The expected results are as follows:

- The TX buffer descriptor (Ep0) CONTROL/STATUS field must contain 0x3800.
- The TX buffer descriptor (Ep0) DATA LENGTH field must contain 0x0003.
- The TX buffer descriptor (Ep1) CONTROL/STATUS field must contain 0x3c80.
- The TX buffer descriptor (Ep1) DATA LENGTH field must contain 0x0003.
- The RX buffer descriptor (Ep0) CONTROL/STATUS field must contain 0x3c00.
- The RX buffer descriptor (Ep0) DATA LENGTH field must contain 0x0005.
- The RX buffer descriptor (Ep0) DATA BUFFER field must contain 0xabcd122b, 0x42xxxxxx.

16.11 THE SERIAL MANAGEMENT CONTROLLERS

The serial management controllers (SMCs) consist of two full-duplex ports that can be independently configured to support any one of three protocols or modes—UART, Transparent, or general-circuit interface (GCI). Simple UART operation is used to provide a debug/monitor port in an application, which allows a serial communication controller (SCCx) to be free for other purposes. The serial management controller clock can be derived from one of the four internal baud rate generators or from a 16× external clock pin.

The functionality and performance provided by the SMCx in UART mode is generally less than that provided by an SCCx in UART mode. The SMCx implementation does not support such features as special character recognition and detection.

In totally Transparent mode, a serial management controller can use the TDM channel to connect to a T1 line or directly to the SMC's set of pins. However, SMC2 does not have its own set of dedicated pins, so the time-slot assigner pins are its only option. The receive and transmit clocks are derived from the time-division multiplex (TDM) channel, the internal baud rate generators, or from an external 1× clock. The Transparent protocol allows the transmitter and receiver to use the external synchronization pin.

Each serial management controller supports the circuit interface and monitor channels of the GCI bus. In which case, the serial management controller is connected to the TDM channel in the serial interface. For testing purposes, the serial management controllers support loopback and echo modes. The SMCx receiver and transmitter are double-buffered, which provides an effective FIFO size (latency) of two characters. Refer to **Section 16.7 The Serial Interface with Time-Slot Assigner** for details about configuring the GCI interfaces.

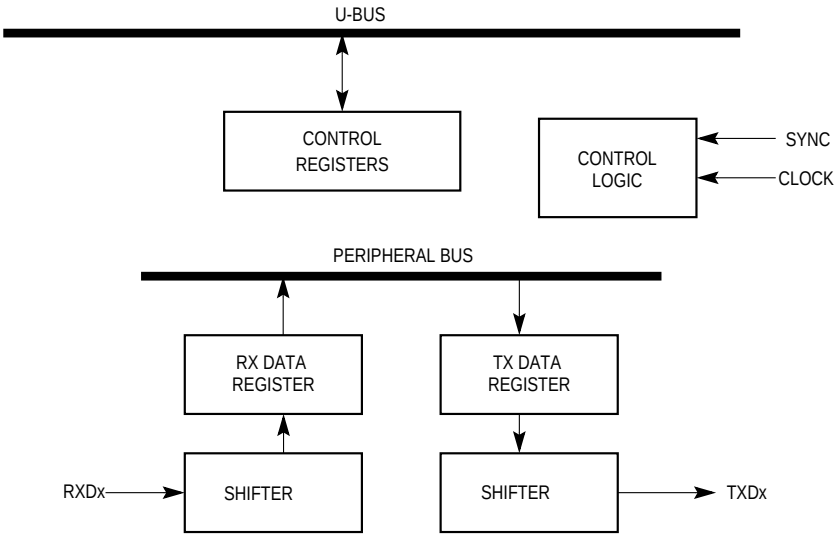


Figure 16-113. Serial Management Controller Block Diagram

The receive data source for the two serial management controller channels have different pin options for each channel. SMC1 can either use the L1RXDA pin of the serial interface or the SMRXD1 pin if it is connected to the NMSI. SMC2 can also use the L1RXDA pin of the serial interface, but if you use the SMRXD2 pin the serial interface time-slot assigner function is not available. Likewise, the transmit data source for SMC1 can be the L1TXDA pin if a serial management controller is connected to the TDM or the SMTXD1 pin if it is connected to the NMSI. SMC2 transmit data source can also be L1TXDA pin if the serial management controller is connected to the TDM, but if you use the SMTXD2 pin, the serial interface time-slot assigner function is not available.

If the serial management controllers are connected to the TDM, the SMCx receive and transmit clocks can be independent from each other, as defined in the CRTA bit of the SIMODE register. Refer to **Section 16.7.5.2 Serial Interface Mode Register** for more information. However, if a serial management controller is connected to the NMSI, the SMCx receive and transmit clocks must be connected to a single clock source called SMCLK, which is an internal signal name for a clock that is generated from the bank of clocks. SMCLK originates from an external pin or one of the two internal baud rate generators. Refer to **Section 16.7.8 Nonmultiplexed Serial Interface Configuration** for more details.

If the serial management controllers are connected to the TDM, it gets its synchronization pulse from the time-slot assigner. Otherwise, if a serial management controller is connected to the NMSI and the Transparent protocol is selected, the serial management controller can use the $\overline{\text{SMSYNx}}$ pin as a synchronization pin to determine when it must start transmitting and receiving. The $\overline{\text{SMSYNx}}$ pin is not used, however, when a serial communication controller is in UART mode.

16.11.1 Features

The following is a list of the serial management controllers' main features:

- Each serial management controller can implement the UART protocol on its own pins
- Each serial management controller can implement a totally Transparent protocol on a multiplexed or nonmultiplexed line. If SMC2 uses a nonmultiplexed line, the serial interface time-slot assigner is not available.
- Each SMCx channel fully supports the circuit interface and monitor channels of the GCI (IOM-2) in ISDN applications
- Two serial management controllers support the two sets of circuit interface and monitor channels in the SCIT channels 0 and 1
- Full-duplex operation
- Local loopback and echo capability for testing

16.11.2 General SMCx Mode Register

The operating mode of each serial management controller port is defined by the 16-bit, memory-mapped, read/write general SMCx mode register (SMCMR). Refer to each specific SMCx protocol section for information about this register and Table 16-2 (page 16-11) for specific commands.

16.11.3 SMCx Buffer Descriptor Operation

When the serial management controllers are configured to operate in GCI mode, their memory structure is predefined as one half-word long for transmit and one half-word long for receive. For more information on these half-word structures, refer to **Section 16.11.8 The SMCx in GCI Mode**.

In UART and Transparent modes, the serial management controllers have a memory structure that is similar to the serial communication controllers. Each buffer is referenced by a buffer descriptor and organized in a buffer descriptor ring located in the dual-port RAM, as illustrated in Figure 16-114.

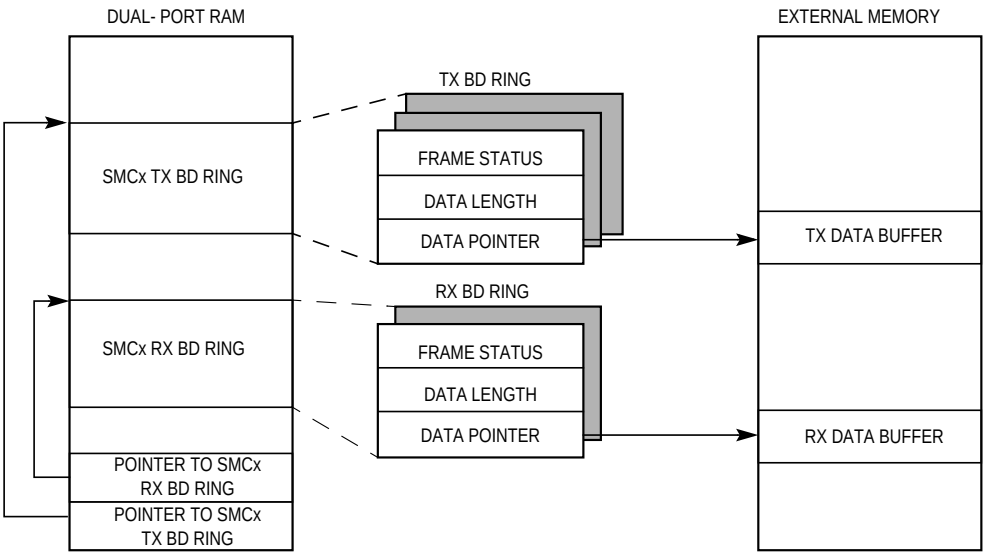


Figure 16-114. SMCx Memory Format

The buffer descriptor ring allows you to define buffers for transmission and reception and each ring forms a circular queue. Using the buffer descriptors, the communication processor module confirms reception and transmission so that the microprocessor knows the buffers have been serviced. The data buffers can reside in external or internal memory and the data buffers reside in the parameter area of an SCCx or SMCx if that channel is not enabled.

16.11.4 SMC General Parameter RAM Memory Map

Each SMCx parameter RAM area begins at the same offset from each base. The protocol-specific portions of the SMCx parameter RAM are discussed in each mode. The SMCx general parameter RAM shared by the UART and transparent protocols is described in Table 16-36. The GCI protocol has its own parameter RAM.

You must initialize certain parameter RAM values before a serial management controller is enabled. Other values are initialized or written by the communication processor module. Once initialized, most parameter RAM values do not need to be accessed in your software since most of the activity is centered around the transmit and receive buffer descriptors and not the parameter RAM. However, if you access the parameter RAM, note the following restrictions.

- The parameter RAM values that pertain to the SMCx transmitter can only be written when the TEN bit in the SMCMR is zero or after the **STOP TRANSMIT** command is issued, but before the **RESTART TRANSMIT** command is issued.
- The parameter RAM values that pertain to the SMCx receiver can only be written when:
  - ❏ The REN bit in the SMCMR is zero.
  - ❏ The receiver is previously enabled after the **ENTER HUNT MODE** command is issued.
  - ❏ The **CLOSE RX BD** command is issued before the REN bit is set.

Table 16-36. SMCx (UART and Transparent) Parameter RAM Memory Map

ADDRESS	NAME	WIDTH	DESCRIPTION
SMCx Base + 00	**RBASE**	Half-Word	RX Buffer Descriptor Base Address
SMCx Base + 02	**TBASE**	Half-Word	TX Buffer Descriptor Base Address
SMCx Base + 04	**RFCR**	Byte	RX Function Code
SMCx Base + 05	**TFCR**	Byte	TX Function Code
SMCx Base + 06	**MRBLR**	Half-Word	Maximum Receive Buffer Length
SMCx Base + 08	RSTATE	Word	RX Internal State
SMCx Base + 0C	RPTR	Word	RX Internal Data Pointer
SMCx Base + 10	RBPTR	Half-Word	RX Buffer Descriptor Pointer
SMCx Base + 12	RCNT	Half-Word	RX Internal Byte Count
SMCx Base + 14	RTMP	Word	RX Temp
SMCx Base + 18	TSTATE	Word	TX Internal State
SMCx Base + 1C	TPTR	Word	TX Internal Data Pointer
SMCx Base + 20	TBPTR	Half-Word	TX Buffer Descriptor Pointer
SMCx Base + 22	TCNT	Half-Word	TX Internal Byte Count
SMCx Base + 24	TTMP	Word	TX Temp
SMCx Base + 28			First Word of Protocol-Specific Area
SMCx Base + 36			Last Word of Protocol-Specific Area

NOTE: You are only responsible for initializing the items in bold.
SMCx Base = (IMMR & 0xFFFF0000) + 0x3E80 (SMC1) and 0x3F80 (SMC2).

- RBASE and TBASE—These entries are used by the dual-port RAM starts the SMCx receive and transmit buffer descriptors. They provide a great deal of flexibility for partitioning the buffer descriptors for a serial management controller. By selecting RBASE and TBASE entries for a serial management controller and by setting the W bit in the last buffer descriptor in each buffer descriptor list, you can select the number of buffer descriptors to allocate for the transmit and receive side of the serial management controller. However, you must initialize these entries before enabling the corresponding channel. Furthermore, you must not configure buffer descriptor tables of two enabled serial management controllers to overlap or erratic operation will occur.

Note: RBASE and TBASE must contain a value that is divisible by eight.

- RFCR and TFCR—There are separate function code registers for the two SMCx channels. One for receive data buffers (RFCR) and one for transmit data buffers (TFCR). The function code entry contains the value that you want to appear on the AT pins when the associated SDMA channel accesses memory. It also controls the byte-ordering convention that is used in the transfers.

RFCR

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED			BO		AT1-AT3		
RESET	0			0		0		
R/W	R/W			R/W		R/W		
ADDR	SMC1 AND SMC2 BASE + 0x04							

Bits 0–2—Reserved

These bits are reserved and must be set to 0.

BO—Byte Ordering

Set these bits to select the required byte ordering of the data buffer. If this field is modified on-the-fly, it takes effect at the beginning of the next frame or the next buffer descriptor.

00 = The DEC/Intel convention is used for byte ordering (swapped operation) and is also called little-endian byte ordering. The transmission order of bytes within a buffer word is reversed in comparison to the Motorola mode. This mode is supported only for 32-bit port size memory.

01 = PowerPC little-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the least-significant byte of the buffer double-word contains data to be transmitted earlier than the most-significant byte of the same buffer double-word.

1X = Motorola byte ordering (normal operation) is also called big-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the most-significant byte of the buffer word contains data to be transmitted earlier than the least-significant byte of the same buffer word.

AT—Address Type 1–3

These bits contain the function code value used during the SDMA channel memory access. The AT0 pin is driven with a 1 to identify this SDMA channel access as a DMA type.

TFCR

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED			BO		AT1–AT3		
RESET	0			0		0		
R/W	R/W			R/W		R/W		
ADDR	SMC1 AND SMC2 BASE + 0x05							

Bits 0–2—Reserved

These bits are reserved and must be set to 0.

BO—Byte Ordering

Set these bits to select the required byte ordering of the data buffer. If this field is modified on-the-fly, it takes effect at the beginning of the next frame or the next buffer descriptor.

00 = The DEC/Intel convention is used for byte ordering (swapped operation) and is also called little-endian byte ordering. The transmission order of bytes within a buffer word is reversed in comparison to the Motorola mode. This mode is supported only for 32-bit port size memory.

01 = PowerPC little-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the least-significant byte of the buffer double-word contains data to be transmitted earlier than the most-significant byte of the same buffer double-word.

1X = Motorola byte ordering (normal operation) is also called big-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the most-significant byte of the buffer word contains data to be transmitted earlier than the least-significant byte of the same buffer word.

AT—Address Type 1–3

These bits contain the function code value used during this SDMA channel memory access. The AT0 pin is driven with a 1 to identify this SDMA channel access as a DMA type.

- MRBLR—Each serial management controller has one maximum receive buffer length register that defines the length of the receive buffer. The MRBLR defines the maximum number of bytes that the MPC823 writes to a receive buffer on a serial management controller before it moves on to the next buffer. The MPC823 can write fewer bytes to the buffer than MRBLR if a condition, such as an error or end-of-frame occurs, but it never writes more bytes than the MRBLR value. It follows then, that the buffers you supply must always be at least as long as the MRBLR. The transmit buffers for a serial management controller are not affected in any way by the value programmed into the MRBLR. Each transmit buffer can have a different length. You can choose the number of bytes to be transmitted by programming the DATA LENGTH field in the TX buffer descriptor.

Note: The MRBLR is not intended to be dynamically changed while a serial management controller 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 the receive buffer length can be successfully achieved. This occurs when the communication processor module transfers control to the next RX buffer descriptor in the table. Thus, a change to MRBLR does not have an immediate effect. To guarantee that the change occurs on a particular RX buffer descriptor, you must only change the MRBLR while the SMCx receiver is disabled. The value of MRBLR must be greater than zero and it must be even if the character length of the data is greater than 8 bits.

- RBPTR—The receiver buffer descriptor pointer for each SMCx channel points to the next buffer descriptor the receiver transfers data to when it is idle or to the current buffer descriptor during frame processing. After a reset or when the end of the buffer descriptor table is reached, the communication processor module initializes this pointer to the value programmed in the RBASE entry. Although you will not usually need to write the RBPTR in most applications, you can modify it when the receiver is disabled or when you are sure no receive buffer is currently being used.
- TBPTR—The transmitter buffer descriptor pointer for each SMCx channel points to the next buffer descriptor the transmitter transfers data from when it is idle or to the current buffer descriptor during frame transmission. After a reset or when the end of the buffer descriptor table is reached, the communication processor module initializes this pointer to the value programmed in the TBASE register. Although you will not usually need to write TBPTR in most applications, you can modify it when the receiver is disabled or when you are sure no receive buffer is currently being used.

- Other General Parameters—You do not need to access these parameters during normal operation. They are only listed because they provide helpful debugging information. Additional parameters are listed in Table 16-33 (page 16-354). The RX and TX internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed. TCNT is a down-count value that is initialized with the DATA LENGTH field of the TX buffer descriptor and decremented with every byte read by the SDMA channels. RCNT is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels. RSTATE, TSTATE, RTEMP, TTEMP, and the reserved areas are only used by the RISC microcontroller.

Note: To extract data from a partially full receive buffer, use the **CLOSE RX BD** command.

16.11.5 Disabling the SMCs On-the-Fly

If you do not need a serial management controller, it can be disabled and reenabled later. In which case, you must follow a particular sequence of steps to ensure that the buffers are properly closed and that new data is transferred to or from a new buffer. This sequence is required if the parameters you want to change are not dynamic. If the register or bit description states that dynamic or on-the-fly changes are allowed, you do not have to follow the sequence and the register or bit can can be changed immediately.

Note: A serial management controller does not have to be fully disabled for you to modify the parameter RAM. Refer to the parameter RAM description for details about modifying the parameter RAM values. If you want to disable the SCCs, USB, SMCs, SPI, and I^2C, use the CPM command register (described in **Section 16.2.6.1 CPM Command Register**) to reset the communication processor module with a single command.

