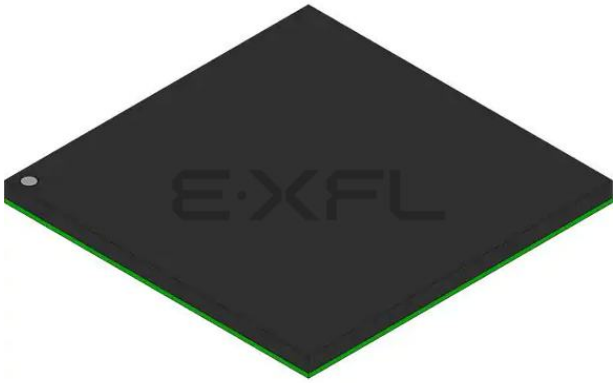


Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

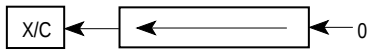
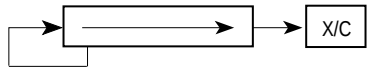
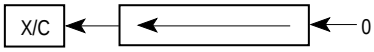
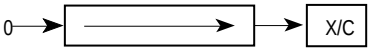
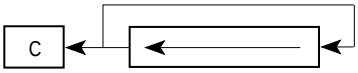
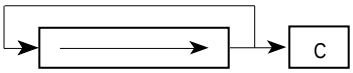
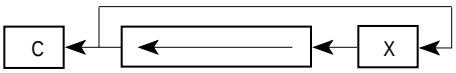
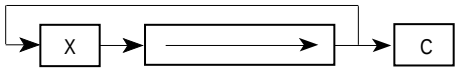
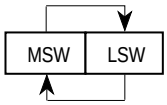
Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Active
Core Processor	CPU32+
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	Communications; CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10Mbps (1)
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	357-BBGA
Supplier Device Package	357-PBGA (25x25)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68360czq25l

Table 5-7. Shift and Rotate Operations

Instruction	Operand Syntax	Operand Size	Operation
ASL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
SWAP	Dn	16	

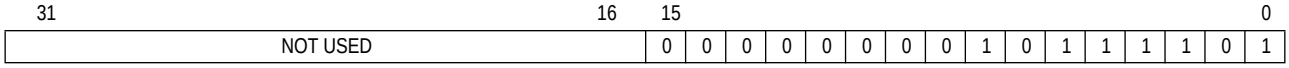
5.3.3.6 BIT MANIPULATION INSTRUCTIONS. Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). All bit manipulation operations can be performed on either registers or memory. The bit number is specified as immediate data or in a data register. Register operands are 32 bits, and memory operands are 8 bits. Table 5-8 is a summary of bit manipulation instructions.

Table 5-15. T8-Bit Independent Variable Entries

X (Subroutine)	X (Instruction)	Y
0	0	0
1	256	16
2	512	32
3	768	48
4	1024	64
5	1280	80
6	1536	96
7	1792	112
8	2048	128
9	2304	112
10	2560	96
11	2816	80
12	3072	64
13	3328	48
14	3584	32
15	3840	16
16	4096	0

The first column is the value passed to the subroutine, the second column is the value expected by the table instruction, and the third column is the result returned by the subroutine.

The following value has been calculated for independent variable X:



Since X is an 8-bit value, the upper four bits are used as a table offset, and the lower four bits are used as an interpolation fraction. The following results are obtained from the subroutine:

Table Entry Offset ⇒ Dx [4:7] = $B = 11

Interpolation Fraction ⇒ Dx [0:3] = $D = 13

Thus, Y is calculated as follows:

Y = 80 + (13 (64 – 80)) / 16 = 67

If the 8-bit value for X were used directly by the table instruction, interpolation would be incorrectly performed between entries 0 and 1. Data must be shifted to the left four places before use:

LSL.W #4, Dx

"not ready/come again" response. Once the receive data latch has been loaded, the CPU is released to act on the new data. Response data overwrites the "not ready" response when the CPU has completed the current operation.

Data written into the output shift register appears immediately on the DSO signal. In general, this action changes the state of the signal from a high ("not ready" response status bit) to a low (valid data status bit) logic level. However, this level change only occurs if the command completes successfully. Error conditions overwrite the "not ready" response with the appropriate response that also has the status bit set.

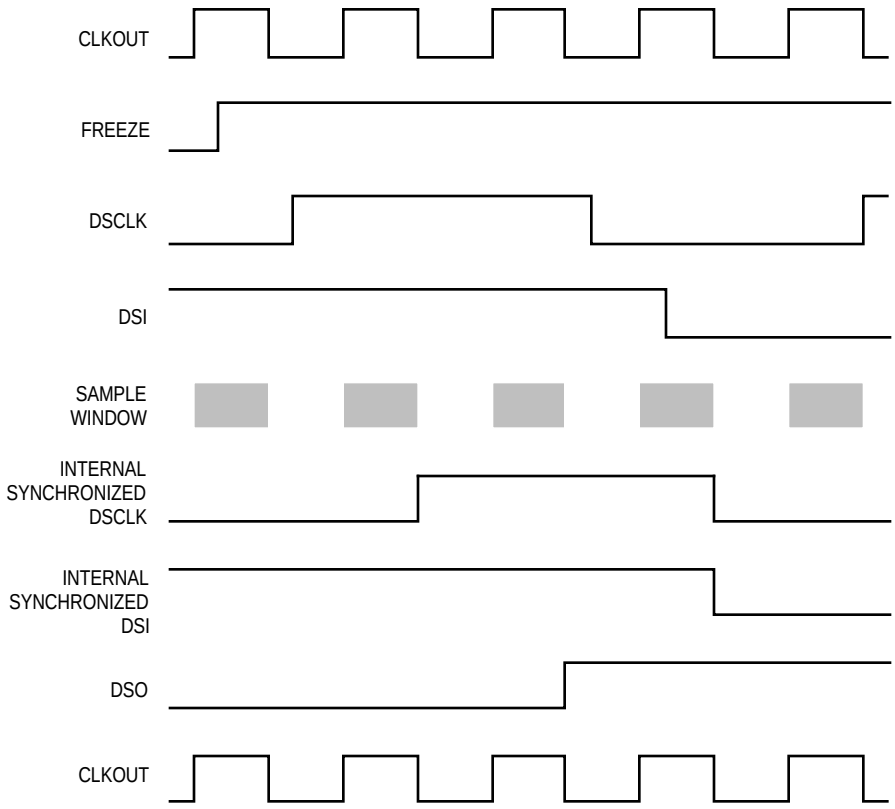


Figure 5-23. Serial InterfaceTiming Diagram

A user can use the state change on DSO to signal hardware that the next serial transfer may begin. A timeout of sufficient length to trap error conditions that do not change the state of DSO should also be incorporated into the design. Hardware interlocks in the CPU prevent result data from corrupting serial transfers in progress.

