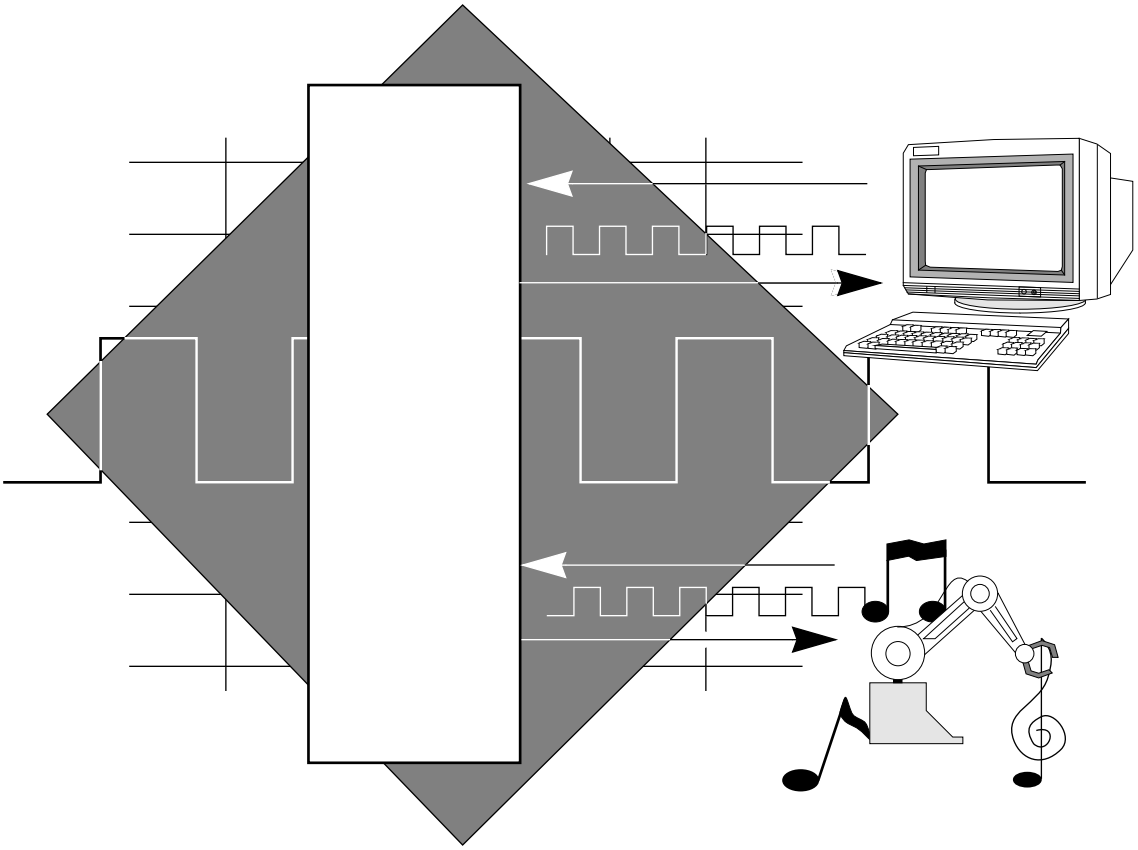
SECTION 6

PORT C



SECTION CONTENTS

6.1 INTRODUCTION

Port C is a triple-function I/O port with nine pins (see Figure 6-1). Three of the nine pins can be configured as general-purpose I/O or as the serial communication interface (SCI) pins. The other six pins can also be configured as GPIO, or they can be configured as the synchronous serial interface (SSI) pins.

When configured as general-purpose I/O, port C can be used for device control. When the pins are configured as serial interfaces, port C provides a convenient connection to other DSPs, processors, codecs, digital-to-analog and analog-to-digital converters, and any of several transducers. This section describes all three port C functions as well as examples of how to configure and use each function.

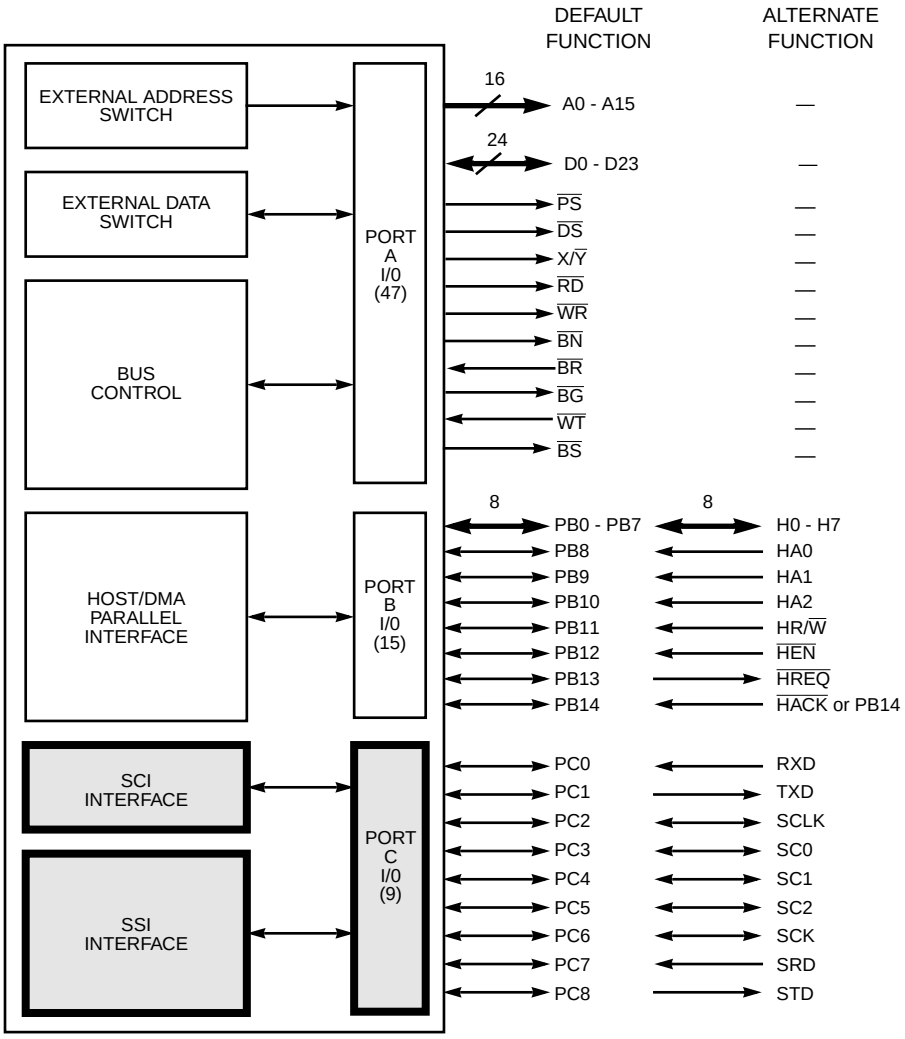


Figure 6-1 Port C Interface

6.2 GENERAL-PURPOSE I/O (PORT C)

When it is configured as GPIO, Port C can be viewed as nine I/O pins (see Figure 6-2), which are controlled by three memory-mapped registers. These registers are the Port C control register (PCC), Port C data direction register (PCDDR), and Port C data register (PCD) (see Figure 6-3).

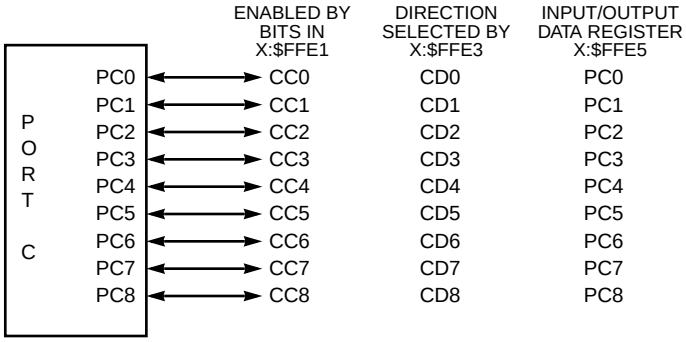
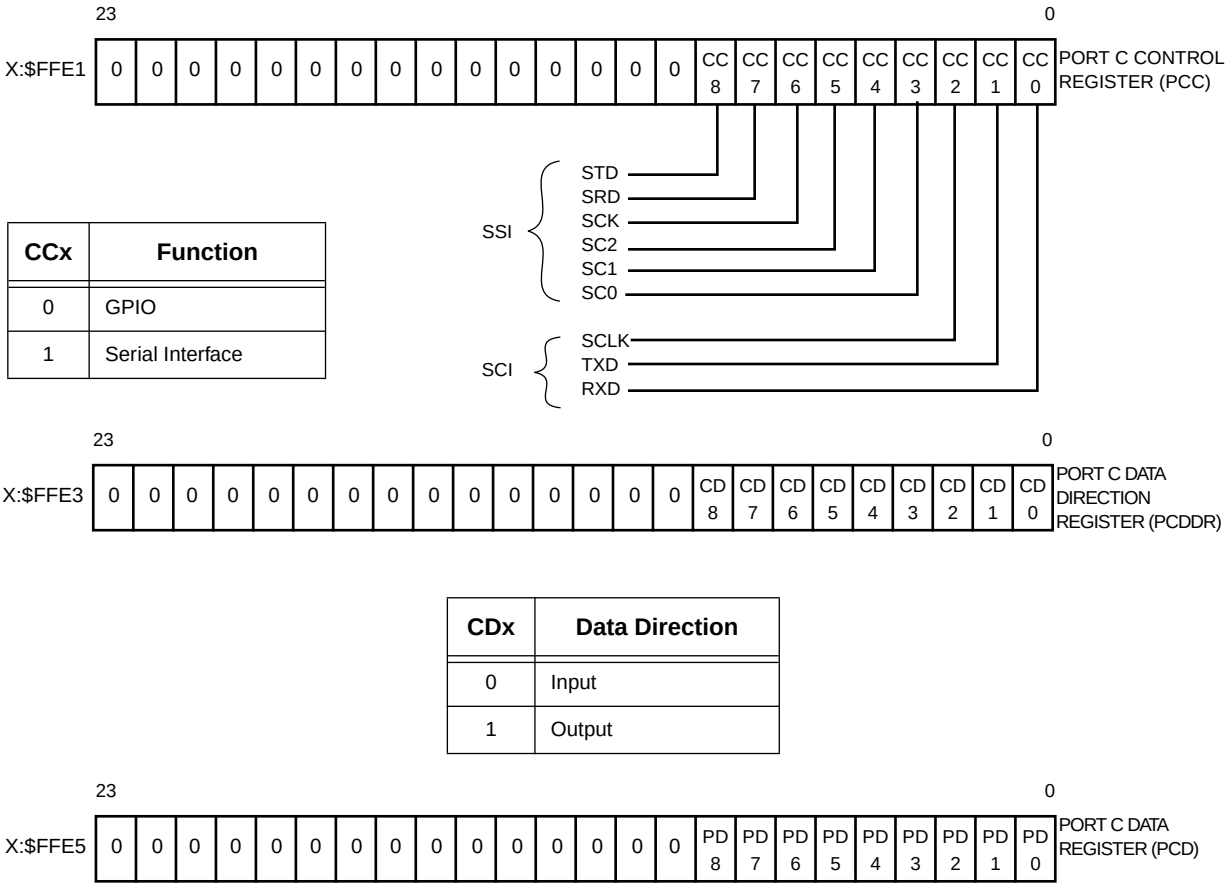


Figure 6-2 Port C GPIO Control

Reset clears PCC and PCDDR to configure Port C as general-purpose I/O with all nine pins as inputs. (External circuitry connected to these pins may need pullups until the pins are configured for operation.) Each Port C pin may be individually programmed as a general-purpose I/O pin or as a dedicated on-chip peripheral pin under software control. Pin selection between general-purpose I/O and SCI or SSI is made by setting the appropriate PCC bit (memory location X:$FFE1) to zero for general-purpose I/O or to one for serial interface.

The PCDDR (memory location X:$FFE3) programs each pin corresponding to a bit in the PCD (memory location X:$FFE5) as an input pin (if PCDDR=0) or as an output pin (if PCDDR=1).

If a pin is configured as a GPIO **input** (as shown in Figure 6-4) and the processor reads the PCD, the processor sees the logic level on the pin. If the processor writes to the PCD, the data is latched there, but does not appear on the pin because the buffer is in the high-impedance state.



CCx	Function
0	GPIO
1	Serial Interface

CDx	Data Direction
0	Input
1	Output

NOTE: Hardware and software reset clears PCC and PCDDR.

Figure 6-3 Port C GPIO Registers

If a pin is configured as a GPIO **output** and the processor reads the PCD, the processor sees the contents of the PCD rather the logic level on the pin, which allows the PCD to be used as a general purpose 15-bit register. If the processor writes to the PCD, the data is latched there and appears on the pin during the following instruction cycle (see **6.2.2**).

If a pin is configured as a **serial interface** (SCI or SSI) pin, the Port C GPIO registers can be used to help in debugging the serial interface. If the PCDDR bit for a given pin is cleared (configured as an input), the PCD will show the logic level on the pin, regardless of whether the serial interface function is using the pin as an input or an output. If the PCDDR is set (configured as an output) for a given serial interface pin, when the processor reads the PCD, it sees the contents of the PCD rather than the logic level on the pin — another case which allows the PCD to act as a general purpose register.

Port Control Register Bit	Data Direction Register Bit	Pin Function
0	0	Port Input Pin

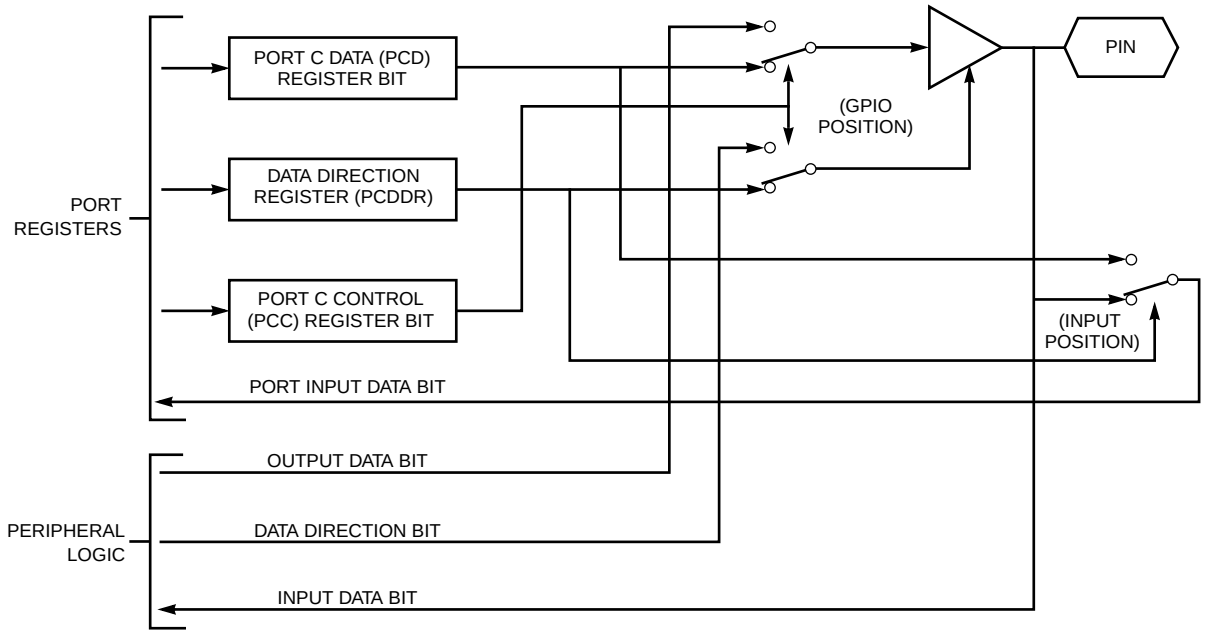


Figure 6-4 Port C I/O Pin Control Logic

6.2.1 Programming General Purpose I/O

Port C and all the DSP56002 peripherals are memory mapped (see Figure 6-5). The standard MOVE instruction transfers data between Port C and a register; as a result, performing a memory-to-memory data transfer takes two MOVE instructions and a register. The MOVEP instruction is specifically designed for I/O data transfer as shown in Figure 6-6. Although the MOVEP instruction may take twice as long to execute as a MOVE instruction, only one MOVEP is required for a memory-to-memory data transfer, and MOVEP does not use a temporary register. Using the MOVEP instruction allows a fast interrupt to move data to/from a peripheral to memory and execute one other instruction or to move the data to an absolute address. MOVEP is the only memory-to-memory move instruction; however, one of the operands must be in the top 64 locations of either X: or Y: memory. The bit-oriented instructions which use I/O short addressing (BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, and JSSET) can also be used to address individual bits for faster

```
    :
    :
MOVEP #$0,X:$FFE1          ;Select Port C to be general-purpose I/O
MOVEP #$01F0,X:$FFE3       ;Select pins PC0–PC3 to be inputs
    :                      ;and pins PC4–PC8 to be outputs
    :
MOVEP #data_out,X:$FFE5    ;Put bits 4–8 of “data_out” on pins
                           ;PB4–PB8 bits 0–3 are ignored.
MOVEP X:$FFE0,#data_in     ;Put PB0–PB3 in bits 0–3 of “data_in”
```

Figure 6-6 Write/Read Parallel Data with Port C

I/O processing.

The DSP does not have a hardware data strobe to strobe data out of the GPIO port. If a data strobe is needed, it can be implemented using software to toggle one of the GPIO pins.

Figure 6-7 shows the process of programming Port C as general-purpose I/O. Normally, it is not good programming practice to activate a peripheral before programming it. However, reset activates the Port C general-purpose I/O as all inputs, and the alternative is to configure the port as an SCI and/or SSI, which may not be desirable. In this case, it is probably better to insure that Port C is initially configured for general-purpose I/O and then configure the data direction and data registers. It may be better in some situations to program the data direction or the data registers first to prevent two devices from driving one signal. The order of steps 1, 2, and 3 in Figure 6-7 is optional and can be changed as needed.