16.11.5.1 DISABLING THE ENTIRE SMCx TRANSMITTER. Follow these steps to fully enable or disable the SMCx transmitter:

1. Issue the **STOP TRANSMIT** command. This command is recommended when a serial management controller is transmitting data since it stops transmission smoothly. This command is not required if a serial management controller is not transmitting, if you overwrite the TBPTR or if you execute the **INIT TX PARAMETERS** command.
2. Clear the TEN bit in the SMCMR. This disables the SMCx transmitter and puts it in a reset state.
3. Modify the SMCx transmit parameters, including the parameter RAM. If you prefer to switch protocols or restore the SMCx transmit parameters to their initial state, issue the **INIT TX PARAMETERS** command.
4. Issue the **RESTART TRANSMIT** command. You must do this if you did not issue the **INIT TX PARAMETERS** command in step 3.
5. Set the TEN bit in the SMCMR. When the R bit is set in the TX buffer descriptor, transmission begins using the TX buffer descriptor indicated by the TBPTR value.

16.11.5.2 DISABLING PART OF THE SMCx TRANSMITTER. Follow this shorter sequence if you prefer to reinitialize the transmit parameters to the state they were in after reset.

1. Clear the TEN bit in the SMCMR.
2. Issue the **INIT TX PARAMETERS** command and make any additional modifications.
3. Set the TEN bit in the SMCMR.

16.11.5.3 DISABLING THE ENTIRE SMCx RECEIVER. Follow these steps to fully enable or disable the receiver:

1. Clear the REN bit in the SMCMR. Reception is aborted immediately, which disables the SMCx receiver and puts it in a reset state.
2. Modify the SMCx receive parameters, including the parameter RAM. If you prefer to switch protocols or restore the SMCx receive parameters to their initial state, issue the **INIT RX PARAMETERS** command.
3. Issue the **CLOSE RX BD** command. You must do this if you did not issue the **INIT RX PARAMETERS** command in step 2.
4. Set the REN bit in the SMCMR. When the E bit is set in the RX buffer descriptor, reception begins immediately using the RX buffer descriptor indicated by the RBPTR.

16.11.5.4 DISABLING PART OF THE SMCx RECEIVER. Follow this shorter sequence to reinitialize the receive parameters to the state they were in after reset.

1. Clear the REN bit in the SMCMR.
2. **INIT RX PARAMETERS** command and make any additional modifications.
3. Set the REN bit in the SMCMR.

16.11.5.5 SWITCHING PROTOCOLS. To switch a protocol that is being executed by a serial management controller without resetting the board or affecting any other serial management controller:

1. Clear the TEN and REN bits in the SMCMR.
2. Issue the **INIT TX AND RX PARAMS** command to initialize the transmit and receive parameters. Make any additional modifications in the SMCMR.
3. Set the TEN and REN bits in the SMCMR. The serial management controller is now enabled with the new protocol.

Tip: You can save power by clearing the TEN and REN bits of a serial management controller.

16.11.6 The SMCx in UART Mode

Compared to an SCCx in UART mode, the serial management controllers are designed to support simple debug/monitor ports instead of full-featured UART controllers. The following is a list of the features that the SMCx in UART mode does not support.

- $\overline{\text{RTSx}}$, $\overline{\text{CTSx}}$, and $\overline{\text{CDx}}$ pins
- Receive and transmit sections clocked at different rates
- Fractional stop bits
- Built-in multidrop modes
- Freeze mode for implementing flow control
- Isochronous operation (1× clock)
- Interrupts on special control character reception
- Ability to transmit data on demand using the transmit on demand register

A serial management controller in UART mode has one feature that an SCCx in UART mode does not. Data length on an SMCx can be a maximum of 14 bits, whereas, a serial communication controller only allows 8 bits. A serial management controller in UART mode is also referred to as a SMCx UART controller.

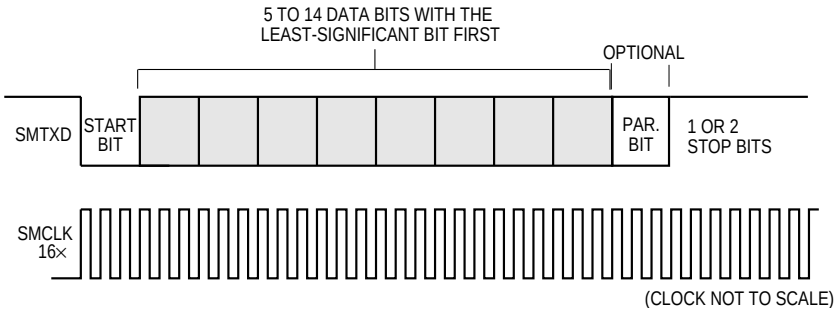


Figure 16-115. SMCx UART Frame Format

16.11.6.1 FEATURES. The following list summarizes the main features of the SMCx in UART mode:

- Flexible message-oriented data structure
- Programmable data length (5–14 bits)
- Programmable 1 or 2 stop bits
- Even/odd/no parity generation and checking
- Frame error, break, and idle detection
- Transmit preamble and break sequences
- Received break character length indication
- Continuous receive and transmit modes

16.11.6.2 SMCx UART CHANNEL TRANSMISSION PROCESS. The UART transmitter is designed to work with almost no intervention from the core. When the core enables the SMCx transmitter, it starts transmitting idles. The SMCx UART controller immediately polls the first buffer descriptor in the transmit channel buffer descriptor ring and once every character time after that, depending on the character length. When there is a message to transmit, the SMCx UART controller fetches the data from memory and starts transmitting the message.

When the buffer descriptor data is completely written to the transmit FIFO, the SMCx UART controller writes the message status bits into the buffer descriptor and clears the R bit. An interrupt is issued if the I bit in the buffer descriptor is set. If the next TX buffer descriptor is ready, the data from its data buffer is appended to the previous data and transmitted out on the transmit pin, without any gaps between the buffers. If the next TX buffer descriptor is not ready, the SMCx UART controller starts transmitting idles and waits for the next TX buffer descriptor to be ready.

By appropriately setting the I bit in each buffer descriptor, interrupts can be generated after the transmission of each buffer, a specific buffer, or each block. The SMCx UART controller then proceeds to the next buffer descriptor in the table. If the CM bit is set in the TX buffer descriptor, the R bit is not cleared, allowing the associated data buffer to be automatically retransmitted next time the communication processor module accesses this data buffer. For instance, if a single TX buffer descriptor is initialized with the CM and W bits set, the data buffer is continuously transmitted until you clear the R bit of the buffer descriptor.

16.11.6.3 SMCx UART CHANNEL RECEPTION PROCESS. When the core enables the SMCx receiver in UART mode, it enters hunt mode and waits for the first character to arrive. Once the first character arrives, the communication processor module checks the first receive buffer descriptor to see if it is empty and then starts storing characters in the associated data buffer.

When the data buffer is filled or the MAX_IDL timer expires (if it is enabled) the SMCx UART controller clears the E bit in the buffer descriptor and generates an interrupt if the I bit in the buffer descriptor is set. If the incoming data exceeds the length of the data buffer, the SMCx UART controller fetches the next buffer descriptor in the table and, if it is empty, continues transferring data to the associated data buffer. If the CM bit is set in the receive buffer descriptor, the E bit is not cleared, which allows the associated data buffer to be automatically overwritten next time the communication processor module accesses this data buffer.

16.11.6.4 SMCx UART PARAMETER RAM MEMORY MAP. When a serial management controller is configured to operate in UART mode, the SMCx UART controller overlays the structure used in Table 16-36 with the parameters described in Table 16-37.

Table 16-37. SMCx UART Parameter RAM Memory Map

ADDRESS	NAME	WIDTH	DESCRIPTION
SMCx Base + 28	**MAX_IDL**	Half-word	Maximum Idle Characters
SMCx Base + 2A	IDLC	Half-word	Temporary Idle Counter
SMCx Base + 2C	BRKLN	Half-word	Last Received Break Length
SMCx Base + 2E	**BRKEC**	Half-word	Receive Break Condition Counter
SMCx Base +30	**BRKCR**	Half-word	Break Count Register (Transmit)
SMCx Base +32	R_MASK	Half-word	Temporary Bit Mask

NOTE: You are only responsible for initializing the items in bold.
SMCx Base = (IMMR & 0xFFFF0000) + 0x3E80 (SMC1) and 0x3F80 (SMC2).

- MAX_IDL—Once a character of data is received on the line, the SMCx UART controller starts counting any idle characters received. If a MAX_IDL number of idle characters is received before the next data character, an idle timeout occurs and the buffer closes. This, in turn, produces an interrupt request to the core to receive the data from the buffer. MAX_IDL provides a convenient way to demarcate frames in SMCx UART mode. But if you do not want to use MAX_IDL, you must program MAX_IDL to 0x0000 and the buffer will never close, regardless of the number of idle characters received. The number of bits in an idle character is calculated as follows—1 + character data length (5 to 14) + 1 (if parity bit is used) + number of stop bits (1 or 2). For example, for a character data length of 8, no parity, and 1 stop bit, the idle character length is 10 bits.
- IDLC—This value is used by the RISC microcontroller to store the current idle counter value in the MAX_IDL timeout process. IDLC is a down-counter that you do not need to initialize or access.
- BRKLN—This value is used to store the length of the last break character received and is the bit length of that character. For example, if the receive pin is low for 257 bit times, BRKLN shows the value 0x0101 and its accuracy comes within one character unit of bits. For 8 data bits, no parity, 1 stop bit, and 1 start bit, BRKLN is accurate within 10 bits.
- BRKEC—This counter counts the number of break conditions that occur on the line. One break condition can last for hundreds of bit times, yet this counter is only incremented once during that period.
- BRKCR—This value indicates when the SMCx UART controller sends a break character sequence after a **STOP TRANSMIT** command is issued. The number of break characters sent by the SMCx UART controller is determined by the value in BRKCR. For 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character is 10 bits long and consists of all zeros.
- R_MASK—This value is a temporary bit mask used internally by the SMCx UART controller.

16.11.6.5 PROGRAMMING THE SMCx UART CONTROLLER. The SMCx UART controller's data structure supports multibuffer operation and allows you to transmit break and preamble sequences. Overrun, parity, and framing errors are reported via the buffer descriptors. In its simplest form, the SMCx UART controller functions in a character-oriented environment. Each character is transmitted with the stop bits and parity that you configure. Characters are received into separate 1-byte buffers. A maskable interrupt can be generated when each buffer is filled.

Many applications may want to take advantage of the message-oriented capabilities that the SMCx UART controller supports through linked buffers for reception or transmission. You can handle data in a message-oriented environment and work on entire messages rather than on a character-by-character basis. A message can span several linked buffers and each one can be transmitted and received as a linked list of buffers without any intervention from the core, which makes it easy to program and saves processor overhead. In a message-oriented environment, the idle sequence is used as the message delimiter. The transmitter can generate an idle sequence before starting a new message and the receiver can close a buffer when an idle sequence is found.

16.11.6.6 SMCx UART COMMANDS. You can program the CPM command register (CPCR) with the following commands to transmit data.

- **STOP TRANSMIT**—This command disables the transmission of characters on the transmit channel. If the SMCx UART controller receives this command while transmitting a message, it stops transmitting. The SMCx UART controller finishes transmitting any data that has already been transferred to its FIFO and shift register and then stops transmitting data. The TBPTR is not advanced when this command is issued. The SMCx UART controller transmits a programmable number of break sequences and then transmits idles. The number of break sequences, which can be zero, must be written to the BRKCR entry before this command is issued to the SMCx UART controller.
- **RESTART TRANSMIT**—This command enables characters to be transmitted on the transmit channel. The SMCx UART controller expects it after disabling the channel in its SMCMR and after issuing the **STOP TRANSMIT** command. The SMCx UART controller resumes transmission from the current TBPTR in the channel's transmit buffer descriptor table.
- **INIT TX PARAMETERS**—This command initializes all the transmit parameters in this serial channel's parameter RAM to their reset state and must only be issued when the transmitter is disabled. The **INIT TX AND RX PARAMS** command can also be used to reset the transmit and receive parameters.

You can program the CPCR with the following commands to receive data.

- **ENTER HUNT MODE**—This command cannot be used for an SMCx UART channel. Issue the **CLOSE RX BD** command instead.
- **CLOSE RX BD**—This command is used to force a serial management controller to close the current receive buffer descriptor if it is currently being used and to use the next buffer descriptor in the list for any subsequently received data. If a serial management controller is not in the process of receiving data, no action is taken by this command.
- **INIT RX PARAMETERS**—This command initializes all the receive parameters in this serial channel parameter RAM to their reset state. This command must only be issued when the receiver is disabled. The **INIT TX AND RX PARAMS** command can also be used to reset the receive and transmit parameters.

16.11.6.7 SENDING A BREAK. A break is an all-zeros character without stop bits and it is sent by issuing the **STOP TRANSMIT** command. The SMCx UART controller finishes transmitting any outstanding data and then sends a character with consecutive zeros. The number of zero bits in this character is the sum of the character length, plus the number of start, parity, and stop bits. The SMCx UART controller transmits a programmable number of break characters according to the BRKCR entry and then reverts to idle or sends data if the **RESTART TRANSMIT** command was issued before completion. When the break is completed, the transmitter sends at least one idle character before transmitting any data to guarantee recognition of a valid start bit.

16.11.6.8 SENDING A PREAMBLE. A preamble sequence provides a convenient way for you to ensure that the line is idle before you start a new message. The preamble sequence is constructed of consecutive ones that are one character long. If the preamble bit in a buffer descriptor is set, a serial management controller sends a preamble sequence before transmitting that data buffer. For 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of 10 ones would be sent before the first character in the buffer. If no preamble sequence is sent, data from two ready transmit buffers can be transmitted without causing a delay on the transmit pin between the two transmit buffers.

16.11.6.9 SMCx UART CONTROLLER ERRORS. The SMCx UART controller reports character reception error conditions via the channel buffer descriptors and the SMCx UART event register. The SMCx UART controller has no transmission errors, which means you cannot stop the transmission of characters in SMCx UART mode.

- Overrun Error—The SMCx UART controller maintains a two-character length FIFO for receiving data. The data is moved to the buffer after the first character is received into the FIFO and if a receiver FIFO overrun occurs, the channel writes the received character into the internal FIFO. Then the channel writes the received character to the buffer, closes it, sets the OV bit in the receive buffer descriptor, and generates a receive interrupt if it is enabled. Reception then continues as normal.

Note: The SMCx UART controller may occasionally get an overrun error when the line is idle, in which case, ignore the error.

- Parity Error—When this error occurs, the channel writes the received character to the buffer, closes it, sets the PR bit in the buffer descriptor, and generates the receive interrupt if it is enabled. Reception then continues as normal.
- Idle Sequence Receive Error—An idle is detected when one character consisting of all ones is received. Once an idle is received, the channel counts the number of consecutive idle characters. If the count reaches the MAX_IDL value, the buffer is closed, and an receive interrupt is generated. If no receive buffer is open, this event does not generate an interrupt or any status information. The idle counter is reset every time a character is received.
- Framing Error—The SMCx UART controller receives this error when it receives a character with no stop bit. When this error occurs, the channel writes the received character to the buffer, closes the buffer, sets the FR bit in the buffer descriptor, and generates the receive interrupt if it is enabled. When this error occurs, parity is not checked for the character.
- Break Sequence Error—This error occurs when the SMCx UART receiver receives an all-zero character with a framing error. When it occurs, the channel increments the BRKEC entry and generates a maskable BRK interrupt in the SMCE–UART register. The channel also measures the length of the break sequence and stores this value in the BRKLN counter. If the channel was in the middle of buffer processing when the break was received, the buffer is closed with the BR bit in the receive buffer descriptor set and the receive interrupt is generated if it is enabled.

16.11.6.10 SMCx UART MODE REGISTER. When a serial management controller is in UART mode, the 16-bit, memory-mapped, read/write SMCx mode register is referred to as the SMCx UART mode register (SMCMR–UART). The functionality of bits 8-15 is common to each SMCx protocol, but bits 0-7 vary according to the protocol selected by the SM field. This register is cleared by reset.

SMCMR–UART

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RES	CLEN				SL	PEN	PM	RES		SM		DM		TEN	REN
RESET	0	0				0	0	0	0		0		0		0	0
R/W	R/W	R/W				R/W	R/W	R/W	R/W		R/W		R/W		R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA82 (SMC1), 0xA92 (SMC2)															

Bits 0 and 8–9—Reserved

These bits are reserved and must be set to 0.

CLEN—Character Length

This field must be programmed with the total number of bits in the character minus one. The total number of bits in the character is calculated as the sum of 1 (start bit always present) + number of data bits (5 to 14) + number of parity bits (0 or 1) + number of stop bits (1 or 2). For example, for 8 data bits, no parity, and 1 stop bit, the total number of bits in the character is 1 + 8 + 0 + 1 = 10. In this case, CLEN must be programmed to 9.

The number of data bits in the character ranges from 5 to 14 bits. If the data bit length is less than 8 bits, the most-significant bits of each byte in memory are not used on transmission and are written with zeros on reception. On the other hand, if the data bit length is more than 8 bits, the most-significant bits of each 16-bit word in memory are not used on transmit and are written with zeros on receive.

Note: The total number of bits in the character must never exceed 16. Thus, if you choose a 14-bit data length, set SL to one stop bit and disable parity. If you choose a 13-bit data length and parity is enabled, set SL to one stop bit. To prevent erratic behavior, do not write the values 0-3 to CLEN.

SL—Stop Length

0 = One stop bit.
1 = Two stop bits.

PEN—Parity Enable

0 = No parity.
1 = Parity is enabled for the transmitter and receiver, depending on the PM bit setting.

PM—Parity Mode

0 = Odd parity.
1 = Even parity.

SM—SMCx Mode

00 = GCI or SCIT support.
01 = Reserved.
10 = UART mode (must be selected for SMCx UART operation).
11 = Totally Transparent mode.

DM—Diagnostic Mode

00 = Normal mode.
01 = Local loopback mode.
10 = Echo mode.
11 = Reserved.

