Read/Write (R/W) 20-2
20.2.4 Transfer Acknowledge (TA) 20-2
20.2.5 Transfer Error Acknowledge (TEA) 20-3
20.3 Bus Exception: Double Bus Fault 20-3
20.4 Bus Characteristics 20-3
20.5 Data Transfer Mechanism 20-4
20.5.1 Bus Sizing 20-4
20.6 External Bus Interface Types 20-7
20.6.1 Interface for FLASH/SRAM Devices with Byte Strobes 20-8
20.6.2 Interface for FLASH/SRAM Devices without Byte Strobes 20-12
20.7 Burst Data Transfers 20-17
20.8 Misaligned Operands 20-18
20.9 Interrupt Cycles 20-19
20.10 Bus Errors 20-19
20.11 Bus Arbitration 20-21

Table 2-6. Notational Conventions (continued)

Instruction	Operand Syntax
<ea>y,<ea>x	Source and destination effective addresses, respectively
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
bc	Instruction cache
# <vector>	Identifies the 4-bit vector number for trap instructions
<>	identifies an indirect data address referencing memory
<xxx>	identifies an absolute address referencing memory
dn	Signal displacement value, *n* bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication
/	Arithmetic division
~	Invert; operand is logically complemented
&	Logical AND
|	Logical OR
^	Logical exclusive OR
<<	Shift left (example: D0 << 3 is shift D0 left 3 bits)
>>	Shift right (example: D0 >> 3 is shift D0 right 3 bits)
→	Source operand is moved to destination operand
←→	Two operands are exchanged
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion
If <condition> then <operations> else <operations>	Test the condition. If the condition is true, the operations in the then clause are performed. If the condition is false and the optional else clause is present, the operations in the else clause are performed. If the condition is false and the else clause is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.

Table 2-11 lists timings for MOVE.L.

Table 2-11. Move Long Execution Times

Source	Destination						
	Rx	**(Ax)**	**(Ax)+**	**–(Ax)**	**(d16,Ax)**	**(d8,Ax,Xi*SF)**	**(xxx).wl**
Dy	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
Ay	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
(Ay)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	3(1/1)	2(1/1)
(Ay)+	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	3(1/1)	2(1/1)
–(Ay)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	3(1/1)	2(1/1)
(d16,Ay)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	—	—
(d8,Ay,Xi*SF)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
(xxx).w	2(1/0)	2(1/1)	2(1/1)	2(1/1)	—	—	—
(xxx).l	2(1/0)	2(1/1)	2(1/1)	2(1/1)	—	—	—
(d16,PC)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	—	—
(d8,PC,Xi*SF)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
#<xxx>	1(0/0)	2(0/1)	2(0/1)	2(0/1)	—	—	—

Table 2-12 gives execution times for MOVE.L instructions accessing program-visible registers of the MAC unit, along with other MOVE.L timings. Execution times for moving contents of the ACC or MACSR into a destination location represent the best-case scenario when the store instruction is executed and no load, MAC, or MSAC instructions are in the MAC execution pipeline. In general, these store operations require only one cycle for execution, but if they are preceded immediately by a load, MAC, or MSAC instruction, the MAC pipeline depth is exposed and execution time is three cycles.

Table 2-12. Move Execution Times

Opcode	<ea>	Effective Address							
		Rn	**(An)**	**(An)+**	**–(An)**	**(d16,An)**	**(d8,An,Xi*SF)**	**(xxx).wl**	**#<xxx>**
move.l	<ea>,ACC	1(0/0)	—	—	—	—	—	—	1(0/0)
move.l	<ea>,MACSR	2(0/0)	—	—	—	—	—	—	2(0/0)
move.l	<ea>,MASK	1(0/0)	—	—	—	—	—	—	1(0/0)
move.l	ACC,Rx	1(0/0)	—	—	—	—	—	—	—
move.l	MACSR,CCR	1(0/0)	—	—	—	—	—	—	—
move.l	MACSR,Rx	1(0/0)	—	—	—	—	—	—	—
move.l	MASK,Rx	1(0/0)	—	—	—	—	—	—	—

5.4.3 Address Breakpoint Registers (ABLR, ABHR)

The address breakpoint low and high registers (ABLR, ABHR), Figure 5-6, define regions in the processor's data address space that can be used as part of the trigger. These register values are compared with the address for each transfer on the processor's high-speed local bus. The trigger definition register (TDR) identifies the trigger as one of three cases:

1. identically the value in ABLR
2. inside the range bound by ABLR and ABHR inclusive
3. outside that same range

	31 … 0
Field	Address
Reset	—
R/W	Write only. ABHR is accessible in supervisor mode as debug control register 0x0C using the WDEBUG instruction and via the BDM port using the RDMREG and WDMREG commands. ABLR is accessible in supervisor mode as debug control register 0x0D using the WDEBUG instruction and via the BDM port using the WDMREG command.
DRc[4–0]	0x0D (ABLR); 0x0C (ABHR)

Figure 5-6. Address Breakpoint Registers (ABLR, ABHR)

Table 5-6 describes ABLR fields.