5.6.2.7.2 Development System Serial Logic. The development system, as the master of the serial data link, must supply the serial clock. However, normal and BDM operations could interact if the clock generator is not properly designed.

Breakpoint requests are made by asserting $\overline{\text{BKPT}}$ to the low state in either of two ways. The primary method is to assert $\overline{\text{BKPT}}$ during a single bus cycle for which an exception is desired. Another method is to assert $\overline{\text{BKPT}}$, then continue to assert it until the CPU32+ responds by asserting FREEZE. This method is useful for forcing a transition into BDM when

1 = Synchronous operation of the memory controller (external MC68030-type master only).

When the SRAM controller is used, $\overline{CS}$ and $\overline{DSACK}$ assertion and negation timings are synchronous. The CSNTQ and the TRLXQ attributes may be set as desired.

When the DRAM controller is used, $\overline{CAS}$ and $\overline{DSACK}$ are negated synchronously to the QUICC clock.

Only when the SYNC bit is set, is parity support possible for an external MC68030-type master.

Table 6-12 summarizes the effects of the various combinations of the SYNC bit in the GMR and the BSTM bit in the MCR.)

Table 6-12. SYNC-BSTM Bit Combination Summary (MC68030-Type External Master)

SYNC-BSTM	Result
00	MC68030-type master and QUICC can be asynchronous. Lowest performance, since the external AS signal is synchronized prior to being used. Parity support is not available.
01	External MC68030-type master is running synchronously with the QUICC, and the user desires to make external-to-external SRAM accesses as fast as possible. The CSNTQ and TRLXQ attributes may not be used. Does not affect DRAM performance. Parity support is not available.
10	Do not use.
11	Not as fast as case 01, but CSNTQ and TRLXQ attributes may be used. Parity support is available.

NOTES:
If Synchronous bus mode is selected, glue logic is required for external MC68030-type bus master (including MC68360) ensuring that proper set up time for address strobe assertion is met

OPAR—Odd Parity

This attribute is used to program odd or even parity. It may also be used to generate parity errors for testing purposes by writing the DRAM/SRAM with OPAR = 1 and reading the DRAM/SRAM with OPAR = 0.

0 = Even parity
1 = Odd parity

PBEE—Parity Bus Error Enable

This attribute is used to enable an internal bus error if a parity error is detected. It is applicable only when the QUICC is the bus master; if in slave mode the $\overline{PERR}$ will be asserted if the parity function is enabled but it will not cause bus error regardless of the setting of this bit. The $\overline{BERR}$ signal will be internally asserted on the memory read cycle.

0 = Disable internal bus error.
1 = Enable internal bus error.

NOTE

Using the internal bus error requires a longer data setup time for read cycles.

TSS40—$\overline{TS}$ Sample (MC68EC040)

This attribute is used to control the MC68EC040 cycles. When the MC68EC040 address to clock setup timing does not meet the memory controller decoding time, the memory

7.6.5 IDMA Examples

The following paragraphs provide IDMA examples.

7.6.5.1 SINGLE BUFFER EXAMPLES. To see three examples of single buffer operation, see the 7.6.4.6.1 Dual Address Mode.

7.6.5.2 BUFFER CHAINING EXAMPLE. •The following example shows the setup required to initialize IDMA channel 1 to perform three buffer transfers using the buffer chaining mode. This example will move 16 bytes from address 0 to address $1000, then 16 bytes from address $100 to $1100, and then 16 bytes from address 200 to $1200.

1. Initialize basic IDMA channel 1 registers:
2. ICCR = $0720. Recommended normal configuration.
3. FCR1 = $89. Source function code is 1000; destination function code is 1001.
4. SAPR1 not initialized. Will be initialized later by RISC controller.
5. DAPR1 not initialized. Will be initialized later by RISC controller.
6. BCR1 not initialized. Will be initialized later by RISC controller.
7. CSR1 = $FF. Clear any CSR bits that are currently set.
8. CMAR1 = $00. Disable interrupts for this example.
9. CMR1 = $530C. The RISC controls the IDMA activity (RCI bit is set). The IDMA channel uses 12.5% of the bus bandwidth. The source and destination size are long word. Do not set the STR bit yet.
10. Issue the INIT_IDMA command to the RISC controller. This command is not required unless the IDMA was reset with the CMR RST bit while in the buffer chaining or auto buffer modes.
11. CR = $0591. Issue INIT_IDMA command to IDMA channel 1.
12. Initialize the IDMA channel 1 parameter RAM:
13. IBASE = $0000. This points the beginning of the IDMA BDs. The value of $0000 means that the first IDMA BD is located at the beginning of the internal dual-port RAM.
14. Initialize the first IDMA BD:
15. BD1_STATUS = $0000. This is offset 0 from the BD. Set up all bits except the V-bit.
16. BD1_Data_Length = $00000010. Transfer 16 bytes.
17. BD1_Source_Pointer = $00000000. Source address.
18. BD1_Destination_Pointer = $00001000. Destination address.
19. BD1_STATUS = $8000. Set the V-bit. It is good practice to set the V-bit last; however, in this example the IDMA channel is not yet enabled, so it could have been set earlier.
20. Initialize the second IDMA BD:
21. BD2_STATUS = $0000. This is offset 0 from the BD. Set up all bits except the