TEN—SMCx Transmit Enable

0 = SMCx transmitter disabled.
1 = SMCx transmitter enabled.

REN—SMCx Receive Enable

0 = SMCx receiver disabled.
1 = SMCx receiver enabled.

16.11.6.11 SMCx UART RECEIVE BUFFER DESCRIPTOR. Using the buffer descriptors, the communication processor module reports information about the received data on a per-buffer basis. It then closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer when one of the following events occurs:

- An error is received while a message is being processed.
- A full receive (RX) buffer is detected.
- A programmable number of consecutive idle characters are received.

Note: The communication processor module sets all the status bits in this buffer descriptor, but you must clear them before submitting the buffer descriptor to the communication processor module.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	RES	**W**	**I**	RESERVED		**CM**	ID	RESERVED		BR	FR	PR	RES	OV	RES
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: You are only responsible for initializing the items in bold.

E—Empty

0 = The data buffer associated with this RX buffer descriptor is filled with received data or data reception is aborted due to an error condition. The core is free to examine or write to any fields of this RX buffer descriptor. The communication processor module does not use this buffer descriptor as long as the E bit is zero.

1 = The data buffer associated with this buffer descriptor is empty or reception is currently in progress. This RX buffer descriptor and its associated receive buffer are owned by the communication processor module. Once the E bit is set, the core must not write any fields of this RX buffer descriptor.

Bits 1, 4–5, 8–9, 13, and 15—Reserved

These bits are reserved and must be set to 0.

W—Wrap (Final Buffer Descriptor in Table)

0 = This is not the last buffer descriptor in the RX buffer descriptor table.

1 = This is the last buffer descriptor in the RX buffer descriptor table. After this buffer is used, the communication processor module receives incoming data into the first buffer descriptor that RBASE points to in the table. The number of RX buffer descriptors in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer is filled.

1 = The RX bit in the event register is set when this buffer is completely filled by the communication processor module, indicating the need for the core to process the buffer. The RX bit can cause an interrupt if it is enabled.

CM—Continuous Mode

0 = Normal operation.

1 = The E bit is not cleared by the communication processor module after this buffer descriptor is closed, thus allowing the associated data buffer to be automatically overwritten next time the communication processor module accesses this buffer descriptor. However, the E bit is cleared if an error occurs during reception, regardless of how the CM bit is set.

ID—Buffer Closed on Reception of Idles

This bit indicates that the buffer has closed because a programmable number of consecutive idle sequences have been received. The communication processor module writes this bit after the received data is in the associated data buffer.

BR—Buffer Closed on Reception of Break

This bit indicates that the buffer has closed because a break sequence has been received. The communication processor module writes this bit after the received data is in the associated data buffer.

0 = No break sequence is received.

1 = A break sequence is received and the buffer closes.

FR—Framing Error

This bit indicates that a character with a framing error has been received and is located in the last byte of this buffer. A framing error is a character without a stop bit. A new receive buffer is used to receive additional data. The communication processor module writes this bit after the received data is in the associated data buffer.

PR—Parity Error

This bit indicates that a character with a parity error has been received and is located in the last byte of this buffer. A new receive buffer is used to receive additional data. The communication processor module writes this bit after the received data is in the associated data buffer.

OV—Overrun

This bit indicates that a receiver overrun has occurred during message reception. The communication processor module writes this bit after the received data is in the associated data buffer.

DATA LENGTH

This field represents the number of octets the communication processor module writes into this buffer descriptor data buffer. After the data is received in the associated data buffer, the communication processor module writes this field once the buffer descriptor closes.

Note: The actual amount of memory allocated for this buffer must be greater than or equal to the MRBLR entry.

RX DATA BUFFER POINTER

This field always points to the first location of the associated data buffer and must be even. The buffer can reside in internal or external memory. The communication processor module writes this bit after the received data is in the associated data buffer.

Figure 16-116 illustrates an example of the RX buffer descriptor process. It shows the resulting state of the RX buffer descriptors after they receive 10 characters, an idle period, and five characters (one with a framing error). The example assumes that MRBLR = 8 in the SMCx parameter RAM.

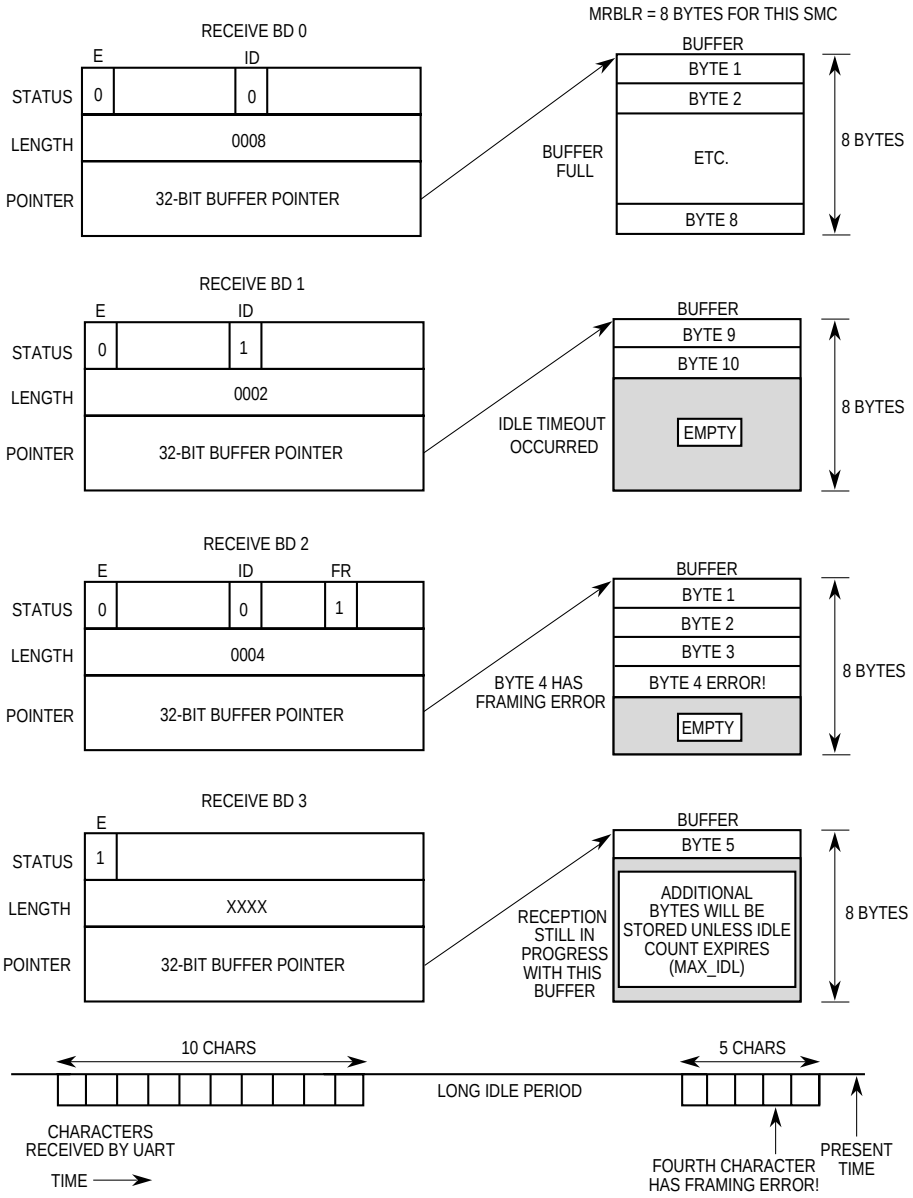


Figure 16-116. SMCx UART Receive Buffer Descriptor Example

16.11.6.12 SMCx UART TRANSMIT BUFFER DESCRIPTOR. Data is sent to the communication processor module for transmission on an SMCx channel by arranging it in buffers referenced by the channel's transmit (TX) buffer descriptor ring. Using the buffer descriptors, the communication processor module confirms transmission or indicates error conditions so that the processor knows the buffers have been serviced.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**R**	RES	**W**	**I**	RESERVED		**CM**	**P**	RESERVED							
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: You are only responsible for initializing the items in bold.

Note: The communication processor module sets all the status bits in this buffer descriptor, but you must clear them before submitting the buffer descriptor to the communication processor module.

R—Ready

0 = The data buffer associated with this buffer descriptor is not ready for transmission, but you are free to manipulate this buffer descriptor or its associated data buffer. The communication processor module clears this bit after the buffer has been transmitted or an error condition is encountered.

1 = The data buffer, which you prepare for transmission, is not transmitted yet or is currently being transmitted. You cannot write any fields of this buffer descriptor once this bit is set.

Bits 1, 4–5, and 8–15—Reserved

These bits are reserved and must be set to 0.

W—Wrap (Final Buffer Descriptor in Table)

0 = This is not the last buffer descriptor in the TX buffer descriptor table.

1 = This is the last buffer descriptor in the TX buffer descriptor table. After this buffer is used, the communication processor module receives incoming data into the first buffer descriptor that TBASE points to in the table. The number of TX buffer descriptors in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer is serviced.

1 = The TX bit in the SMCE–UART register is set when this buffer is serviced. Transmission can cause an interrupt if it is enabled.

CM—Continuous Mode

0 = Normal operation.
1 = The R bit is not cleared by the communication processor module after this buffer descriptor is closed, thus allowing the associated data buffer to be automatically retransmitted next time the communication processor module accesses this buffer descriptor.

P—Preamble

0 = No preamble sequence is sent.
1 = The SMCx UART controller sends one all-ones character before it sends the data so that the other end detects an idle line before the data is received. If this bit is set and the data length of this buffer descriptor is zero, only a preamble is sent.

DATA LENGTH

This field represents the number of octets that the communication processor module must transmit from this buffer descriptor data buffer. However, it is never modified by the communication processor module. Normally, this value must be greater than zero, but it can be equal to zero with the P bit set if only a preamble is sent.

If the number of data bits in the SMCx UART character is greater than 8, then the data length must be even. For example, to transmit three UART characters of 8-bit data, 1 start, and 1 stop, the DATA LENGTH field must be initialized to 3. However, to transmit three SMCx UART characters of 9-bit data, 1 start, and 1 stop, the DATA LENGTH field must be initialized to 6, since the three 9-bit data fields occupy three half-words in memory (the 9 least-significant bits of each half-word).

TX DATA BUFFER POINTER

This field always points to the first location of the associated data buffer. It can be even or odd, unless the number of actual data bits in the SMCx UART character is greater than 8 bits, in which this field is even. For instance, the pointer to 8-bit data, 1 start, and 1 stop characters can be even or odd, but the pointer to 9-bit data, 1 start, and 1 stop characters must be even. The buffer can reside in internal or external memory.

16.11.6.13 SMCx UART EVENT REGISTER. When a serial management controller is in UART mode, the 8-bit memory-mapped SMCx event register is referred to as the SMCx UART event (SMCE–UART) register. It is used to generate interrupts and report events recognized by the SMCx UART channel. When an event is recognized, the SMCx UART controller sets the corresponding bit in this register.

A bit is cleared by writing a 1 (writing a zero has no effect) and more than one bit can be cleared at a time. All unmasked bits must be cleared before the communication processor module clears the internal interrupt request. This register is cleared by reset and can be read at any time. An example of the timing of various events in the SMCE–UART register is illustrated in Figure 16-117.

SMCE –UART

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED	BRKE	RESERVED	BRK	RESERVED	BSY	TX	RX
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA86 (SMC1), 0xA96 (SMC2)							

Bit 0, 2, and 4—Reserved

These bits are reserved and must be set to 0.

BRKE—Break End

This bit indicates that an end of break sequence has been detected. It occurs after one idle bit is received after a break sequence.

BRK—Break Character Received

This bit indicates that a break character has been received. If a very long break sequence occurs, this interrupt only occurs once after the first all-zeros character is received.

BSY—Busy Condition

This bit indicates that a character has been received and discarded due to a lack of buffers. It is be set in the middle of the last stop bit of the first receive character for which there is no available buffer. Reception continues when an empty buffer is provided.

TX—TX Buffer

This bit indicates that a buffer has been transmitted over the SMCx UART channel. It is set once the transmit data of the last character in the buffer is written to the transmit FIFO. You must wait two character times to be sure that the data is completely sent over the transmit pin.

RX—Receive Buffer

This bit indicates that a buffer has been received and its associated RX buffer descriptor is now closed. It is set in the middle of the last stop bit of the last character that is written to the receive buffer.

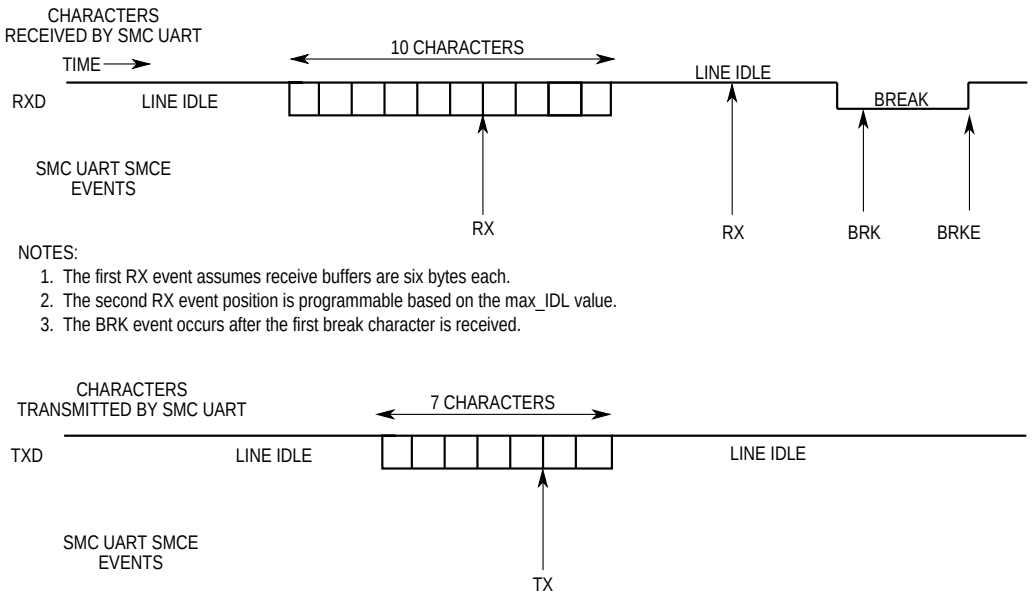


Figure 16-117. SMCx UART Interrupt Example

16.11.6.14 SMCx UART MASK REGISTER. When a serial management controller is in UART mode, the 8-bit read/write SMCx mask register is referred to as the SMCx UART mask (SMCM–UART) register. It has the same bit format as the SMCE–UART register. If a bit in this register is a 1, the corresponding interrupt in the SMCE–UART register is enabled. If the bit is zero, the corresponding interrupt in the SMCE–UART register is masked.

SMCM–UART

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED	BRKE	RESERVED	BRK	RESERVED	BSY	TX	RX
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA8A (SMC1), 0xA94 (SMC2)							

16.11.6.15 SMC1 UART CONTROLLER PROGRAMMING EXAMPLE. The following is an initialization sequence for 9,600 baud, 8 data bits, no parity, and 1 stop bit operation of an SMC1 UART controller assuming a 25MHz system frequency. BRG1 and SMC1 are used.

1. Configure the port B pins to enable SMTXD1 and SMRXD1. Write PBPAR bits 25 and 24 with ones and then PBDIR and PBODR bits 25 and 24 with zeros.
2. Configure the BRG1. Write 0x010144 to BRGC1. The DIV16 bit is not used and the divider is 162 (decimal). The resulting BRG1 clock is 16× the preferred bit rate of the SMC1 UART controller.
3. Connect the BRG1 clock to SMC1 using the serial interface. Write the SMC1 bit in SIMODE with a 0 and the SMC1CS field in SIMODE register with 0x000.
4. Write RBASE and TBASE in the SMC1 parameter RAM to point to the RX buffer descriptor and TX buffer descriptor in the dual-port RAM. Assuming one RX buffer descriptor at the beginning of dual-port RAM and one TX buffer descriptor following that RX buffer descriptor, write RBASE with 0x2000 and TBASE with 0x2008.
5. Program the CPCR to execute the **INIT RX AND TX PARAMS** command. Write 0x0091 to the CPCR.
6. Write 0x0001 to the SDCR to initialize the SDMA configuration register.
7. Write 0x18 to the RFCR and TFCR for normal operation.
8. Write MRBLR with the maximum number of bytes per receive buffer. Assume 16 bytes, so MRBLR = 0x0010.
9. Write MAX_IDL with 0x0000 in the SMC1 UART parameter RAM to disable the MAX_IDL functionality for this example.
10. Clear BRKLN and BRKEC in the SMC1 UART parameter RAM for the clarity.
11. Set BRKCR to 0x0001, so that if a **STOP TRANSMIT** command is issued, one break character is sent.

12. Initialize the RX buffer descriptor. Assume the RX data buffer is at 0x00001000 in main memory. Write 0xB000 to RX_BD_Status, 0x0000 to RX_BD_Length (not required), and 0x00001000 to RX_BD_Pointer.
13. Initialize the TX buffer descriptor. Assume the TX data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Then write 0xB000 to TX_BD_Status, 0x0005 to TX_BD_Length, and 0x00002000 to TX_BD_Pointer.
14. Write 0xFF to the SMCE–UART register to clear any previous events.
15. Write 0x17 to the SMCM–UART register to enable all possible serial management controller interrupts.
16. Write 0x00000010 to the CIMR so SMC1 can generate a system interrupt. The CICR must also be initialized.
17. Write 0x4820 to SMCMR to configure normal operation (not loopback), 8-bit characters, no parity, 1 stop bit. Notice that the transmitter and receiver are not enabled yet.
18. Write 0x4823 to SMCMR to enable the SMC1 transmitter and receiver. This additional write ensures that the TEN and REN bits are enabled last.