Table 5-6. 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 5-7 describes ABHR fields.

Table 5-7. 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.

Breakpoint registers must be carefully configured in a development system if the processor is executing. The debug module contains no hardware interlocks, so TDR should be disabled while breakpoint registers are loaded, after which TDR can be written to define the exact trigger. This prevents spurious breakpoint triggers.

Because there are no hardware interlocks in the debug unit, no BDM operations are allowed while the CPU is writing the debug's registers (DSCLK must be inactive).

Note that the debug module requires the use of the internal bus to perform BDM commands. In Revision A, if the processor is executing a tight loop that is contained within a single aligned longword, the processor may never grant the internal bus to the debug module, for example:

```
        align4
label1: nop
        bra.b label1
```

or

```
        align4
label2: bra.w label2
```

The processor grants the internal bus if these loops are forced across two longwords.

5.7 Processor Status, DDATA Definition

This section specifies the ColdFire processor and debug module's generation of the processor status (PST) and debug data (DDATA) output on an instruction basis. In general, the PST/DDATA output for an instruction is defined as follows:

PST = 0x1, {PST = [0x89B], DDATA= operand}

where the {...} definition is optional operand information defined by the setting of the CSR.

The CSR provides capabilities to display operands based on reference type (read, write, or both). A PST value {0x8, 0x9, or 0xB} identifies the size and presence of valid data to follow on the DDATA output {1, 2, or 4 bytes}. Additionally, for certain change-of-flow branch instructions, CSR[BTB] provides the capability to display the target instruction address on the DDATA output {2, 3, or 4 bytes} using a PST value of {0x9, 0xA, or 0xB}.

5.7.1 User Instruction Set

Table 5-22 shows the PST/DDATA specification for user-mode instructions. Rn represents any {Dn, An} register. In this definition, the 'y' suffix generally denotes the source and 'x' denotes the destination operand. For a given instruction, the optional operand data is displayed only for those effective addresses referencing memory.The 'DD' nomenclature refers to the DDATA outputs.

Chapter 10
DMA Controller

The MCF5272 has a one-channel DMA controller that supports memory-to-memory DMA transfers that can be used for block data moves. This chapter describes in detail its signals, registers, and operating modes.

10.1 DMA Data Transfer Types

A source and destination address must be specified for any dual-address DMA transfer. These addresses can have different data transfer types, but there is always a read from the source address followed by a write to the destination address. On the MCF5272, sources and destinations can be SDRAM, external SRAM, or on-chip peripheral in any combination.

NOTE

Memory-to-memory DMA transfers run to completion if the assume request bit in the system configuration register, SCR[AR], is set. This generally prevents the CPU from recognizing interrupts and blocks bus accesses by other on-chip bus masters. It is best not to enable SCR[AR] when the DMA controller is in use. When AR = 0, the DMA controller allows the CPU and Ethernet controller to obtain bus access.

Table 10-1. DMA Data Transfer Matrix

Transfer Type	Source or Destination Address		
	SDRAM	External SRAM	On-Chip Peripheral
Byte (8 bits)	Yes	Yes	Yes
Word (16 bits)	Yes	Yes	Yes
Longword (32 bits)	Yes	Yes	Yes
Burst (4 x longword)	Yes	No	No

Note that transfers to on-chip peripherals are limited by the transfer type supported by a specific peripheral.

10.3.4 DMA Destination Address Register (DDAR)

The DDAR provides a 32-bit address that the DMA controller drives onto the internal address bus for all of the channel’s write accesses. The address is altered after each write access according to the addressing mode.