6.2.2 Port C General Purpose I/O Timing

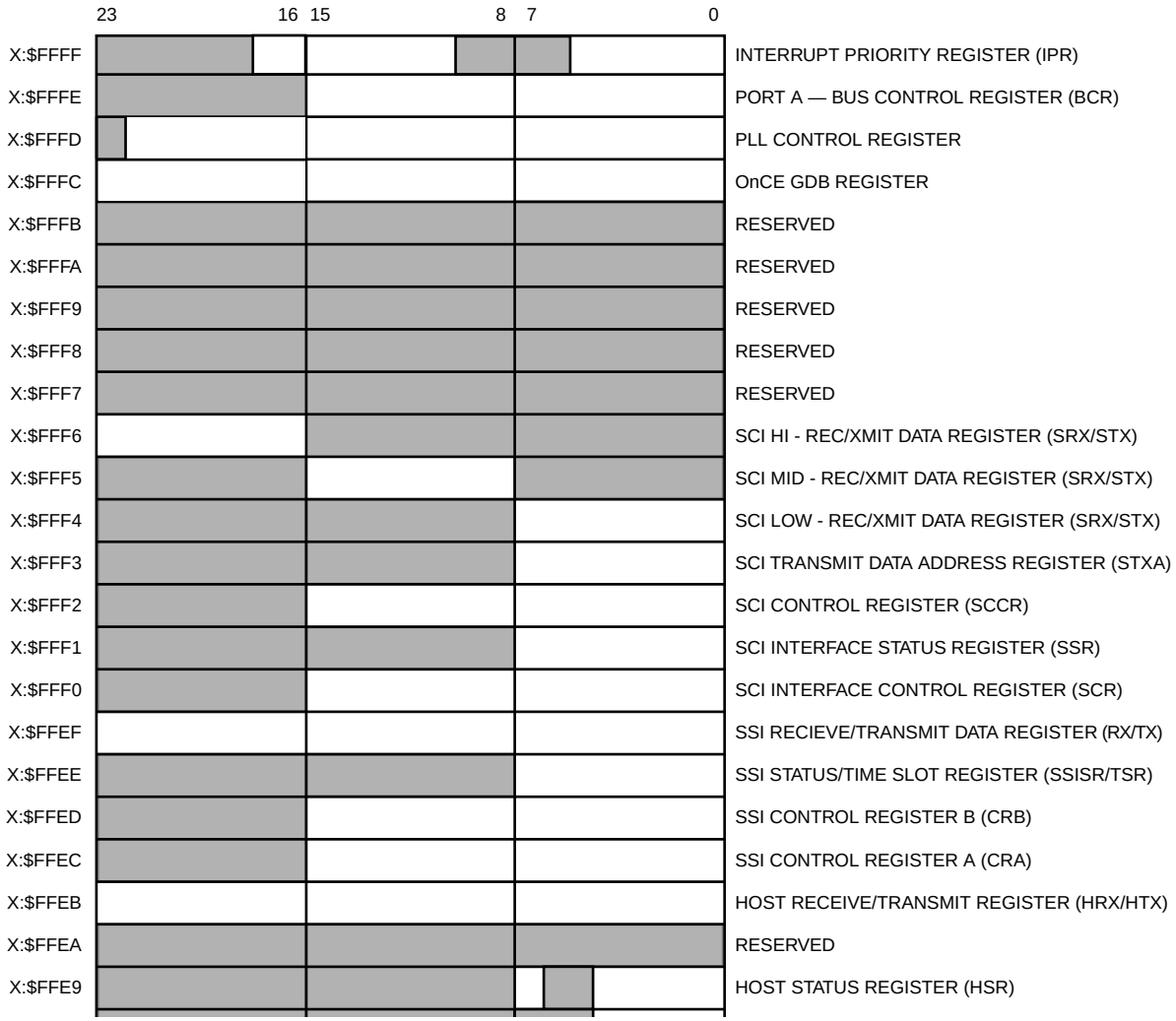
Parallel data written to Port C is delayed by one instruction cycle. For example, the following instruction:

```
MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN
```

1. writes nine bits of data to the Port C register, but the output pins do not change until the following instruction cycle
2. writes 24 bits of data to the external Y memory, which appears on Port A during T2 and T3 of the current instruction

As a result, if it is necessary to synchronize the Port A and Port C outputs, two instructions must be used:

```
MOVE    DATA9,X:PORTC
NOP                      DATA24,Y:EXTERN
```



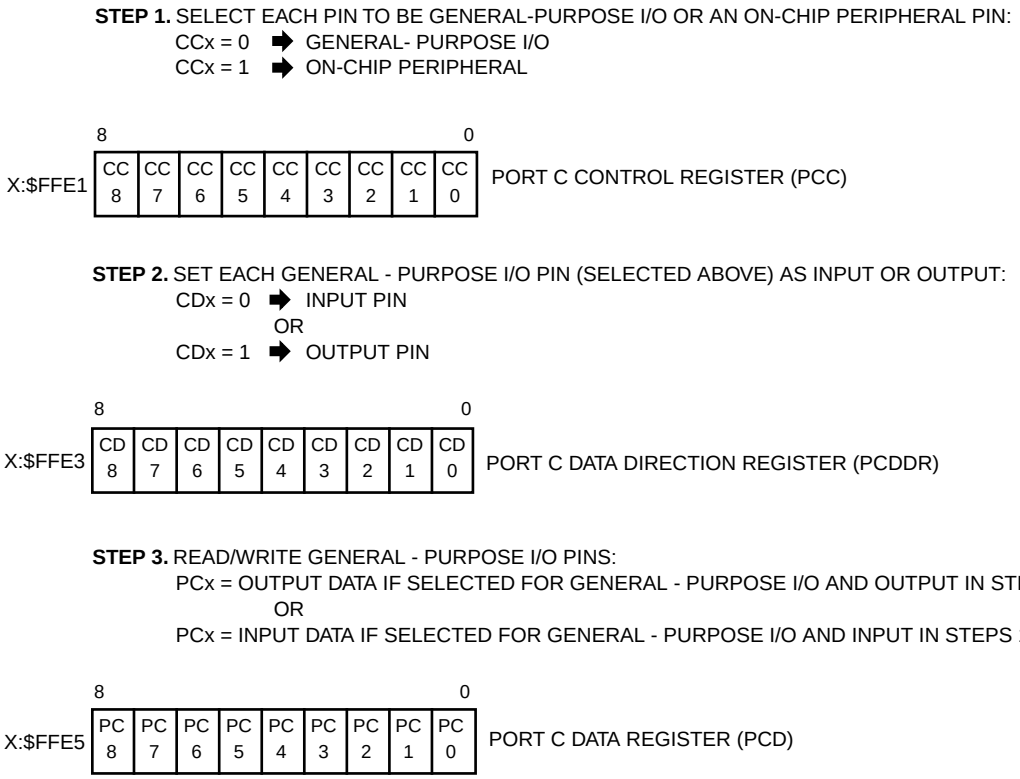


Figure 6-7 I/O Port C Configuration

The NOP can be replaced by any instruction that allows parallel moves. Inserting one or more “MOVE DATA15,X:PORTC DATA24,Y:EXTERN” instructions between the first and second instruction produces an external 33-bit write each instruction cycle with only one instruction cycle lost in setup time:

```
MOVE    DATA9,X:PORTC
MOVE    DATA9,X:PORTC     DATA24,Y:EXTERN
MOVE    DATA9,X:PORTC     DATA24,Y:EXTERN
  :
  :
MOVE    DATA9,X:PORTC     DATA24,Y:EXTERN
NOP                       DATA24,Y:EXTERN
```

One application of this technique is to create an extended address for Port A by concatenating the Port A address bits (instead of data bits) to the Port C general-purpose output bits. The Port C general-purpose I/O register would then work as a base address register, allowing the address space to be extended from 64K words (16 bits) to 33.5 million words

(16 bits+ 9 bits=25 bits).

Port C uses the DSP central processing unit (CPU) four-phase clock for its operation. Therefore, if wait states are inserted in the DSP CPU timing, they also affect Port C timing. As a result, Port A and Port C in the previous synchronization example will always stay synchronized, regardless of how many wait states are used.

6.3 SERIAL COMMUNICATION INTERFACE (SCI)

The SCI provides a full-duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. The communication can be TTL-level signals or, with additional logic, RS232C, RS422, etc.

This interface uses three dedicated pins: transmit data (TXD), receive data (RXD), and SCI serial clock (SCLK). It supports industry-standard asynchronous bit rates and protocols as well as high-speed (up to 5 Mbps for a 40-MHz clock) synchronous data transmission. The asynchronous protocols include a multidrop mode for master/slave operation with wakeup on idle line and wakeup on address bit capability.

The SCI consists of separate transmit and receive sections whose operations can be asynchronous with respect to each other. A programmable baud-rate generator provides the transmit and receive clocks. An enable vector and an interrupt vector have been included so that the baud-rate generator can function as a general-purpose timer when it is not being used by the SCI peripheral or when the interrupt timing is the same as that used by the SCI. The following is a short list of SCI features:

- Three-Pin Interface:
  - TXD – Transmit Data
  - RXD – Receive Data
  - SCLK – Serial Clock
- 625 Kbps NRZ Asynchronous Communications Interface (40-MHz System Clock)
- 5.0 Mbps Synchronous Serial Mode (40-MHz System Clock)
- Multidrop Mode for Multiprocessor Systems:
  - Two Wakeup Modes: Idle Line and Address Bit
  - Wired-OR Mode
- On-Chip or External Baud Rate Generation/Interrupt Timer
- Four Interrupt Priority Levels
- Fast or Long Interrupts

6.3.1 SCI I/O Pins

The three SCI pins can be configured as either general-purpose I/O or as a specific SCI pin. Each pin is independent of the other two, so that if only TXD is needed, RXD and SCLK can be programmed for general-purpose I/O. However, at least one of the three pins must be selected as an SCI pin to release the SCI from reset.

SCI interrupts may be enabled by programming the SCI control registers before any of the SCI pins are programmed as SCI functions. In this case, only one transmit interrupt can be generated because the transmit data register is empty. The timer and timer interrupt will operate as they do when one or more of the SCI pins is programmed as an SCI function.

6.3.1.1 Receive Data (RXD)

This input receives byte-oriented serial data and transfers the data to the SCI receive shift register. Asynchronous input data is sampled on the positive edge of the receive clock (1 × SCLK) if SCKP equals zero. See the DSP56002 Technical Data Sheet for detailed timing information. RXD may be programmed as a general-purpose I/O pin (PC0) when the SCI RXD function is not being used.

6.3.1.2 Transmit Data (TXD)

This output transmits serial data from the SCI transmit shift register. Data changes on the negative edge of the asynchronous transmit clock (SCLK) if SCKP equals zero. This output is stable on the positive edge of the transmit clock. See the DSP56002 Technical Data Sheet for detailed timing information. TXD may be programmed as a general-purpose I/O pin (PC1) when the SCI TXD function is not being used.

6.3.1.3 SCI Serial Clock (SCLK)

This bidirectional pin provides an input or output clock from which the transmit and/or receive baud rate is derived in the asynchronous mode and from which data is transferred in the synchronous mode. SCLK may be programmed as a general-purpose I/O pin (PC2) when the SCI SCLK function is not being used. This pin may be programmed as PC2 when data is being transmitted on TXD since, in the asynchronous mode, the clock need not be transmitted. There is no connection between programming the PC2 pin as SCLK and data coming out the TXD pin because SCLK is independent of SCI data I/O.

6.3.2 SCI Programming Model

The resources available in the SCI are described before discussing specific examples of how the SCI is used. The registers comprising the SCI are shown in Figure 6-8 and Figure 6-9. These registers are the SCI control register (SCR), SCI status register (SSR), SCI clock control register (SCCR), SCI receive data registers (SRX), SCI transmit data registers (STX), and the SCI transmit data address register (STXA). The SCI programming model can be viewed as three types of registers: 1) control – SCR and SCCR in Figure 6-8; 2) status – SSR in Figure 6-8; and 3) data transfer – SRX, STX, and STXA in Figure 6-9. The following paragraphs describe each bit in the programming model.

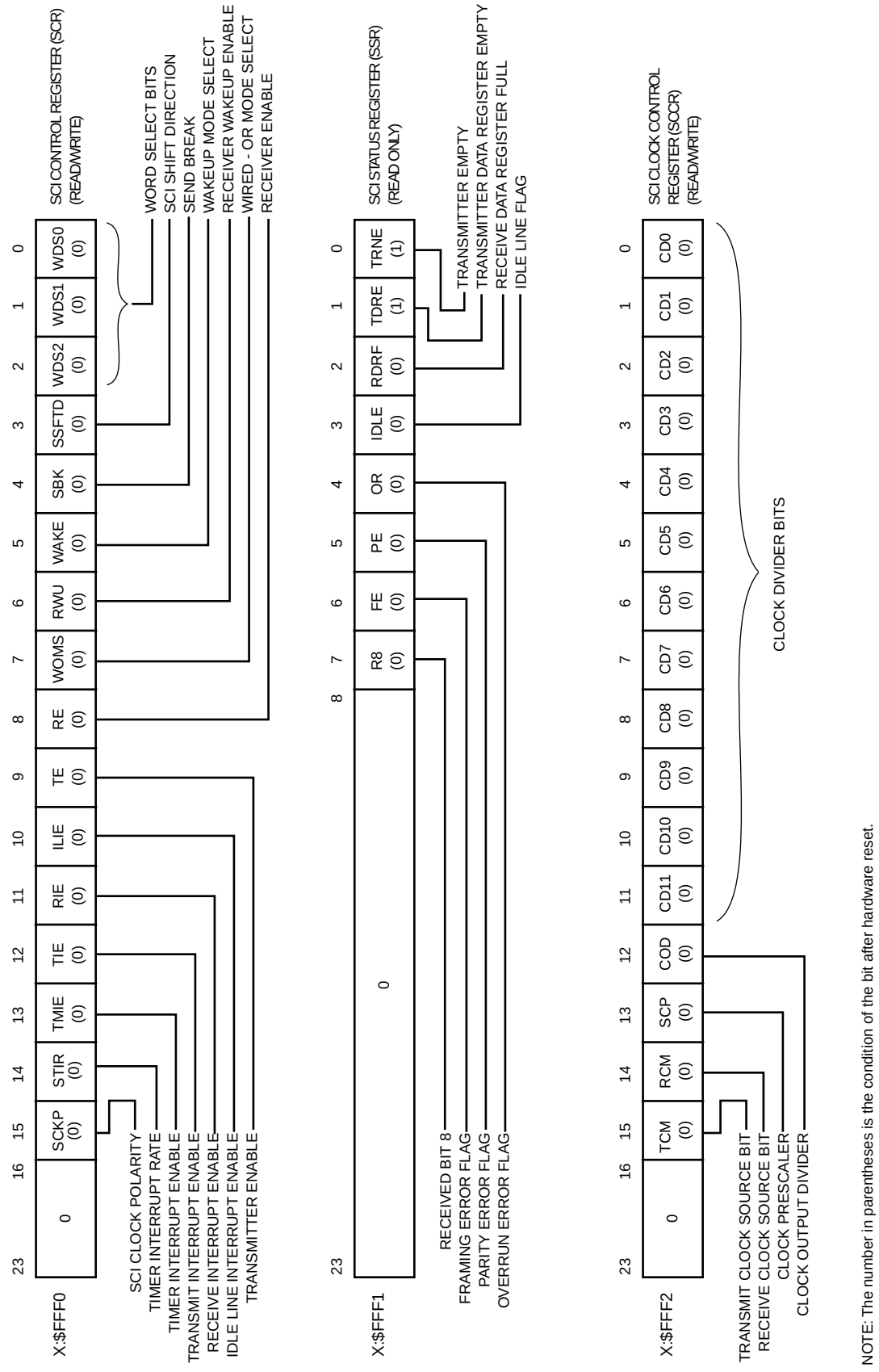
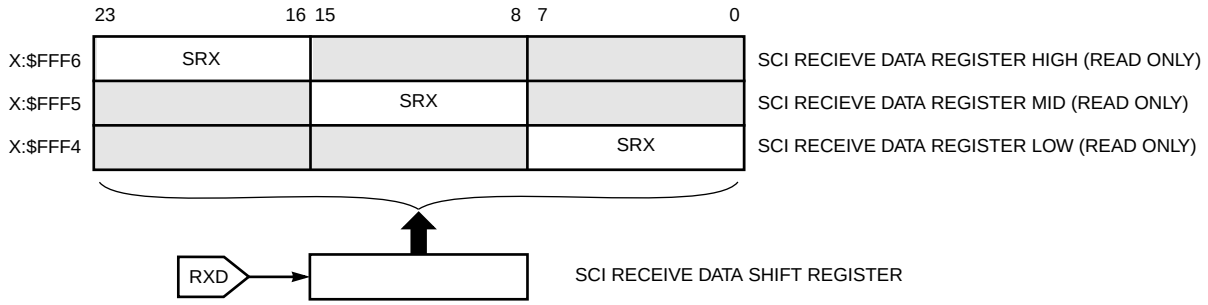
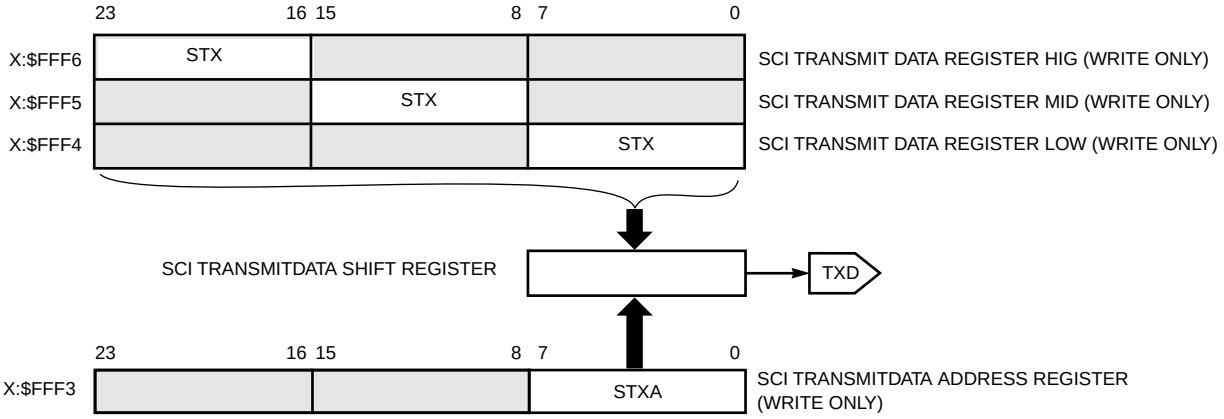


Figure 6-8 SCI Programming Model – Control and Status Registers



NOTE: SRX is the same register decoded at three different addresses.

(a) Receive Data Register



NOTES:
1. Bytes are masked on the fly.
2. STX is the same register decoded at three different addresses.

(b) Transmit Data Register

Figure 6-9 SCI Programming Model

6.3.2.1 SCI Control Register (SCR)

The SCR is a 16-bit read/write register that controls the serial interface operation. Each bit is described in the following paragraphs.

6.3.2.1.1 SCR Word Select (WDS0, WDS1, WDS2) Bits 0, 1, and 2

The three word-select bits (WDS0, WDS1, WDS2) select the format of the transmit and receive data. The formats include three asynchronous, one multidrop asynchronous mode, and an 8-bit synchronous (shift register) mode. The asynchronous modes are compatible with most UART-type serial devices and support standard RS232C communication links.

The multidrop asynchronous modes are compatible with the MC68681 DUART, the M68HC11 SCI interface, and the Intel 8051 serial interface.

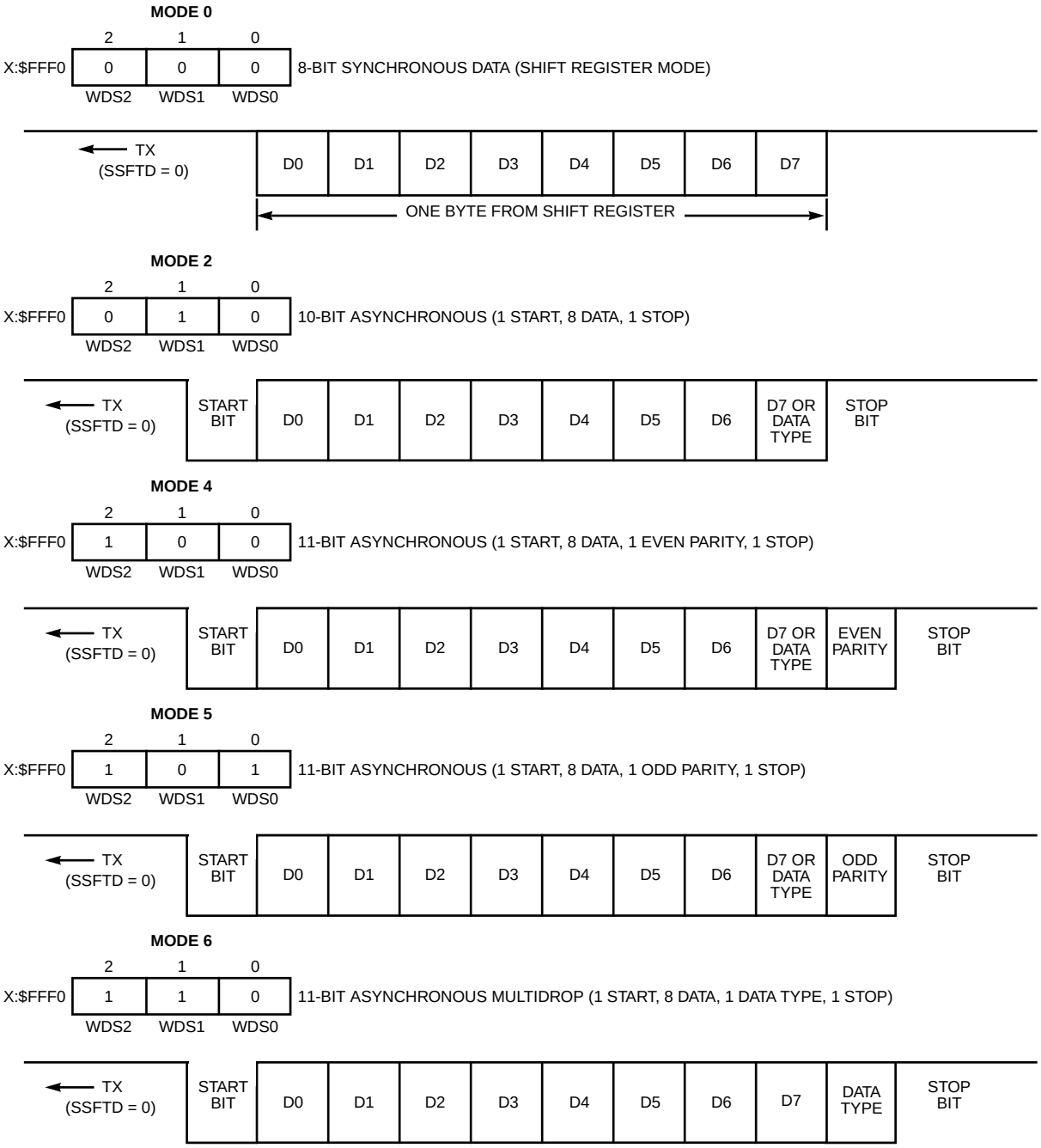
The synchronous data mode is essentially a high-speed shift register used for I/O expansion and stream-mode channel interfaces. A gated transmit and receive clock that is compatible with the Intel 8051 serial interface mode 0 accomplishes data synchronization. The word formats are shown in Table 6-1 (also see Figure 6-10 (a) and (b)).