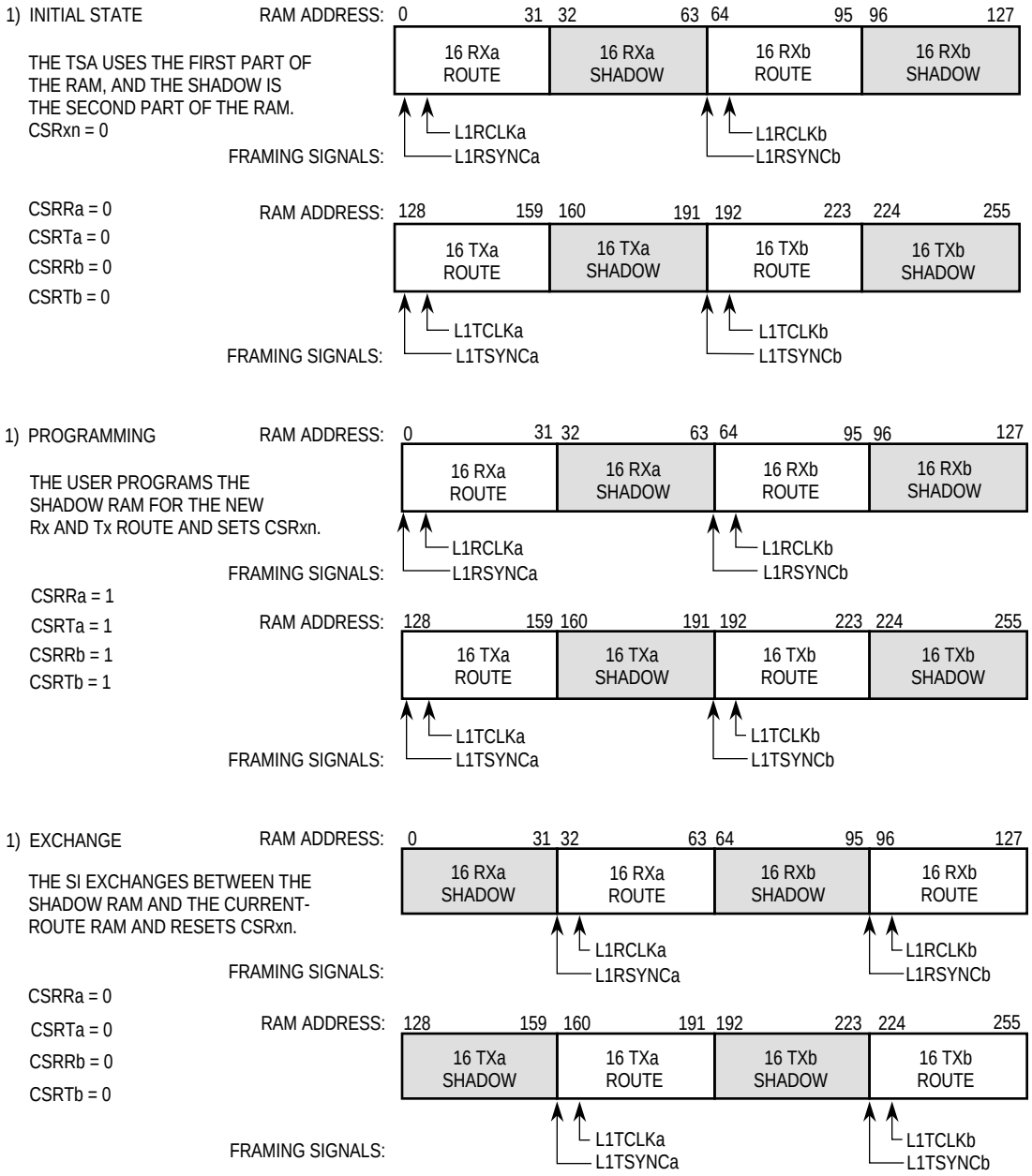


Figure 7-28. SI RAM Dynamic Changes

7.8.5 SI Registers

The following paragraphs describe the SI registers.

7.8.5.1 SI GLOBAL MODE REGISTER (SIGMR). The 8-bit SIGMR defines the RAM division modes. The SIGMR appears to the user as a memory-mapped, read-write register and is cleared at reset.

7	6	5	4	3	2	1	0
—				ENb	ENa	RDM1–RDM0	

Bits 7–4—Reserved

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. It is suggested that the user write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

7.10.7.3 MAXIMUM RECEIVE BUFFER LENGTH REGISTER (MRBLR). Each SCC has one MRBLR to define the receive buffer length for that SCC. MRBLR defines the maximum number of bytes that the QUICC will write to a receive buffer on that SCC before moving to the next buffer. The QUICC may write fewer bytes to the buffer than MRBLR if a condition such as an error or end-of-frame occurs, but it will never write more bytes than the MRBLR value. It follows, then, that buffers supplied by the user for use by the QUICC should always be of size MRBLR (or greater) in length.

The transmit buffers for an SCC are not affected in any way by the value programmed into MRBLR. Transmit buffers may be individually chosen to have varying lengths, as needed. The number of bytes to be transmitted is chosen by programming the data length field in the Tx BD.

NOTE

MRBLR is not intended to be changed dynamically while an SCC is operating. However, if it is modified in a single bus cycle with one 16-bit move (NOT two 8-bit bus cycles back-to-back), then a dynamic change in receive buffer length can be successfully achieved. This takes place when the CP moves control to the next Rx BD in the table. Thus, a change to MRBLR will not have an immediate effect. To guarantee the exact Rx BD on which the change will occur, the user should change MRBLR only while the SCC receiver is disabled.

The MRBLR value should be greater than zero for all modes.

For Ethernet and HDLC the MRBLR should be evenly divisible by 4. In totally transparent mode, MRBLR should also be divisible by 4, unless the receive FIFO width (RFW) bit in GSMR is set to 8 bits.

7.10.7.4 RECEIVER BD POINTER (RBPTR). The RBPTR for each SCC channel points to the next BD that the receiver will transfer data to when it is in idle state or to the current BD during frame processing. After a reset or when the end of the BD table is reached, the CP initializes this pointer to the value programmed in the RBASE entry. Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled or when the user is sure that no receive buffer is currently in use.

7.10.7.5 TRANSMITTER BD POINTER (TBPTR). The TBPTR for each SCC channel points to the next BD that the transmitter will transfer data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of the BD table is reached, the CP initializes this pointer to the value programmed in the TBASE entry.