	31 0
Field	DESTADR
Reset	0000_0000_0000_0000_0000_0000_0000_0000
R/W	R/W
Addr	MBAR + 0x00F0

Figure 10-4. DMA Destination Address Register (DDAR)

10.3.5 DMA Byte Count Register (DBCR)

The DBCR is initially loaded with the number of bytes to be transferred during the DMA. After each transfer, the DBCR decrements by the number of bytes transferred. DIR[ASC] is set when the byte counter reaches zero. The user must ensure that the bytes remaining to be transferred and the transfer size are such that the byte counter decrements to zero or wraps around without setting the ASC flag.

	31 24	23 0
Field	—	BYTCNT
Reset	R/W	
R/W	0000_0000_0000_0000_0000_0000_0000_0000	
Addr	MBAR + 0x00E8	

Figure 10-5. DMA Byte Count Register (DBCR)

11.5.20 Pointer-to-Transmit Descriptor Ring (ETDSR)

This register provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be long-word aligned. However, it is recommended it be aligned on a 16 byte boundary (address evenly divisible by 16) in order to improve bus utilization. Bits 1 and 0 should be set to 0 by the user. Non-zero values in these two bit positions are ignored by the hardware.

This register is not reset and must be initialized by the user prior to operation.

	31–16
Field	X_DES_START
Reset	Undefined
R/W	Read/Write

	15–2	1	0
Field	X_DES_START	0	0
Reset	Undefined		
R/W	Read/Write		
Addr	MBAR + 0xC14		

Figure 11-25. Pointer-to-Transmit Descriptor Ring (ETDSR)

Table 11-28. ETDSR Field Descriptions

Bits	Name	Description
31–2	X_DES_START	Pointer to start of transmit buffer descriptor queue.
1–0	—	Reserved, should be cleared.

11.5.21 Receive Buffer Size Register (EMRBR)

The EMRBR register dictates the maximum size of all receive buffers. Note that because receive frames are truncated at 2k-1 bytes, only bits 10–4 are used. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, R_BUFF_SIZE must be set to MAX_FL or larger. The R_BUFF_SIZE must be evenly divisible by 16. To insure this, bits 3–0 are forced low. To minimize bus utilization (descriptor fetches), it is recommended that R_BUFF_SIZE be at least 256 bytes.

This register, Figure 11-26, is not reset and must be initialized by the user prior to operation.

	31–16
Field	—
Reset	0000_0000_0000_0000
R/W	Read/Write

	15–11	10–4	3–0
Field	—	R_BUFF_SIZE	—
Reset	0000_0		0000
R/W	Read/Write		
Addr	MBAR + 0xC18		

Figure 11-26. Receive Buffer Size (EMRBR)

Table 11-29. EMRBR Field Descriptions

Bits	Name	Description
31–11	—	Reserved, should be cleared.
10–4	R_BUFF_SIZE	Receive buffer size.
3–0	—	Reserved, should be cleared.

11.6.1 FEC Buffer Descriptor Tables

The data for the fast Ethernet frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. Each buffer has a pointer to it in a buffer descriptor (BD). In addition to pointing to the buffer, the BD contains the current state of the buffer. To permit maximum user flexibility, the BDs are also located in external memory.

Software defines buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] produces the buffer. Software writing to either TDAR or RDAR tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and processes the buffers after they have been defined. After the data DMA is complete and the BDs have been written by the DMA engine, RxBD[E] or TxBD[R] are cleared by hardware to indicate that the buffer has been processed. Software may poll the BDs to detect when the buffers have been processed or may rely on the buffer/frame interrupts.

The ETHER_EN signal operates as a reset to the BD/DMA logic. When ETHER_EN is negated, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive BD must be initialized by software (write 0x0000_0000 to the most significant word of buffer descriptor) before the ETHER_EN bit is set.

The BDs are organized in two separate rings, one for receive buffers and one for transmit buffers. ERDSR defines the starting address of the receive BDs and ETDSR the same for the transmit BDs. The last buffer descriptor in each ring is defined by the wrap (W) bit. When set, W indicates that the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffers descriptor rings must start on a double-word boundary.

The format of the transmit and receive buffer descriptors are given in Figure 11-27 and Figure 11-28.

11.6.1.1 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first word and the pointer in the second word. When the buffer has been sent as a DMA, the FEC will modify the E, L, M, LG, NO, SH, CR, and OV bits and write the length of the used portion of the buffer in the first word. The M, LG, NO, SH, CR, and OV bits in the first word of the buffer descriptor are modified by the FEC only when the L bit is set.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0	E	RO1	W	RO2	L	—		M	BC	MC	LG	NO	SH	CR	OV	TR
+2	DATA LENGTH															
+4	Rx Data Buffer Pointer A[31–16]															
+6	Rx Data Buffer Pointer A[15–0]															

Figure 11-27. Receive Buffer Descriptor (RxBD)

The first word of the RxBD contains control and status bits. Its format is detailed below.