Table 6-1 Word Formats

WDS2	WDS1	WDS0	Word Formats
0	0	0	8-Bit Synchronous Data (shift register mode)
0	0	1	Reserved
0	1	0	10-Bit Asynchronous (1 start, 8 data, 1 stop)
0	1	1	Reserved
1	0	0	11-Bit Asynchronous (1 start, 8 data, 1 even parity, 1 stop)
1	0	1	11-Bit Asynchronous (1 start, 8 data, 1 odd parity, 1 stop)
1	1	0	11-Bit Multidrop (1 start, 8 data, 1 data type, 1 stop)
1	1	1	Reserved

When odd parity is selected, the transmitter will count the number of bits in the data word. If the total is not an odd number, the parity bit is made equal to one and thus produces an odd number. If the receiver counts an even number of ones, an error in transmission has occurred. When even parity is selected, an even number must result from the calculation performed at both ends of the line or an error in transmission has occurred.

The word-select bits are cleared by hardware and software reset.

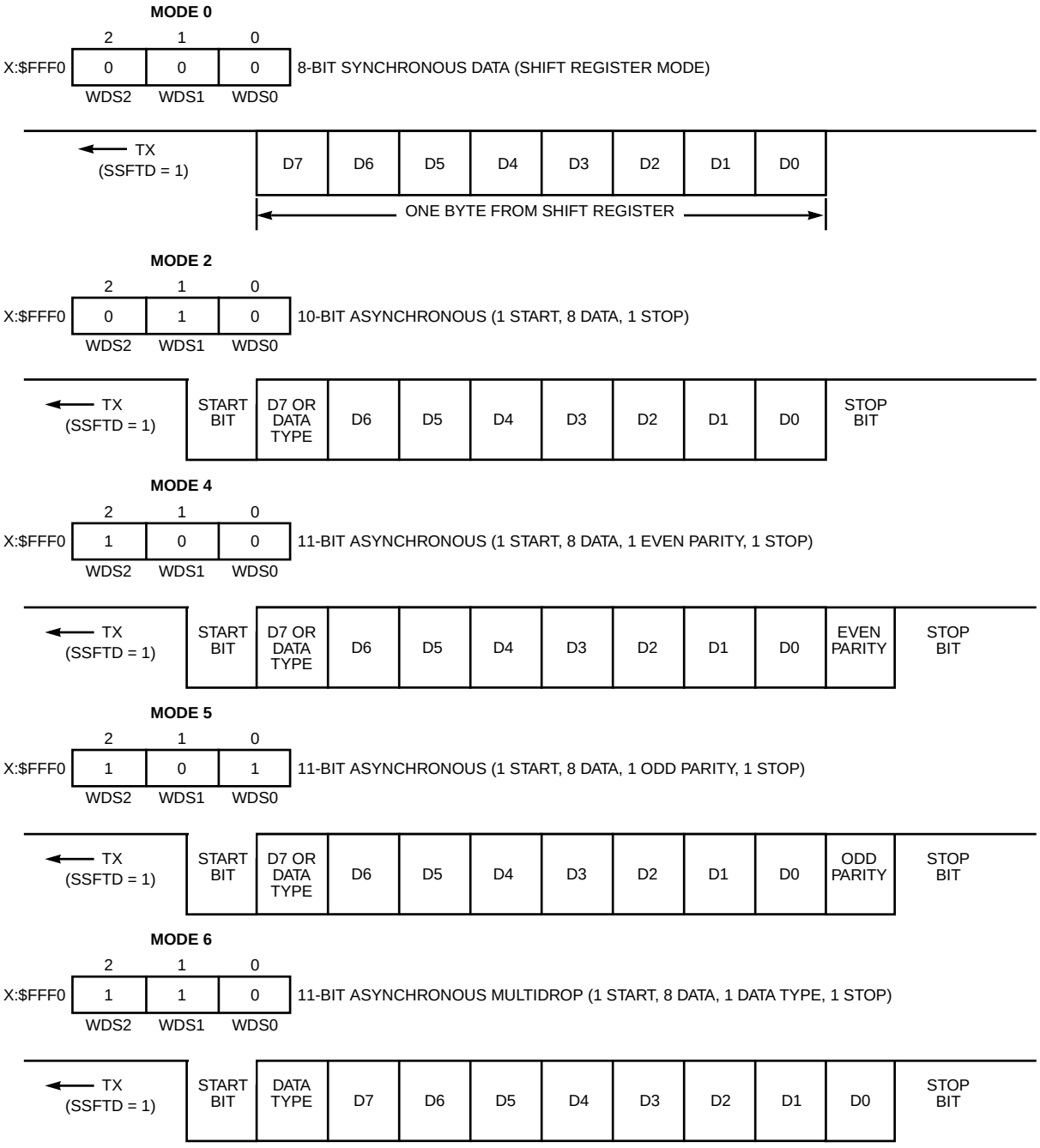


Data Type: 1 = Address Byte
0 = Data Byte

NOTES:
1. Modes1, 3, and 7 are reserved.
2. D0 =LSB;D7 = MSB
3. Data is transmitted and received LSB first if SSFTD = 0 or MSB first if SSFTD = 1.

(a) SSFTD = 0

Figure 6-10 Serial Formats (Sheet 1 of 2)



Data Type: 1 = Address Byte
0 = Data Byte

NOTES:
1. Modes 1, 3, and 7 are reserved.
2. D0 = LSB;D7 = MSB
3. Data is transmitted and received LSB first if SSFTD = 0 or MSB first if SSFTD = 1.

(b) SSFTD = 1

Figure 6-10 Serial Formats (Sheet 2 of 2)

6.3.2.1.2 SCR SCI Shift Direction (SSFTD) Bit 3

The SCI data shift registers can be programmed to shift data in/out either LSB first if SSFTD equals zero, or MSB first if SSFTD equals one. The parity and data type bits do not change position and remain adjacent to the stop bit. SSFTD is cleared by hardware and software reset.

6.3.2.1.3 SCR Send Break (SBK) Bit 4

A break is an all-zero word frame – a start bit zero, a character of all zeros (including any parity), and a stop bit zero: i.e., 10 or 11 zeros depending on the WDS mode selected. If SBK is set and then cleared, the transmitter completes transmission of any data, sends 10 or 11 zeros, and reverts to idle or sending data. If SBK remains set, the transmitter will continually send whole frames of zeros (10 or 11 bits with no stop bit). At the completion of the break code, the transmitter sends at least one high bit before transmitting any data to guarantee recognition of a valid start bit. Break can be used to signal an unusual condition, message, etc. by forcing a frame error, which is caused by a missing stop bit. Hardware and software reset clear SBK.

6.3.2.1.4 SCR Wakeup Mode Select (WAKE) Bit 5

When WAKE equals zero, an idle line wakeup is selected. In the idle line wakeup mode, the SCI receiver is re-enabled by an idle string of at least 10 or 11 (depending on WDS mode) consecutive ones. The transmitter's software must provide this idle string between consecutive messages. The idle string cannot occur within a valid message because each word frame contains a start bit that is a zero.

When WAKE equals one, an address bit wakeup is selected. In the address bit wakeup mode, the SCI receiver is re-enabled when the last (eighth or ninth) data bit received in a character (frame) is one. The ninth data bit is the address bit (R8) in the 11-bit multidrop mode; the eighth data bit is the address bit in the 10-bit asynchronous and 11-bit asynchronous with parity modes. Thus, the received character is an address that has to be processed by all sleeping processors – i.e., each processor has to compare the received character with its own address and decide whether to receive or ignore all following characters. WAKE is cleared by hardware and software reset.

6.3.2.1.5 SCR Receiver Wakeup Enable (RWU) Bit 6

When RWU equals one and the SCI is in an asynchronous mode, the wakeup function is enabled – i.e., the SCI is put to sleep waiting for a reason (defined by the WAKE bit) to wakeup. In the sleeping state, all receive flags, except IDLE, and interrupts are disabled. When the receiver wakes up, this bit is cleared by the wakeup hardware. The programmer may also clear the RWU bit to wake up the receiver.

RWU can be used by the programmer to ignore messages that are for other devices on a multidrop serial network. Wakeup on idle line (WAKE=0) or wakeup on address bit (WAKE=1) must be chosen.

1. When WAKE equals zero and RWU equals one, the receiver will not respond to data on the data line until an idle line is detected.
2. When WAKE equals one and RWU equals one, the receiver will not respond to data on the data line until a data byte with bit 9 equal to one is detected.

When the receiver wakes up, the RWU bit is cleared, and the first byte of data is received. If interrupts are enabled, the CPU will be interrupted, and the interrupt routine will read the message header to determine if the message is intended for this DSP.

1. If the message is for this DSP, the message will be received, and RWU will again be set to one to wait for the next message.
2. If the message is not for this DSP, the DSP will immediately set RWU to one. Setting RWU to one causes the DSP to ignore the remainder of the message and wait for the next message.

RWU is cleared by hardware and software reset. RWU is a don't care in the synchronous mode.

6.3.2.1.6 SCR Wired-OR Mode Select (WOMS) Bit 7

When the WOMS bit is set, the SCI TXD driver is programmed to function as an open-drain output and may be wired together with other TXD pins in an appropriate bus configuration such as a master-slave multidrop configuration. An external pullup resistor is required on the bus. When the WOMS is cleared, the TXD pin uses an active internal pullup. This bit is cleared by hardware and software reset.

6.3.2.1.7 SCR Receiver Enable (RE) Bit 8

When RE is set, the receiver is enabled. When RE is cleared, the receiver is disabled, and data transfer is inhibited to the receive data register (SRX) from the receive shift register. If RE is cleared while a character is being received, the reception of the character will be completed before the receiver is disabled. RE does not inhibit RDRF or receive interrupts. RE is cleared by a hardware and software reset.

6.3.2.1.8 SCR Transmitter Enable (TE) Bit 9

When TE is set, the transmitter is enabled. When TE is cleared, the transmitter will complete transmission of data in the SCI transmit data shift register; then the serial output is

forced high (idle). Data present in the SCI transmit data register (STX) will not be transmitted. STX may be written and TDRE will be cleared, but the data will not be transferred into the shift register. TE does not inhibit TDRE or transmit interrupts. TE is cleared by a hardware and software reset.

Setting TE will cause the transmitter to send a preamble of 10 or 11 consecutive ones (depending on WDS). This procedure gives the programmer a convenient way to ensure that the line goes idle before starting a new message. To force this separation of messages by the minimum idle line time, the following sequence is recommended:

1. Write the last byte of the first message to STX
2. Wait for TDRE to go high, indicating the last byte has been transferred to the transmit shift register
3. Clear TE and set TE back to one. This queues an idle line preamble to immediately follow the transmission of the last character of the message (including the stop bit)
4. Write the first byte of the second message to STX

In this sequence, if the first byte of the second message is not transferred to the STX prior to the finish of the preamble transmission, then the transmit data line will simply mark idle until STX is finally written.

6.3.2.1.9 SCR Idle Line Interrupt Enable (ILIE) Bit 10

When ILIE is set, the SCI interrupt occurs when IDLE is set. When ILIE is clear, the IDLE interrupt is disabled. ILIE is cleared by hardware and software reset.

An internal flag, the shift register idle interrupt (SRIINT) flag, is the interrupt request to the interrupt controller. SRIINT is not directly accessible to the user.

When a valid start bit has been received, an idle interrupt will be generated if both IDLE (SCI Status Register bit 3) and ILIE equals one. The idle interrupt acknowledge from the interrupt controller clears this interrupt request. The idle interrupt will not be asserted again until at least one character has been received. The result is as follows:

1. The IDLE bit shows the real status of the receive line at all times.
2. Idle interrupt is generated once for each idle state, no matter how long the idle state lasts.

6.3.2.1.10 SCR SCI Receive Interrupt Enable (RIE) Bit 11

The RIE bit is used to enable the SCI receive data interrupt. If RIE is cleared, receive interrupts are disabled, and the RDRF bit in the SCI status register must be polled to determine if the receive data register is full. If both RIE and RDRF are set, the SCI will request an SCI receive data interrupt from the interrupt controller.

One of two possible receive data interrupts will be requested:

1. Receive without exception will be requested if PE, FE, and OR are all clear (i.e., a normal received character).
2. Receive with exception will be requested if PE, FE, and OR are not all clear (i.e., a received character with an error condition).

RIE is cleared by hardware and software reset.

6.3.2.1.11 SCR SCI Transmit Interrupt Enable (TIE) Bit 12

The TIE bit is used to enable the SCI transmit data interrupt. If TIE is cleared, transmit data interrupts are disabled, and the transmit data register empty (TDRE) bit in the SCI status register must be polled to determine if the transmit data register is empty. If both TIE and TDRE are set, the SCI will request an SCI transmit data interrupt from the interrupt controller. TIE is cleared by hardware and software reset.

6.3.2.1.12 SCR Timer Interrupt Enable (TMIE) Bit 13

The TMIE bit is used to enable the SCI timer interrupt. If TMIE is set (enabled), the timer interrupt requests will be made to the interrupt controller at the rate set by the SCI clock register. The timer interrupt is automatically cleared by the timer interrupt acknowledge from the interrupt controller. This feature allows DSP programmers to use the SCI baud clock generator as a simple periodic interrupt generator if the SCI is not in use, if external clocks are used for the SCI, or if periodic interrupts are needed at the SCI baud rate. The SCI internal clock is divided by 16 (to match the 1 × SCI baud rate) for timer interrupt generation. This timer does not require that any SCI pins be configured for SCI use to operate. TMIE is cleared by hardware and software reset.

6.3.2.1.13 SCR SCI Timer Interrupt Rate (STIR) Bit 14

This bit controls a divide by 32 in the SCI Timer interrupt generator. When this bit is cleared, the divide by 32 is inserted in the chain. When the bit is set, the divide by 32 is bypassed, thereby increasing the timer resolution by 32 times. This bit is cleared by hardware and software reset.

6.3.2.1.14 SCR SCI Clock Polarity (SCKP) Bit 15

The clock polarity, sourced or received on the clock pin (SCLK), can be inverted using this bit, eliminating the need for an external inverter. When bit 15 equals zero, the clock polarity is positive; when bit 15 equals one, the clock polarity is negative. In the synchronous mode, positive polarity means that the clock is normally positive and transitions negative during data valid; whereas, negative polarity means that the clock is normally negative and transitions positive during valid data. In the asynchronous mode, positive polarity means that the rising edge of the clock occurs in the center of the period that data is valid; negative polarity means that the falling edge of the clock occurs during the center of the period that data is valid. SCKP is cleared on hardware and software reset.

6.3.2.2 SCI Status Register (SSR)

The SSR is an 8-bit read-only register used by the DSP CPU to determine the status of the SCI. When the SSR is read onto the internal data bus, the register contents occupy the low-order byte of the data bus and all high-order portions are zero filled. The status bits are described in the following paragraphs.

6.3.2.2.1 SSR Transmitter Empty (TRNE) Bit 0

The TRNE flag is set when both the transmit shift register and data register are empty to indicate that there is no data in the transmitter. When TRNE is set, data written to one of the three STX locations or to the STXA will be transferred to the transmit shift register and be the first data transmitted. TRNE is cleared when TDRE is cleared by writing data into the transmit data register (STX) or the transmit data address register (STXA), or when an idle, preamble, or break is transmitted. The purpose of this bit is to indicate that the transmitter is empty; therefore, the data written to STX or STXA will be transmitted next – i.e., there is not a word in the transmit shift register presently being transmitted. This procedure is useful when initiating the transfer of a message (i.e., a string of characters). TRNE is set by the hardware, software, SCI individual, and stop reset.

6.3.2.2.2 SSR Transmit Data Register Empty (TDRE) Bit 1

The TDRE bit is set when the SCI transmit data register is empty. When TDRE is set, new data may be written to one of the SCI transmit data registers (STX) or transmit data address register (STXA). TDRE is cleared when the SCI transmit data register is written. TDRE is set by the hardware, software, SCI individual, and stop reset.

In the SCI synchronous mode, when using the internal SCI clock, there is a delay of up to 5.5 serial clock cycles between the time that STX is written until TDRE is set, indicating the data has been transferred from the STX to the transmit shift register. There is a two to four serial clock cycle delay between writing STX and loading the transmit shift register;

in addition, TDRE is set in the middle of transmitting the second bit. When using an external serial transmit clock, if the clock stops, the SCI transmitter stops. TDRE will not be set until the middle of the second bit transmitted after the external clock starts. Gating the external clock off after the first bit has been transmitted will delay TDRE indefinitely.

In the SCI asynchronous mode, the TDRE flag is not set immediately after a word is transferred from the STX or STXA to the transmit shift register nor when the word first begins to be shifted out. TDRE is set two cycles of the 16× clock after the start bit – i.e., two 16× clock cycles into to transmission time of the first data bit.

6.3.2.2.3 SSR Receive Data Register Full (RDRF) Bit 2

The RDRF bit is set when a valid character is transferred to the SCI receive data register from the SCI receive shift register. RDRF is cleared when the SCI receive data register is read or by the hardware, software, SCI individual, and stop reset.

6.3.2.2.4 SSR Idle Line Flag (IDLE) Bit 3

IDLE is set when 10 (or 11) consecutive ones are received. IDLE is cleared by a start-bit detection. The IDLE status bit represents the status of the receive line. The transition of IDLE from zero to one can cause an IDLE interrupt (ILIE). IDLE is cleared by the hardware, software, SCI individual, and stop reset.

6.3.2.2.5 SSR Overrun Error Flag (OR) Bit 4

The OR flag is set when a byte is ready to be transferred from the receive shift register to the receive data register (SRX) that is already full (RDRF=1). The receive shift register data is not transferred to the SRX. The OR flag indicates that character(s) in the receive data stream may have been lost. The only valid data is located in the SRX. OR is cleared when the SCI status register is read, followed by a read of SRX. The OR bit clears the FE and PE bits – i.e., overrun error has higher priority than FE or PE. OR is cleared by the hardware, software, SCI individual, and stop reset.

6.3.2.2.6 SSR Parity Error (PE) Bit 5

In the 11-bit asynchronous modes, the PE bit is set when an incorrect parity bit has been detected in the received character. It is set simultaneously with RDRF for the byte which contains the parity error – i.e., when the received word is transferred to the SRX. If PE is set, it does not inhibit further data transfer into the SRX. PE is cleared when the SCI status register is read, followed by a read of SRX. PE is also cleared by the hardware, software, SCI individual, or stop reset. In the 10-bit asynchronous mode, the 11-bit multidrop mode,

and the 8-bit synchronous mode, the PE bit is always cleared since there is no parity bit in these modes. If the byte received causes both parity and overrun errors, the SCI receiver will only recognize the overrun error.

6.3.2.2.7 SSR Framing Error Flag (FE) Bit 6

The FE bit is set in the asynchronous modes when no stop bit is detected in the data string received. FE and RDRE are set simultaneously – i.e., when the received word is transferred to the SRX. However, the FE flag inhibits further transfer of data into the SRX until it is cleared. FE is cleared when the SCI status register is read followed by reading the SRX. The hardware, software, SCI individual, and stop reset also clear FE. In the 8-bit synchronous mode, FE is always cleared. If the byte received causes both framing and overrun errors, the SCI receiver will only recognize the overrun error.

6.3.2.2.8 SSR Received Bit 8 Address (R8) Bit 7

In the 11-bit asynchronous multidrop mode, the R8 bit is used to indicate whether the received byte is an address or data. R8 is not affected by reading the SRX or status register. The hardware, software, SCI individual, and stop reset clear R8.

6.3.2.3 SCI Clock Control Register (SCCR)

The SCCR is a 16-bit read/write register which controls the selection of the clock modes and baud rates for the transmit and receive sections of the SCI interface. The control bits are described in the following paragraphs. The SCCR is cleared by hardware reset.

The basic points of the clock generator are as follows:

1. The SCI core always uses a 16 × internal clock in the asynchronous modes and always uses a 2 × internal clock in the synchronous mode. The maximum internal clock available to the SCI peripheral block is the oscillator frequency divided by 4. With a 40-MHz crystal, this gives a maximum data rate of 625 Kbps for asynchonous data and 5 Mbps for synchronous data. These maximum rates are the same for internally or externally supplied clocks.
2. The 16 × clock is necessary for the asynchronous modes to synchronize the SCI to the incoming data (see Figure 6-11).
3. For the asynchronous modes, the user must provide a 16 × clock if he wishes to use an external baud rate generator (i.e., SCLK input).
4. For the asynchronous modes, the user may select either 1 × or 16 × for the output clock when using internal TX and RX clocks (TCM=0 and RCM=0).

5. The transmit data on the TXD pin changes on the negative edge of the 1 × serial clock and is stable on the positive edge (SCKP=0). For SCKP equals one, the data changes on the positive edge and is stable on the negative edge.
6. The receive data on the RXD pin is sampled on the positive edge (if SCKP=0) or on the negative edge (if SCKP=1) of the 1 × serial clock.
7. For the asynchronous mode, the output clock is continuous.
8. For the synchronous mode, a 1 × clock is used for the output or input baud rate. The maximum 1 × clock is the crystal frequency divided by 8.
9. For the synchronous mode, the clock is gated.
10. For both the asynchronous and synchronous modes, the transmitter and receiver are synchronous with each other.