Note: After 5 bytes are transmitted, the TX buffer descriptor is closed. The receive buffer is closed after 16 bytes are received. Any data received after 16 bytes causes a busy (out-of-buffers) condition since only one RX buffer descriptor is prepared.

16.11.6.16 HANDLING INTERRUPTS IN THE SMCx UART CONTROLLER. Follow these steps to handle an interrupt in a serial management controller:

1. Once an interrupt occurs, read the SMCE–UART register to discover the cause of the interrupt. To clear the SMCE bits, write ones to them.
2. Process the TX buffer descriptor to reuse it if the TX bit is set in the SMCE–UART register. Extract data from the RX buffer descriptor if the RX bit is set in the SMCE–UART. To transmit another buffer, set the R bit in the TX buffer descriptor.
3. Clear the SMCx bit in the CISR.
4. Execute the **rfi** instruction.

16.11.7 The SMCx in Transparent Mode

A serial managment controller in Transparent mode is also referred to as the SMCx Transparent controller. The following is a list of the features that the SMCx Transparent controller does **not** support:

- Independent transmit and receive clocks, unless it is connected to the TDM channel of the serial interface
- CRC generation and checking
- Full $\overline{\text{RTSx}}$, $\overline{\text{CTSx}}$, and $\overline{\text{CDx}}$ pins. Instead, there is a SMSYN pin for each SMC.
- Ability to transmit data on demand using the transmit on demand register
- Receiver/transmitter in transparent mode while executing another protocol
- 4-, 8-, or 16-bit Sync recognition
- Internal DPLL support

An SMCx in Transparent mode has one feature that an SCCx in Transparent mode does not. The serial management controllers allow a data character length option of 4 to 16 bits, whereas the SCCs allow 8 or 32 bits, depending on how the RFW bit is set in the GSMR_H (described in **Section 16.9.2 The General SCCx Mode Registers**).

16.11.7.1 FEATURES. The following list summarizes the features of a serial management controller in Transparent mode:

- Flexible data buffers
- Connects to the TDM bus using the time-slot assigner in the serial interface
- Transmits and receives transparently on its own set of pins using a sync pin to synchronize the beginning of transmission and reception to an external event
- Programmable 4-16 character length
- Reverse data mode
- Continuous transmission and reception modes
- Four available commands

16.11.7.2 SMCx TRANSPARENT CHANNEL TRANSMISSION PROCESS. The SMCx Transparent transmitter is designed to work with almost no intervention from the core. When the core enables the SMCx transmitter in transparent mode, it starts transmitting idles. The serial management controllers immediately poll the first buffer descriptor in the transmit channel buffer descriptor ring once every character time, depending on the character length (every 4 to 16 serial clocks). When there is a message to transmit, a serial management controller fetches the data from memory and starts transmitting the message after synchronization is achieved.

Synchronization can be achieved in two ways. First, when the transmitter is connected to the TDM channel, it can be synchronized to a time-slot. Once the frame sync is received, the transmitter waits for the first bit of its time-slot to occur before it starts transmitting. Data is only transmitted during the time-slots defined by the time-slot assigner. Secondly, when working with its own set of pins, the transmitter starts transmitting when the $\overline{\text{SMSYNx}}$ signal is asserted.

When buffer descriptor data is completely written to the transmit FIFO, the L bit is checked and if it is set, a serial management controller writes the message status bits into the buffer descriptor and clears the R bit. It then starts transmitting idles. When the end of the current buffer descriptor is reached and the L bit is not set, only the R bit is cleared. In both cases, an interrupt is issued according to the I bit in the buffer descriptor. By appropriately setting the I bit in each buffer descriptor, interrupts can be generated after each buffer, a specific buffer, or each block is transmitted. The serial management controller then proceeds to the next buffer descriptor in the table. If no additional buffers have been presented to the serial management controller for transmission and the L bit was cleared, an underrun is detected and the serial management controller begins transmitting idles.

If the CM bit is set in the transmit buffer descriptor, the R bit is not cleared, thus allowing the associated data buffer to be automatically retransmitted next time the communication processor module accesses this data buffer. For instance, if a single transmit buffer descriptor is initialized with the CM bit and the W bit set, the data buffer is continuously transmitted until you clear the R bit of the buffer descriptor.

16.11.7.3 SMCx TRANSPARENT CHANNEL RECEPTION PROCESS. When the core enables the SMCx receiver in transparent mode, it waits for synchronization before receiving data. Once synchronization is achieved, the receiver transfers incoming data into memory according to the first receive buffer descriptor in the ring. Synchronization can be achieved in two ways. First, when the receiver is connected to the TDM channel, it can be synchronized to a time-slot. Once the frame sync is received, the receiver waits for the first bit of its time-slot to occur before reception begins. Data is only received during the time-slots defined by the time-slot assigner. Secondly, when working with its own set of pins, the receiver starts receiving when the $\overline{\text{SMSYNx}}$ signal is asserted.

When the data buffer is filled, a SMCx Transparent controller clears the E bit in the buffer descriptor and generates an interrupt if the I bit in the buffer descriptor is set. If the incoming data exceeds the length of the data buffer, a serial management controller fetches the next buffer descriptor in the table and, if it is empty, continues transferring data to the associated data buffer. If the CM bit is set in the receive buffer descriptor, the E bit is not cleared, thus allowing the associated data buffer to be automatically overwritten next time the communication processor module accesses this data buffer.

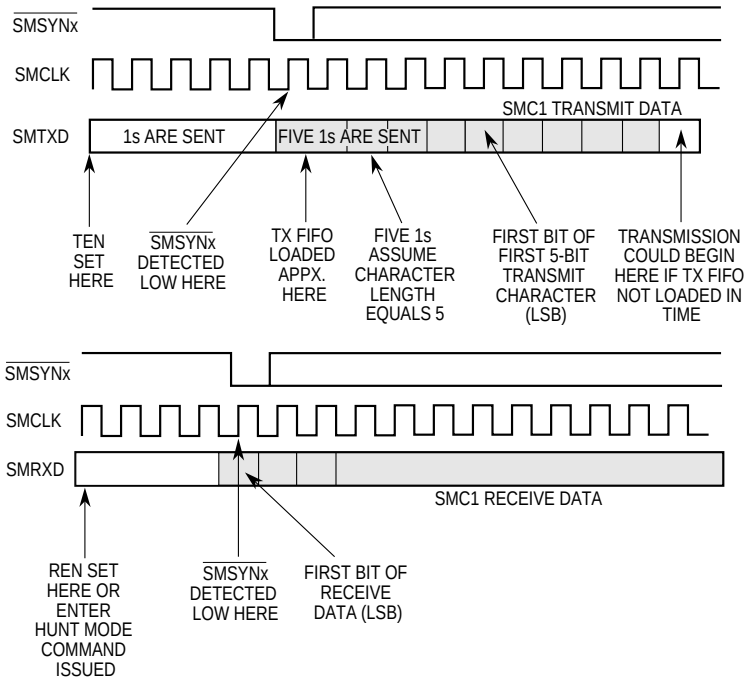
16.11.7.4 USING THE SMSYNx PIN FOR SYNCHRONIZATION. The SMSYNx pin offers a method to externally synchronize a SMCx Transparent channel. This method differs somewhat from the synchronization options available in the serial communication controllers and must be studied carefully. See Figure 16-118 for an example.

Once the REN bit is set in the SMCMR, the first rising edge of the SMCLK signal that finds the SMSYNx pin low causes the SMCx receiver to achieve synchronization. Data starts being received or latched on the same rising edge of SMCLK that latched SMSYNx. This is the first bit of data received. The receiver never loses synchronization again, regardless of the state of SMSYNx, until you clear the REN bit.

Once the TEN bit is set in the SMCMR, the first rising edge of the SMCLK signal that finds the SMSYNx pin low causes the SMCx transmitter to achieve synchronization. The SMCx transmitter begins transmitting ones asynchronously from the falling edge of SMSYNx. After one character of ones is transmitted, if the transmit FIFO is loaded (the transmit buffer descriptor is ready with data), data starts being transmitted on the next falling edge of SMCLK after some multiple of all-ones (preamble) characters are transmitted. If the transmit FIFO is loaded at some later time, the data starts transmitting after some multiple number of all-ones characters is transmitted.

Note: Regardless of whether the transmitter or receiver uses the SMSYNx signal, it must make glitch-free transitions from high to low or low to high. Glitches on SMSYNx can cause a serial management controller to behave erratically.

The transmitter never loses synchronization again, regardless of the state of SMSYNx, until you clear the TEN bit or issue the **ENTER HUNT MODE** command.



NOTES:
1. SMCLK is an internal clock derived from an external clock pin or a baud rate generator.
2. This example shows the SMC receiver and transmitter enabled separately. If the REN and TEN bits were set at the same time, a single falling edge of SMSYNx would synchronize both.

Figure 16-118. SMSYNx Pin Synchronization

If both the REN and TEN bits are set in the SMCMR, the first falling edge of the SMSYNx pin causes both the transmitter and receiver to achieve synchronization. To resynchronize the transmitter or receiver, the SMCx transmitter/receiver can be disabled and reenabled and the SMSYNx pin can be used again to resynchronize the transmitter or receiver. Refer to **Section 16.11.5 Disabling the SMCs On-the-Fly** for a description of how to safely disable and reenable a serial management controller. Simply clearing and setting the TEN bit may not be sufficient.

16.11.7.5 USING THE TIME-SLOT ASSIGNER FOR SYNCHRONIZATION. The time-slot assigner offers a method to internally synchronize a SMCx Transparent channel without using the $\overline{\text{SMSYNx}}$ pin. This method is similar to that of the $\overline{\text{SMSYNx}}$ pin, except that the synchronization event is not the falling edge of the $\overline{\text{SMSYNx}}$ signal, but the first time-slot for this SMCx receiver/transmitter after the frame sync indication. Refer to **Section 16.7 The Serial Interface with Time-Slot Assigner** for further information about configuring time-slots for the SMCs and SCCs.

The time-slot assigner allows the SMCx receiver and transmitter to be enabled simultaneously and synchronized, a capability that the $\overline{\text{SMSYNx}}$ pin does not provide. Refer to Figure 16-119 for an example of synchronization using the time-slot assigner.

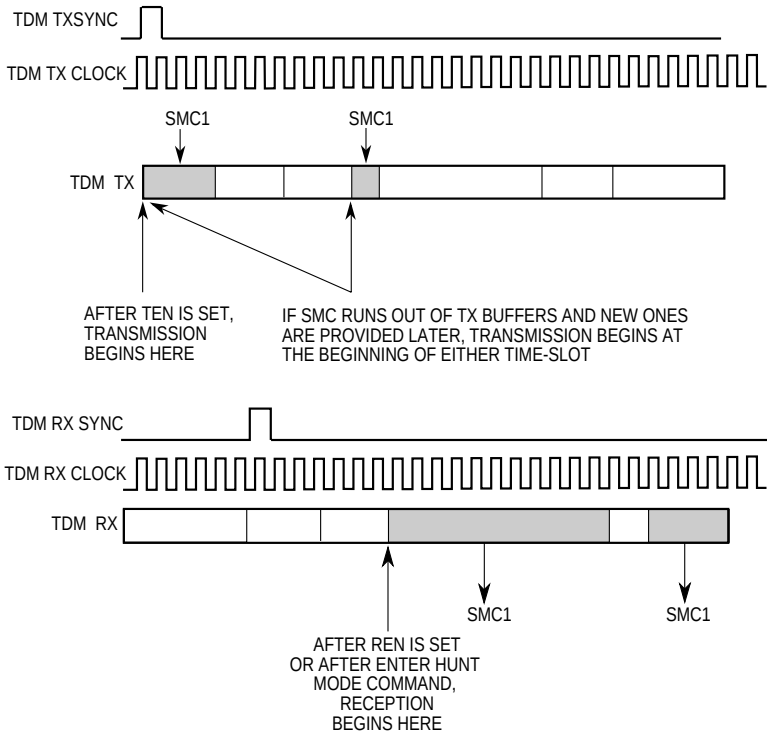


Figure 16-119. Time-Slot Assigner Synchronization

Once the REN bit is set in the SMCMR, the first time-slot after frame sync causes the SMCx receiver to achieve synchronization. Data is received immediately, but only during the defined receive time-slots. The receiver continues receiving data during its defined time-slots until you clear the REN bit. If the **ENTER HUNT MODE** command is issued, the receiver loses synchronization, closes the current buffer, and resynchronizes to the first time-slot after the frame sync.

Once the TEN bit is set in the SMCMR, an SMCx Transparent controller waits for the transmit FIFO to be loaded before trying to achieve synchronization. Once the transmit FIFO is loaded, synchronization and transmission begins, depending on the following situations:

- If a buffer is made ready when an SMCx Transparent controller is enabled, then the first byte will be placed in time-slot 1 if the CLEN field in SMCMR is set to 8 and slot 2 if CLEN is set to 16.
- If a buffer has an SMCx Transparent controller enabled, then the first byte in the next buffer can appear in any time-slot associated with this channel.
- If a buffer is ended with the L bit set, then the next buffer can appear in any time-slot associated with this channel.

If an SMCx Transparent controller runs out of transmit buffers and a new transmit buffer is provided later, idles are transmitted during the gap between data buffers. Data transmission from the later data buffer begins at the beginning of an SMCx Transparent controller time-slot, but not necessarily the first time-slot after the frame sync. So if you want to maintain a certain bit alignment beginning with the first time-slot, make sure that at least one TX buffer descriptor is always ready and that no underrun occurs. Otherwise, the SMCx Transparent transmitter must be disabled and reenabled. Refer to **Section 16.11.5 Disabling the SMCs On-the-Fly** for a description of how to safely disable and reenable the SMCx Transparent controller. Simply clearing and setting TEN may not be sufficient.

16.11.7.6 SMCx TRANSPARENT CONTROLLER PARAMETER RAM MEMORY MAP. There is no protocol-specific parameter RAM for the SMCx Transparent controller. Only the general SMCx parameter RAM is used. See **Section 16.11.4 SMC General Parameter RAM Memory Map** for more information.

16.11.7.7 SMCx TRANSPARENT COMMANDS. You can program the CPCR (described in **Section 16.2.6.1 CPM Command Register**) with the following commands to transmit data.

- **STOP TRANSMIT**—After the hardware or software is reset and the channel is enabled in the SMCM–Transparent register, the channel is in transmit enable mode and starts polling the first buffer descriptor in the table. This command disables the transmission of frames on the transmit channel. If the transparent controller receives this command while transmitting a frame, it stops after the contents of the FIFO are transmitted (up to 2 characters). The TBPTR is not advanced to the next buffer descriptor, no new buffer descriptor is accessed, and no new buffers are transmitted for this channel. The transmitter sends idles until the **RESTART TRANSMIT** command is issued.
- **RESTART TRANSMIT**—This command is used to begin or continue transmission from the current TBPTR in the channel's TX buffer descriptor table. When the channel receives this command, it starts polling the R bit in the TX buffer descriptor. A serial management controller expects this command after a **STOP TRANSMIT** command is issued and the channel in the SMCMR is disabled or after a transmitter error occurs.
- **INIT TX PARAMETERS**—This command initializes all the transmit parameters in this serial channel parameter RAM to their reset state and must only be issued when the transmitter is disabled. The **INIT TX AND RX PARAMS** command can also be used to reset the transmit and receive parameters.

You can program the CPCR with the following commands to receive data.

- **ENTER HUNT MODE**—This command forces a serial management controller to close the current receive buffer descriptor if it is currently being used and to use the next buffer descriptor in the list for any subsequently received data. If a serial management controller is not in the process of receiving data, the buffer is not closed. Additionally, this command causes the receiver to wait for a resynchronization before further reception continues.
- **CLOSE RX BD**—This command is used to force a serial management controller to close the current receive buffer descriptor if it is being used and to use the next buffer descriptor in the list for any subsequently received data. If a serial management controller is not in the process of receiving data, no action is taken by this command.
- **INIT RX PARAMETERS**—This command initializes all the receive parameters in this serial channel's parameter RAM to their reset state. It must only be issued when the receiver is disabled. The **INIT TX AND RX PARAMS** command can also be used to reset the receive and transmit parameters.

16.11.7.8 SMCx TRANSPARENT CONTROLLER ERRORS. The serial management controllers report message reception and transmission error conditions using the channel buffer descriptors and the SMCE–Transparent register. The following transmission errors can be detected by the SMCx Transparent controller.

- Underrun Error—When this error occurs, the channel stops transmitting the buffer, closes it, sets the UN bit in the buffer descriptor, sets the TXE bit in the event register and causes an interrupt if the TXE bit is set in the mask register. The channel resumes transmission after it receives the **RESTART TRANSMIT** command. Underrun cannot occur between frames.
- Overrun Error—A serial management controller maintains an internal FIFO for receiving data. The communication processor module begins programming the SDMA channel if the data buffer is in external memory when the first character is received into the FIFO. If a FIFO overrun occurs, a serial management controller writes the received data character to the internal FIFO over the previously received character. The previous character and its status bits are lost. Then the channel closes the buffer, sets the OV bit in the receive buffer descriptor, and generates the receive interrupt if it is enabled. Reception then continues as normal.

16.11.7.9 SMCx TRANSPARENT MODE REGISTER. When a serial management controller is in transparent mode, the 16-bit, memory-mapped, read/write SMCx mode register is referred to as the SMCx transparent mode register (SMCMR–Transparent). The function of bits 8-15 is common to each SMCx protocol, but bits 0-7 vary according to the protocol selected by the SM bits of this register.

SMCMR–TRANSPARENT

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RES	CLEN				RES	BS	REVD	RESERVED		SM		DM		TEN	REN
RESET	0	0				0	0	0	0		0		0		0	0
R/W	R/W	R/W				R/W	R/W	R/W	R/W		R/W		R/W		R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA82 (SMC1), 0xA92 (SMC2)															

Bits 0, 5, and 8–9—Reserved

These bits are reserved and must be set to 0.