- Supports remote wakeup
- Detects start-of-frame and missed start-of-frame for isochronous endpoint synchronization
- Notification of start-of-frame, reset, suspend, and resume events

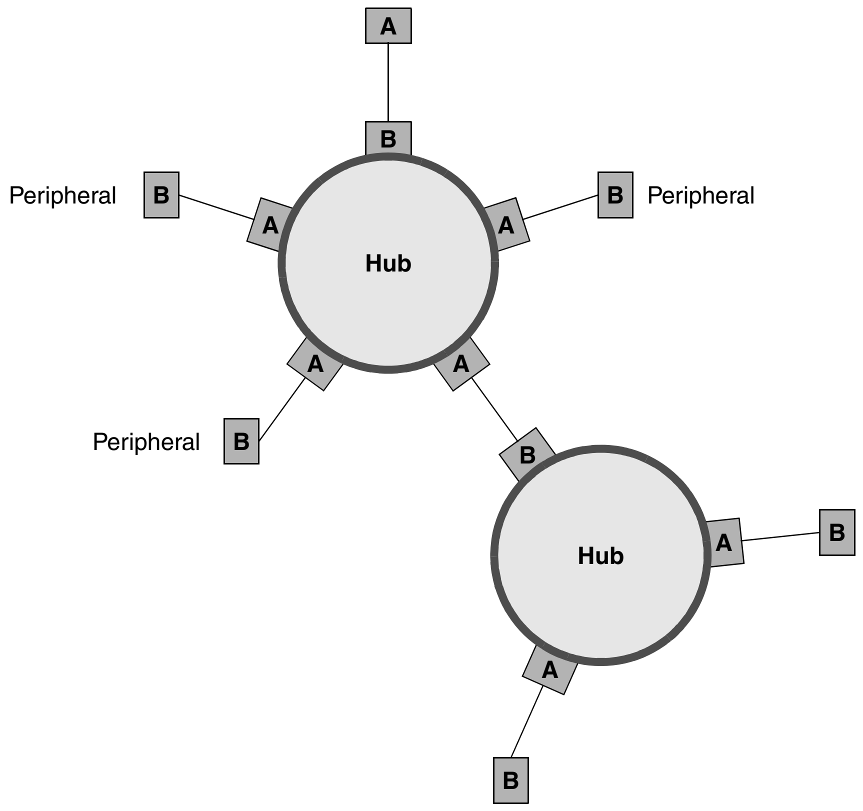


Figure 12-1. The USB "Tiered Star" Topology

12.2 Module Operation

The MCF5272 USB system consists of a protocol state machine which controls the transmitter and receiver modules. The state machine implements only the USB function state diagram. The MCF5272 USB controller can serve as a USB function endpoint, but cannot serve as a USB host.

12.2.1 USB Module Architecture

A block diagram of the USB module is shown in Figure 12-2. The module is partitioned into five functional blocks. These blocks are USB internal transceiver, clock generator, USB control logic, USB request processor, and endpoint controllers.

15.3.5 Timer Event Registers (TER0–TER3)

TERs are used to report events recognized by the timer. On recognition of an event, the timer sets the appropriate TER*n* bit, regardless of the corresponding interrupt enable bits (ORI and CE) in the TMR*n*. Writing a 1 to a bit clears it; writing 0 has no effect. Both bits must be cleared before the timer can negate the request to the interrupt controller. Both bits may be cleared simultaneously.

	15–2	1	0
Field	—	REF	CAP
Reset	0000_0000_0000_0000		
R/W	Read/Write		
Addr	MBAR + 0x210 (TER0); 0x230 (TER1); 0x250 (TER2); 0x270 (TER3)		

Figure 15-6. Timer Event Registers (TER0–TER3)

Table 15-2 describes TER*n* fields.

Table 15-2. TER*n* Field Descriptions

Bits	Name	Description
15–2	—	Reserved, should be cleared.
1	REF	Output reference event. 0 The counter has not reached the TRR value 1 The counter reached the TRR value. TMR[ORI] is used to enable the interrupt request caused by this event. Write a 1 to this bit to clear the event condition.
0	CAP	Capture event. 0 The counter value has not been latched into the TCAP. 1 The counter value is latched in the TCAP. TMR[CE] is used to enable capture and the interrupt request caused by this event. Write a 1 to this bit to clear the event condition.