6.3.2.3.1 SCCR Clock Divider (CD11–CD0) Bits 11–0

The clock divider bits (CD11–CD0) are used to preset a 12-bit counter, which is decremented at the I_{cyc} rate (crystal frequency divided by 2). The counter is not accessible to the user. When the counter reaches zero, it is reloaded from the clock divider bits. Thus, a value of 0000 0000 0000 in CD11–CD0 produces the maximum rate of I_{cyc}, and a value of 0000 0000 0001 produces a rate of $I_{cyc}/2$. The lowest rate available is $I_{cyc}/4096$. Figure 6-12 and Figure 6-35 show the clock dividers. Bits CD11–CD0 are cleared by hardware and software reset.

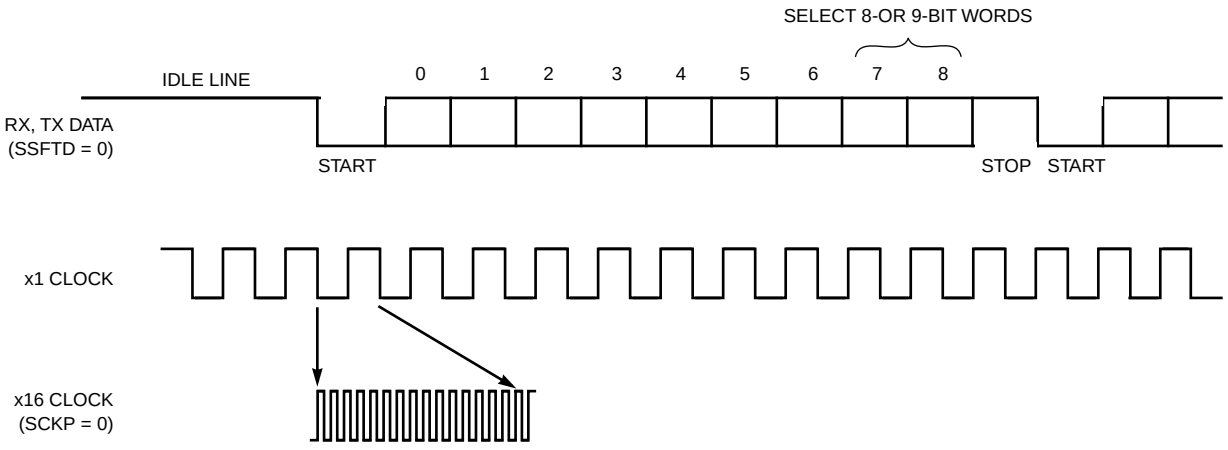


Figure 6-11 16 x Serial Clock

6.3.2.3.2 SCCR Clock Out Divider (COD) Bit 12

Figure 6-12 and Figure 6-35 show the clock divider circuit. The output divider is controlled by COD and the SCI mode. If the SCI mode is synchronous, the output divider is fixed at divide by 2; if the SCI mode is asynchronous, and

1. If COD equals zero and SCLK is an output (i.e., TCM and RCM=0), the SCI clock is divided by 16 before being output to the SCLK pin; thus, the SCLK output is a 1 × clock
2. If COD equals one and SCLK is an output, the SCI clock is fed directly out to the SCLK pin; thus, the SCLK output is a 16 × baud clock

The COD bit is cleared by hardware and software reset.

6.3.2.3.3 SCCR SCI Clock Prescaler (SCP) Bit 13

The SCI SCP bit selects a divide by 1 (SCP=0) or divide by 8 (SCP=1) prescaler for the clock divider. The output of the prescaler is further divided by 2 to form the SCI clock. Hardware and software reset clear SCP. Figure 6-12 and Figure 6-35 show the clock divider diagram.

6.3.2.3.4 SCCR Receive Clock Mode Source Bit (RCM) Bit 14

RCM selects internal or external clock for the receiver (see Figure 6-35). RCM equals zero selects the internal clock; RCM equals one selects the external clock from the SCLK pin. Hardware and software reset clear RCM.

6.3.2.3.5 SCCR Transmit Clock Source Bit (TCM) Bit 15

The TCM bit selects internal or external clock for the transmitter (see Figure 6-35). TCM equals zero selects the internal clock; TCM equals one selects the external clock from the SCLK pin. Hardware and software reset clear TCM.

6.3.2.4 SCI Data Registers

The SCI data registers are divided into two groups: receive and transmit. There are two receive registers – a receive data register (SRX) and a serial-to-parallel receive shift register. There are also two transmit registers – a transmit data register (called either STX or STXA) and a parallel-to-serial transmit shift register.

6.3.2.4.1 SCI Receive Registers

Data words received on the RXD pin are shifted into the SCI receive shift register. When the complete word has been received, the data portion of the word is transferred to the byte-wide SRX. This process converts the serial data to parallel data and provides double

buffering. Double buffering provides flexibility and increased throughput since the programmer can save the previous word while the current word is being received.

The SRX can be read at three locations: X:$FFF4, X:$FFF5, and X:$FFF6 (see Figure 6-13). When location X:$FFF4 is read, the contents of the SRX are placed in the lower byte of the data bus and the remaining bits on the data bus are written as zeros. Similarly, when X:$FFF5 is read, the contents of SRX are placed in the middle byte of the bus, and when X:$FFF6 is read, the contents of SRX are placed in the high byte with the remaining bits zeroed. Mapping SRX as described allows three bytes to be efficiently packed into

TCM	RCM	TX Clock	RX Clock	SCLK Pin	Mode
0	0	Internal	Internal	Output	Synchronous/Asynchronous
0	1	Internal	External	Input	Asynchronous Only
1	0	External	Internal	Input	Asynchronous Only
1	1	External	External	Input	Synchronous/Asynchronous

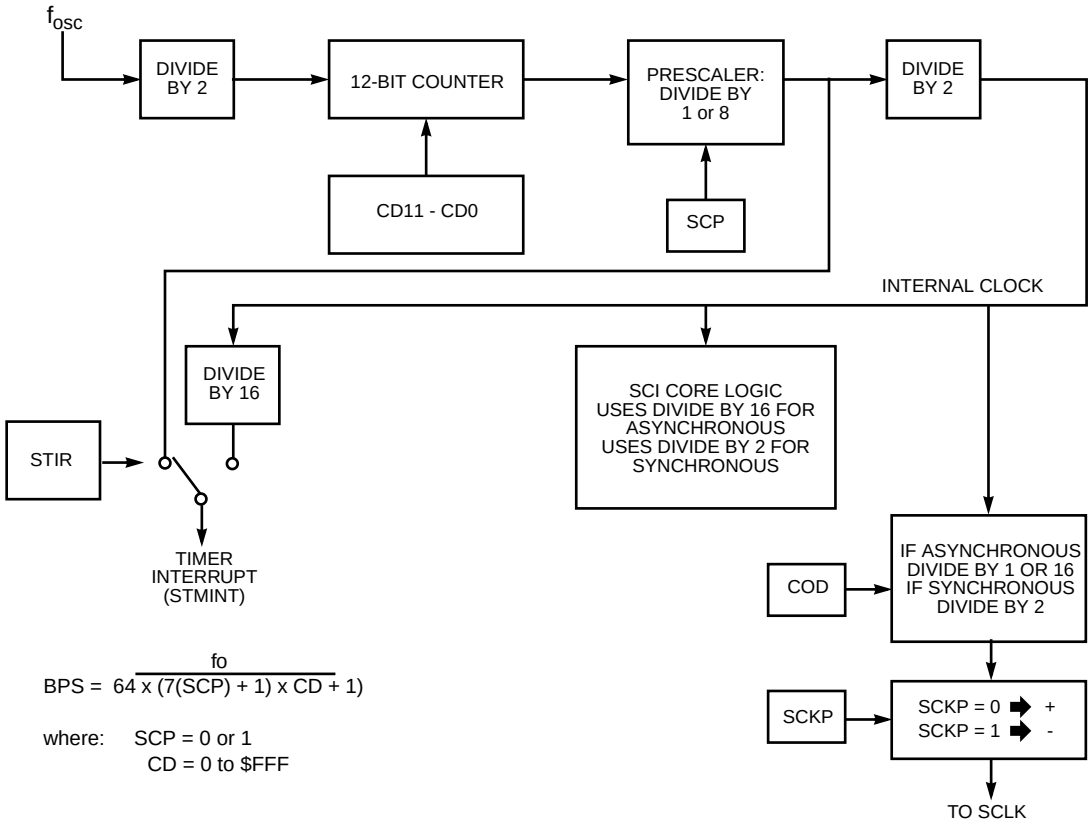


Figure 6-12 SCI Baud Rate Generator

one 24-bit word by “OR”-ing three data bytes read from the three addresses. The following code fragment requires that R0 initially points to X:$FFF4, register A is initially cleared, and R3 points to a data buffer. The only programming trick is using BCLR to test bit 1 of the packing pointer to see if it is pointing to X:$FFF6 and clearing bit 1 to point to X:$FFF4 if it had been pointing to X:$FFF6. This procedure resets the packing pointer after receiving three bytes.

```
        MOVE    X:(R0),X0   ;Copy received data to temporary register
        BCLR    #$1,R0      ;Test for last byte
                            ;reset pointer if it is the last byte
        OR      X0,A        ;Pack the data into register A
        MOVE    (R0)+       ;and increment the packing pointer
        JCS     FLAG        ;Jump to clean up routine if last byte
        RTI                 ;Else return until next byte is received
FLAG    MOVE    A,(R3)+     ;Move the packed data to memory
        CLR     A           ;Prepare A for packing next three bytes
        RTI                 ;Return until the next byte is received
```

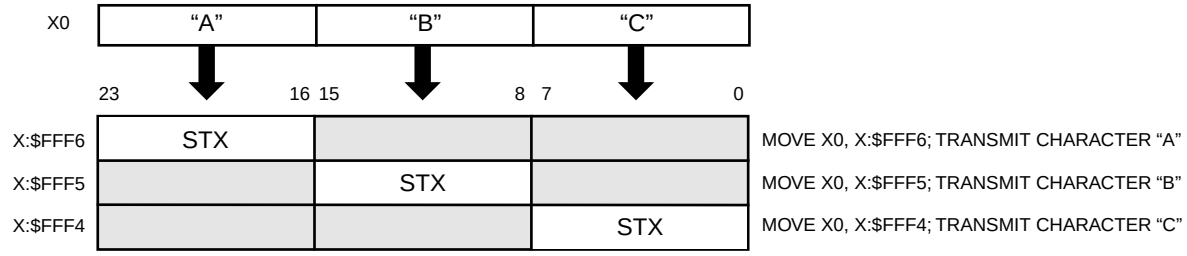
The length and format of the serial word is defined by the WDS0, WDS1, and WDS2 control bits in the SCI control register. In the synchronous modes, the start bit, the eight data bits with LSB first, the address/data indicator bit and/or the parity bit, and the stop bit are received in that order for SSFTD equals zero (see Figure 6-10 (a)). For SSFTD equals one, the data bits are transmitted MSB first (see Figure 6-10(b)). The clock source is defined by the receive clock mode (RCM) select bit in the SCR. In the synchronous mode, the synchronization is provided by gating the clock. In either mode, when a complete word has been clocked in, the contents of the shift register can be transferred to the SRX and the flags; RDRF, FE, PE, and OR are changed appropriately. Because the operation of the SCI receive shift register is transparent to the DSP, the contents of this register are not directly accessible to the programmer.

6.3.2.4.2 SCI Transmit Registers

The transmit data register is one byte-wide register mapped into four addresses: X:$FFF3, X:$FFF4, X:$FFF5, and X:$FFF6. In the asynchronous mode, when data is to be transmitted, X:$FFF4, X:$FFF5, and X:$FFF6 are used, and the register is called STX. When X:$FFF4 is written, the low byte on the data bus is transferred to the STX; when X:$FFF5 is written, the middle byte is transferred to the STX; and when X:$FFF6 is written, the high byte is transferred to the STX. This structure (see Figure 6-9) makes it easy for the programmer to unpack the bytes in a 24-bit word for transmission. Location X:$FFF3 should be written in the 11-bit asynchronous multidrop mode when the data is

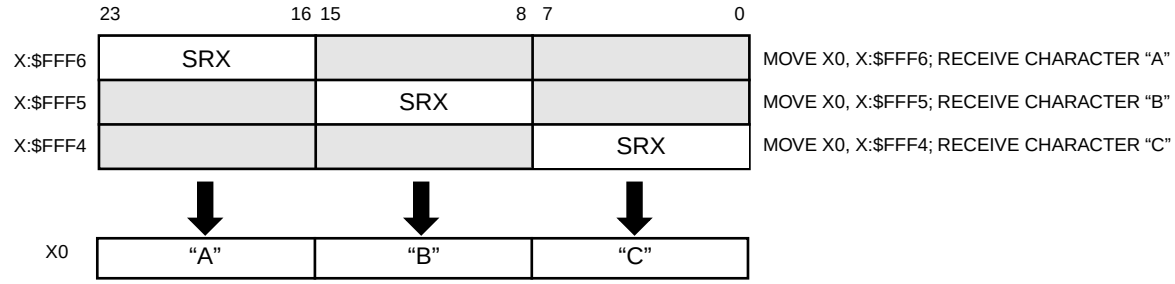
an address and it is desired that the ninth bit (the address bit) be set. When X:$FFF3 is written, the transmit data register is called STXA, and data from the low byte on the data bus is stored in STXA. The address data bit will be cleared in the 11-bit asynchronous multidrop mode when any of X:$FFF4, X:$FFF5, or X:$FFF6 is written. When either STX or STXA is written, TDRE is cleared.

The transfer from either STX or STXA to the transmit shift register occurs automatically, but not immediately, when the last bit from the previous word has been shifted out – i.e., the transmit shift register is empty. Like the receiver, the transmitter is double buffered. However, there will be a two to four serial clock cycle delay between when the data is transferred from either STX or STXA to the transmit shift register and when the first bit appears on the TXD pin. (A serial clock cycle is the time required to transmit one data bit).



NOTE: STX is the same register decoded at three different addresses.

(a) Unpacking



NOTE: SRX is the same register decoded at three different addresses.

(b) Packing

Figure 6-13 Data Packing and Unpacking

The transmit shift register is not directly addressable, and a dedicated flag for this register does not exist. Because of this fact and the two to four cycle delay, two bytes cannot be written consecutively to STX or STXA without polling. The second byte will overwrite the first byte. The TDRE flag should always be polled prior to writing STX or STXA to prevent overruns unless transmit interrupts have been enabled. Either STX or STXA is usually written as part of the interrupt service routine. Of course, the interrupt will only be generated if TDRE equals one. The transmit shift register is indirectly visible via the TRNE bit in the SSR.

In the synchronous modes, data is synchronized with the transmit clock, which may have either an internal or external source as defined by the TCM bit in the SCCR. The length and format of the serial word is defined by the WDS0, WDS1, and WDS2 control bits in the SCR. In the asynchronous modes, the start bit, the eight data bits (with the LSB first if SSFTD=0 and the MSB first if SSFTD=1), the address/data indicator bit or parity bit, and the stop bit are transmitted in that order (see Figure 6-10).

The data to be transmitted can be written to any one of the three STX addresses. If SCKP equals one and SSHTD equals one, the SCI synchronous mode is equivalent to the SSI operation in the 8-bit data on-demand mode.

6.3.2.5 Preamble, Break, and Data Transmission Priority

It is possible that two or three transmission commands are set simultaneously:

1. A preamble (TE was toggled)
2. A break (SBK was set or was toggled)
3. There is data for transmission (TDRE=0)

After the current character transmission, if two or more of these commands are set, the transmitter will execute them in the following priority:

1. Preamble
2. Break
3. Data

6.3.3 Register Contents After Reset

There are four methods to reset the SCI. Hardware or software reset clears the port control register bits, which configure all I/O as general-purpose input. The SCI will remain in the reset state while all SCI pins are programmed as general-purpose I/O (CC2, CC1, and CC0=0); the SCI will become active only when at least one of the SCI I/O pins is programmed as not general-purpose I/O.

During program execution, the CC2, CC1, and CC0 bits may be cleared (individual reset), which will cause the SCI to stop serial activity and enter the reset state. All SCI status bits will be set to their reset state; however, the contents of the interface control register are not affected, allowing the DSP program to reset the SCI separately from the other internal peripherals.

The STOP instruction halts operation of the SCI until the DSP is restarted, causing the SSR to be reset. No other SCI registers are affected by the STOP instruction. Table 6-2 illustrates how each type of reset affects each register in the SCI.

6.3.4 SCI Initialization

The correct way to initialize the SCI is as follows:

1. Hardware or software reset
2. Program SCI control registers
3. Configure SCI pins (at least one) as not general-purpose I/O

Figure 6-14 and Figure 6-15 show how to configure the bits in the SCI registers. Figure 6-14 is the basic initialization procedure showing which registers must be configured. (1) A hardware or software reset should be used to reset the SCI and prevent it from doing anything unexpected while it is being programmed. (2) Both the SCI interface control register and the clock control register must be configured for any operation using the SCI. (3) The pins to be used must then be selected to release the SCI from reset and (4) begin operation. If interrupts are to be used, the pins must be selected, and interrupts must be enabled and unmasked before the SCI will operate. The order does not matter; any one of these three requirements for interrupts can be used to finally enable the SCI.

Figure 6-15 shows the meaning of the individual bits in the SCR and SCCR. The figures below do not assume that interrupts will be used; they recommend selecting the appropriate pins to enable the SCI. Programs shown in Figures Figure 6-20, Figure 6-21, Figure 6-28, Figure 6-34, and Figure 6-36 control the SCI by enabling and disabling interrupts. Either method is acceptable.

Table 6-2 SCI Registers after Reset

Register Bit	Bit Mnemonic	Bit Number	Reset Type			
			HW Reset	SW Reset	IR Reset	ST Reset
SCR	SCKP	15	0	0	–	–
	STIR	14	0	0	–	–
	TMIE	13	0	0	–	–
	TIE	12	0	0	–	–
	RIE	11	0	0	–	–
	ILIE	10	0	0	–	–
	TE	9	0	0	–	–
	RE	8	0	0	–	–
	WOMS	7	0	0	–	–
	RWU	6	0	0	–	–
	WAKE	5	0	0	–	–
	SBK	4	0	0	–	–
	SSFTD	3	0	0	–	–
	WDS (2–0)	2–0	0	0	–	–
SSR	R8	7	0	0	0	0
	FE	6	0	0	0	0
	PE	5	0	0	0	0
	OR	4	0	0	0	0
	IDLE	3	0	0	0	0
	RDRF	2	0	0	0	0
	TDRE	1	1	1	1	1
	TRNE	0	1	1	1	1
SCCR	TCM	15	0	0	–	–
	RCM	14	0	0	–	–
	SCP	13	0	0	–	–
	COD	12	0	0	–	–
	CD (11–0)	11–0	0	0	–	–
SRX	SRX (23–0)	23–16, 15–8, 7–0	–	–	–	–
STX	STX (23–0)	23–0	–	–	–	–
SRSH	SRS (8–0)	8–0	–	–	–	–
STSH	STS (8–0)	8–0	–	–	–	–

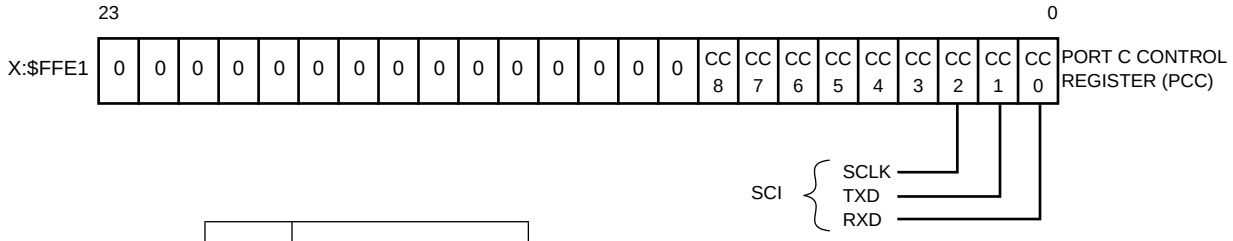
NOTES:
SRSH – SCI receive shift register, STSH – SCI transmit shift register
HW – Hardware reset is caused by asserting the external RESET pin.
SW – Software reset is caused by executing the RESET instruction.
IR – Individual reset is caused by clearing PCC (bits 0–2) (configured for general-purpose I/O).
ST – Stop reset is caused by executing the STOP instruction.
1 – The bit is set during the xx reset.
0 – The bit is cleared during the xx reset.
– – The bit is not changed during the xx reset.