CLEN—Character Length

This field is programmed with a value between 3 and 15 to obtain 4 to 16 bits per character. If the character length is less than 8 bits, the MSBs of the byte in buffer memory are not used on transmit and are written with zeros on receive. On the other hand, if the character length is more than 8 bits but less than 16 bits, the MSBs of the half-word in buffer memory are not used on transmit and are written with zeros on receive.

Note: Erratic behavior will occur if you do not write the values 0 to 2 to CLEN. Larger character lengths increase the potential performance of the SMCx channel and lower the performance impact of other channels. For instance, using 16-bit characters, rather than 8-bit characters is recommended if 16-bit characters are acceptable in the final application.

BS—Byte Sequence

This bit controls the sequence of byte transmission if the REVD bit is set for a character length greater than 8 bits. It must be set to zero to maintain behavior compatibility with the MC68360 QUICC microprocessor.

0 = Normal mode. This must be selected if the character length is less than or equal to 8 bits.
1 = Transmit lower address byte first.

REVD—Reverse Data

0 = Normal mode.
1 = Reverse the character bit order. The MSB is transmitted first.

SM—SMCx Mode

00 = GCI or SCIT mode.
01 = Reserved.
10 = UART mode.
11 = Totally transparent mode. This must be selected for SMCx transparent operation.

DM—Diagnostic Mode

00 = Normal mode.
01 = Local loopback mode.
10 = Echo mode.
11 = Reserved.

TEN—SMCx Transmit Enable

0 = SMCx transmitter disabled.
1 = SMCx transmitter enabled.

Note: Once the TEN bit is cleared, it must not be reenabled for at least three serial clocks.

REN—SMCx Receive Enable

0 = SMCx receiver disabled.
1 = SMCx receiver enabled.

16.11.7.10 SMCx TRANSPARENT RECEIVE BUFFER DESCRIPTOR. Using receive (RX) buffer descriptors, the communication processor module reports information about the received data for each buffer and closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events occurs:

- An overrun error occurs
- A full receive buffer is detected
- The **ENTER HUNT MODE** command is issued

Note: The communication processor module sets all the status bits in this buffer descriptor, but you must clear them before submitting the buffer descriptor to the communication processor module.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	RES	**W**	**I**	RESERVED		**CM**	RESERVED							OV	RES
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: You are only responsible for initializing the items in bold.

E—Empty

0 = The data buffer associated with this RX buffer descriptor is filled with received data or data reception has been aborted due to an error condition. The core is free to examine or write to any fields of this RX buffer descriptor. The communication processor module does not use this buffer descriptor as long as the E bit is zero.
1 = The data buffer associated with this buffer descriptor is empty or is currently receiving data. This RX buffer descriptor and its associated receive buffer are owned by the communication processor module. Once the E bit is set, the core must not write any fields of this RX buffer descriptor.

Bits 1, 4–5, 7–13, and 15—Reserved

These bits are reserved and must be set to 0.

W—Wrap (Final Buffer Descriptor in Table)

0 = This is not the last buffer descriptor in the RX buffer descriptor table.
1 = This is the last buffer descriptor in the RX buffer descriptor table. After this buffer is used, the communication processor module receives incoming data into the first buffer descriptor that RBASE points to in the table. The number of RX buffer descriptors in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer is filled.
1 = The RX bit in the SMCE–Transparent register is set when this buffer is completely filled by the communication processor module, thus indicating that the core needs to process the buffer. The RX bit can cause an interrupt if it is enabled.

CM—Continuous Mode

0 = Normal operation.
1 = The E bit is not cleared by the communication processor module after this buffer descriptor is closed, thus allowing the associated data buffer to be automatically overwritten next time the communication processor module accesses this buffer descriptor. However, the E bit is cleared if an error occurs during reception, regardless of how the CM bit is set.

OV—Overrun

This bit indicates that a receiver overrun has occurred during message reception. The communication processor module writes this bit after the received data is placed into the associated data buffer.

DATA LENGTH

This field represents the number of octets that the communication processor module writes into this data buffer. It is only written once by the communication processor module as the buffer is closed. The communication processor module writes this field after the received data is placed into the associated data buffer.

Note: The actual amount of memory allocated for this buffer must be greater than or equal to the MRBLR entry.

RX DATA BUFFER POINTER

This field always points to the first location of the associated data buffer, must be even, and can reside in internal or external memory. The communication processor module writes these bits after the received data is placed into the associated data buffer.

16.11.7.11 SMCx TRANSPARENT TRANSMIT BUFFER DESCRIPTOR. Data is sent to the communication processor module for transmission on an SMCx channel by arranging it in buffers referenced by the channel's transmit (TX) buffer descriptor table. Using the buffer descriptors, the communication processor module confirms transmission or indicates error conditions so that the processor knows the buffers have been serviced.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**R**	RES	**W**	**I**	**L**	RES	**CM**	RESERVED							UN	RES
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: You are only responsible for initializing the items in bold.

Note: The communication processor module sets all the status bits in this buffer descriptor, but you must clear them before submitting the buffer descriptor to the communication processor module.

R—Ready

0 = The data buffer associated with this buffer descriptor is not ready for transmission and you are free to manipulate it or its associated data buffer. The communication processor module clears this bit after the buffer is transmitted or after an error condition is encountered.

1 = The data buffer, which you prepare for transmission, is not transmitted yet or is currently being transmitted. You cannot write any fields of this buffer descriptor once this bit is set.

Bits 1, 5, 7–13, and 15—Reserved

These bits are reserved and must be set to 0.

W—Wrap (Final Buffer Descriptor in Table)

0 = This is not the last buffer descriptor in the TX buffer descriptor table.

1 = This is the last buffer descriptor in the TX buffer descriptor table. After this buffer is used, the communication processor module receives incoming data into the first buffer descriptor that TBASE points to in the table. The number of TX buffer descriptors in this table is programmable and determined by the W bit and overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer is serviced.

1 = The TX and TXE bits in the SMCE–Transparent register are set when this buffer is serviced. TX and TXE can cause interrupts if they are enabled.

L— Last in Message

0 = The last byte in the buffer is not the last byte in the transmitted transparent frame. Data from the next transmit buffer (if ready) is transmitted immediately following the last byte of this buffer.

1 = The last byte in this buffer is the last byte in the transmitted transparent frame. After this buffer is transmitted, the transmitter requires synchronization before the next buffer is transmitted.

CM—Continuous Mode

0 = Normal operation.

1 = The communication processor module does not clear the R bit after this buffer descriptor is closed, thus allowing the associated data buffer to be automatically retransmitted next time the communication processor module accesses this buffer descriptor. However, the R bit will be cleared if an error occurs during transmission, regardless of how the CM bit is set.

UN—Underrun

This bit indicates that a serial management controller has encountered a transmitter underrun condition while transmitting the associated data buffer.

DATA LENGTH

This field represents the number of octets that the communication processor module must transmit from this data buffer and it is never modified by the communication processor module. This field can be even or odd, but if the number of bits in the transparent character is greater than 8, this field must be even. For example, to transmit three transparent 8-bit characters, this field must be initialized to 3. However, to transmit three transparent 9-bit characters, this field must be initialized to 6 since the three 9-bit characters occupy three half-words in memory.

TX DATA BUFFER POINTER

This field always points to the first byte of the associated data buffer. They can be even or odd, unless the character length is greater than 8 bits, in which case this field must be even. For instance, the pointer to 8-bit transparent characters can be even or odd, but the pointer to 9-bit transparent characters must be even. The buffer can reside in internal or external memory.

16.11.7.12 SMCx TRANSPARENT EVENT REGISTER. When a serial management controller is in transparent mode, the 8-bit memory-mapped SMCx event register is referred to as the SMCx transparent event (SMCE–Transparent) register. It is used to generate interrupts and report events recognized by the SMCx channel. When an event is recognized, the serial management controller sets the corresponding bit in this register. Interrupts generated by this register can be masked in the SMCM–Transparent register.

A bit is cleared by writing a 1 (writing a zero has no effect) and more than one bit can be cleared at a time. All unmasked bits must be cleared before the communication processor module clears the internal interrupt request. This register is cleared at reset and can be read at any time.

SMCE–TRANSPARENT

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED			TXE	RES	BSY	TX	RX
RESET	0			0	0	0	0	0
R/W	R/W			R/W	R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA86 (SMC1), 0xA96 (SMC2)							

Bits 0–2 and 4—Reserved

These bits are reserved and must be set to 0.

TXE—TX Error

This bit indicates that an underrun error has occurred on the transmitter channel.

BSY—Busy Condition

This bit indicates that a character has been received and discarded due to a lack of buffers. Reception begins after a new buffer is provided. You can execute an **ENTER HUNT MODE** command to make the receiver wait for resynchronization.

TX—TX Buffer

This bit indicates that a buffer has been transmitted. If the L bit of the TX buffer descriptor is set, this bit is set when the last data character starts being transmitted and you must wait one character time to be sure that the data is completely sent over the transmit pin. If the L bit of the TX buffer descriptor is cleared, this bit is set when the last data character is written to the transmit FIFO and you must wait two character times to be sure that the data is completely sent over the transmit pin.

RX—RX Buffer

This bit indicates that a data buffer has been received on the SMCx channel and its associated RX buffer descriptor is now closed. This bit is set after the last character is written to the buffer.

16.11.7.13 SMCx TRANSPARENT MASK REGISTER. When a serial management controller is in transparent mode, the 8-bit read/write SMCx mask register is referred to as the SMCx transparent mask (SMCM–Transparent) register. It has the same bit format as the SMCE–Transparent register. If a bit in this register is a 1, the corresponding interrupt in the SMCE–Transparent register is enabled. If the bit is zero, the corresponding interrupt is masked.

SMCM–TRANSPARENT

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED			TXE	RES	BSY	TX	RX
RESET	0			0	0	0	0	0
R/W	R/W			R/W	R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA8A (SMC1), 0xA9A (SMC2)							

16.11.7.14 SMCx TRANSPARENT NMSI PROGRAMMING EXAMPLE. The following is an example initialization sequence for an SMC1 transparent channel over its own set of pins. The transmit and receive clocks are provided by the CLK3 pin and the SMSYN1 pin is used to obtain synchronization.

1. Configure the port B pins to enable the SMTXD1, SMRXD1, and SMSYN1 pins. Write PBPAR bits 25, 24, and 23 with ones and then PBDIR and PBODR bits 25, 24, and 23 with zeros.
2. Configure the port A pins to enable CLK3. Write PAPAR bit 5 with a one and PADIR bit 5 with a zero. The other functions of this pin are the timers or the time-slot assigner. These alternate functions cannot be used on this pin.
3. Connect the CLK3 clock to SMC1 using the serial interface. Write the SMC1 bit in the SIMODE register with a 0 and the SMC1CS field in the SIMODE register with 110.
4. Write RBASE and TBASE in the SMCx parameter RAM to point to the RX buffer descriptor and TX buffer descriptor in the dual-port RAM. Assuming one RX buffer descriptor at the beginning of the dual-port RAM and one TX buffer descriptor following that RX buffer descriptor, write RBASE with 0x2000 and TBASE with 0x2008.
5. Program the CPCR to execute the **INIT RX AND TX PARAMS** command. Write 0x0091 to the CPCR.
6. Write 0x0001 to the SDCR to initialize the SDMA configuration register.
7. Write 0x18 to RFCR and TFCR for normal operation.
8. Write MRBLR with the maximum number of bytes per receive buffer. Assume 16 bytes, so MRBLR = 0x0010.
9. Initialize the RX buffer descriptor and assume the RX data buffer is at 0x00001000 in main memory. Write 0xB000 to RX_BD_Status, 0x0000 to RX_BD_Length (optional), and 0x00001000 to RX_BD_Pointer.

10. Initialize the TX buffer descriptor and assume the TX data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Write 0xB000 to TX_BD_Status, 0x0005 to TX_BD_Length, and 0x00002000 to TX_BD_Pointer.
11. Write 0xFF to the SMCE–Transparent register to clear any previous events.
12. Write 0x13 to the SMCM–Transparent register to enable all possible serial management controller interrupts.
13. Write 0x00000010 to the CIMR to allow SMC1 to generate a system interrupt. The CICR must also be initialized.
14. Write 0x3830 to the SMCMR to configure 8-bit characters, unreversed data, and normal operation (not loopback). Notice that the transmitter and receiver have not been enabled yet.
15. Write 0x3833 to the SMCMR to enable the SMCx transmitter and receiver. This additional write ensures that the TEN and REN bits are enabled last.

Note: After 5 bytes are transmitted, the TX buffer descriptor is closed and after 16 bytes are received the receive buffer is closed too. Any data received after 16 bytes causes a busy (out-of-buffers) condition since only one RX buffer descriptor is prepared.

16.11.7.15 SMC1 TRANSPARENT TSA PROGRAMMING EXAMPLE. The following is an example initialization sequence for the SMC1 transparent channel over the time-slot assigner. It is assumed that the time-slot assigner and TDM pins have already been set up to route time-slot data to the SMC1 transmitter and receiver. Refer to **Section 16.7 The Serial Interface with Time-Slot Assigner** for examples of how to configure the time-slot assigner. The transmit and receive clocks and synchronization signals are provided internally from the time-slot assigner.

1. Write RBASE and TBASE in the SMC1 parameter RAM to point to the RX buffer descriptor and TX buffer descriptor in the dual-port RAM. Assuming one RX buffer descriptor at the beginning of the dual-port RAM and one TX buffer descriptor following that RX buffer descriptor, write RBASE with 0x2000 and TBASE with 0x2008.
2. Program the CPCR to execute the **INIT TX AND RX PARAMS** command. Write 0x0091 to the CPCR.
3. Write 0x0001 to the SDCR to initialize the SDMA configuration register.
4. Write 0x18 to RFCR and TFCR for normal operation.
5. Write MRBLR with the maximum number of bytes per receive buffer. Assume 16 bytes, so MRBLR = 0x0010.
6. Initialize the RX buffer descriptor and assume the RX data buffer is at 0x00001000 in main memory. Write 0xB000 to RX_BD_Status, 0x0000 to RX_BD_Length (optional), and 0x00001000 to RX_BD_Pointer.

7. Initialize the TX buffer descriptor and assume the TX data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Write 0xB000 to TX_BD_Status, 0x0005 to TX_BD_Length, and 0x00002000 to TX_BD_Pointer.
8. Write 0xFF to the SMCE–Transparent register to clear any previous events.
9. Write 0x13 to the SMCM–Transparent register to enable all possible serial management controller interrupts.
10. Write 0x00000010 to the CIMR so that SMC1 can generate a system interrupt. The CICR must also be initialized.
11. Write 0x3830 to the SMCMR–Transparent to configure 8-bit characters, unreversed data, and normal operation (not loopback). Notice that the transmitter and receiver are not enabled yet.
12. Write 0x3833 to the SMCMR–Transparent to enable the SMC1 transmitter and receiver. This additional write ensures that the TEN and REN bits are enabled last.

16.11.7.16 HANDLING INTERRUPTS IN THE SMCx. Follow these steps to handle an interrupt in the serial management controller:

1. Once an interrupt occurs, read the SMCE register to discover the cause of the interrupts. The SMCE bits are usually cleared at this time.
2. Process the TX buffer descriptor to reuse it if the TX bit is set in the SMCE–Transparent register. Extract data from the RX buffer descriptor if the RX bit is set in the SMCE–Transparent register. To transmit another buffer, simply set the R bit in the RX buffer descriptor.
3. Clear the SMC1 bit in the CISR.
4. Execute the **rfi** instruction.

16.11.8 The SMCx in GCI Mode

The serial management controllers can be used to control the circuit interface and monitor channels of the general circuit interface (GCI) frame. When using the SCIT configuration of a general circuit interface, one serial management controller can handle SCIT channel 0, and the other serial management controller can handle SCIT channel 1. The main features of a serial management controller in GCI mode are as follows:

- Each SMCx channel supports the circuit interface and monitor channels of the GCI (IOM-2) in ISDN applications
- Two serial management controllers support the two sets of circuit interface and monitor channels in SCIT channels 0 and 1
- Full-duplex operation
- Local loopback and echo capability for testing

To use the SMCx GCI channels properly, the time-slot assigner in the serial interface must be configured to route the monitor and circuit interface channels to the serial management controller you prefer. Refer to **Section 16.7 The Serial Interface with Time-Slot Assigner** for more details on how to program this configuration. A serial management controller in GCI mode is also referred to as the SMCx GCI controller.

16.11.8.0.1 SMCx GCI Monitor Channel Transmission Process. The monitor channel 0 is used to exchange data with a layer 1 device (reading and writing internal registers and transferring the S and Q bits). Monitor channel 1 is used for programming and controlling voice/data modules, such as CODECs. The core writes the data byte into the transmit (TX) buffer descriptor. The serial management controller transmits the data on the monitor channel and handles the A and E control bits according to the GCI monitor channel protocol. You can issue the **TIMEOUT** command to solve deadlocks when errors in the A and E bit occur on the data line.

16.11.8.0.2 SMCx GCI Monitor Channel Reception Process. The serial management controller receives data and handles the A and E control bits according to the GCI monitor channel protocol. When the communication processor module stores a received data byte in the SMCx receive (RX) buffer descriptor, a maskable interrupt is generated. You can issue the **TRANSMIT ABORT REQUEST** command and the MPC823 transmits an abort request on the E bit.

16.11.8.1 HANDLING THE SMCx CIRCUIT INTERFACE CHANNEL. The circuit interface channel is used to control the layer 1 device. The layer 2 device in the TE sends commands and receives indication to or from the upstream layer 1 device via circuit interface channel 0. In the SCIT configuration, circuit interface channel 1 is used to convey real-time status information between the layer 2 device and nonlayer 1 peripheral devices (CODECs).

16.11.8.1.1 SMCx GCI Circuit Interface Channel Transmission Process. The core writes the data byte into the circuit interface TX buffer descriptor and the serial management controller transmits the data continuously on the circuit interface channel to the physical layer device.