Table 16-1. UART Module Programming Model

MBAR Offset		[31:24]	[23:16]	[15:8]	[7:0]
UART0	UART1				
0x100	0x140	UART mode registers[1]—(UMR1*n*) [p. 16-4], (UMR2*n*) [p. 16-6]	—		
0x104	0x144	(Read) UART status registers—(USR*n*) [p. 16-7]	—		
		(Write) UART clock-select register[1]—(UCSR*n*) [p. 16-8]	—		
0x108	0x148	(Read) Do not access.[2]	—		
		(Write) UART command registers—(UCR*n*) [p. 16-9]	—		
0x10C	0x14C	(UART/Read) UART receiver buffers—(URB*n*) [p. 16-10]	—		
		(UART/Write) UART transmitter buffers—(UTB*n*) [p. 16-11]	—		
0x110	0x150	(Read) UART input port change registers—(UIPCR*n*) [p. 16-11]	—		
		(Write) UART auxiliary control registers[1]—(UACR*n*) [p. 16-12]	—		
0x114	0x154	(Read) UART interrupt status registers—(UISR*n*) [p. 16-12]	—		
		(Write) UART interrupt mask registers—(UIMR*n*) [p. 16-12]	—		
0x118	0x158	(Read) Do not access.[2]	—		
		UART divider upper registers—(UDU*n*) [p. 16-14]	—		
0x11C	0x15C	(Read) Do not access.[2]	—		
		UART divider lower registers—(UDL*n*) [p. 16-14]	—		
0x120	0x160	(Read) UART autobaud register MSB—(UABU*n*) [p. 16-17]	—		
		(Write) Do not access.[2]	—		
0x124	0x164	(Read) UART autobaud register LSB—(UABL*n*) [p. 16-17]	—		
		(Write) Do not access[2]	—		
0x128	0x168	UART Transmitter FIFO registers—(UTF*n*) [p. 16-4]	—		

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*)

An internal interrupt request signal ($\overline{IRQ}$) is provided to notify the interrupt controller of an interrupt condition. The output is the logical NOR of unmasked UISR*n* bits. The interrupt levels of the UART modules are programmed in SIM register ICR2. See Section 7.2, "Interrupt Controller Registers."

Table 16-15 briefly describes the UART module signals.

NOTE

The terms 'assertion' and 'negation' are used to avoid confusion between active-low and active-high signals. 'Asserted' indicates that a signal is active, independent of the voltage level; 'negated' indicates that a signal is inactive.

Table 16-15. UART Module Signals

Signal	Description
Transmitter Serial Data Output (URT_TxD)	URT_TxD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out on URT_TxD on the falling edge of the clock source, with the least significant bit (lsb) sent first.
Receiver Serial Data Input (URT_RxD)	Data received on URT_RxD is sampled on the rising edge of the clock source, with the lsb received first.
Clear-to-Send ($\overline{\text{URT\_CTS}}$)	This input can generate an interrupt on a change of state.
Request-to-Send ($\overline{\text{URT\_RTS}}$)	This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. It can control serial data flow when connected to a transmitter's $\overline{\text{CTS}}$.
Clock (URT_CLK)	The UART's external clock source. It can be used in 1x or 16x mode. When both the transmitter and receiver use the timer as the clock source (UCR = 0xDD), the 16x clock is driven out on UARTCLK. If either the transmitter or receiver use an external clock (1x or 16x), URT_CLK is an input.

Figure 16-22 shows a signal configuration for a UART/RS-232 interface.

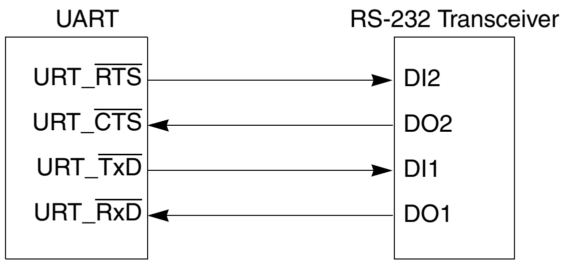


Figure 16-22. UART/RS-232 Interface

16.5 Operation

This section describes operation of the clock source generator, transmitter, and receiver.