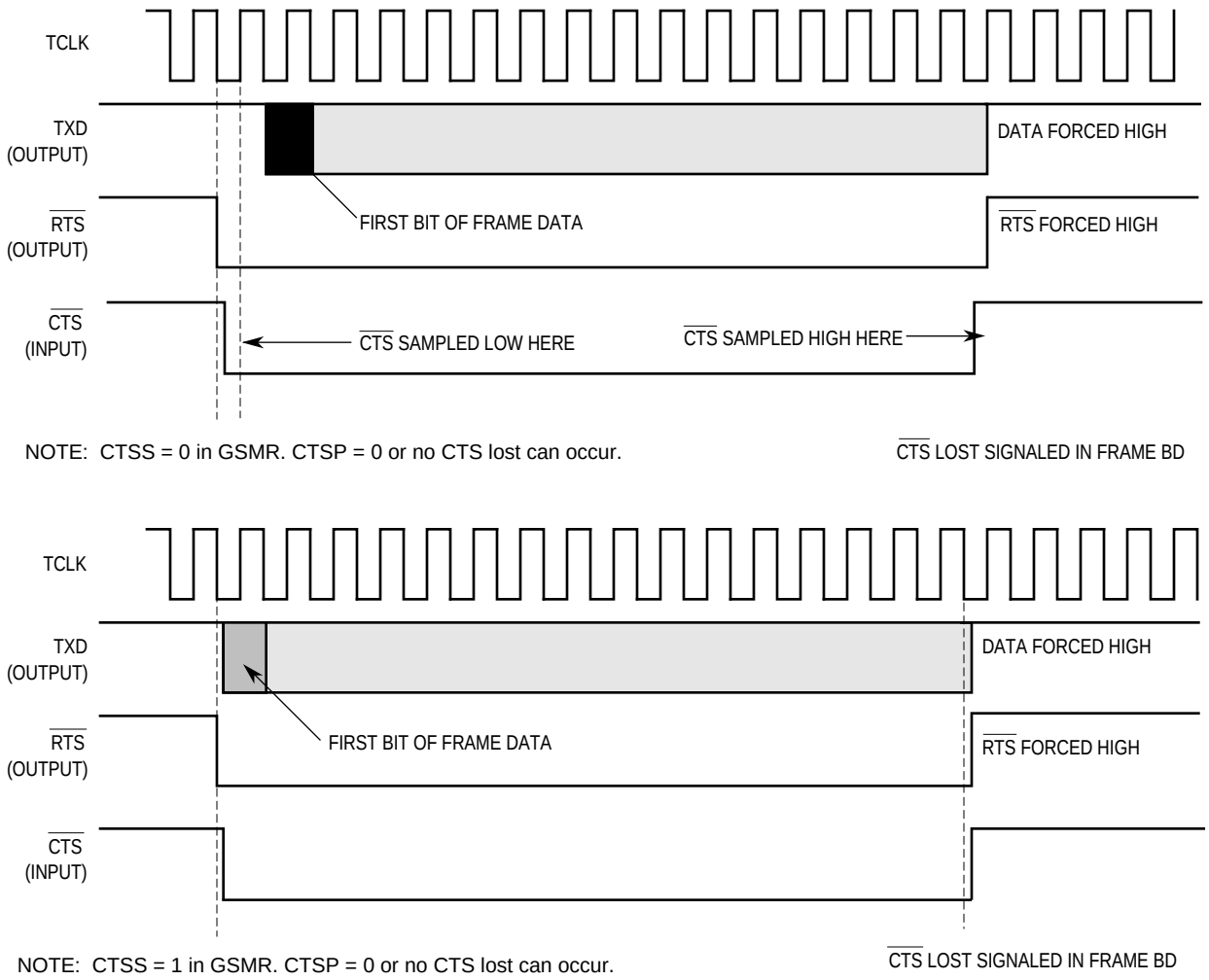
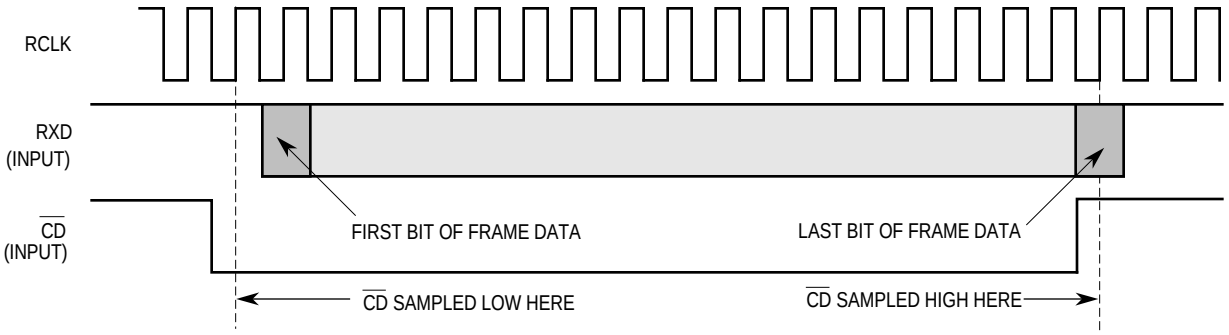


Figure 7-41. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

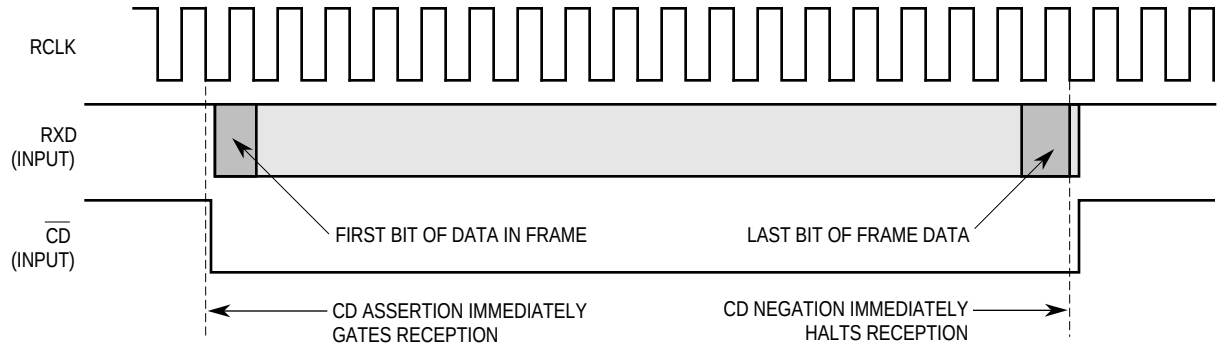
NOTE

If the CTSS bit in GSMR is set, all $\overline{\text{CTS}}$ transitions must occur while the transmit clock is low.

Reception delays are determined by the $\overline{\text{CD}}$ pin as shown in Figure 7-42. If the CDS bit in GSMR is zero, then the $\overline{\text{CD}}$ pin is sampled on the rising receive clock edge prior to data being received. If the CDS bit in GSMR is one, then the $\overline{\text{CD}}$ pin transitions cause data to be immediately gated into the receiver.



NOTES:
1. CDS = 0 in GSMR; CDP = 0.
2. If CD is negated prior to the last bit of the receive frame, CD LOST is signaled in the frame BD.
3. If CDP = 1, CD LOST cannot occur, and CD negation has no effect on reception.



NOTES:
1. CDS = 1 in GSMR; CDP = 0.
2. If CD is negated prior to the last bit of the receive frame, CD lost is signaled in the frame BD.
3. If CDP = 1, CD lost cannot occur, and CD negation has no effect on reception.

Figure 7-42. Using CD to Control Reception of Synchronous Protocols

If the CD pin is programmed to envelope the data, the CD pin must remain asserted during frame transmission, or a CD lost error occurs. The negation of the CD pin terminates reception. If the CDS bit in the GSMR is zero, the CD pin must be sampled by the SCC before a CD lost is recognized. Otherwise, the negation of CD immediately causes the CD lost condition.

NOTE

If the CDS bit in GSMR is set, all CD transitions must occur while the receive clock is low.

7.10.11.2 ASYNCHRONOUS PROTOCOLS. The RTS pin is asserted when the SCC data is loaded into the transmit FIFO and a falling transmit clock occurs. The CD and CTS pins may be used to control reception and transmission in the same manner as the synchronous protocols. The first bit of data transmission in an asynchronous protocol is the start bit of the first character. In addition, the UART protocol has an option for CTS flow control as described in 7.10.16 UART Controller.

occur; a zero represents a masked bit position. Upon an address match, the address and the data following are written into the data buffers. When the addresses are not matched and the frame is error-free, the nonmatching address received counter (NMARC) is incremented.