Table 6-3 (a) through Table 6-4 (b) provide the settings for common baud rates for

1. PERFORM HARDWARE OR SOFTWARE RESET.

2. PROGRAM SCI CONTROL REGISTERS:
   a) SCI INTERFACE CONTROL REGISTER — X:$FFF0
   b) SCI CLOCK CONTROL REGISTER — X:$FFF2

3. CONFIGURE AT LEAST ONE PORT C CONTROL BIT AS SCI.



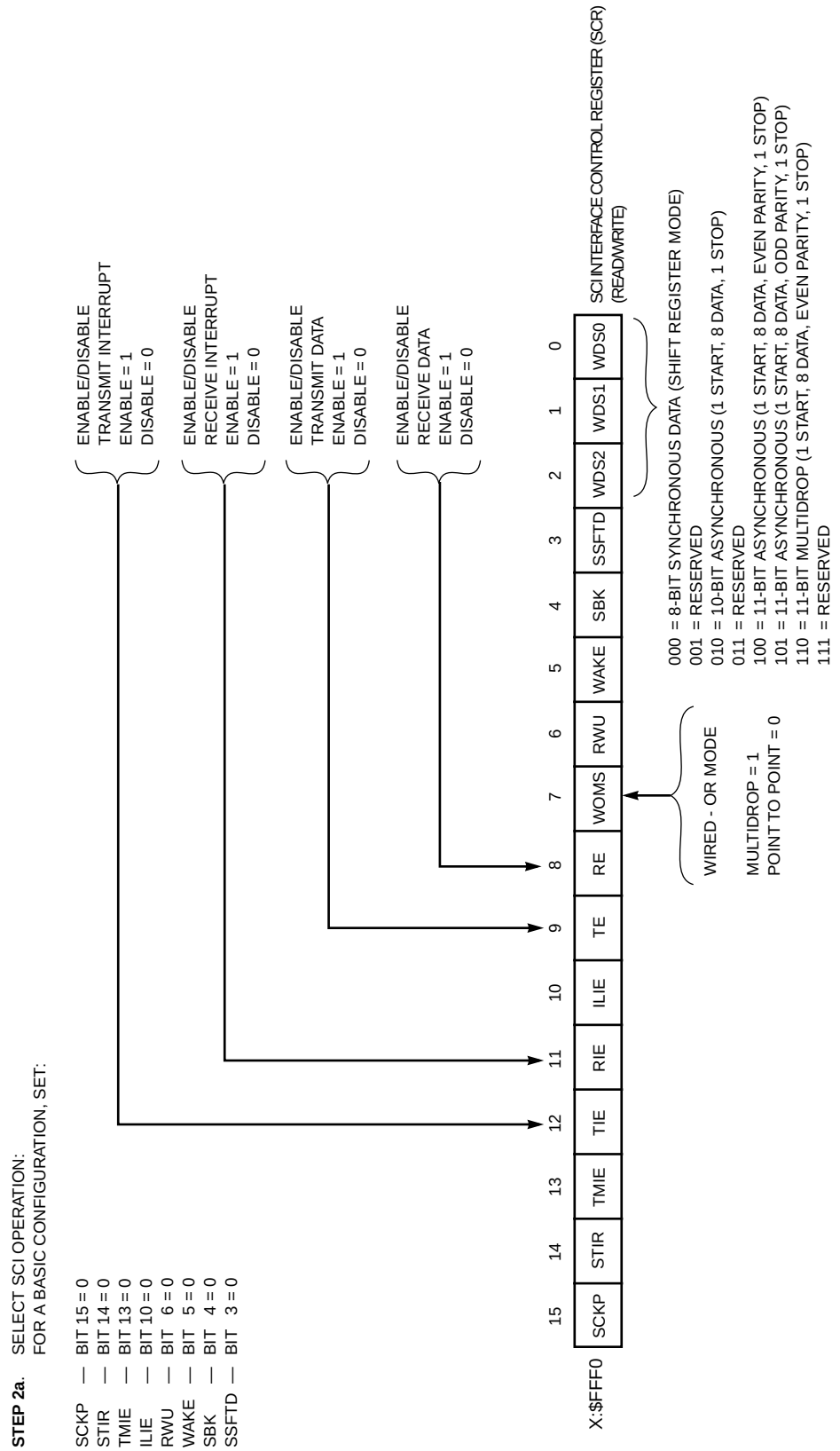
CCx	Function
0	GPIO
1	Serial Interface

4. SCI IS NOW ACTIVE.

Figure 6-14 SCI Initialization Procedure

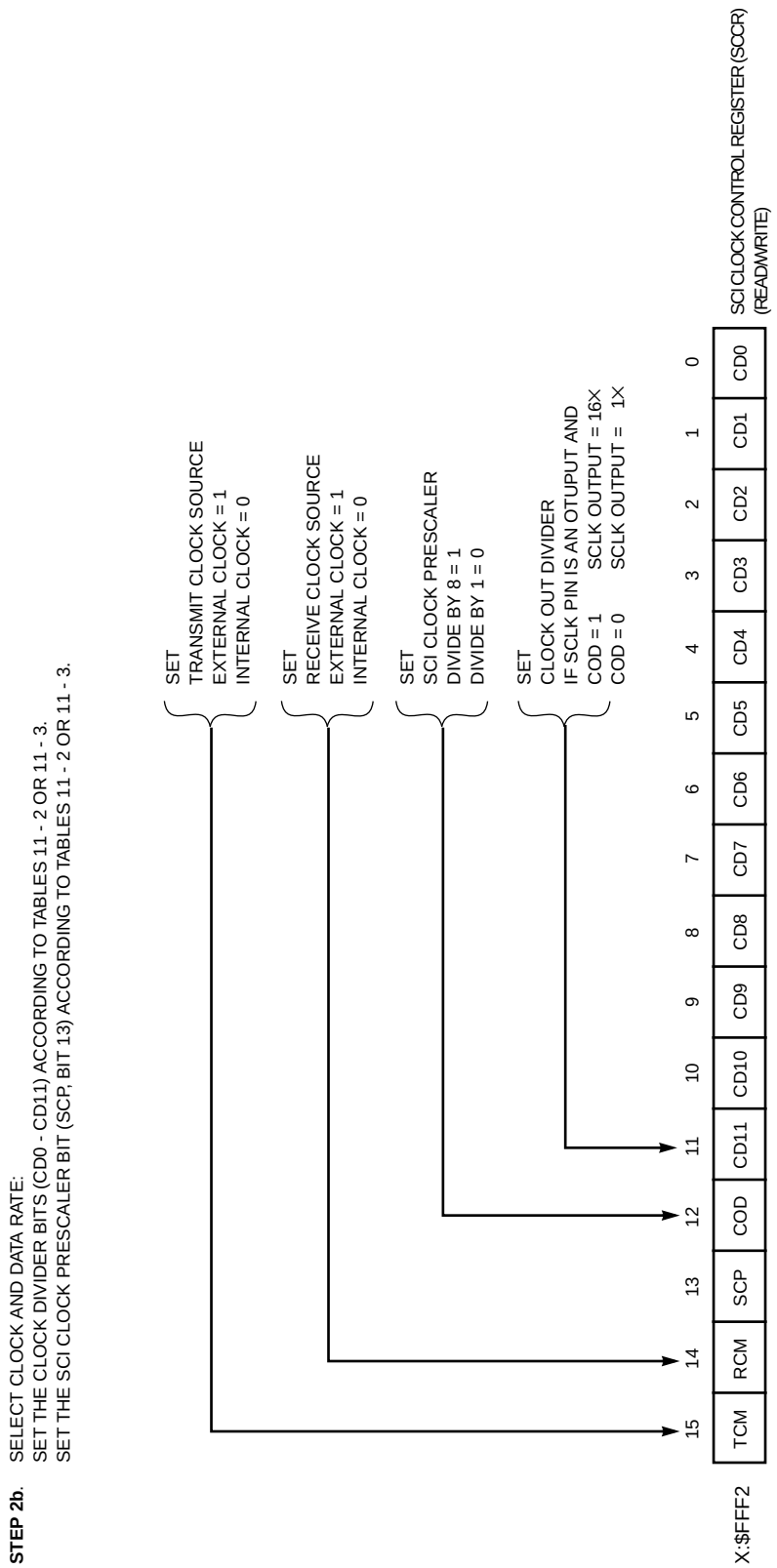
the SCI. The asynchronous SCI baud rates show a baud rate error for the fixed oscillator frequency (see Table 6-3 (a)). These small-percentage baud rate errors should allow most UARTs to synchronize. The synchronous applications usually require exact frequencies, which require that the crystal frequency be chosen carefully (see Table 6-4 (a) and Table 6-4 (b)).

An alternative to selecting the system clock to accommodate the SCI requirements is to provide an external clock to the SCI. For example, a 2.048 MHz bit rate requires a CPU clock of 32.768 MHz. An application may need a 40 MHz CPU clock and an external clock for the SCI.



Step 2a

Figure 6-15 SCI General Initialization Detail – Step 2 (Sheet 1 of 2)



Step 2b

Figure 6-15 SCI General Initialization Detail – Step 2 (Sheet 2 of 2)

Table 6-3 (a) Asynchronous SCI Bit Rates for a 40-MHz Crystal

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Bit Rate Error, Percent
625.0K	0	$000	0
56.0K	0	$00A	+1.46
38.4K	0	$00F	+1.72
19.2K	0	$020	-1.36
9600	0	$040	+0.16
8000	0	$04D	+0.15
4800	0	$081	+0.15
2400	1	$020	-1.38
1200	1	$040	+0.08
600	1	$081	0
300	1	$103	0

BPS= $f_0 \div \prod$ (64X (7(SCP) + 1) X (CD + 1)); f_0=40 MHz
SCP=0 or 1
CD=0 to $FFF

Table 6-3 (b) Frequencies for Exact Asynchronous SCI Bit Rates

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Crystal Frequency
9600	0	$040	39,936,000
4800	0	$081	39,936,000
2400	0	$103	39,936,000
1200	0	$207	39,936,000
300	0	$822	39,993,000
9600	1	$007	39,321,600
4800	1	$00F	39,321,600
2400	1	$01F	39,321,600
1200	1	$040	39,360,000
300	1	$103	39,936,000

f0=BPS X 64X (7(SCP) + 1)X(CD + 1))
SCP=0 or 1
CD=0 to $FFF

Table 6-4 (a) Synchronous SCI Bit Rates for a 32.768-MHz Crystal

Baud Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Baud Rate Error, Percent
4.096M	0	$000	0
128K	0	$01F	0
64K	0	$03F	0
56K	0	$048	-0.195
32K	0	$07F	0
16K	0	$0FF	0
8000	0	$1FF	0
4000	0	$3FF	0
2000	0	$7FF	0
1000	0	$FFF	0

BPS= $f_0 \div (8 \times (7(SCP) + 1) \times (CD + 1))$; f_0=32.768 MHz
SCP=0 or 1
CD=0 to $FFF

Table 6-4 (b) Frequencies for Exact Synchronous SCI Bit Rates

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Crystal Frequency
2.048M	0	$001	32.768 MHz
1.544M	0	$002	37.056 MHz
1.536M	0	$002	36.864 MHz

f_0=BPS $\times 8 \times (7(SCP) + 1) \times (CD + 1)$
SCP=0 or 1
CD=0 to $FFF

6.3.5 SCI Exceptions

The SCI can cause five different exceptions in the DSP (see Figure 6-53). These exceptions are as follows:

1. SCI Receive Data – caused by receive data register full with no receive error conditions existing. This error-free interrupt may use a fast interrupt service

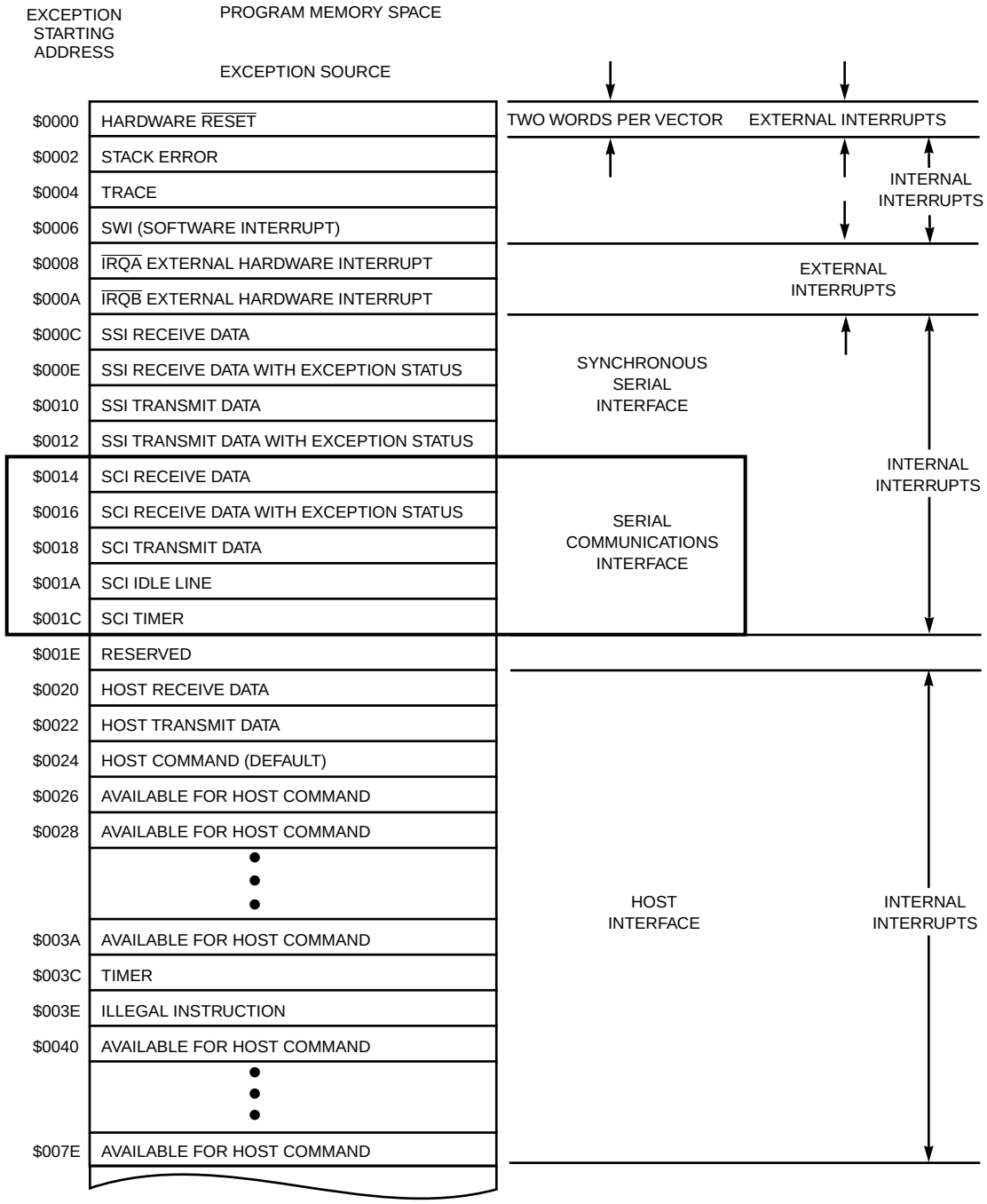


Figure 6-16 SCI Exception Vector Locations

routine for minimum overhead. This interrupt is enabled by SCR bit 11 (RIE).

2. SCI Receive Data with Exception Status – caused by receive data register full with a receiver error (parity, framing, or overrun error). The SCI status register must be read to clear the receiver error flag. A long interrupt service routine should be used to handle the error condition. This interrupt is enabled by SCR bit 11 (RIE).
3. SCI Transmit Data – caused by transmit data register empty. This error-free interrupt may use a fast interrupt service routine for minimum overhead. This interrupt is enabled by SCR bit 12 (TIE).
4. SCI Idle Line – occurs when the receive line enters the idle state (10 or 11 bits of ones). This interrupt is latched and then automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 10 (ILIE).
5. SCI Timer – caused by the baud rate counter underflowing. This interrupt is automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 13 (TMIE).

6.3.6 Synchronous Data

The synchronous mode (WDS=0, shift register mode) is designed to implement serial-to-parallel and parallel-to-serial conversions. This mode will directly interface to 8051/8096 synchronous (mode 0) buses as both a controller (master) or a peripheral (slave) and is compatible with the SSI mode if SCKP equals one. In synchronous mode, the clock is always common to the transmit and receive shift registers.

As a controller (synchronous master) shown in Figure 6-17, the DSP puts out a clock on the SCLK pin when data is present in the transmit shift register (a gated clock mode). The master mode is selected by choosing internal transmit and receive clocks (setting TCM and RCM=0). The example shows a 74HC165 parallel-to-serial shift register and 74HC164 serial-to-parallel shift register being used to convert eight bits of serial I/O to eight bits of parallel I/O. The load pulse latches eight bits into the 74HC165 and then SCLK shifts the RXD data into the SCI (these data bits are sample bits 0-7 in the timing diagram). At the same time, TXD shifts data out (B0-B7) to the 74HC164. When using the internal clock, data is transmitted when the transmit shift register is full. Data is valid on both edges of the output clock, which is compatible with an 8051 microprocessor. Received data is sampled in the middle of the clock low time if SCKP equals zero or in the middle of the clock high time if SCKP equals one. There is a window during which STX must be written with the next byte to be transmitted to prevent a gap between words. This

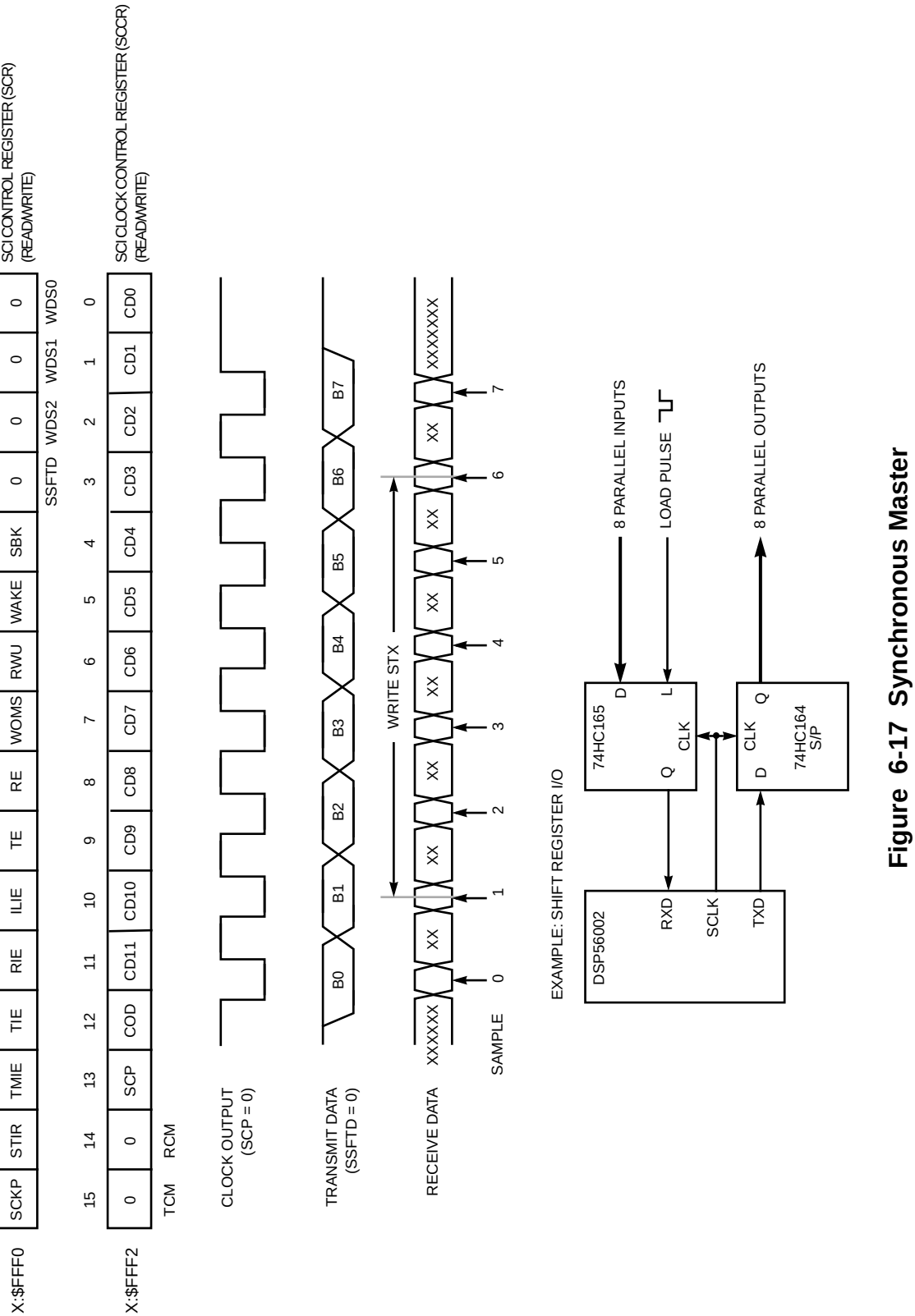


Figure 6-17 Synchronous Master

window is from the time TDRE goes high halfway into transmission of bit 1 until the middle of bit 6 (see Figure 6-19(a)).