16.5.1 Transmitter/Receiver Clock Source

CLKIN serves as the basic timing reference for the clock source generator logic, which consists of a clock generator and a programmable 16+4-bit divider (UDU, UDL, UFPD) dedicated to the UART. The clock generator cannot produce standard baud rates if CLKIN is used, so the 16-bit divider should be used.

19.16.1.5 UART1 CTS ($\overline{\text{URT1\_CTS}}$/QSPI_CS2)

UART1: $\overline{\text{URT1\_CTS}}$ is the clear-to-send input indicating to the UART1 module that it can begin data transmission.

QSPI mode: This output pin provides a QSPI peripheral chip select, QSPI_CS2, when in Master mode. QSPI_CS2 can be programmed to be active high or low.

19.16.1.6 UART1 RTS ($\overline{\text{URT1\_RTS}}$/$\overline{\text{INT5}}$)

Interrupt mode: This pin can be used as $\overline{\text{INT5}}$.

UART1: The $\overline{\text{URT1\_RTS}}$ output is an automatic request to send output from the UART1 module. It can be configured to be asserted and negated as a function of the RxFIFO level.

19.16.1.7 Serial Data Output (DOUT0/URT1_TxD)

IDL mode: The DOUT0 output is for clocking data out of IDL port 0. Data is clocked out of DOUT0 on the rising edge of DCL0.

GCI mode: The DOUT0 output is for clocking data out of GCI port 0. DCL0 is twice the bit rate (two clocks per data bit).

UART1: URT1_TxD is the transmitter serial data output for the UART1 module. The output is held high ('mark' condition) when the transmitter is disabled, idle, or operating in the local loop back mode. Data is shifted out, least significant bit first, on this pin at the falling edge of the serial clock source.

19.16.1.8 D-Channel Request(DREQ0/PA10)

IDL mode: This pin can be independently configured as the DREQ0 output for signaling to a layer-1 S/T transceiver that a frame of data is ready to be sent on the port 0 D channel.

Port A mode: In GCI or IDL modes this pin can be independently configured as PA10.

19.16.1.9 QSPI Chip Select 1 (QSPI_CS1/PA11)

QSPI mode: QSPI_CS1 is a QSPI peripheral chip select.

Port A mode: In GCI or IDL modes this pin can be independently configured as PA11.

19.16.2 GCI/IDL TDM Port 1

Physical Layer Interface port 1 is an additional GCI/IDL port. Also internally connected to these pins are GCI/IDL serial ports 2 and 3.

19.16.2.1 GCI/IDL Data Clock (DCL1/GDCL1_OUT)

IDL mode: DCL1 is the data clock used to clock data in and out of the DIN1 and DOUT1 pins for IDL port 1. Data is clocked in to DIN1 on the falling edge of DCL1. Data is clocked out of DOUT1 on the rising edge of DCL1.

Table 19-8. MCF5272 Bus Width Selection

WSEL	Data Bus Width
0	32 bits
1	16 Bits

Table 19-9. MCF5272 $\overline{\text{CS0}}$ Memory Bus Width Selection

BUSW1	BUSW0	$\overline{\text{CS0}}$ Bus Width
0	0	32 bits
0	1	8 bits
1	0	16 bits
1	1	Reserved, do not use.

Table 19-10. MCF5272 High Impedance Mode Selection

HI-Z	Data Bus Width
0	Enable HI-Z mode, all pins are high impedance
1	Normal operation

19.19 Power Supply Pins

The VDD/GND pins are connected to 3.3 V supply and associated ground. When the on-chip USB transceiver is used, the USB_VDD and USB_GND pins are connected to the 3.3 V device supply and associated ground. When the on-chip USB transceiver is not used, USB_GND should be connected to the device GND and USB_VDD left unconnected. Refer to Section 12.2.1.1 USB Transceiver Interface.