NOTE

For 8-bit addresses, mask out (clear) the eight high-order bits in the HMASK register.

The eight low-order bits of HMASK and HADDRx should contain the address byte that immediately follows the opening flag. Example: To recognize a frame that begins $7E (Flag), $68, $AA, using 16-bit address recognition, HADDRx should contain $AA68, and HMASK should contain $FFFF (see Figure 7-51).

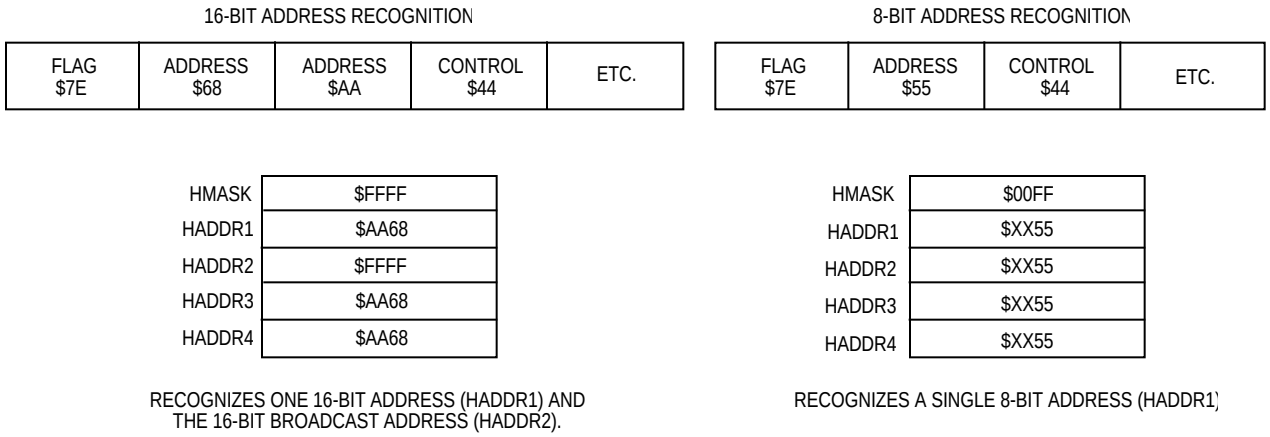


Figure 7-51. HDLC Address Recognition Example

RFTHR. The received frames threshold value is used to reduce the interrupt overhead that might otherwise occur when a series of short HDLC frames arrives, each causing an RXF interrupt. By setting the RFTHR value, the user can limit the frequency of RXF interrupts. The RXF interrupt will only occur when the RFTHR value is reached. RFCNT is a down-counter used to implement this feature.

NOTE

The user should provide enough empty Rx BDs to receive the number of frames specified in RFTHR.

7.10.17.5 HDLC PROGRAMMING MODEL. The CPU32+ core configures each SCC to operate in one of the protocols by the MODE bits in the GSMR. The HDLC controller uses the same data structure as in all other modes. This data structure supports multibuffer operation and address comparisons.

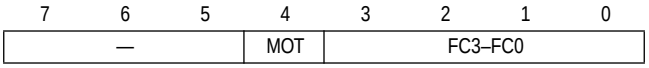
sponding channel. Furthermore, the user should not configure BD tables of two enabled SMCs to overlap, or erratic operation will occur.

NOTE

RBASE and TBASE should contain a value that is divisible by 8.

7.11.4.2 SMC FUNCTION CODE REGISTERS (RFCR, TFCR). There are four separate function code registers for the two SMC channels: two for receive data buffers (RFCRx) and two for transmit data buffers (TFCRx). The FC entry contains the value that the user would like to appear on the function code pins FC3–FC0 when the associated SDMA channel accesses memory. It also controls the byte-ordering convention to be used in the transfers.

Receive Function Code Register



Bits 7–5—Reserved

MOT—Motorola

This bit should be set by the user to achieve normal operation. MOT *must be set* if the data buffer is located in external memory and has a 16-bit wide memory port size.

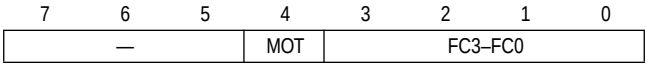
0 = DEC (and Intel) convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering—normal operation. It is also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. The user should write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

Transmit Function Code Register



Bits 7–5—Reserved

Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled or when the user is sure that no receive buffer is currently in use.

7.11.4.5 TRANSMITTER BUFFER DESCRIPTOR POINTER (TBPTR). The TBPTR for each SMC channel points to the next BD that the transmitter will transfer data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of the BD table is reached, the CP initializes this pointer to the value programmed in the TBASE entry. Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled or when the user is sure that no transmit buffer is currently in use (e.g., after a STOP TRANSMIT command is issued, or after a GRACEFUL STOP TRANSMIT command is issued, and the frame completes its transmission.)

7.11.4.6 OTHER GENERAL PARAMETERS. Additional parameters are listed in Table 7-12. These parameters do not need to be accessed by the user in normal operation, and are listed only because they may provide helpful information for experienced users and for debugging.

The Rx and Tx internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.

The Tx internal byte count is a down-count value that is initialized with the Tx BD data length and decremented with every byte read by the SDMA channels. The Rx internal byte count is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.

NOTE

To extract data from a partially full receive buffer, the CLOSE Rx BD command may be used.

The Rx internal state, Tx internal state, Rx temp, Tx temp, and reserved areas are for RISC use only.

7.11.5 Disabling the SMCs on the Fly

If an SMC is not needed for a period of time, it may be disabled and re-enabled later. In this case, a sequence of operations is followed.

This sequence ensures that any buffers in use will be properly closed and that new data will be transferred to/from a new buffer. Such a sequence is required if the parameters that must be changed are not allowed to be changed dynamically. If the register or bit description states that dynamic (on-the-fly) changes are allowed, the following sequences are not required, and the register or bit may be changed immediately. In all other cases, the sequence should be used. For instance, the baudrate generators allow on-the-fly changes.

NOTES

The modification of parameter RAM does not require a full disabling of the SMC. See the parameter RAM description for details on when parameter RAM values may be modified.

If the user desires to disable all SCCs, SMCs, and SPI, then the CR may be used to reset the entire CP with a single command.