16.11.8.1.2 SMCx GCI Circuit Interface Channel Reception Process. The SMCx receiver continuously monitors the circuit interface channel and when it recognizes a change in the data and this value is received in two successive frames, it is interpreted as valid data. This is referred to as the double last-look method. The received data byte is stored by the communication processor module in the circuit interface RX buffer descriptor and a maskable interrupt is generated. If the serial management controller is configured to support SCIT channel 1, the double last-look method is not used.

16.11.8.2 SMCx GCI PARAMETER RAM MEMORY MAP. The SMCx GCI parameter RAM area begins at the same offset from each SMCx base area. The SMCx in GCI mode has a very different parameter RAM memory map than the SMCx in UART or transparent mode. In GCI mode, the general-purpose parameter RAM contains the buffer descriptors, instead of pointers, to the buffer descriptors. You can see the difference when you compare Table 16-38 with Table 16-36. The SMCx in GCI mode contains no protocol-specific parameter RAM.

Table 16-38. SMCx GCI Parameter RAM Memory Map

ADDRESS	NAME	WIDTH	DESCRIPTION
SMCx Base + 00	**M_RXBD**	Half-word	Monitor Channel RX Buffer Descriptor
SMCx Base + 02	**M_TXBD**	Half-word	Monitor Channel TX Buffer Descriptor
SMCx Base + 04	**CI_RXBD**	Half-word	Circuit Interface Channel RX Buffer Descriptor
SMCx Base + 06	**CI_TXBD**	Half-word	Circuit Interface Channel TX Buffer Descriptor
SMCx Base + 08	Temp1	Half-word	
SMCBase + 0A	Temp2	Half-word	
SMCx Base + 0C	Temp3	Half-word	
SMCx Base + 0E	Temp4	Half-word	

NOTE: You are only responsible for initializing the items in bold.
SMCx Base = (IMMR & 0xFFFF0000) + 0x3E80 (SMC1) and 0x3F80 (SMC2).

- M_RXBD—The SMCx monitor channel receive buffer descriptor is used by the communication processor module to report information about the monitor channel receive byte.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	L	**RE**	**MS**	RESERVED				DATA							

NOTE: You are only responsible for initializing the items in bold.

E—Empty

When a serial management controller uses the monitor channel protocol, it waits until the core sets this bit before acknowledging the monitor channel data.

0 = The communication processor module clears this bit to indicate that the data byte associated with this buffer descriptor is now available to the core.

1 = The core sets this bit to indicate that the data byte associated with this buffer descriptor has been read.

L—Last (EOM)

This bit is only valid when a serial management controller implements the monitor channel protocol and is set when the EOM indication is received on the E bit. When this bit is set, the data byte is invalid.

ER—Error Condition

This bit is only valid when a serial management controller implements the monitor channel protocol and is set when an error condition occurs on the monitor channel protocol. A new byte is transmitted before a serial management controller acknowledges the previous byte.

MS—Data Mismatch

This bit is only valid when a serial management controller implements the monitor channel protocol. It is set when two different consecutive bytes are received and it is cleared when the last two consecutive bytes match. A serial management controller waits for the reception of two identical consecutive bytes before writing new data to the RX buffer descriptor.

Bits 4–7—Reserved

These bits are reserved and must be set to 0.

DATA—Data

This field contains the monitor channel data byte that a serial management controller received.

- M_TXBD—The SMCx monitor channel transmit buffer descriptor is used by the communication processor module to report information about the monitor channel transmit byte.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**R**	**L**	**AR**	RESERVED					**DATA**							

NOTE: You are only responsible for initializing the items in bold.

R—Ready

0 = This bit is cleared by the communication processor module after transmission. The TX buffer descriptor is now available to the core.

1 = The core sets this bit to indicate that the data byte associated with this buffer descriptor is ready for transmission.

L—Last (EOM)

This bit is only valid when a serial management controller implements the monitor channel protocol. When it is set, a serial management controller first transmits the buffer data and then transmits the EOM indication on the E bit.

AR—Abort Request

This bit is only valid when a serial management controller uses the monitor channel protocol and it is set by a serial management controller when an abort request is received on the A bit. The SMCx transmitter transmits the EOM on the E bit after an abort request is received.

Bits 3–7—Reserved

These bits are reserved and must be set to 0.

DATA—Data Field

This field contains the data to be transmitted by a serial management controller on the monitor channel.

- C/I_RXBD—The SMCx circuit interface channel receive (RX) buffer descriptor is used by the communication processor module to report information about the circuit interface channel receive byte.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	E	RES							C/I DATA						RESERVED	

E—Empty

0 = The communication processor module clears this bit to indicate that the data byte associated with this buffer descriptor is now available to the core.

1 = The core set this bit to indicate that the data byte associated with this buffer descriptor has been read.

Note: Additional data received is discarded until the E bit is set.

Bits 1–7 and 14–15—Reserved

These bits are reserved and must be set to 0.

C/I DATA—Command/Indication Data Bits

This field represents a 4-bit data field for circuit interface channel 0 and a 6-bit data field for circuit interface channel 1. It contains the data received from the circuit interface channel. For circuit interface channel 0, bits 10-13 contain the 4-bit data field and bits 8 and 9 are always written with zeros. For circuit interface channel 1, bits 8-13 contain the 6-bit data field.

- C/I_TXBD—The SMCx circuit interface channel transmit (TX) buffer descriptor is used by the communication processor module to report information about the circuit interface channel transmit byte.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	R	RESERVED							C/I DATA						RESERVED	

R—Ready

0 = The communication processor module clears this bit after transmission to indicate that the buffer descriptor is now available to the core.

1 = The core sets this bit to indicate that the data associated with this buffer descriptor is ready for transmission.

Bits 1–7 and 14–15—Reserved

These bits are reserved and must be set to 0.

C/I DATA—Command/Indication Data Bits

This field represents a 4-bit data field for circuit interface channel 0 and a 6-bit data field for circuit interface channel 1. It contains the data to be transmitted onto the circuit interface channel. For circuit interface channel 0, bits 10-13 contain the 4-bit data field and bits 8 and 9 are always written with zeros. For circuit interface channel 1, bits 8-13 contain the 6-bit data field.

- TEMP1–4—These bits are used internally by the RISC microcontroller.

16.11.8.3 SMCx GCI COMMANDS. The following commands are issued to the CPM command register.

- **INIT TX AND RX PARAMS—**This command initializes the transmit and receive parameters in the parameter RAM to their reset state and it is especially useful when switching protocols on a given serial channel.
- **TRANSMIT ABORT REQUEST**—This receiver command can be issued when the MPC823 implements the monitor channel protocol. When it is issued, the MPC823 sends an abort request on the A bit.
- **TIMEOUT**—This transmitter command can be issued when the MPC823 implements the monitor channel protocol and it is usually issued because the device is not responding or A bit errors are detected. The MPC823 sends an abort request on the E bit at the time this command is issued.

16.11.8.4 SMCx GCI MODE REGISTER. When a serial management controller is in GCI mode, the 16-bit, memory-mapped, read/write SMCx mode register is referred to as the SMCx GCI mode (SMCM–GCI) register. The functions of bits 8–15 are common to each SMCx protocol, but bits 0–7 vary according to the protocol selected by the SM bits.

SMCM–GCI

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RES	CLEN				ME	RES	C#	RES		SM		DM		TEN	REN
RESET	0	0				0	0	0	0		0		0		0	0
R/W	R/W	R/W				R/W	R/W	R/W	R/W		R/W		R/W		R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA82 (SMCMR1), 0xA92 (SMCMR2)															

Bits 0, 6, and 8–9—Reserved

These bits are reserved and must be set to 0.

CLEN—Character Length

This field is used to define the total number of bits in the circuit interface and monitor channels of the SCIT channels 0 or 1. CLEN ranges from 0 to 15 and specifies values from 1 to 16 bits. CLEN must be written with 13 for the SCIT channel 0 or GCI (8 data bits, plus A and E bits, plus 4 circuit interface bits = 14 bits) or with 15 for the SCIT channel 1 (8 data, bits, plus A and E bits, plus 6 circuit interface bits = 16 bits).

ME—Monitor Enable

0 = The serial management controller does not support the monitor channel.
1 = The serial management controller supports the monitor channel with the monitor channel protocol.

C#—SCIT Channel Number

0 = SCIT channel 0.
1 = SCIT channel 1. Required for Siemens ARCOFI and SGS S/T chips.

SM—SMCx Mode

00 = GCI or SCIT mode. Required for SMCx GCI or SCIT operation.
01 = Reserved.
10 = UART mode.
11 = Totally transparent mode.

DM—Diagnostic Mode

00 = Normal mode.
01 = Local loopback mode.
10 = Echo mode.
11 = Reserved.

TEN—SMCx Transmit Enable

0 = SMCx transmitter disabled.
1 = SMCx transmitter enabled.

REN—SMCx Receive Enable

0 = SMCx receiver disabled.
1 = SMCx receiver enabled.

16.11.8.5 SMCx GCI EVENT REGISTER. When a serial management controller is in GCI mode, the 8-bit memory-mapped SMCx event register is referred to as the SMCx GCI event (SMCE–GCI) register. It is used to generate interrupts and report events recognized by the SMCx channel. When an event is recognized, a serial management controller sets the corresponding bit in this register. Interrupts generated by this register can be masked in the SMCM–GCI register. A bit is cleared by writing a 1 (writing a zero has no effect) more than one bit can be cleared at a time. All unmasked bits must be cleared before the communication processor module clears the internal interrupt request to the CPM interrupt controller. This register is cleared by reset and can be read at any time.

SMCE–GCI

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED				CTXB	CRXB	MTXB	MRXB
RESET	0				0	0	0	0
R/W	R/W				R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA86 (SMC1), 0xA96 (SMC2)							

Bits 0–3—Reserved

These bits are reserved and must be set to 0.

CTXB—Circuit Interface Channel Buffer Transmitted

This bit indicates that the circuit interface transmit buffer is now empty.

CRXB—Circuit Interface Channel Buffer Received

This bit indicates when the circuit interface receive buffer is full.

MTXB—Monitor Channel Buffer Transmitted

This bit indicates that the monitor transmit buffer is now empty.

MRXB—Monitor Channel Buffer Received

This bit indicates when the monitor receive buffer is full.

16.11.8.6 SMCx GCI MASK REGISTER. When a serial management controller is in GCI mode, the 8-bit, memory-mapped, read/write SMCx mask register is referred to as the SMCx GCI mask (SMCM–GCI) register. It has the same bit format as the SMCE–GCI register. If a bit in this register is a 1, the corresponding interrupt in the SMCE–GCI is enabled. If the bit is zero, the corresponding interrupt in the SMCE–GCI is masked.

SMCM–GCI

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED				CTXB	CRXB	MTXB	MRXB
RESET	0				0	0	0	0
R/W	R/W				R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xA8A (SMC1), 0xA9A(SMC2)							

16.12 THE SERIAL PERIPHERAL INTERFACE

The serial peripheral interface (SPI) allows the MPC823 to exchange data between other MPC8xx microprocessors, the MC68328, MC68360, and MC68302 embedded microprocessors, as well as the MC68HC11 and MC68HC05 microcontroller families and a variety of peripheral devices.

The serial peripheral interface is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock and slave select). The SPI block consists of transmitter and receiver sections, an independent baud rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is transmitted and received simultaneously.

Because the SPI receiver and transmitter are double-buffered, as illustrated in the block diagram below, the effective FIFO size is 2 characters. You can program the MPC823 serial peripheral interface to shift out the most- or least-significant bit first. When the serial peripheral interface is not enabled in the SPMODE register, it consumes very little power.

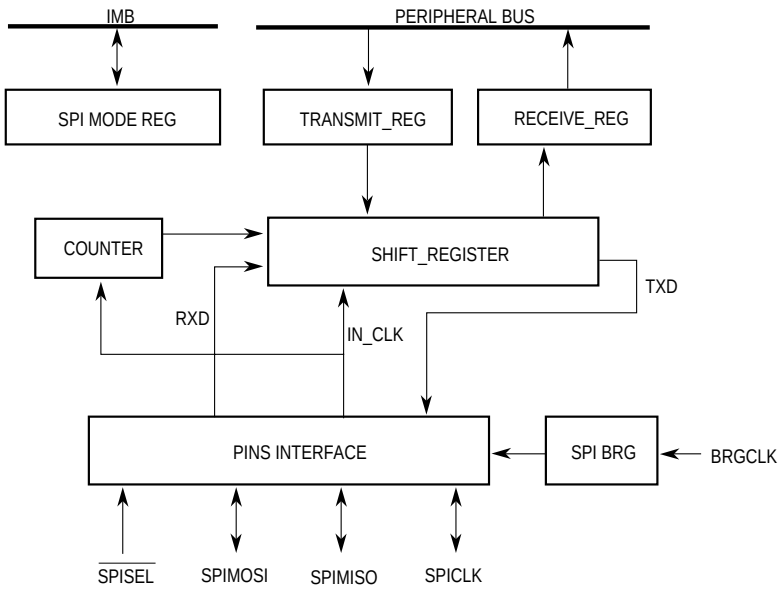


Figure 16-120. SPI Block Diagram

16.12.1 Features

The following is a list of the serial peripheral interface's main features:

- Four-Wire Interface (SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$)
- Full-Duplex Operation
- Works with Data Characters from 4 to 16 Bits Long
- Supports Back-to-Back Character Transmission and Reception
- Master or Slave SPI Modes Supported
- Multimaster Environment Support
- Continuous Transfer Mode for Autoscanning Peripherals
- Supports Maximum Clock Rates of 6.25MHz in Master Mode and 12.5MHz in Slave Mode, Assuming a 25MHz System Clock is Used
- Independent Programmable Baud Rate Generator
- Programmable Clock Phase and Polarity
- Open-Drain Output Pins Support Multimaster Configuration
- Local Loopback Capability for Testing

16.12.2 SPI Clocking and Pin Functions

You can configure the serial peripheral interface as a master for the serial channel or as a slave. You can also use it in a multimaster environment. When the serial peripheral interface is a master, use the SPI baud rate generator to generate the SPI transmit and receive clocks. It takes its input from the BRGCLK, which is generated in the clock synthesizer of the MPC823, specifically for the SPI baud rate generator and the other three baud rate generators in the communication processor module.

The SPIMISO pin is an input in master mode and an output in slave mode. It follows then, that the SPIMOSI pin is an output in master mode and an input in slave mode. The SPIMOSI and SPIMISO pins change functionality between master and slave mode to support a multimaster configuration that allows communication from one serial peripheral interface to another with the same hardware configuration.

When the serial peripheral interface is in master mode, 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. You can implement this by using one of the MPC823 general-purpose I/O pins. The $\overline{\text{SPISEL}}$ pin must not be asserted while the serial peripheral interface is in master mode or an error will occur.

When the serial peripheral interface is in slave mode, 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 MPC823 is the enable input to the SPI slave. When the serial peripheral interface is operating in a multimaster environment, the $\overline{\text{SPISEL}}$ pin is still an input and is used to detect an error condition when more than one master is operating.

Using the fields in the SPI mode register, you can select any of the four combinations of the gated SPICLK phase and polarity. The SPI pins can also be configured as open-drain pins to support a multimaster configuration in which the same SPI pin is driven by the MPC823 or an external SPI device.

16.12.3 The SPI Transmission and Reception Process

When the serial peripheral interface is in master mode, it transmits a message to the peripheral or slave, which sends back an immediate reply. When the MPC823 has more than one slave, it can use the general-purpose parallel I/O pins to selectively enable different slaves. To start the data exchange process, the core writes the data to be transmitted into a data buffer, configures a transmit (TX) buffer descriptor with its R bit set, and configures one or more receive (RX) buffer descriptors. The core then sets the STR bit in the SPCOM register to start transmitting data, which starts when the SDMA channel loads the transmit FIFO with data.

The serial peripheral interface 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 serial peripheral interface shifts received data in from the SPIMISO pin. This received data is written into a receive buffer using the next available RX buffer descriptor. The serial peripheral interface continues transmitting and receiving characters until the transmit buffer has been completely transmitted or an error has occurred. The communication processor module then clears the R and E bits in the TX and RX buffer descriptors and may issue a maskable interrupt to the CPM interrupt controller.

When multiple TX buffer descriptors are ready to be transmitted, the L bit in the TX buffer descriptor determines whether or not the serial peripheral interface must continue transmitting without waiting for the STR bit to be set again. If the L bit is cleared, the data from the next TX buffer descriptor begins transmitting after data from the first TX buffer descriptor is transmitted. If the L bit is set, transmission stops after data from this TX buffer descriptor has finished transmitting. In addition, the current RX buffer descriptor that is used to receive data is closed after transmission stops, even if the receive buffer is not full. This means that you do not need to provide receive buffers that are the same length as the transmit buffers. If the serial peripheral interface is the only master in a system, then the $\overline{\text{SPISEL}}$ pin can be used as a general-purpose I/O, and the internal $\overline{\text{SPISEL}}$ signal to the serial peripheral interface is always forced internally inactive, thus eliminating the possibility of a multimaster error.

When the serial peripheral interface is in slave mode, it receives messages from an SPI master and sends back an immediate reply. The $\overline{\text{SPISEL}}$ pin must be asserted before receive clocks are recognized and once $\overline{\text{SPISEL}}$ is asserted, the SPICLK pin becomes an input from the master to the slave. SPICLK can be any frequency from the DC to the BRGCLK/2, which is 12.5MHz for a 25MHz system.

Before the data is exchanged, the core writes the data to be transmitted into a data buffer, configures a TX buffer descriptor with its R bit set, and configures one or more RX buffer descriptors. The core then sets the STR bit in the SPCOM register to enable the serial peripheral interface so it will prepare the data for transmission and wait for the $\overline{\text{SPISEL}}$ pin to be asserted. Data is shifted out from the slave on the SPIMISO pin and shifted in through the SPIMOSI pin. A maskable interrupt is issued when a full buffer finishes receiving and transmitting or after an error occurs. Using the next RX buffer descriptor in the ring, the serial peripheral interface continues reception until it runs out of receive buffers or the $\overline{\text{SPISEL}}$ pin is negated.