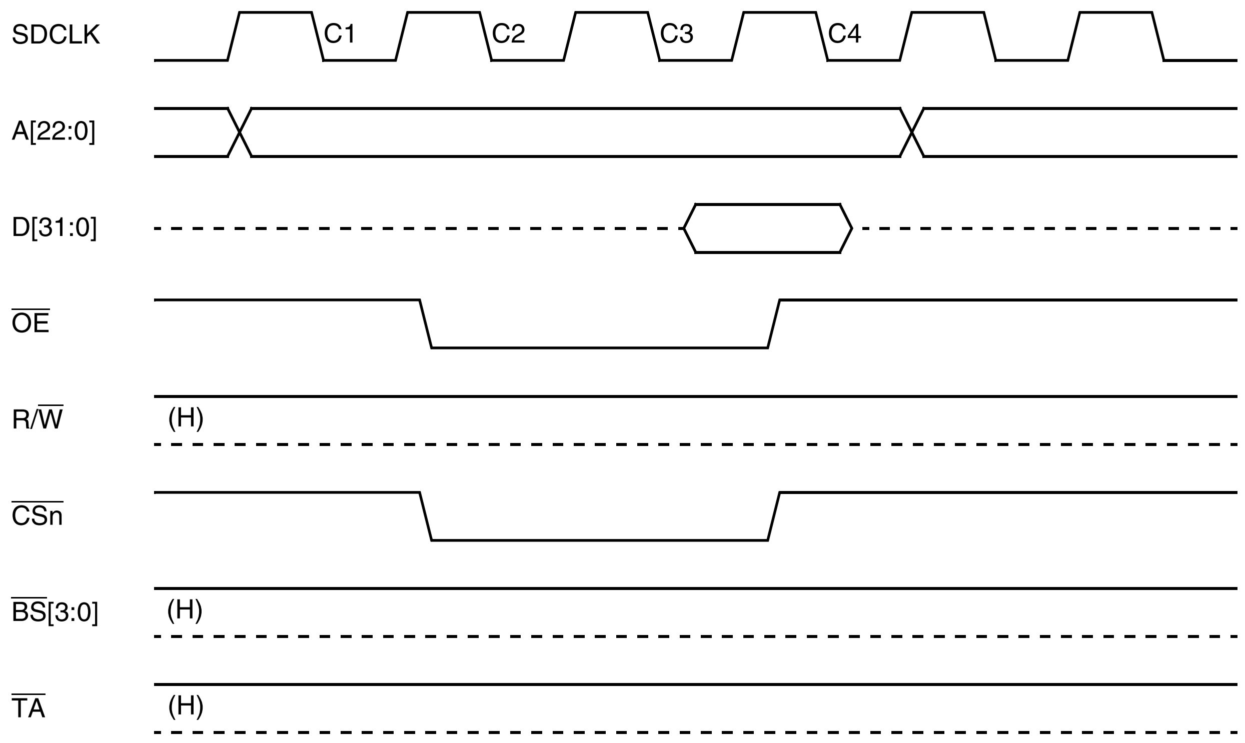


Figure 20-16. Longword Read with Address Setup and Address Hold; EBI = 11; 32-Bit Port, Internal Termination

20.12.2 Normal Reset

External normal resets should be performed anytime it is important to maintain the data stored in SDRAM during a reset. An external normal reset is performed when an external device asserts RSTI while negating DRESETEN. At power on reset both RSTI and DRESETEN must be asserted simultaneously. If DRESETEN is not asserted at the same time as RSTI at power up, the SDRAMC cannot be initialized by software.

During an external normal reset, RSTI must be asserted for a minimum of six CLKINs. Figure 20-22 is a functional timing diagram of external normal reset operation, illustrating relationships among RSTI, DRESETEN, RSTO, mode selects, and bus signals. RSTI and DRESETEN are internally synchronized on consecutive falling and rising clocks before being used and must meet the specified setup and hold times to the falling edge of CLKIN only if recognition by a specific falling edge is required