As a peripheral (synchronous slave) shown in Figure 6-18, the DSP accepts an input clock from the SCLK pin. If SCKP equals zero, data is clocked in on the rising edge of SCLK, and data is clocked out on the falling edge of SCLK. If SCKP equals one, data is clocked in on the falling edge of SCLK, and data is clocked out on the rising edge of SCLK. The slave mode is selected by choosing external transmit and receive clocks (TCM and RCM=1). Since there is no frame signal, if a clock is missed due to noise or any other reason, the receiver will lose synchronization with the data without any error signal being generated. Detecting an error of this type can be done with an error detecting protocol or with external circuitry such as a watchdog timer. The simplest way to recover synchronization is to reset the SCI.

The timing diagram in Figure 6-18 shows transmit data in the normal driven mode. Bit B7 is essentially one-half SCI clock long ($T_{SCI}/2 + 1.5\ T_{EXTAL}$) The last data bit is truncated so that the pin is guaranteed to go to its reset state before the start of the next data word, thereby delimiting data words. The 1.5 crystal clock cycles provide sufficient hold time to satisfy most external logic requirements. The example diagram requires that the WOMS bit be set in the SCR to wired-OR RXD and TXD, which causes TXD to be three-stated when not transmitting. Collisions (two devices transmitting simultaneously) must be avoided with this circuit by using a protocol such as alternating transmit and receive periods. In the example, the 8051 is the master device because it controls the clock. There is a window during which STX must be written with the next byte to be transmitted to prevent the current word from being retransmitted. This window is from the time TDRE goes high, which is halfway into the transmission of bit 1, until the middle of bit 6 (see Figure 6-19(b)). Of course, this assumes the clock remains continuous – i.e., there is a second word. If the clock stops, the SCI stops.

The DSP is initially configured according to the protocol to either receive data or transmit data. If the protocol determines that the next data transfer will be a DSP transmit, the DSP will configure the SCI for transmit and load STX (or STXA). When the master starts SCLK, data will be ready and waiting. If the protocol determines that the next data transfer will be a DSP receive, the DSP will configure the SCI for receive and will either poll the SCI or enable interrupts. This methodology allows multiple slave processors to use the same data line. Selection of individual slave processors can be under protocol control or by multiplexing SCLK.

Note: TCM=0, RCM=1 and TCM=1,RCM=0 are not allowed in the synchronous mode. The results are undefined.

The assembly program shown in Figure 6-20 uses the SCI synchronous mode to transmit only the low byte of the Y data ROM contents. The program sets the reset vector to run the program after a hardware reset, puts the MOVEP instruction at the SCI transmit inter-

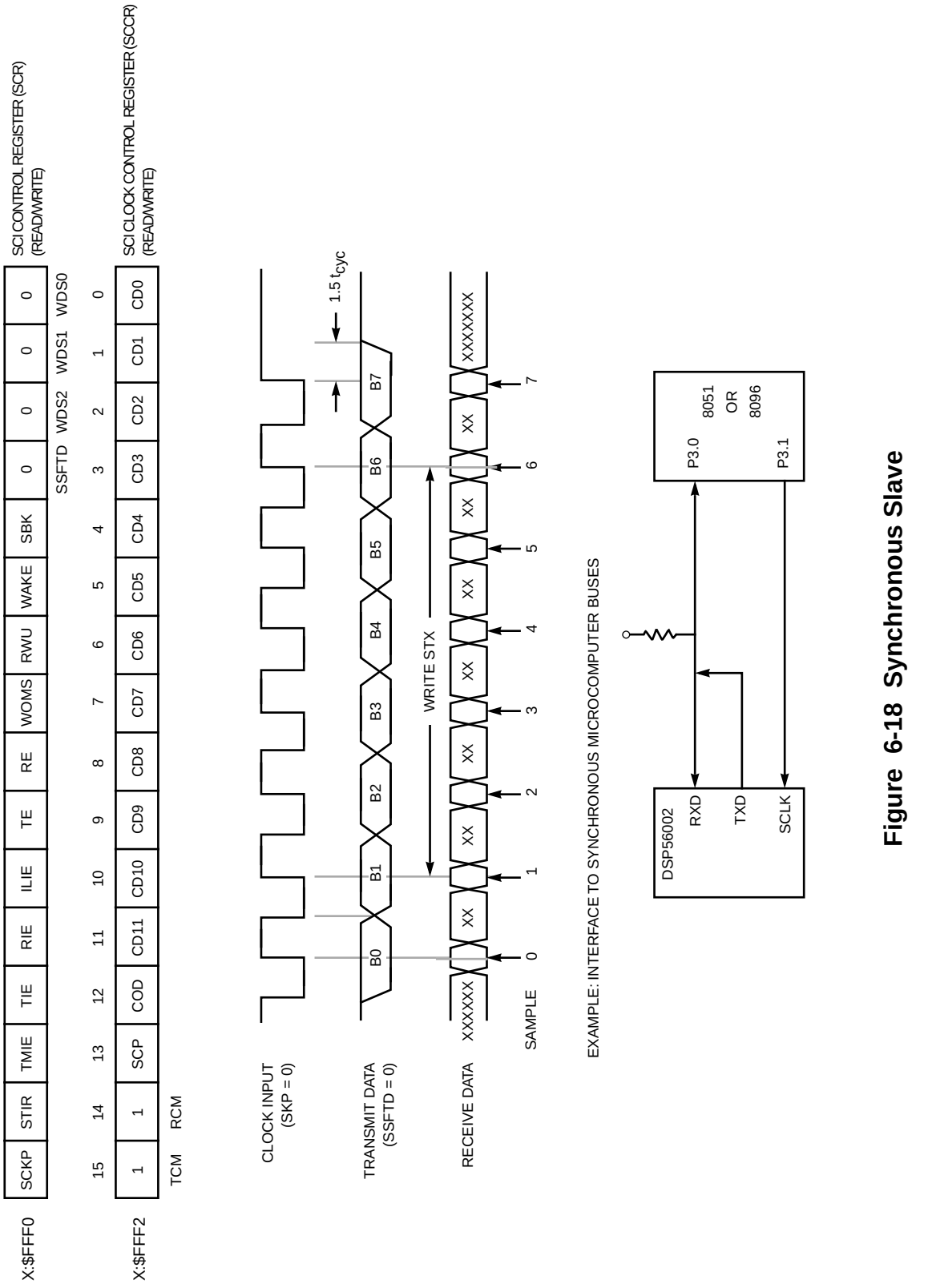


Figure 6-18 Synchronous Slave

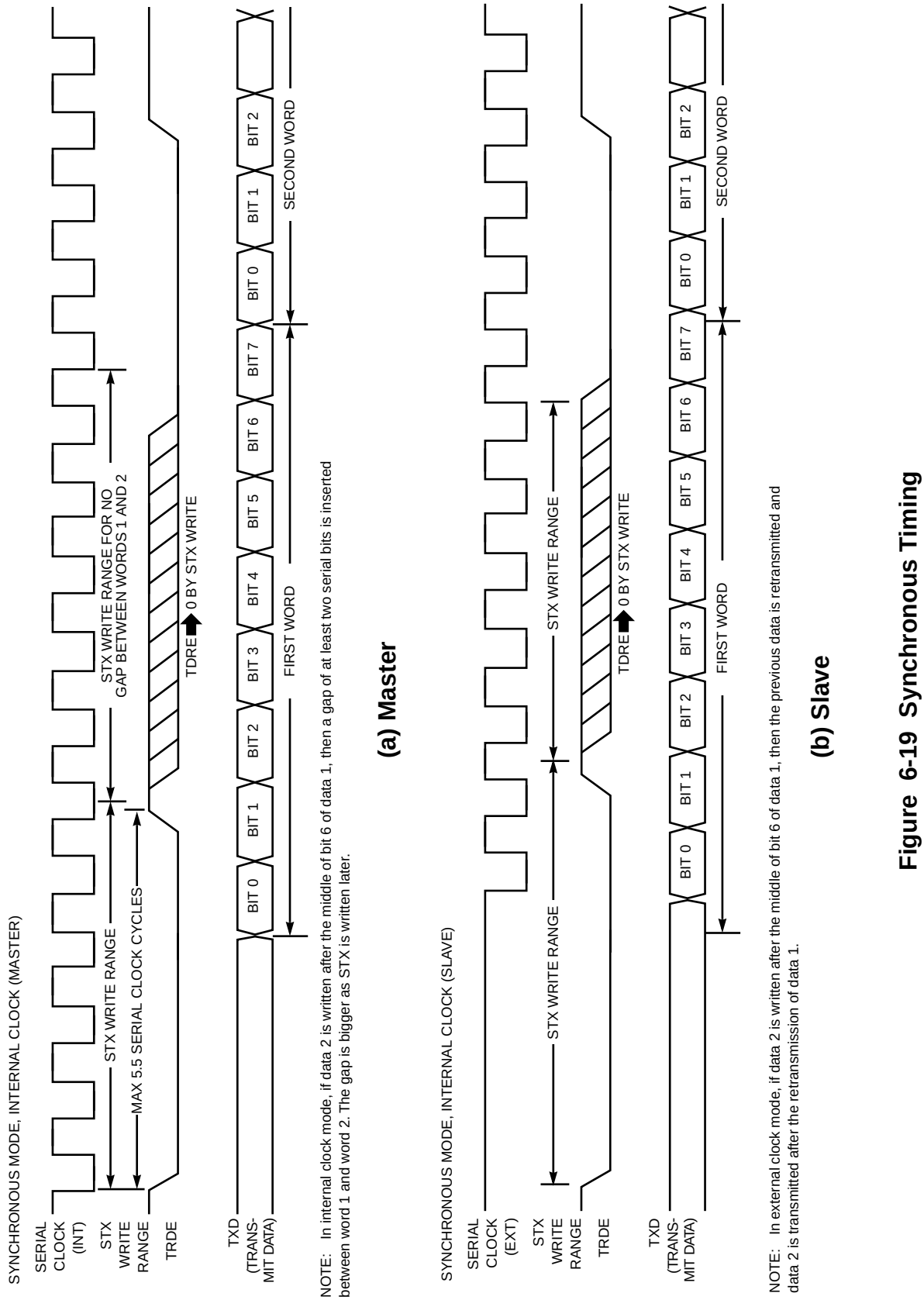


Figure 6-19 Synchronous Timing

rupt vector location, sets the memory wait states to zero, and configures the memory

```
        ORG     P:0             ;Reset vector
        JMP     $40             ;
        ORG     P:$18           ;SCI transmit interrupt vector
        MOVEP   Y:(R0)+,X:$FFF4 ;Transmit low byte of data

        ORG     P:$40
        MOVEP   #0,X:$FFFE      ;Clear BCR
        MOVE    #$100,R0        ;Data ROM start address
        MOVE    #$FF,M0         ;Size of data ROM - Wraps around at $200
        MOVEC   #6,OMR          ;Change operating mode to enable data ROM
        MOVEP   #$C000,X:$FFFF  ;Interrupt priority register
        MOVEP   #$1200,X:$FFF0  ;8-bit synchronous mode
        MOVEP   #7,X:$FFE1      ;Port C control register – enable SCI
        MOVEC   #0,SR           ;Unmask interrupts
LAB0    JMP     LAB0            ;Wait in loop for interrupts
```

Figure 6-20 SCI Synchronous Transmit

pointers, operating mode register, and the IPR.

The SCI is then configured and the interrupts are unmasked, which starts the data transfer. The jump-to-self instruction (LAB0 JMP LAB0) is used to wait while interrupts transfer the data.

The program shown in Figure 6-21 is the program for receiving data from the program presented in Figure 6-20. The program sets the reset vector to run the program after hardware reset, puts the MOVEP instruction to store the data in a circular buffer starting at $100 at the SCI receive interrupt vector location, puts another MOVEP instruction at the SCI receive interrupt vector location, sets the memory wait states to zero, and configures the memory pointers and IPR. The SCI is then configured and the interrupts are unmasked, which starts the data transfer. The jump-to-self instruction (LAB0 JMP LAB0) is used to wait while interrupts transfer the data.

6.3.7 Asynchronous Data

Asynchronous data uses a data format with embedded word sync, which allows an unsynchronized data clock to be synchronized with the word if the clock rate and number of bits per word is known. Thus, the clock can be generated by the receiver rather than requiring a separate clock signal. The transmitter and receiver both use an internal clock that is 16 × the data rate to allow the SCI to synchronize the data. The data format requires that each data byte have an additional start bit and stop bit. In addition, two of the word formats have a parity bit. The multidrop mode used when SCIs are on a common

```
        ORG     P:0             ;Reset vector
        JMP     $40             ;

        ORG     P:$14           ;SCI receive data vector
        MOVEP   X:$FFF4,Y:(R0)+ ;Receive low byte of data
        NOP                     ;Fast interrupt response

        MOVEP   X:$FFF1,X0      ;Receive with exception. Read status register
        MOVEP   X:$FFF4,Y:(R0)+ ;Receive low byte of data

        ORG     P:$40
        MOVEP   #0,X:$FFFE      ;Clear BCR
        MOVE    #$100,R0        ;Data ROM start address
        MOVE    #$FF,M0         ; Size of data ROM – wraps around at $200
        MOVEP   #$C000,X:$FFFF  ;Interrupt priority register
        MOVEP   #$900,X:$FFF0   ; 8-bit synchronous mode receive only
        MOVEP   #$C000,X:$FFF2  ;Clock control register external clock
        MOVEP   #7,X:$FFE1      ;Port C control register – enable SCI
        MOVEC   #0,SR           ;Unmask interrupts
LAB0    JMP     LAB0            ;Wait in loop for interrupts
```

Figure 6-21 SCI Synchronous Receive

bus has an additional data type bit. The SCI can operate in full-duplex or half-duplex modes since the transmitter and receiver are independent. The SCI transmitter and receiver can use either the internal clock (TCM=0 and/or RCM=0) or an external clock (TCM=1 and/or RCM=1) or a combination. If a combination is used, the transmitter and receiver can run at different data rates.

6.3.7.1 Asynchronous Data Reception

Figure 6-22 illustrates initializing the SCI data receiver for asynchronous data. The first step (1) resets the SCI to prevent the SCI from transmitting or receiving data. Step two (2) selects the desired operation by programming the SCR. As a minimum, the word format (WDS2, WDS1, and WDS0) must be selected, and (3) the receiver must be enabled (RE=1). If (4) interrupts are to be used, set RIE equals one. Use Table 6-3 (a) through Table 6-4 (b) to set (5) the baud rate (SCP and CD0–CD11 in the SCCR). Once the SCI is completely configured, it is enabled by (6) setting the RXD bit in the PCC.

The receiver is continually sampling RDX at the 16 × clock rate to find the idle-start-bit transition edge. When that edge is detected (1) the following eight or nine bits, depending on the mode, are clocked into the receive shift register (see Figure 6-23). Once a complete byte is received, (2) the character is latched into the SRX, and RDRF is set as well

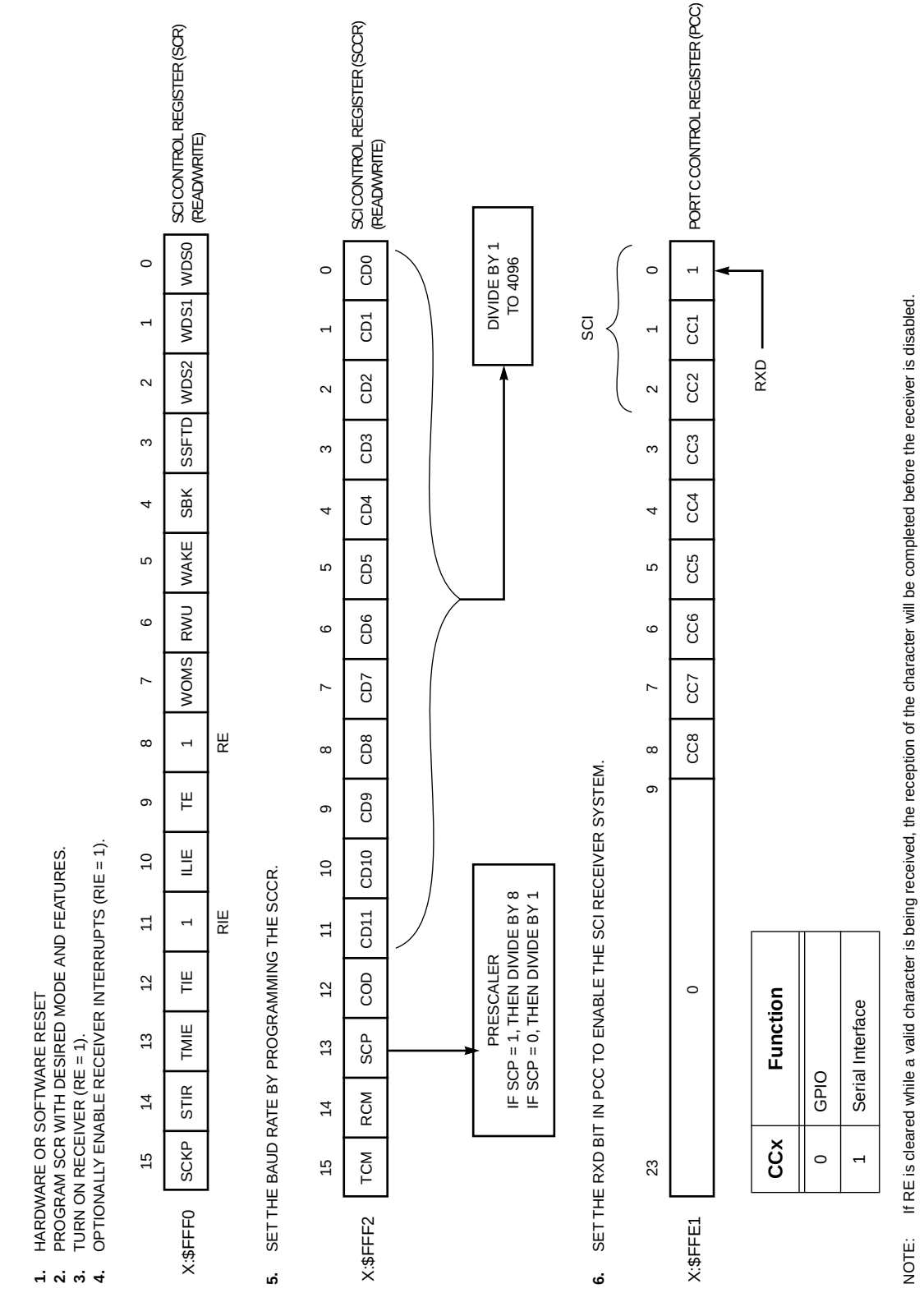


Figure 6-22 Asynchronous SCI Receiver Initialization

as the error flags, OR, PE, and FE. If (3) interrupts are enabled, an interrupt is generated.

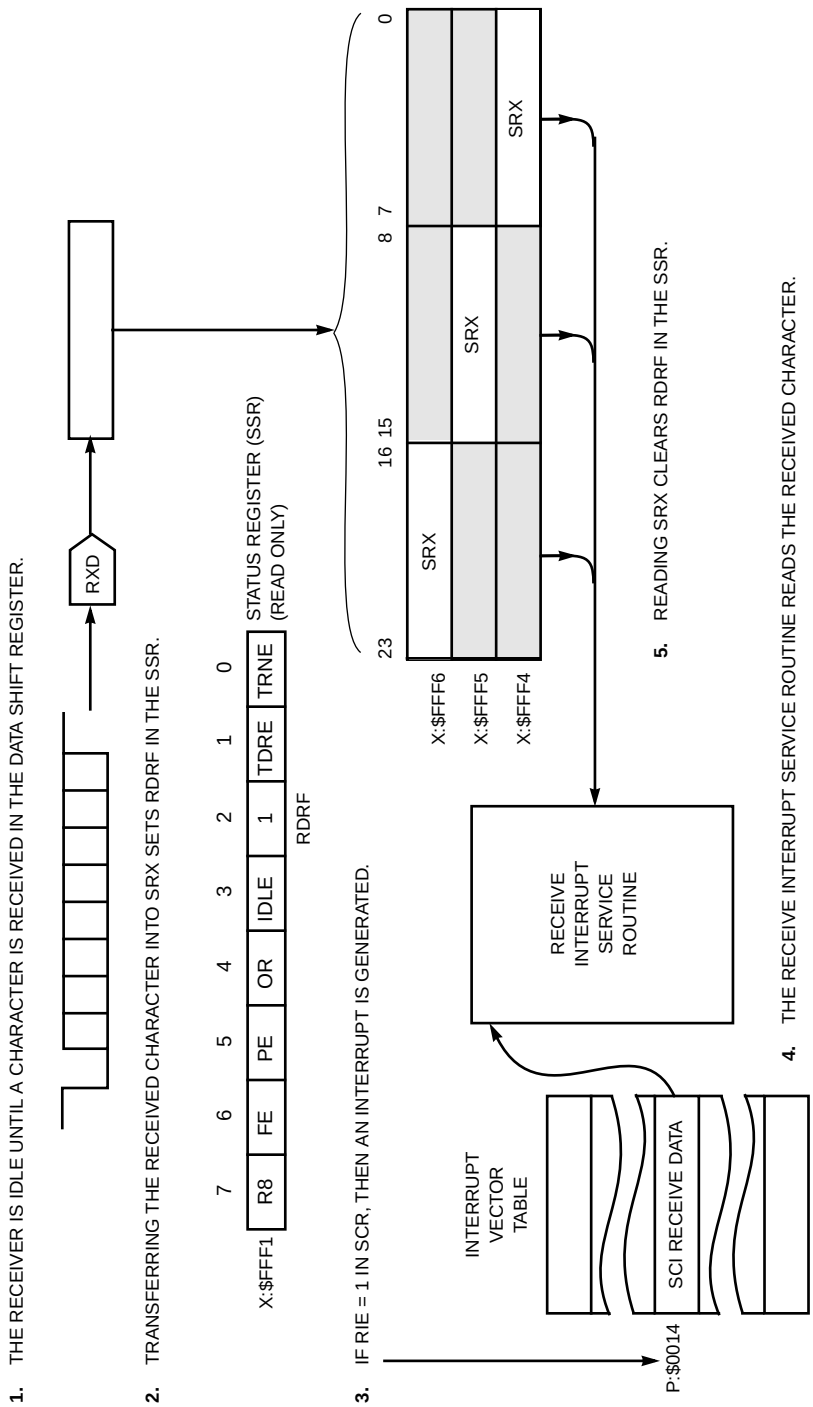


Figure 6-23 SCI Character Reception

The interrupt service routine, which can be a fast interrupt or a long interrupt, (4) reads the received character. Reading the SRX (5) automatically clears RDFR in the SSR and makes the SRX ready to receive another byte.