Transmission continues until no more data is available or the $\overline{\text{SPISEL}}$ pin is negated. If the pin is negated before all the data is transmitted, it stops, but the TX buffer descriptor stays open. Further transmission continues once the $\overline{\text{SPISEL}}$ pin is reasserted and SPICLK begins toggling. After the characters in the TX buffer descriptor are transmitted, the serial peripheral interface transmits ones if $\overline{\text{SPISEL}}$ is not negated.

16.12.3.1 MULTIMASTER OPERATION. The serial peripheral interface can operate in a multimaster environment in which more than one SPI device is connected to the same bus. In this configuration, the SPIMOSI, SPIMISO, and SPICLK pins of all SPIs are connected together and the $\overline{\text{SPISEL}}$ input pins are connected separately. In this environment, only one SPI device can be in master mode and all the others must be in slave mode. When the serial peripheral interface is configured as a master and its $\overline{\text{SPISEL}}$ signal goes active or low, a multimaster error will occur because more than one SPI device is a bus master. The serial peripheral interface sets the MME bit in the SPIE register and a maskable interrupt is issued to the core. It also disables SPI operation and the output drivers of the SPI pins. The core must clear the EN bit the SPMODE register before using the serial peripheral interface again. After the problems are corrected, clear the MME bit and enable the serial peripheral interface the same way you would after a reset.

Note: The maximum sustained data rate that the serial peripheral interface supports is SYSTEMCLK/50. However, the serial peripheral interface can transfer a single character at much higher rates. For instance, SYSTEMCLK/4 in master mode and SYSTEMCLK/2 in slave mode. If multiple characters are to be transmitted, you must insert gaps between them so that it will not exceed the maximum sustained data rate.

16.12.3.2 SPI PARAMETER RAM MEMORY MAP. The SPI parameter RAM area begins at the SPI base address and is used for the general SPI parameters. Notice that it is similar to the SCCx general-purpose parameter RAM. You must initialize certain parameter RAM values before the serial peripheral interface is enabled. The communication processor module initilizes the other values. Once initialized, the parameter RAM values do not usually need to be accessed by your software. They only need to be modified when there is no serial peripheral interface activity in progress.

Table 16-39. SPI Parameter RAM Memory Map

ADDRESS	NAME	WIDTH	DESCRIPTION
SPI Base + 00	**RBASE**	Half-word	RX Buffer Descriptor Base Address
SPI Base+ 02	**TBASE**	Half-word	TX Buffer Descriptor Base Address
SPI Base+ 04	**RFCR**	Byte	RX Function Code
SPI Base+ 05	**TFCR**	Byte	TX Function Code
SPI Base+ 06	**MRBLR**	Half-word	Maximum Receive Buffer Length
SPI Base+ 08	RSTATE	Word	RX Internal State
SPI Base+ 0C	RPTR	Word	RX Internal Data Pointer
SPI Base + 10	RBPTR	Half-word	RX Buffer Descriptor Pointer
SPI Base + 12	RCNT	Half-word	RX Internal Byte Count
SPI Base + 14	RTMP	Word	RX Temp
SPI Base + 18	TSTATE	Word	TX Internal State
SPI Base + 1C	TPTR	Word	TX Internal Data Pointer
SPI Base + 20	TBPTR	Half-word	TX Buffer Descriptor Pointer
SPI Base + 22	TCNT	Half-word	TX Internal Byte Count
SPI Base + 24	TTMP	Word	TX Temp

NOTE: You are only responsible for initializing the items in bold.
SPI Base = (IMMR & 0xFFFF0000) + 0x3D80.
SCCx Ethernet parameter RAM space overlaps the SPI parameter RAM space. The address range for SCCx space is 0x1000 through 0x1CA3. You need a microcode patch to run SPI and Ethernet concurrently.

- RBASE and TBASE—The dual-port RAM starts receiving and transmitting data for the RX and TX buffer descriptors in the RBASE and TBASE entries. They provide a great deal of flexibility for partitioning buffer descriptors for a serial peripheral interface. You must initialize these entries before enabling the corresponding channel. You must not configure the SPI buffer descriptor tables to overlap with the tables of the USB, SMCx, and SCCx or erratic operation will occur. RBASE and TBASE must contain a value that is divisible by eight.

- RFCR and TFCR—The receive and transmit function code register entries contain the value that you want to appear on the AT pins when the associated SDMA channel accesses memory. This register controls the byte-ordering convention used in the transfers.

RFCR

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED			BO		AT		
RESET	0			0		0		
R/W	R/W			R/W		R/W		
ADDR	SPI BASE + 0x04							

Bits 0–2—Reserved

These bits are reserved and must be set to 0.

BO—Byte Ordering

You must set these bits to select the required byte ordering of the data buffer.

00 = The DEC/Intel convention is used for byte ordering (swapped operation) and is also called little-endian byte ordering. The transmission order of bytes within a buffer word is reversed in comparison to the Motorola mode. This mode is supported only for 32-bit port size memory.

01 = PowerPC little-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the least-significant byte of the buffer double-word contains data to be transmitted earlier than the most-significant byte of the same buffer double-word.

1X = Motorola byte ordering (normal operation) is also called big-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the most-significant byte of the buffer word contains data to be transmitted earlier than the least-significant byte of the same buffer word.

AT—Address Type 1–3

These bits contain the function code value used during the SDMA channel memory access. AT0 is driven with a 1 to identify this SDMA channel access as a DMA-type access.

TFCR

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED			BO		AT		
RESET	0			0		0		
R/W	R/W			R/W		R/W		
ADDR	SPI BASE + 0x05							

Bits 0–2—Reserved

These bits are reserved and must be set to 0.

BO—Byte Ordering

You must set these bits to select the required byte ordering of the data buffer. If this bit field is modified on-the-fly, it takes effect at the beginning of the next frame or at the beginning of the next buffer descriptor.

00 = The DEC/Intel convention is used for byte ordering (swapped operation) and is also called little-endian byte ordering. The transmission order of bytes within a buffer word is reversed in comparison to the Motorola mode. This mode is supported only for 32-bit port size memory.
01 = PowerPC little-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the least-significant byte of the buffer double-word contains data to be transmitted earlier than the most-significant byte of the same buffer double-word.
1X = Motorola byte ordering (normal operation) is also called big-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the most-significant byte of the buffer word contains data to be transmitted earlier than the least-significant byte of the same buffer word.

AT—Address Type 1–3

These bits contain the function code value used during the SDMA channel memory access. AT0 is driven with a 1 to identify this SDMA channel access as a DMA-type access.

- MRBLR—The serial peripheral interface has one maximum receive buffer length entry to define its receive buffer length and it defines the maximum number of bytes that the MPC823 writes to a receive buffer on the serial peripheral interface before moving to the next buffer. The MPC823 can write fewer bytes to the buffer than the MRBLR value if an error or end-of-frame occurs, but it never writes more bytes than the MRBLR value. Buffers you supply for the MPC823 to use must always be at least as long as MRBLR. The transmit buffers for a serial peripheral interface are not affected by the value you program into MRBLR and they can have different lengths, as needed. You can choose the number of bytes to be transmitted by programming the DATA LENGTH field in the TX buffer descriptor.

Note: The MRBLR is not intended to be dynamically changed while a serial management controller 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 the receive buffer length can be successfully achieved. This occurs when the communication processor module transfers control to the next RX buffer descriptor in the table. Thus, a change to MRBLR does not have an immediate effect. To guarantee that the change occurs on a particular RX buffer descriptor, you must only change the MRBLR while the SMCx receiver is disabled. The value of MRBLR must be greater than zero and it must be even if the character length of the data is greater than 8 bits.

- RRBPTR—The RX buffer descriptor pointer entry for each SPI channel points to the next buffer descriptor that the receiver transfers data to when it is idle or to the current buffer descriptor during frame processing. After a reset or when the end of the buffer descriptor table is reached, the communication processor module initializes this pointer to the value programmed in the RBASE entry. Although in most applications you must not write RBPTR, it can be modified when the receiver is disabled or when you are sure that no receive buffer is currently in use.
- TBPTR—The TX buffer descriptor pointer entry for each SPI channel points to the next buffer descriptor that the transmitter transfers data from when it is idle or to the current buffer descriptor during frame transmission. After a reset or when the end of buffer descriptor table is reached, the communication processor module initializes this pointer to the value programmed in the TBASE entry. Although in most applications you must not write TBPTR, it can be modified when the transmitter is disabled or when you are sure that no transmit buffer is currently in use.
- Other General Parameters—For normal operation, you do not need to access these parameters. They are only listed here because they provide helpful information for debugging purposes. Additional parameters are listed in Table 16-39. RPTR and TPTR are updated by the SDMA channels to show the next address in the buffer to be accessed. TCNT is a down-count value initialized with the DATA LENGTH field of the TX buffer descriptor and decremented with every byte read by the SDMA channels. The RCNT is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write. The RSTATE, TSTATE, RTMP, TMP, and reserved areas can only be used by the RISC microcontroller.

Note: To extract data from a partially full buffer, issue the **CLOSE RX BD** command.

16.12.3.3 SPI COMMANDS. The following transmit and receive commands are issued to the CPM command register (CPCR).

- **INIT TX PARAMETERS**—This command initializes all transmit parameters in the serial channel parameter RAM to their reset state and must only be issued when the transmitter is disabled. The **INIT TX AND RX PARAMS** command can also be used to reset the transmit and receive parameters.
- **CLOSE RX BD**—This command is used to force the SPI controller to close the current RX buffer descriptor if it is currently being used and to use the next buffer descriptor for any subsequently received data. If the SPI controller is not in the process of receiving data, no action is taken by this command.
- **INIT RX PARAMETERS**—This command initializes all the receive parameters in this serial channel parameter RAM to their reset state and must only be issued when the receiver is disabled. The **INIT TX AND RX PARAMS** command can also be used to reset the receive and transmit parameters.

16.12.3.4 SPI BUFFER DESCRIPTOR RING. The data associated with the serial peripheral interface is stored in buffers, which are referenced by buffer descriptors organized in a buffer descriptor ring located in the dual-port RAM. This ring has the same basic configuration as the SCCx, SMCx, USB, and I^2C controllers.

The buffer descriptor ring forms a circular queue that helps you organize the buffers you want to transmit or receive. Using the buffer descriptors, the communication processor module confirms reception and transmission or indicates error conditions so that the processor knows the buffers have been serviced. The actual buffers can reside in either external or internal memory and the data buffers can reside in the parameter area of another unused controller if it is not enabled.

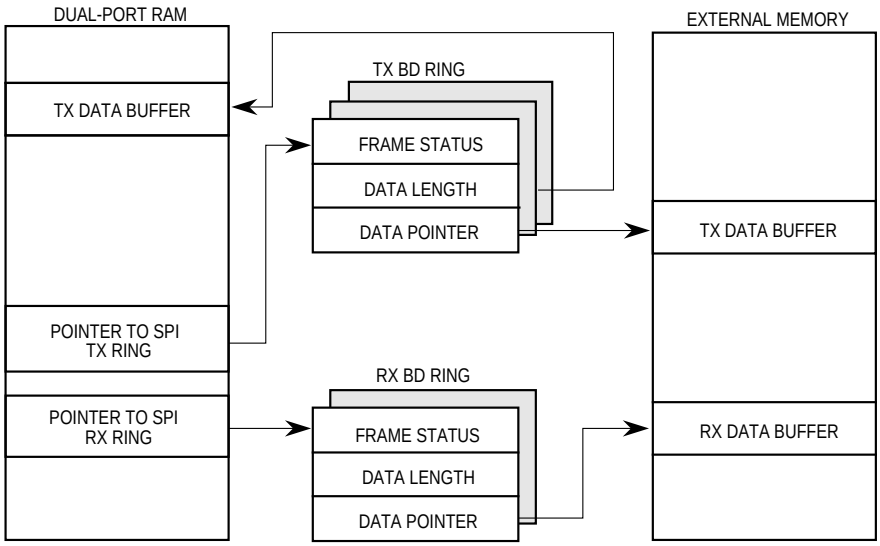


Figure 16-121. SPI Memory Format

16.12.4 Programming the Serial Peripheral Interface

16.12.4.1 SPI MODE REGISTER. The read/write SPI mode (SPMODE) register controls both the serial peripheral interface operation mode and clock source. Table 16-2 contains more information on commands that can be used with this register.

SPMODE

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RES	LOOP	CI	CP	DIV16	REV	M/S	EN	LEN				PM			
RESET	0	0	0	0	0	0	0	0	0				0			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W				R/W			
ADDR	(IMMR & 0xFFFF0000) + 0xAA0															

Bit 0—Reserved

This bit is reserved and must be set to 0.

LOOP—Loop Mode

When set, this bit selects the local loopback operation. The transmitter output is internally connected to the receiver input. The receiver and transmitter operate normally, except that the externally received data is ignored.

0 = Normal operation.
1 = The serial peripheral interface is in loopback mode.

CI—Clock Invert

This bit inverts the SPI clock polarity. See Figure 16-122 and Figure 16-123 for details.

0 = The inactive state of SPICLK is low.
1 = The inactive state of SPICLK is high.

CP—Clock Phase

This bit selects one of two fundamentally different transfer formats. See Figure 16-122 and Figure 16-123 for details.

0 = SPICLK starts toggling at the middle of the data transfer.
1 = SPICLK starts toggling at the beginning of the data transfer.

DIV16—Divide by 16

This bit selects the clock source for the SPI baud rate generator when configured as an SPI master. In slave mode, the clock source is the SPICLK pin.

0 = Use the BRGCLK as the input to the SPI baud rate generator.
1 = Use the BRGCLK/16 as the input to the SPI baud rate generator.

REV—Reverse Data

This bit determines the receive and transmit character bit order.

0 = Reverse data. Least-significant bit of the character transmitted and received first.
1 = Normal operation. Most-significant bit of the character transmitted and received first.

M/S—Master/Slave

This bit configures the serial peripheral interface to operate as a master or slave.

0 = The serial peripheral interface is a slave.
1 = The serial peripheral interface is a master.

EN—Enable SPI

This bit enables serial peripheral interface operation. Configure the SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$ pins to connect to the serial peripheral interface, as described in **Section 16.14.6 The Port B Registers**. When the EN bit is cleared, the serial peripheral interface is in a reset state and consumes minimal power, which means the SPI baud rate generator is not functioning and the input clock is disabled.

0 = The serial peripheral interface is disabled.
1 = The serial peripheral interface is enabled.

Note: You must not modify other bits of the SPMODE register when the EN bit is set.

LEN—Character Length

This field specifies the number of bits in a character. These values must be between 4 and 16 bits. If you program a value less than 4 bits, erratic behavior will occur. If the value of LEN is less than or equal to a byte, there will be LEN number of valid bits in every byte (8 bits) in memory. On the other hand, if the value of LEN is greater than a byte, there is LEN number of valid bits in every half-word (16 bits) in memory.

0011 = 4-bit character length.
0100 = 5-bit character length.
0101 = 6-bit character length.
0110 = 7-bit character length.
0111 = 8-bit character length.
•
•
•
1111 = 16-bit character length.

PM—Prescale Modulus Select

This field specifies the divide ratio of the prescale divider in the SPI clock generator. The BRGCLK is divided by 4 * ([PM0–PM3] + 1), thus giving a clock divide ratio of 4 to 64. The clock has a 50% duty cycle.

16.12.4.1.1 SPI Examples With Different LEN Values. The programming examples below illustrate the effect of the LEN field and the REV bit in the SPMODE register on output from the SPI controller. They illustrate the master mode output from the SPI controller as the LEN varies. To help map the output process, make *g* through *v* the binary symbols, use *x* to indicate a deleted bit, use __ to indicate original byte boundaries, and use _ to indicate original nibble (4-bit) boundaries.

The initial pattern for all examples is `ghij_klmn__opqr_stuv`.

Example 1 `LEN = 0x4` `(Data Size = 5)`

Data Selected:	`xxxj_klmn_xxxr_stuv`
Data Transmitted for REV=0:	`nmlk_j__vuts_r`
Data Transmitted for REV=1:	`j_nmlk__r_stuv`

Example 2 `LEN = 0x7` `(Data Size = 8)`

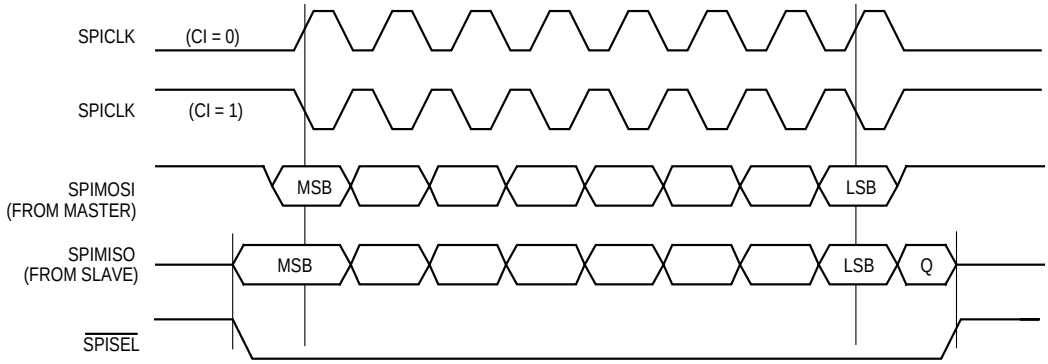
Data Selected:	`ghij_klmn_opqr_stuv`
Data Transmitted for REV=0:	`nmlk_jihg__vuts_rqpo`
Data Transmitted for REV=1:	`ghij_klmn__opqr_stuv`

Example 3 `LEN = 0xc` `(Data Size = 13)`

Data Selected:	`ghij_klmn_xxxr_stuv`
Data Transmitted for REV=0:	`nmlk_jihg__vuts_r`
Data Transmitted for REV=1:	`r_stuv__ghij_klmn`

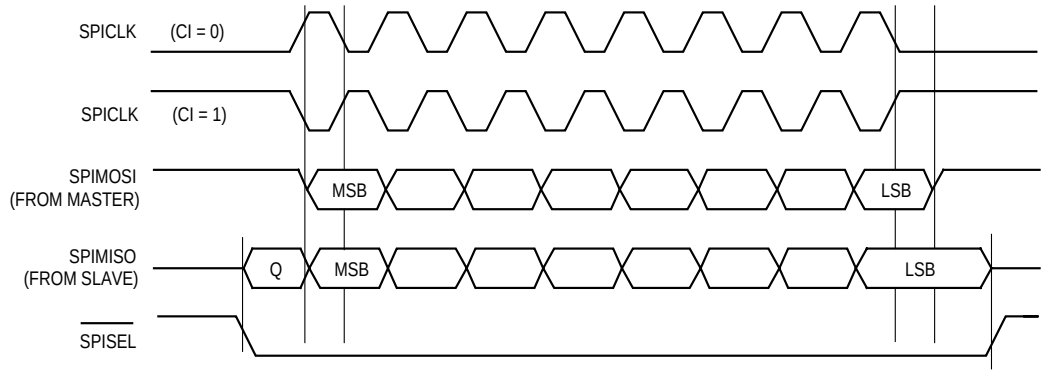
Example 4 `LEN = 0xf` `(Data Size = 16)`

Data Selected:	`ghij_klmn_opqr_stuv`
Data Transmitted for REV=0:	`nmlk_jihg__vuts_rqpo`
Data Transmitted for REV=1:	`opqr_stuv__ghij_klmn`



NOTE: Q = Undefined Signal

Figure 16-122. SPI Transfer Format If CP is Set to 0



NOTE: Q = Undefined Signal

Figure 16-123. SPI Transfer Format If CP is Set to 1

16.12.4.1.2 SPI Receive Buffer Descriptor. Using receive (RX) buffer descriptors, the communication processor module reports information about each buffer of received data, closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer once the current buffer is full. Additionally, it closes the buffer when the serial peripheral interface is configured as a slave and the $\overline{\text{SPISEL}}$ pin goes inactive, thus indicating that the reception process has stopped.