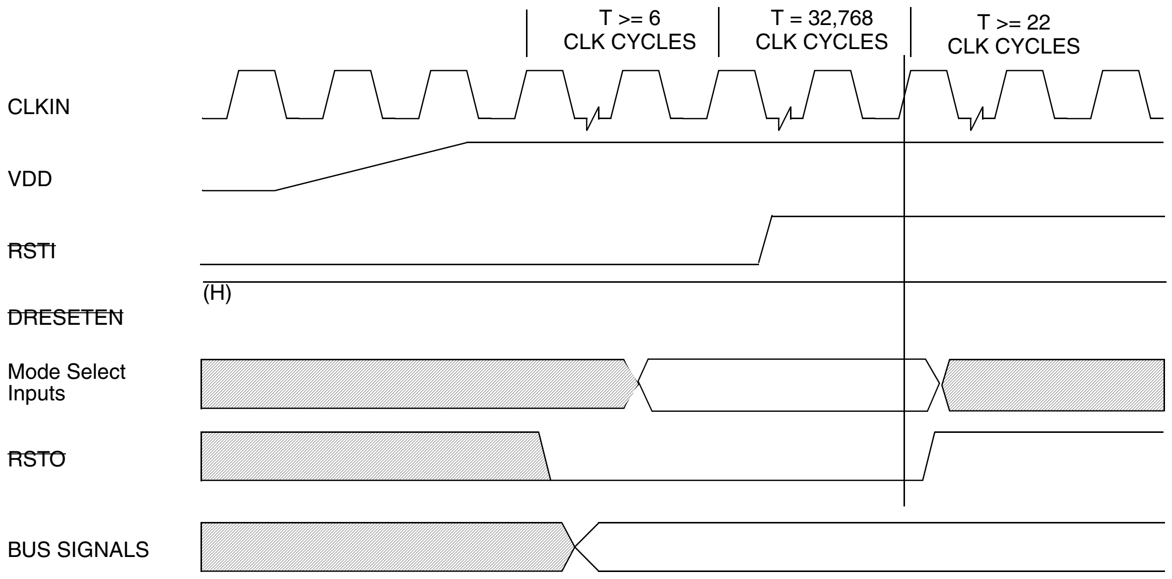


Figure 20-22. Normal Reset Timing

The levels of the mode select inputs, QSPI_Dout/WSEL, QSPI_CLK/BUSW1, QSPI_CS0/BUSW0, and HiZ are sampled when RSTO negates and they select the port size of CS0 and the physical data bus width after a master reset occurs. RSTO is asserted as long as RSTI is asserted and remains asserted for 32,768 CLKIN cycles after RSTI is negated. For proper normal reset operation, DRESETEN must be negated as long as RSTI is asserted.

The levels of the mode select inputs, QSPI_Dout/WSEL, QSPI_CLK/BUSW1, and QSPI_CS0/BUSW0, are sampled when RSTO negates and they select the port size of CS0 and the physical data bus width after a master reset occurs. The INTx signals are synchronized and are registered on the last falling edge of CLKIN where RSTI is asserted.

During the normal reset period, all outputs are driven to their default levels. Once RSTO negates, all bus signals continue to remain in this state until the ColdFire core begins the first bus cycle for reset exception processing.

23.3.2 Processor Bus Input Timing Specifications

Table 23-7 lists processor bus input timings.

NOTE

All processor bus timings are synchronous; that is, input setup/hold and output delay with respect to the rising edge of a reference clock. The reference clock is the SDCLK output.

All other timing relationships can be derived from these values.

Table 23-7. Processor Bus Input Timing Specifications

Name	Characteristic[1]	0–66 MHz		Unit
		Min	Max	
Control Inputs				
B1a [2]	$\overline{\text{RSTI}}$ valid to SDCLK (setup)	6.5	—	nS
B1b [2]	$\overline{\text{TA}}$ valid to SDCLK (setup)	8	—	nS
B1c [2]	$\overline{\text{TEA}}$ valid to SDCLK (setup)	8	—	nS
B1d [2]	$\overline{\text{INTx}}$ valid to SDCLK (setup)	8	—	nS
B1e [2]	$\overline{\text{BKPT}}$ valid to PSTCLK (setup)	10	—	nS
B1f	Mode selects (BUSW[1:0], WSEL, $\overline{\text{HIZ}}$) valid to SDCLK (setup) (when $\overline{\text{RSTI}}$ asserted)	8	—	nS
B2d	SDCLK to asynchronous control inputs ($\overline{\text{RSTI}}$, $\overline{\text{TA}}$, $\overline{\text{TEA}}$, $\overline{\text{INTx}}$) invalid (hold)	2	—	nS
B2e	SDCLK to mode selects (BUSW[1,0], WSEL, $\overline{\text{HIZ}}$) invalid (hold) (when $\overline{\text{RSTI}}$ asserted)	2	—	nS
B2f	PSTCLK to asynchronous control input $\overline{\text{BKPT}}$ invalid (hold)	0	—	nS
B3	$\overline{\text{RSTI}}$ width low	10T	—	nS
Data Inputs				
B4	Data input (D[31:0]) valid to SDCLK (setup)	14	—	nS
B5	SDCLK to data input (D[31:0]) invalid (hold)	0	—	nS

[1] All timing references to SDCLK are given to its rising edge when bit 3 of the SDRAM control register is 0.

[2] $\overline{\text{RSTI}}$, $\overline{\text{TA}}$, $\overline{\text{TEA}}$, and $\overline{\text{INT}x}$ are synchronized internally. The setup time must be met only if recognition is needed on a particular clock edge.