7.11.5.1 SMC TRANSMITTER FULL SEQUENCE. For the SMC transmitter, the full disable and enable sequence is as follows:

1. STOP TRANSMIT command. This command is recommended if the SMC is currently in the process of transmitting data since it stops transmission in an orderly way. If the SMC is not transmitting (e.g., no Tx BDs are ready), then the STOP TRANSMIT command is not required. Furthermore, if the TBPTR will be overwritten by the user or the INIT TX PARAMETERS command will be executed, this command is not required.
2. Clear the TEN bit in the SMCMR. This disables the SMC transmitter and puts it in a reset state.
3. Make modifications. The user may make modifications to the SMC transmit parameters including the parameter RAM. If the user desires to switch protocols or restore the SMC transmit parameters to their initial state, the INIT TX PARAMETERS command may now be issued.
4. RESTART TRANSMIT command. This command is required if the INIT TX PARAMETERS command was not issued in step 3.
5. Set the TEN bit in the SMCMR. Transmission will now begin using the Tx BD pointed to by the TBPTR value as soon as the Tx BD R-bit is set.

7.11.5.2 SMC TRANSMITTER SHORTCUT SEQUENCE. A shorter sequence is possible if the user desires to reinitialize the transmit parameters to the state they had after reset. This sequence is as follows:

1. Clear the TEN bit in the SMCMR.
2. INIT TX PARAMETERS command. Any additional modifications may now be made.
3. Set the TEN bit in the SMCMR.

7.11.5.3 SMC RECEIVER FULL SEQUENCE. For the receiver, the full disable and enable sequence is as follows:

1. Clear the REN bit in the SMCMR. Reception will be aborted immediately. This disables the receiver of the SMC and puts it in a reset state.
2. Make modifications. The user may make modifications to the SMC receive parameters including the parameter RAM. If the user desires to switch protocols or restore the SMC receive parameters to their initial state, the INIT RX PARAMETERS command may now be issued.
3. CLOSE Rx BD command. This command is required if the INIT RX PARAMETERS command was not issued in step 2.
4. Set the REN bit in the SMCMR. Reception will now begin immediately using the Rx

When the SPI is working as a master, SPICLK is the clock output signal that shifts in the received data from the SPIMISO pin and shifts out the transmitted data to the SPIMOSI pin. Additionally, an SPI master device must provide a slave select signal output to enable the SPI slave devices. This may be implemented using one of the QUICC's general-purpose I/O pins. The $\overline{\text{SPISEL}}$ pin should not be asserted while the SPI is working as a master, or the SPI will indicate an error.

When the SPI is working as a slave, SPICLK is the clock input signal that shifts in the received data from the SPIMOSI pin and shifts out the transmitted data to the SPIMISO pin. The $\overline{\text{SPISEL}}$ pin provided by the QUICC is the enable input to the SPI slave.

When the SPI is working in a multi-master environment, the $\overline{\text{SPISEL}}$ pin is still an input and is used to detect an error condition when more then one master is operating.

SPICLK is a gated clock (i.e., the clock only toggles while data is being transferred). The user can select any of four combinations of SPICLK phase and polarity using two bits in the SPI mode register (SPMODE).

The SPI pins can also be configured as open-drain pins to support a multi-master configuration where the same SPI pin can be driven by the QUICC or an external SPI device.

7.12.4 SPI Transmit/Receive Process

The following paragraphs discuss SPI master, slave, and multi-master operation.

7.12.4.1 SPI MASTER MODE. When the SPI functions in master mode, the SPI transmits a message to the peripheral (SPI slave), which in turn sends back a simultaneous reply. When the QUICC works with more than one slave, it can use the general-purpose parallel I/O pins to selectively enable different slaves.

To begin the data exchange, the CPU32+ core writes the data to be transmitted into a data buffer, configures a Tx BD with its R-bit set and configures one or more Rx BDs. The CPU32+ core should then set the STR bit in the SPCOM to start transmission of data. The data will begin transmission once the SDMA channel has loaded the transmit FIFO with data.

The SPI controller then generates programmable clock pulses on the SPICLK pin for each character and shifts the data out on the SPIMOSI pin. At the same time, the SPI shifts receive data in from the SPIMISO pin. This receive data is written into a receive buffer using the next available Rx BD. The SPI will continue transmitting and receiving characters until the transmit buffer has been completely transmitted or an error has occurred ($\overline{\text{SPISEL}}$ pin unexpectedly asserted). The CP then clears the R and E bits in the Tx BD and Rx BD, and issues a maskable interrupt to the CPM interrupt controller.

When multiple Tx BDs are ready for transmission, the Tx BD L-bit determines whether the SPI continues to transmit without waiting for the STR bit to be set again. If the L-bit is cleared, the data from the next Tx BD will begin its transmission following the transmission of data from the first Tx BD. In most cases, the user should see no delay on the SPIMOSI pin between buffers. If the L-bit is set, transmission will cease after data from this Tx BD has

access as a DMA-type access. Example: FC3-FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

MOT—Motorola

This bit should be set by the user to achieve normal operation.

0 = DEC (and Intel) convention is used for byte ordering. Swapped operation. Also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering. Normal operation. Also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

Res—Reserved. Should be set to zero by the user.

7.13.8.7 TRANSMITTER BUFFER DESCRIPTOR POINTER (TBPTR). The transmitter buffer descriptor pointer (TBPTR) points to the next BD that the transmitter will transfer data from when it is in IDLE state, or to the current BD during frame transmission. After a reset or when the end of BD table is reached, the CP initializes this pointer to the value programmed in the TBASE entry. Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled, or when the user is sure that no transmit buffer is currently in use (i.e. after the STOP TRANSMIT command is issued.)

7.13.8.8 CENTRONICS TRANSMITTER PROGRAMMING MODEL. The host configures the PIP to operate as a Centronics controller by programming the PIP Configuration register (PIPC). Timing attributes (minimum data setup time and strobe pulse width) are set by programming the PIP Timing Parameters register (PTPR). The transmit errors are reported through the Tx BD.

7.13.8.9 CENTRONICS TRANSMITTER COMMAND SET. The Centronics transmitter uses SMC2 transmit commands (i.e, same opcodes and channel number)

7.13.8.9.1 STOP TRANSMIT Command. The channel STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the Centronics controller during frame transmission, transmission of that buffer is aborted and the TBPTR is not advanced to the next BD. No new BD is accessed, and no new buffers are transmitted for this channel. The transmitter will idle until the RESTART TRANSMIT command is given.

7.13.8.9.2 RESTART TRANSMIT Command. The RESTART TRANSMIT command is used to begin or resume transmission from the current Tx BD Pointer (TBPTR) in the channel's Tx BD table. When this command is received by the channel following by the STR bit in the PIP Configuration register (PIPC) being set, it will start processing the current BD. This command is expected by the Centronics controller after a STOP TRANSMIT command, after the disabling of the channel in its mode register, or after a transmitter error occurs.

This register may be read by the user to determine which interrupt requests are currently in progress (i.e., the interrupt handler started execution) for each CPM interrupt source. More than one bit in the CISR may be a one if higher priority CPM interrupts are allowed to interrupt lower priority level interrupts within the same CPM interrupt level. For example, the TIMER2 interrupt routine could interrupt the handling of the TIMER3 routine, using a special nesting technique described earlier. During this time, the user would see both the TIMER3 and the TIMER2 bits simultaneously set in the CISR.