If (1) an FE, PE, or OR occurs while receiving data (see Figure 6-24), (2) RDRF is set because a character has been received; FE, PE, or OR is set in the SSR to indicate that an error was detected. Either (3) the SSR can be polled by software to look for errors, or (4) interrupts can be used to execute an interrupt service routine. This interrupt is different from the normal receive interrupt and is caused only by receive errors. The long interrupt service routine should (5) read the SSR to determine what error was detected and then (6) read the SRX to clear RDRF and all three error flags.

6.3.7.2 Asynchronous Data Transmission

Figure 6-25 illustrates initializing the SCI data transmitter for asynchronous data. The first step (1) resets the SCI to prevent the SCI from transmitting or receiving data. Step two (2) selects the desired operation by programming the SCR. As a minimum, the word format (WDS2, WDS1, and WDS0) must be selected, and (3) the transmitter must be enabled (TE=1). If (4) interrupts are to be used, set TIE equals one. Use Table 6-3 (a) through Table 6-4 (b) to set (5) the baud rate (SCP and CD0–CD11 in the SCCR). Once the SCI is completely configured, it can be enabled by (6) setting the TXD bit in the PCC. Transmission begins with (7) a preamble of ones.

If polling is used to transmit data (see Figure 6-26), the polling routine can look at either TDRE or TRNE to determine when to load another byte into STX. If TDRE is used (1), one byte may be loaded into STX. If TRNE is used (2), two bytes may be loaded into STX if enough time is allowed for the first byte to begin transmission (see **6.3.2.4.2**). If interrupts are used (3), then an interrupt is generated when STX is empty. The interrupt routine, which can be a fast interrupt or a long interrupt, writes (4) one byte into STX. If multidrop mode is being used and this byte is an address, STXA should be used instead of STX. Writing STX or STXA (5) clears TDRE in the SSR. When the transmit data shift register is empty (6), the byte in STX (or STXA) is latched into the transmit data shift register, TRNE is cleared, and TDRE is set.

There is a provision to send a break or preamble. A break (space) consists of a period of zeros with no start or stop bits that is as long or longer than a character frame. A preamble (mark) is an inverted break. A preamble of 10 or 11 ones (depending on the word length selected by WDS2, WDS1, and WDS0) can be sent with the following procedure (see Figure 6-27). (1) Write the last byte to STX and (2) wait for TDRE equals one. This is the byte that will be transmitted immediately before the preamble. (3) Clear TE and then again set it to one. Momentarily clearing TE causes the output to go high for one character frame.

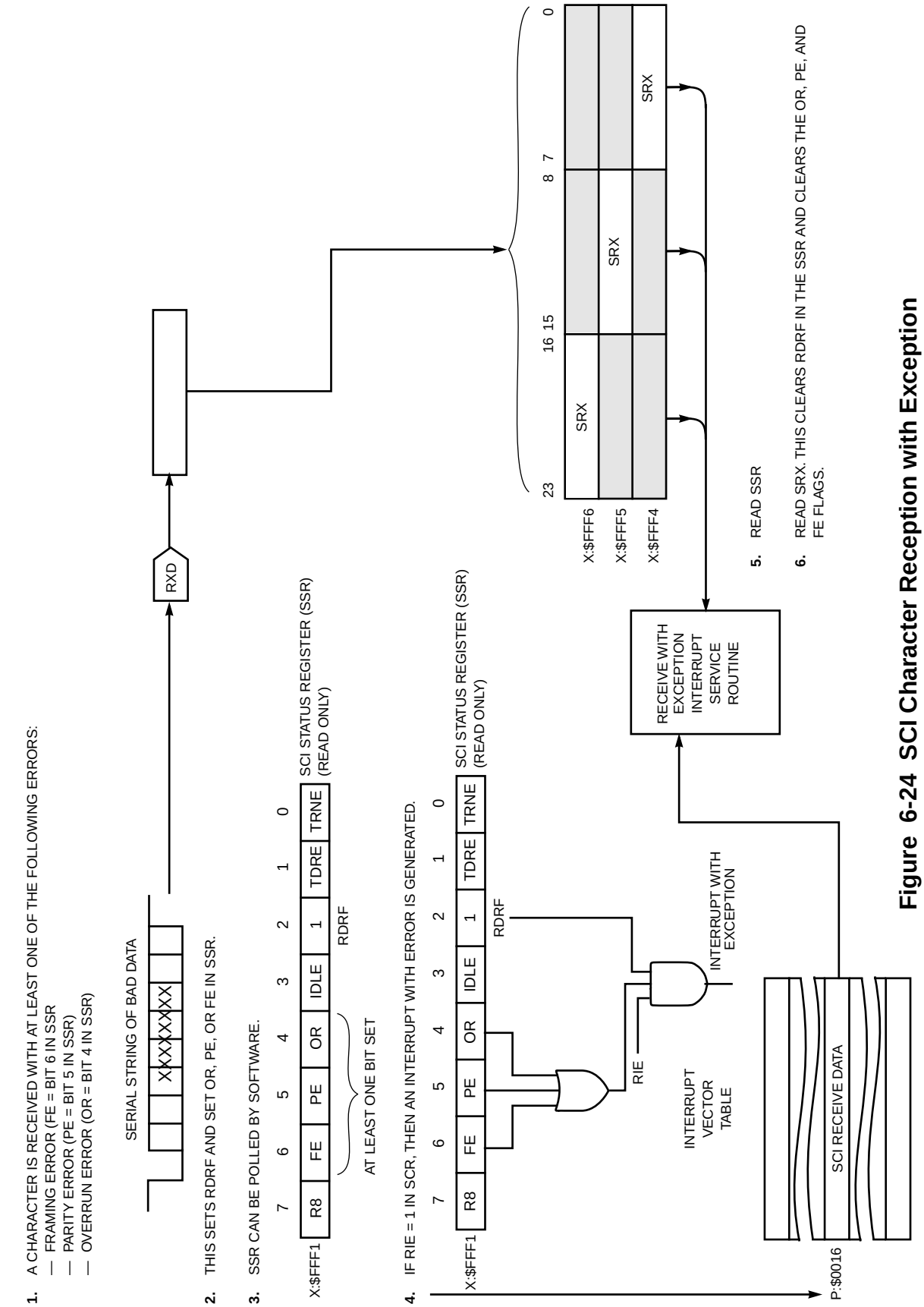


Figure 6-24 SCI Character Reception with Exception

If TE remains cleared for a longer period, the output will remain high for an even number

1. HARDWARE OR SOFTWARE RESET
2. PROGRAM SCR WITH DESIRED MODE AND FEATURES.
3. TURN ON TRANSMITTER (TE = 1).
4. OPTIONALLY ENABLE TRANSMITTER INTERRUPTS (TIE = 1).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
X:$FFF0	SCKP	STIR	TMIE	1	RIE	ILIE	1	RE	WOMS	RWU	WAKE	SBK	SSFTD	WDS2	WDS1	WDS0	SCI CONTROL REGISTER (SCR) (READ/WRITE)
				TIE			TE										

5. SET THE SCI CLOCK PRESCALER BIT AND THE CLOCK DIVIDER BITS IN THE SCCR.
6. SET THE TXD BIT IN PCC TO ENABLE THE SCI TRANSMITTER SYSTEM.

													SCI		
	23	9	8	7	6	5	4	3	2	1	0				
X:$FFE1	0		CC8	CC7	CC6	CC5	CC4	CC3	CC2	1	CC0	PORT C CONTROL REGISTER (PCC)			
										TXD					

CCx	Function
0	GPIO
1	Serial Interface

7. THE TRANSMITTER WILL FIRST BROADCAST A PREAMBLE OF ONES BEFORE BEGINNING DATA TRANSMISSION:
   10 ONES WILL BE TRANSMITTED FOR THE 10-BIT ASYNCHRONOUS MODE.
   11 ONES WILL BE TRANSMITTED FOR THE 11-BIT ASYNCHRONOUS MODE.

NOTE: If TE is cleared while transmitting a character, the transmission of the character will be completed before the transmitter is disabled.

Figure 6-25 Asynchronous SCI Transmitter Initialization

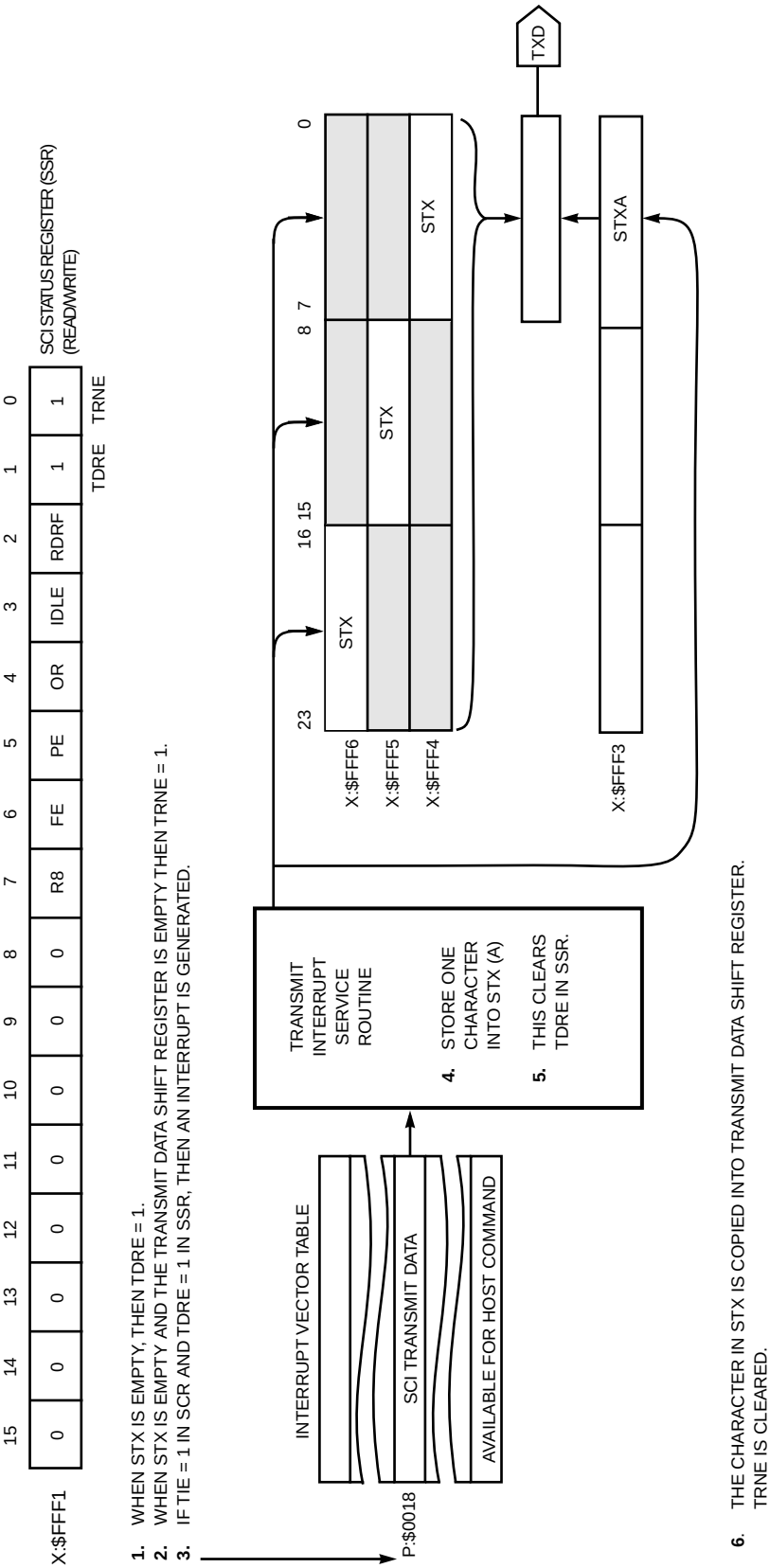
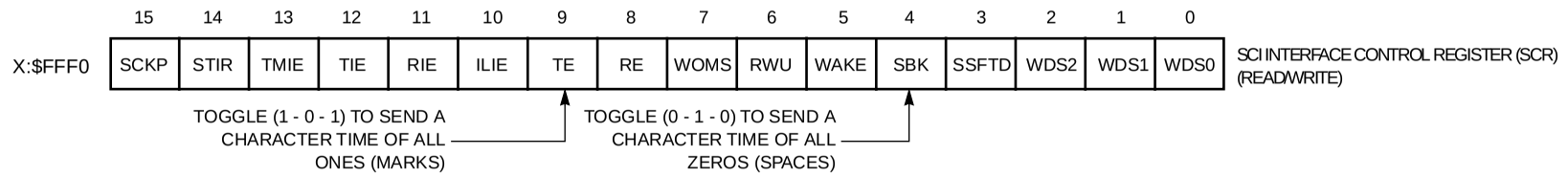


Figure 6-26 Asynchronous SCI Character Transmission

of character frames until TE is set. (4) Write the first byte to follow the preamble into SRX



- 10 OR 11 ONES/ZEROS WILL BE SENT DEPENDING ON THE WORD LENGTH SPECIFIED BY WDS2, WDS1, WDS0.

MARKS (ONES)

1. WRITE THE LAST BYTE TO STX.
2. WAIT FOR TRDE = 1. THE LAST BYTE IS NOW IN THE TRANSMIT SHIFT REGISTER.
3. CLEAR TE AND SET BACK TO ONE. THIS QUEUES THE PREAMBLE TO FOLLOW THE LAST BYTE.
4. WRITE THE FIRST BYTE TO FOLLOW THE PREAMBLE INTO SRX.

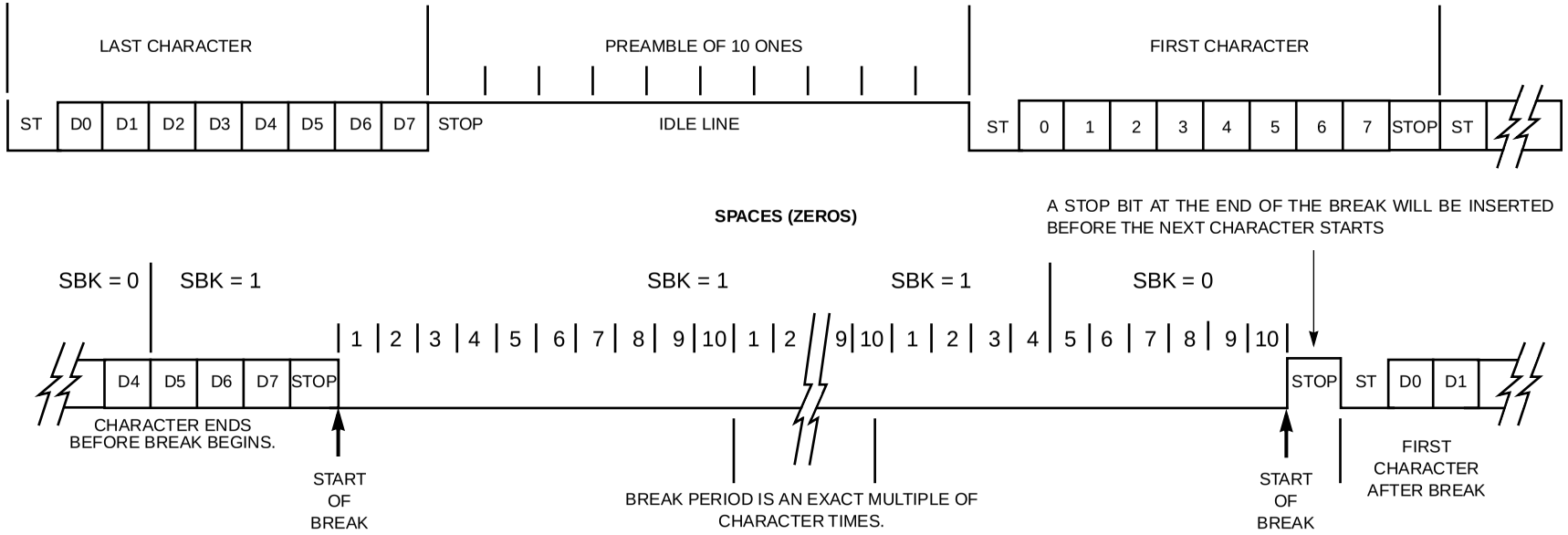


Figure 6-27 Transmitting Marks and Spaces

before the preamble is complete and resume normal transmission. Sending a break follows the same procedure except that instead of clearing TE, SBK is set in the SCR to send breaks and then reset to resume normal data transmission.

The example presented in Figure 6-28 uses the SCI in the asynchronous mode to transfer data into buffers. Interrupts are used, allowing the DSP to perform other tasks while the data transfer is occurring. This program can be tested by connecting the SCI transmit and receive pins. Equates are used for convenience and readability.

The program sets the reset vector to run the program after reset, puts a MOVEP instruction at the SCI receive interrupt vector location, and puts a MOVEP and BCLR at the SCI transmit interrupt vector location so that, after transmitting a byte, the transmitter is disabled until another byte is ready for transmission. The SCI is initialized by setting the interrupt level, which configures the SCR and SCCR, and then is enabled by writing the PCC. The main program begins by enabling interrupts, which allows data to be received. Data is transmitted by moving a byte of data to the transmit register and by enabling interrupts. The jump-to-self instruction (SEND JMP SEND) is used to wait while interrupts transfer the data.

```
;*********************************************************************************************
;       SCI ASYNC WITH INTERRUPTS AND SINGLE BYTE BUFFERS*
;*********************************************************************************************
;***********************************************
;       SCI and other EQUATES*
;***********************************************
START   EQU     $0040   ;Start of program
PCC     EQU     $FFE1   ;Port C control register
SCR     EQU     $FFF0   ;SCI interface control register
SCCR    EQU     $FFF2   ;SCI clock control register
SRX     EQU     $FFF4   ;SCI receive register
STX     EQU     $FFF4   ;SCI transmit register
BCR     EQU     $FFFE   ;Bus control register
IPR     EQU     $FFFF   ;Interrupt priority register
RXBUF   EQU     $100    ;Receive buffer
TXBUF   EQU     $200    ;Transmit buffer
```

Figure 6-28 SCI Asynchronous Transmit/Receive Example (Sheet 1 of 3)

```
;***********************************************
```

```
;       RESET VECTOR*
;************************************************
                ORG         P:$0000
                JMP         START
;************************************************
;       SCI RECEIVE INTERRUPT VECTOR*
;************************************************
                ORG         P:$0014         ;Load the SCI RX interrupt vectors
                MOVEP       X:SRX,Y:(R0)+   ;Put the received byte in the receive
                                            ;buffer. This receive routine is
                                            ;implemented as a fast interrupt.
;************************************************
;       SCI TRANSMIT INTERRUPT VECTOR*
;************************************************
                ORG         P:$0018         ;Load the SCI TX interrupt vectors
                MOVEP       X:(R3)+,X:STX   ;Transmit a byte and
                                            ;increment the pointer in the
                                            ;transmit buffer.
                BCLR        #12,X:SCR       ;Disable transmit interrupts
;*********************************************************************************************
;       INITIALIZE THE SCI PORT AND RX, TX BUFFER POINTERS*
;*********************************************************************************************
                ORG         P:START         ;Start the program at location $40
                ORI         #$03,MR         ;Mask interrupts temporarily
                MOVEP       #$C000,X:IPR    ;Set interrupt priority to 2
                MOVEP       #$0B02,X:SCR    ;Disable TX, enable RX interrupts
                                            ;Enable transmitter, receiver
                                            ;Point to point
                                            ;10-bit asynchronous
                                            ;(1 start, 8 data, 1 stop)
                MOVEP       #$0022,X:SCCR;Use internal TX, RX clocks
                                            ;9600 BPS
                MOVEP       #>$03,X:PCC     ;Select pins TXD and RXD for SCI
                MOVE        RXBUF,R0        ;Initialize the receive buffer
                MOVE        TXBUF,R3        ;Initialize the transmit buffer
```

Figure 6-28 SCI Asynchronous Transmit/Receive Example (Sheet 2 of 3)

```
;************************************************
```

```
;       MAIN PROGRAM*
;***********************************************
                ANDI    #$FC,MR         ;Re-enable interrupts
                MOVE    #>$41,X:(R3)    ;Move a byte to the transmit buffer
                MOVE    R0,X:(R3)
                BSET    #12,X:SCR       ;and enable interrupts so it
                                        ;will be transmitted
SEND            JMP     SEND            ;Normally something more useful
                                        ;would be put here.
                END                     ;End of example.
```

Figure 6-28 SCI Asynchronous Transmit/Receive Example (Sheet 3 of 3)

6.3.8 Multidrop

Multidrop is a special case of asynchronous data transfer. The key difference is that a protocol is used to allow networking transmitters and receivers on a single data-transmission line. Interprocessor messages in a multidrop network typically begin with a destination address. All receivers check for an address match at the start of each message. Receivers with no address match can ignore the remainder of the message and use a wakeup mode to enable the receiver at the start of the next message. Receivers with an address match can receive the message and optionally transmit an acknowledgment to the sender. The particular message format and protocol used are determined by the user's software. These message formats include point-to-point, bus, token-ring, and custom configurations. The SCI multidrop network is compatible with other leading microprocessors.