The first word of the RX buffer descriptor contains status and control bits that you prepare before reception and then the communication processor module sets them after the buffer is closed. The second word contains the data length (in bytes) that is received and the third and fourth words contain a pointer that always points to the beginning of the received data buffer. You must configure the RX buffer descriptor bits before the serial peripheral interface is enabled.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	RES	**W**	**I**	L	RES	**CM**	RESERVED							OV	ME
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: You are only responsible for initializing the items in bold.

Note: The communication processor module sets all the status bits in this buffer descriptor, but you must clear them before submitting the buffer descriptor to the communication processor module.

E—Empty

0 = The data buffer associated with this RX buffer descriptor is filled with data or stops receiving data because an error occurred. The core is free to examine or write to any fields of this RX buffer descriptor, but the communication processor module does not use this buffer descriptor as long as the E bit is zero.

1 = The data buffer associated with this buffer descriptor is empty or is currently receiving data. This RX buffer descriptor and its associated receive buffer are owned by the communication processor module. Once the E bit is set, the core must not write any fields of this RX buffer descriptor.

Bit 1, 5, and 7–13—Reserved

These bits are reserved and must be set to 0.

W—Wrap (Final Buffer Descriptor in Table)

0 = This is not the last buffer descriptor in the RX buffer descriptor table.
1 = This is the last buffer descriptor in the RX buffer descriptor table. After this buffer is used, the communication processor module receives incoming data into the first buffer descriptor that RBASE points to in the table. The number of RX buffer descriptors in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.

I—Interrupt

0 No interrupt is generated after this buffer is filled.
1 The RXB bit in the SPIE register is set when this buffer is completely filled by the communication processor module, indicating the need for the core to process the buffer. The RXB bit can cause an interrupt if it is enabled.

CM—Continuous Mode

This bit is valid only when the serial peripheral interface is in master mode. In slave mode, it must be written as a zero.

0 = Normal operation.
1 = The E bit is not cleared by the communication processor module after this buffer descriptor is closed, thus allowing the associated data buffer to be automatically overwritten next time the communication processor module accesses this buffer descriptor. This allows continuous reception from an SPI slave into one buffer for autoscanning of a serial A/D peripheral with no core overhead.

L—Last

This bit is set by the serial peripheral interface when the buffer is closed because the $\overline{\text{SPISEL}}$ pin was negated. This only occurs when the serial peripheral interface is in slave mode. Otherwise, the ME bit is set. The serial peripheral interface writes this bit after the received data is placed into the associated data buffer.

0 = This buffer does not contain the last character of the message.
1 = This buffer contains the last character of the message.

OV—Overrun

This bit indicates that a receiver overrun has occurred during reception. This can only occur when the serial peripheral interface is in slave mode. The serial peripheral interface writes this bit after the received data is placed into the associated data buffer.

ME—Multimaster Error

This bit indicates that this buffer is closed because the $\overline{\text{SPISEL}}$ pin was asserted when the serial peripheral interface was in master mode. This indicates a synchronization problem between multiple masters on the SPI bus. The serial peripheral interface writes this bit after the received data is placed into the associated data buffer.

DATA LENGTH

This field represents the number of octets that the communication processor module writes into this buffer descriptor data buffer. The communication processor module writes it once as the buffer descriptor is closed. The serial peripheral interface writes these bits after the received data is placed into the associated data buffer. The actual amount of memory allocated for this buffer must be greater than or equal to the MRBLR entry.

RX DATA BUFFER POINTER

This field always points to the first location of the associated data buffer. It must be even and it can reside in internal or external memory. The serial peripheral interface writes these bits after the received data is placed into the associated data buffer.

16.12.4.1.3 SPI Transmit Buffer Descriptor. Data to be transmitted by the serial peripheral interface is sent to the communication processor module by arranging it in buffers referenced by the transmit (TX) buffer descriptor ring. The first word of the TX buffer descriptor contains status and control bits. You must prepare the following bits before transmitting data.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**R**	RES	**W**	**I**	**L**	RES	**CM**	RESERVED							UN	ME
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: You are only responsible for initializing the items in bold.

Note: The communication processor module sets all the status bits in this buffer descriptor, but you must clear them before submitting the buffer descriptor to the communication processor module.

R—Ready

0 = The data buffer associated with this buffer descriptor is not ready for transmission and you are free to manipulate it or its associated data buffer. The communication processor module clears this bit after the buffer is transmitted or after an error occurs.

1 = The data buffer, which you prepare for transmission, is not transmitted yet or is currently being transmitted. You cannot write any fields of this buffer descriptor once this bit is set.

Bits 1, 5, and 7–13—Reserved

These bits are reserved and must be set to 0.

W—Wrap (Final Buffer Descriptor in Table)

0 = This is not the last buffer descriptor in the TX buffer descriptor table.
1 = This is the last buffer descriptor in the TX buffer descriptor table. After this buffer is used, the communication processor module receives incoming data into the first buffer descriptor that TBASE points to in the table. The number of TX buffer descriptors in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer is serviced.
1 = The TXB or TXE bit in the SPI event register is set when this buffer is serviced. TXB and TXE can cause interrupts if they are enabled.

L—Last

0 = This buffer does not contain the last character of the message.
1 = This buffer contains the last character of the message.

CM—Continuous Mode

This bit is only valid when the serial peripheral interface is in master mode. In slave mode, it must be written as a zero.

0 = Normal operation.
1 = The R bit is not cleared by the communication processor module after this buffer descriptor is closed, thus allowing the associated data buffer to be automatically retransmitted next time the communication processor module accesses this buffer descriptor.

UN—Underrun

This bit indicates that the serial peripheral interface has encountered a transmitter underrun condition while transmitting a data buffer. This error condition is only valid when the serial peripheral interface is in slave mode. The serial peripheral interface writes this bit after it finishes transmitting the associated data buffer.

ME—Multimaster Error

This bit indicates that this buffer is closed because the $\overline{\text{SPISEL}}$ pin was asserted when the serial peripheral interface was in master mode. This indicates a synchronization problem between multiple masters on the SPI bus. The serial peripheral interface writes this bit after it finishes transmitting the associated data buffer.

DATA LENGTH

This field indicates the number of octets that the communication processor module must transmit from this buffer descriptor data buffer. However, it is never modified by the communication processor module. Normally, this value is greater than zero, but if the number of data bits in the character is greater than 8, then DATA LENGTH must be even. For example, to transmit three characters of 8-bit data, the DATA LENGTH field must be initialized to 3. However, to transmit three characters of 9-bit data the DATA LENGTH field must be initialized to 6, since the three 9-bit data fields occupy three half-words in memory. The serial peripheral interface writes these bits after it finishes transmitting the associated data buffer.

TX DATA BUFFER POINTER

This field always points to the first location of the associated data buffer. They can be even or odd, unless the number of actual data bits in the character is greater than 8 bits, in which case the transmit buffer pointer must be even. The buffer can reside in internal or external memory. The serial peripheral interface writes these bits after it finishes transmitting the associated data buffer.

16.12.4.2 SPI COMMAND REGISTER. The 8-bit read/write SPI command (SPCOM) register is used to start serial peripheral interface operation.

SPCOM

BIT	0	1	2	3	4	5	6	7
FIELD	STR	RESERVED						
RESET	0	0						
R/W	R/W	R/W						
ADDR	(IMMR & 0xFFFF0000) + 0xAAD							

STR—Start Transmit

When the serial peripheral interface is configured as a master, setting this bit to 1 causes the serial peripheral interface to start transmitting and receiving data to and from the transmit/receive buffers if they are ready. When the serial peripheral interface is in slave mode, setting the STR bit to 1 when the serial peripheral interface is idle causes the serial peripheral interface to load the transmit data register from the SPI transmit buffer and start transmission as soon as the next SPI input clocks and select signal are received. This bit is automatically cleared after one system clock cycle.

Bits 1–7—Reserved.

These bits are reserved and must be set to 0.

16.12.4.3 SPI EVENT REGISTER. The 8-bit memory-mapped SPI event (SPIE) register is used to generate interrupts and report events recognized by the serial peripheral interface. When an event is recognized, the serial peripheral interface sets the corresponding bit in this register. Interrupts generated by this register can be masked in the SPIM register. A bit is cleared by writing a 1 (writing a zero has no effect) and more than one bit can be cleared at a time. However, all unmasked bits must be cleared before the communication processor module clears the internal interrupt request. This register is cleared by reset and can be read at any time.

SPIE

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED		MME	TXE	RES	BSY	TXB	RXB
RESET	0		0	0	0	0	0	0
R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xAA6							

Bits 0–1 and 4—Reserved

These bits are reserved and must be set to 0.

MME—Multi-Master Error

This bit indicates that the serial peripheral interface has discovered that the $\overline{\text{SPISEL}}$ pin was asserted externally while the serial peripheral interface was in master mode.

TXE—TX Error

This bit indicates that an error has occurred during transmission.

BSY—Busy Condition

This bit indicates that received data has been discarded due to a lack of buffers. This bit is set after the first character is received for which there is no receive buffer.

TXB—TX Buffer

This bit indicates that a buffer has been transmitted. It is set once the transmit data of the last character in the buffer is written to the transmit FIFO. You must wait two character times to be sure that the data is completely sent over the transmit pin.

RXB—RX Buffer

This bit indicates that a buffer has been received. It set after the last character is written to the receive buffer and the RX buffer descriptor is closed.

16.12.4.4 SPI MASK REGISTER. The 8-bit read/write SPI mask (SPIM) register has the same bit formats as the SPIE register. If a bit in the SPIM is 1, the corresponding interrupt in the SPIE register is enabled. If the bit is zero, the corresponding interrupt in the SPIE register is masked. This register is cleared by reset.

SPIM

BIT	0	1	2	3	4	5	6	7
FIELD	RESERVED		MIME	TXE	RES	BSY	TXB	RXB
RESET	0		0	0	0	0	0	0
R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W
ADDR	(IMMR & 0xFFFF0000) + 0xAAA							

16.12.5 SPI Master Programming Example

The following is an example initialization sequence using the serial peripheral interface in master mode at high speed.

1. Configure the port B pins to enable the SPIMOSI, SPIMISO, and SPICLK pins. Write PBPAR and PBDIR bits 30, 29, and 28 with ones and then PBODR bits 30, 29, and 28 with zeros.

Note: For multimaster operation, enable the $\overline{\text{SPISEL}}$ pin to internally connect to the serial peripheral interface.

2. Configure a parallel I/O pin to operate as the serial peripheral interface select pin if needed. If PB16 is chosen, write PBODR bit 16 with a zero, PBDIR bit 16 with a one, and PBPAR bit 16 with a zero. Then write PBDAT bit 16 with a zero to constantly assert the select pin.
3. Write RBASE and TBASE in the SPI parameter RAM to point to the RX buffer descriptor and TX buffer descriptor in the dual-port RAM. Assuming one RX buffer descriptor is at the beginning of the dual-port RAM and one TX buffer descriptor follows that RX buffer descriptor, write RBASE with 0x2000 and TBASE with 0x2008.
4. Program the CPM command register (CPCR) to execute the **INIT RX AND TX PARAMS** command. Write 0x0051 to the CPCR.
5. Write 0x0001 to the SDCR to initialize the SDMA configuration register.
6. Write 0x18 to RFCR and TFCR for normal operation.
7. Write MRBLR with the maximum number of bytes per receive buffer. In this case, assume 16 bytes, so MRBLR = 0x0010.
8. Initialize the RX buffer descriptor and assume the RX data buffer is at 0x00001000 in main memory. Write 0xB000 to RX_BD_Status, 0x0000 to RX_BD_Length (optional), and 0x00001000 to RX_BD_Pointer.

9. Initialize the TX buffer descriptor and assume the TX data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Write 0xB800 to TX_BD_Status, 0x0005 to TX_BD_Length, and 0x00002000 to TX_BD_Pointer.
10. Write 0xFF to the SPIE register to clear any previous events.
11. Write 0x37 to the SPIM register to enable all possible serial peripheral interface interrupts.
12. Write 0x00000020 to the CIMR to allow the serial peripheral interface to generate a system interrupt. The CICR must also be initialized.
13. Write 0x0370 to the SPMODE register to enable normal operation (not loopback), master mode, serial peripheral interface enabled, 8-bit characters, and the fastest speed possible.
14. Set the STR bit in the SPCOM register to start the transfer.

Note: After 5 bytes are transmitted, the TX buffer descriptor is closed. Additionally, the receive buffer is closed after 5 bytes are received because the L bit of the TX buffer descriptor is set.

16.12.6 SPI Slave Programming Example

The following is an example initialization sequence to follow when the serial peripheral interface is in slave mode. It is very similar to the master example, except that the $\overline{\text{SPISEL}}$ pin is used instead of a general-purpose I/O pin.

1. Configure the port B pins to enable the SPIMOSI, SPIMISO, $\overline{\text{SPISEL}}$, and SPICLK pins. Write PBPAR bits 31, 30, 29, and 28 with ones and the PBODR bits 31, 30, 29, and 28 with zeros. Write PBDIR bits 30, 29, and 28 with ones and bit 31 with zero.
2. Write RBASE and TBASE in the SPI parameter RAM to point to the RX and TX buffer descriptor in the dual-port RAM. Assuming one RX buffer descriptor at the beginning of the dual-port RAM and one TX buffer descriptor following that RX buffer descriptor, write RBASE with 0x2000 and TBASE with 0x2008.
3. Write 0x18 to RFCR and TFCR for normal operation.
4. Program the CPM command register (CPCR) to execute the **INIT RX AND TX PARAMS** command. Write 0x0051 to the CPCR.
5. Write 0x0001 to the SDCR to initialize the SDMA configuration register.
6. Write MRBLR with the maximum number of bytes per receive buffer. Assume 16 bytes, so MRBLR = 0x0010.
7. Initialize the RX buffer descriptor and assume the RX data buffer is at 0x00001000 in main memory. Write 0xB000 to RX_BD_Status, 0x0000 to RX_BD_Length (optional), and 0x00001000 to RX_BD_Pointer.
8. Initialize the TX buffer descriptor and assume the TX data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Write 0xB800 to TX_BD_Status, 0x0005 to TX_BD_Length, and 0x00002000 to TX_BD_Pointer.

9. Write 0xFF to the SPIE register to clear any previous events.
10. Write 0x37 to the SPIM register to enable all possible serial peripheral interface interrupts.
11. Write 0x00000020 to the CIMR to allow the serial peripheral interface to generate a system interrupt. The CICR must also be initialized.
12. Write 0x0170 to the SPMODE register to enable normal operation (not loopback), master mode, and 8-bit characters. The SPI baud rate generator speed is ignored because the serial peripheral interface is in slave mode.
13. Set the STR bit in the SPCOM register to enable the serial peripheral interface to be ready once the master begins the transfer.

Note: If the master transmits 3 bytes and negates the $\overline{\text{SPISEL}}$ pin, the RX buffer descriptor is closed, but the TX buffer descriptor remains open. If the master transmits 5 or more bytes, the TX buffer descriptor is closed after the fifth byte. If the master transmits 16 bytes and negates the $\overline{\text{SPISEL}}$ pin, the RX buffer descriptor is closed with no errors and no out-of-buffer error occurs. If the master transmits more than 16 bytes, the RX buffer descriptor is closed (completely full) and the out-of-buffer error occurs after the 17th byte is received.

16.12.7 Handling Interrupts in the SPI

The following sequence must be followed to handle interrupts in the serial peripheral interface.

1. Once an interrupt occurs, read the SPIE register to discover the cause of the interrupts. Normally, the SPIE bits must be cleared at this time.
2. Process the TX buffer descriptor to reuse it and the RX buffer descriptor to extract the data from it. To transmit another buffer, simply set the R bit of the TX buffer descriptor, the E bit of the RX buffer descriptor, and the STR bit of the SPCOM register.
3. Clear the SPI bit in the CISR.
4. Execute the **rfi** instruction.