NOTES

The SCC CISR bit positions are NOT affected by the relative priority between SCCs (as determined by the SCxP and SPS bits in the CICR).

If the error vector is taken, no bit in the CISR is set. All undefined bits in the CISR return zeros when read.

The user can control the extent to which CPM interrupts may interrupt other CPM interrupts by selectively clearing the CISR. A new interrupt will be processed if it has a higher priority than the higher priority interrupt having its CISR bit set. Thus, if an interrupt routine lowers the 3-bit mask in the CPU32+ core to the CPM level minus one and also clears its CISR bit at the beginning of the interrupt routine, a lower priority interrupt can interrupt the higher one, as long as the lower priority interrupt is of higher priority than any other CISR bits that are currently set.

7.15.6 Interrupt Handler Examples

The following examples illustrate proper interrupt handling of CPM interrupts. Nesting of interrupts within the CPM interrupt level is not shown in the following examples.

7.15.6.1 EXAMPLE 1—PC6 INTERRUPT HANDLER. In this example, the CPIC hardware clears the PC6 bit in the CIPR during the interrupt acknowledge cycle. This is an example of a handler for an interrupt source without multiple events.

1. Vector to interrupt handler.
2. Handle event associated with a change in the state of the port C6 pin.
3. Clear the PC6 bit in the CISR.
4. Execute the RTE instruction.

7.15.6.2 EXAMPLE 2—SCC1 INTERRUPT HANDLER. In this example, the CIPR bit SCC1 remains set as long as one or more unmasked event bits remain in the SCCE1 register. This is an example of a handler for an interrupt source with multiple events. Note that the bit in CIPR does not need to be cleared by the handler, but the bit in CISR does need to be cleared.

1. Vector to interrupt handler.
2. Immediately read the SCC1 event register (SCCE1) into a temporary location.

9.1.1.10 DOUBLE BUS FAULT. The QUICC double bus fault monitor may be used in the design. No additional hardware is required.

9.1.1.11 JTAG AND THREE-STATE. The QUICC provides a JTAG test access port, commonly known as JTAG. This interface uses five pins: TMS, TDI, TDO, TCK, and $\overline{\text{TRST}}$. TMS and TDI are left unconnected because they have internal pullups. The JTAG port is disabled in this application; however, the capability could be easily added.

When the QUICC is in master mode, it provides a pin ($\overline{\text{TRIS}}$) that allows all outputs on the device to be three-stated. This pin is simply pulled high in this application.

9.1.1.12 QUICC SERIAL PORTS. The functions on QUICC parallel I/O ports A, B, and C may be used as desired in this application, and have no bearing on the design as shown. However, any unused parallel I/O pins should be configured as outputs so they are not left floating.

9.1.2 Memory Interfaces

In this application, a number of memory arrays have been developed for EPROM, flash EPROM, SRAM, EEPROM, and DRAM. Each memory interface can be attached to the system bus as desired.

One issue not discussed is the decision of whether external buffers are needed on the system bus. This issue depends on the number of memory arrays used in the design and the layout (i.e., capacitance) of the system bus. This issue is left to the user for his particular design.

Another issue left to the user is the number of wait states used with each memory system. This depends on the memory speed, whether external buffers are used, and the loading on the system bus pins. (The QUICC provides capacitance de-rating figures to calculate the effect of additional or less capacitance on the AC Timing Specifications.)

9.1.2.1 QUICC MEMORY INTERFACE PINS. In this design, a number of QUICC pins are made available to the memory arrays (see Figure 9-1). Eight chip select or $\overline{\text{RAS}}$ pins are available in the system. In this design, $\overline{\text{CS0}}$ is used for any of the EPROM arrays since this is the global (boot) chip select. $\overline{\text{RAS1}}$ is used for the DRAM arrays because of its double-drive capability. $\overline{\text{CS2}}$/$\overline{\text{RAS2}}$ is not used in the design and is available for other purposes, such as a second DRAM bank. $\overline{\text{CS3}}$ is for SRAM; $\overline{\text{CS4}}$ is for EEPROM. $\overline{\text{CS5}}$, $\overline{\text{CS6}}$, and $\overline{\text{CS7}}$ are unused.

Parity may be supported for both SRAM and DRAM arrays using the 4-byte parity lines PRTY3–PRTY0. In this design, it is shown with only a DRAM. The QUICC is configured in software to generate a bus error when a parity error occurs.

The QUICC provides the address multiplexing for the DRAM arrays internally, which is configured in software. Therefore, the address multiplex pin is not needed, and it can be used as its other function—an output enable ($\overline{\text{OE}}$) pin. The DRAM arrays require the four $\overline{\text{CAS3}}$–$\overline{\text{CAS0}}$ pins provided by the QUICC. The QUICC also provides four write enable ($\overline{\text{WE}}$) pins to select the correct byte during write operations.

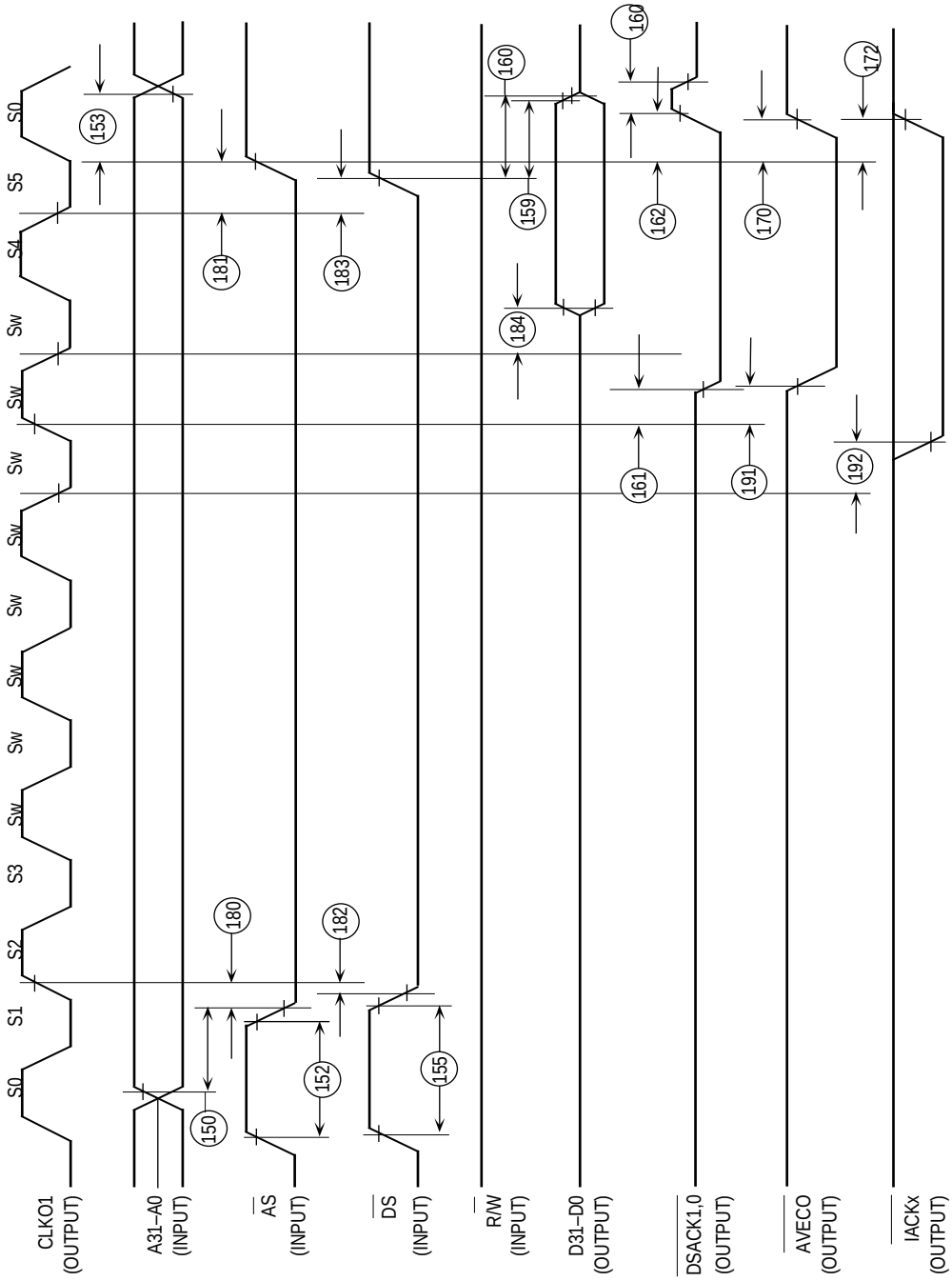


Figure 10-32. External MC68030/MC68360 Synchronous IACK Cycle Timing Diagram

High-Speed Channels		Low-Speed Channels		Comments/Restrictions
Number	Speed	Number	Speed	
1 HDLC	8 Mbps	—	—	
2 HDLC	4 Mbps	—	—	
3 HDLC	2.6 Mbps	—	—	
4 HDLC	2.05 Mbps	—	—	
1 ENET	10 Mbps	3 HDLC	2.05 Mbps	Minor restrictions on HDLC buffer length
1 ENET	10 Mbps	2 HDLC	2.05 Mbps	
2 ENET	10 Mbps	1 HDLC	1 Mbps	
4 UART	625 kbps	—	—	Async SCC
4 UART-S	625 kbps	—	—	Sync UART SCC
4 BISYNC	460 kbps	—	—	
1 TRAN	8 Mbps	—	—	
2 TRAN	4 Mbps	—	—	
3 TRAN	2.6 Mbps	—	—	
4 TRAN	2.05 Mbps	—	—	
2 TRAN	1.56 Mbps	—	—	SMC Channels
2 UART	100 kbps	—	—	SMC Channels

NOTES:

1. These numbers are estimates generated prior to silicon. Additional work will be performed to improve the accuracy of the SCC performance numbers and will be reported in later revisions of this manual.
2. All performance calculations assume a 25-MHz system clock and sync clock for the SCCs. The results scale linearly with different system clock speeds. A change in the sync clock will not affect performance as long as the required 1:2.25 or 1:2.5 ratio is maintained.
3. User should consult with receive and transmit clock timing of the SIA to ensure clock pulse width high and width low are satisfied for high speed serial communications.
4. In all cases, the numbers assume data is stored in external system RAM.
5. The performance is not expected to degrade based on the number of wait states in the system, as long as the system RAM has between 0 and 9 wait states.
6. The performance table is also applicable when the time-slot assigner on the QUICC is used.
7. When the performance of a high-speed channel together with a low-speed channel was measured, the high-speed channel was always SCC1.
8. Performance in *HDLC bus* mode is the same as HDLC performance, for both full duplex and half duplex operation. (Half duplex is the normal configuration of HDLC bus mode, thus HDLC half duplex performance numbers would normally be used.)
9. All results assume that the other RISC features are not operating—i.e., the RISC timer tables, the IDMA auto buffer and buffer chaining modes, the parallel interface port, and the other serial channels. If these features are operating, performance may be slightly reduced. The DRAM refresh controller has no effect on the performance.
10. Except for Ethernet, all table results assume continuous full-duplex operation. Results for half-duplex operation are roughly 2x better.
11. The SMC performance results are without the SCCs operating; otherwise, SMC performance may be reduced. Although the exact SMC performance that can be obtained is determined by many RISC utilization factors and is therefore difficult to estimate, it is expected that 9.6 kbps on both SMC UARTs could be simultaneously supported in most situations.

APPENDIX B
DEVELOPMENT TOOLS AND SUPPORT

Several software development packages are offered as a set of independent modules that provide the following features:

- QUICC Chip Evaluation
- By running the modules on the development board described in B.4 M68360QUADS Development System, it is possible to examine and evaluate the QUICC. Symbolic, user-friendly menus allow control of all parts of the QUICC.
- QUICC Simple Drivers
- Written in C, the source code of the QUICC drivers is available on the Motorola Freeware Bulletin Board (512-891-3733, 8 data, no parity, 1 stop) or through Anonymous FTP to freeware.aus.sps.mot.com under /pub/mcu360. These drivers were developed to support the needs of the general user. Although they are based on the regular QUICC drivers, they provide call-oriented interfaces and self-contained QUICC initialization routines and interrupt handling for a given protocol.
- QUICC Chip Drivers
- Written in C, the source code of the QUICC drivers is available on electronic media. These drivers were developed to support the needs of the protocol implementations.
- Protocol Implementations
- Modules implementing common ISO/OSI layer 2 and 3 protocols are available under license. These are in the form of source code written in C.
- Portability
- All of the higher layer software modules may be ported from source to different QUICC implementations and combined with user-developed code. Software interface documentation is available for all provided modules.

B.1 MOTOROLA SOFTWARE MODULES

Chip driver routines written in C illustrate initialization of the QUICC, interrupt handling, and the management of data transmission and reception on all channels.

In addition to the chip drivers, protocol modules are provided. Layer 2 modules include LAPB and LAPD. The layer 3 module is the X.25 packet layer protocol.

Since the modules require some minimal operating system services, the EDX operating system kernel is provided. EDX is the kernel implemented on the QUADS board (see B.4 M68360QUADS Development System). Use of EDX with the protocol modules is not required as long as some other operating system support is provided by the user.