Figure 6-29 shows a multidrop system with one master and N slaves. The multidrop mode is selected by setting WDS2 equals one, WDS1 equals one, and WDS0 equals zero. One possible protocol is to have a preamble or idle line between messages, followed by an address and then a message. The idle line causes the slaves to wake up and compare the address with their own address. If the addresses match, the slave receives the message. If the addresses do not match, the slave ignores the message and goes back to sleep. It is also possible to generate an interrupt when an address is received, eliminating the need for idle time between consecutive messages and addresses. It is also possible for each slave to look for more than one address, which allows each slave to respond to individual messages as well as broadcast messages (e.g., a global reset).

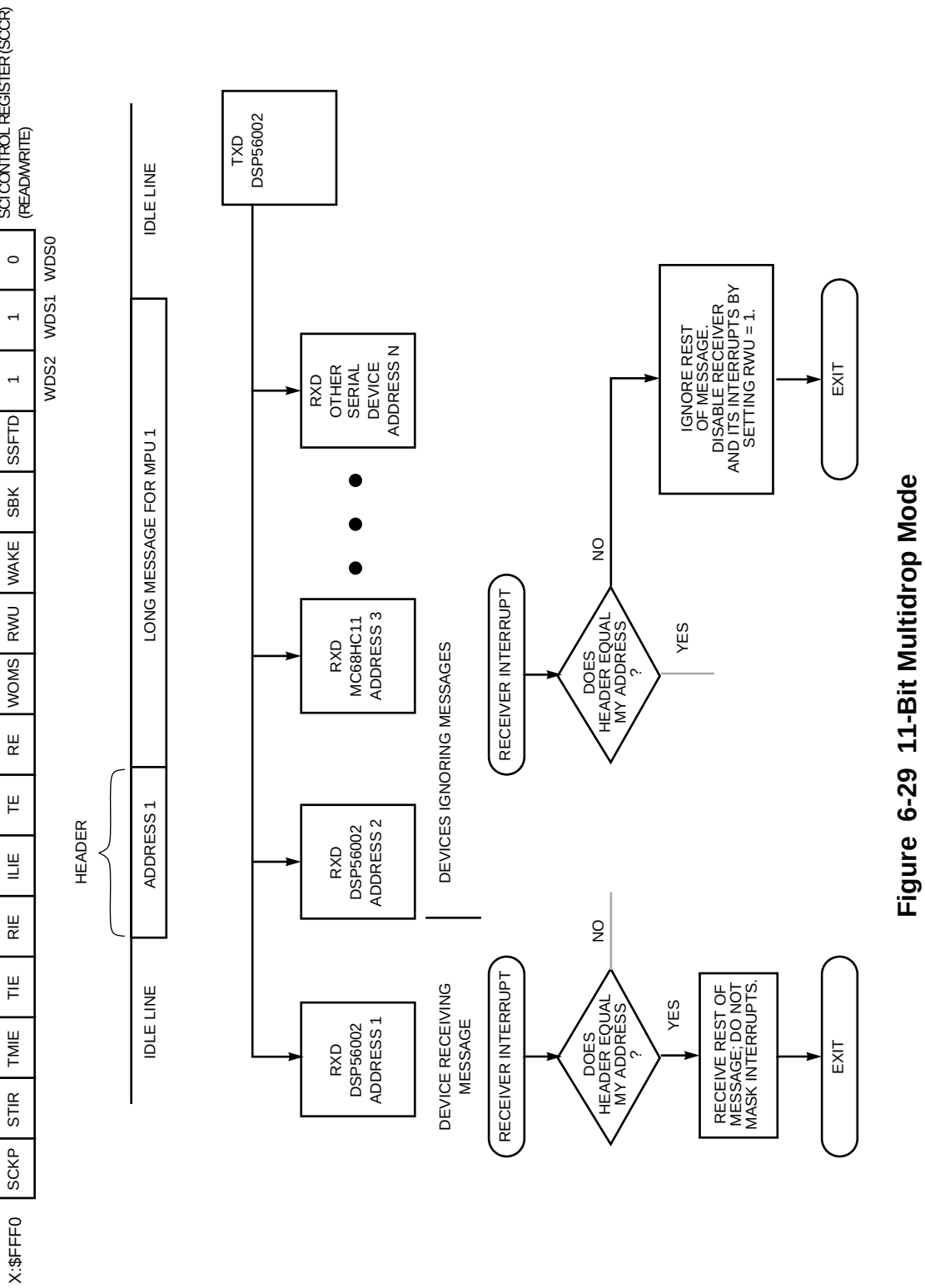


Figure 6-29 11-Bit Multidrop Mode

6.3.8.1 Transmitting Data and Address Characters

Transmitting data and address when the multidrop mode is selected is shown in Figure 6-30. The output sequence shown is idle line, data/address, and the next character. In both cases, an “A” is being transmitted. To send data, TE must be toggled to send the idle line, and then “A” must be sent to STX. Sending the “A” to the STX sets the ninth bit in the frame to zero, which indicates that this frame contains data. If the “A” is sent to STXA instead, the ninth bit in the frame is set to a one, which indicates that this frame contains an address.

6.3.8.2 Wired-OR Mode

Building a multidrop bus network requires connecting multiple transmitters to a common wire. The wired-OR mode allows this to be done without damaging the transmitters when the transmitters are not in use. A protocol is still needed to prevent two transmitters from simultaneously driving the bus. The SCI multidrop word format provides an address field to support this protocol. Figure 6-31 shows a multidrop configuration using wired-OR (set bit 7 of the SCR). The protocol shown consists of an idle line between messages; each message begins with an address character. The message can be any length, depending on the protocol. Each processor in this system has one address that it responds to although each processor can be programmed to respond to more than one address.

6.3.8.3 Idle Line Wakeup

A wakeup mode frees a DSP from reading messages intended for other processors. The usual operational procedure is for each DSP to suspend SCI reception (the DSP can continue processing) until the beginning of a message. Each DSP compares the address in the message header with the DSPs address. If the addresses do not match, the SCI again suspends reception until the next address. If the address matches, the DSP will read and process the message and then suspend reception until the next address.

The idle line wakeup mode wakes up the SCI to read a message before the first character arrives. This mode allows the message to be in any format.

Figure 6-32 shows how to configure the SCI to detect and respond to an idle line. The word format chosen (WDS2, WDS1, and WDS0 in the SCR) must be asynchronous. The WAKE bit must be clear to select idle line wakeup, and RWU must be set to put the SCI to “sleep” and enable the wakeup function. RIE should be set if interrupts are to be used to receive data. If processing must occur when the idle line is first detected, ILIE should be set. The current message is followed by one or more data frames of ones (10 or 11 bits each, depending on which word format is used), which are detected as an idle line. If the word format is multidrop (an 11-bit code), after the 11 ones, the receiver determines the line is idle and (1) clears the RWU, enabling the receiver. The IDLE bit (2) and an internal flag SRIINT (3) are set, indicating the line is idle. The SCI is now ready to receive messages; however, nothing more will happen until the next start bit unless (4) ILIE is set.

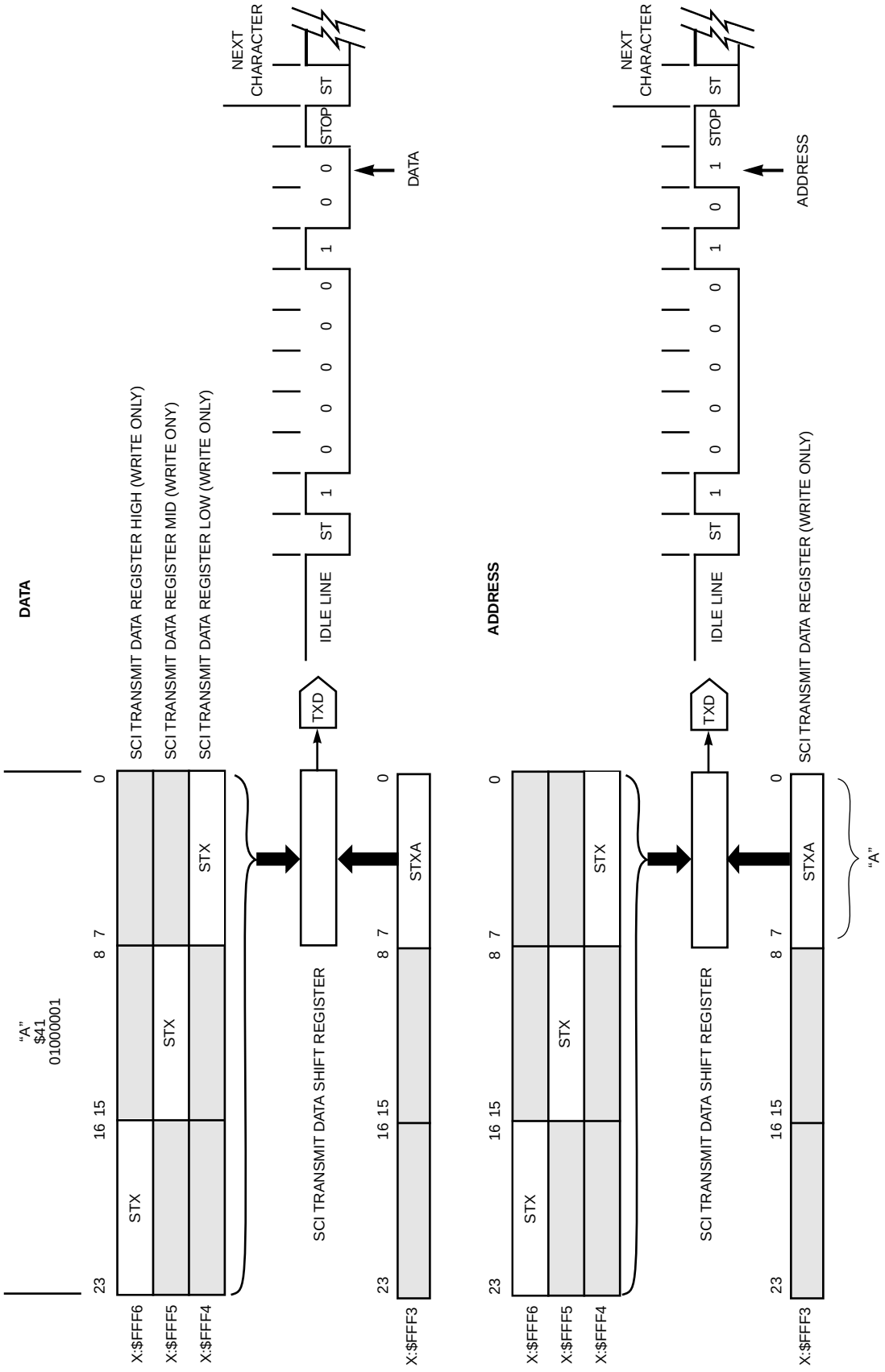


Figure 6-30 Transmitting Data and Address Characters

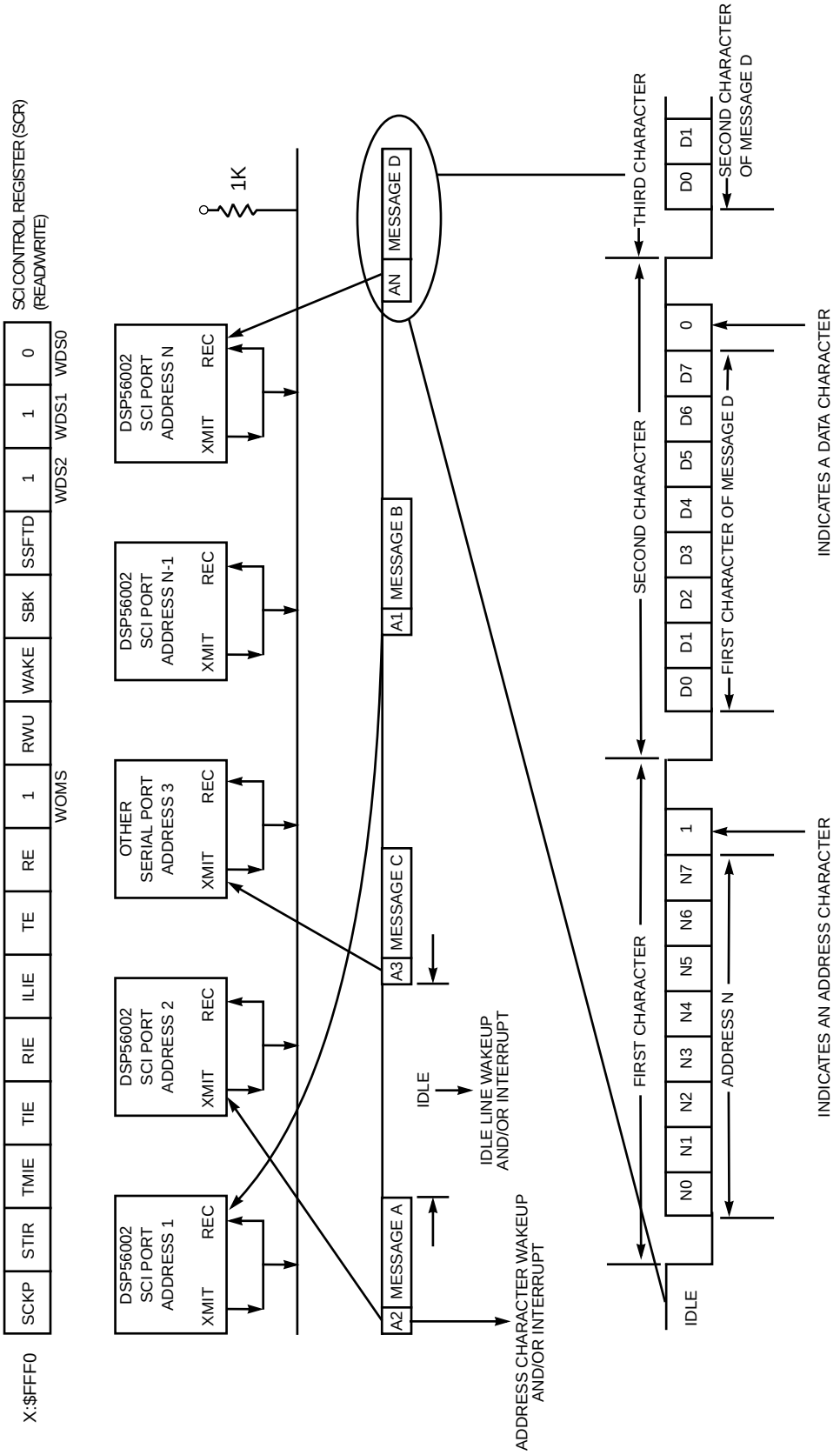


Figure 6-31 Wired-OR Mode

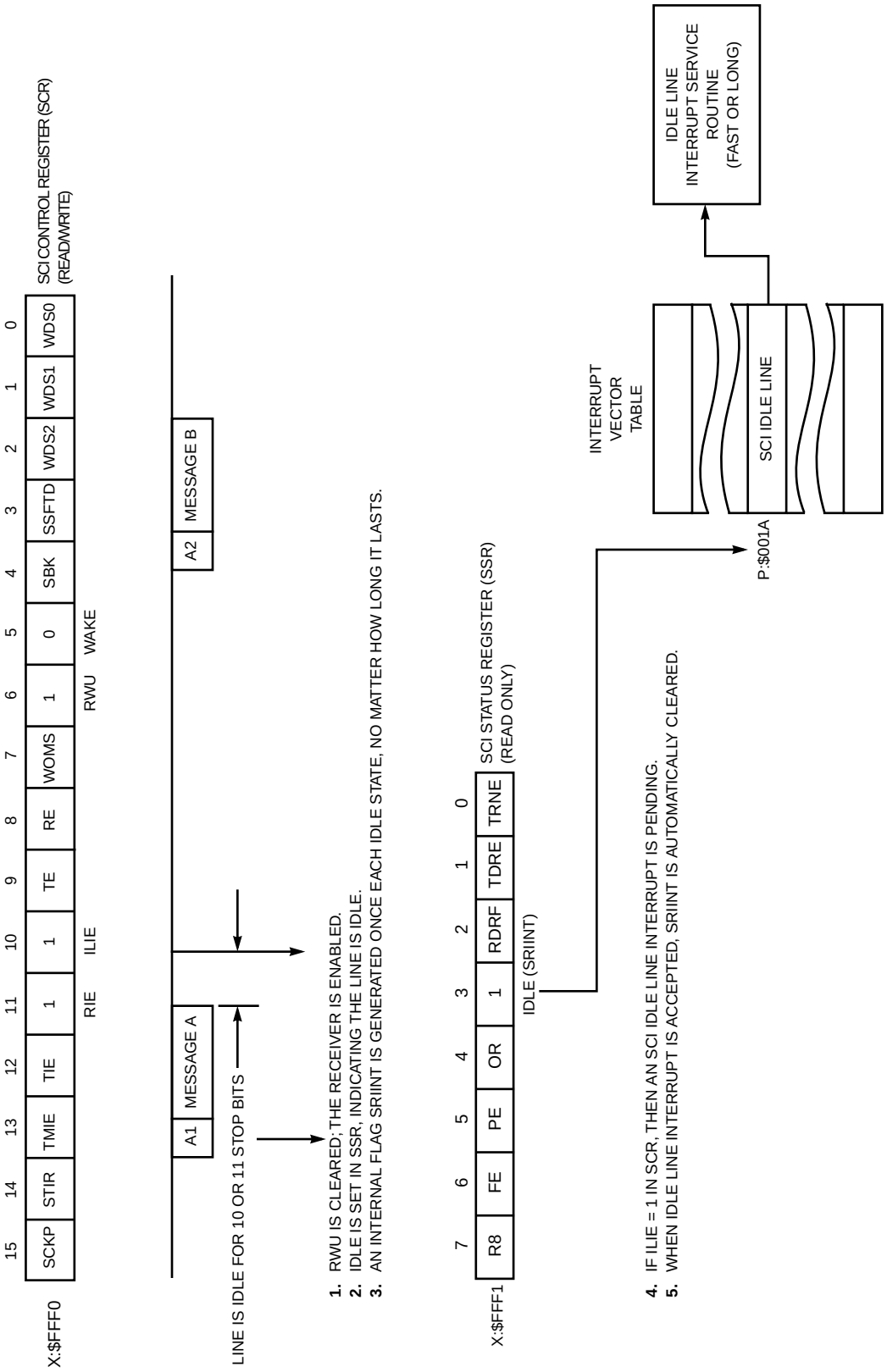


Figure 6-32 Idle Line Wakeup

If ILIE is set, an SCI idle line interrupt will be recognized as pending. When the idle line

interrupt is recognized (5), SRIINT is automatically cleared, and the SCI waits for the first start bit of the next character. Since RIE was set, when the first character is received, an SCI receive data interrupt (or SCI receive data with exception status interrupt if an error is detected) will be recognized as pending. When the receiver has processed the message and is ready to wait for another idle line, RWU must be set to one again.

6.3.8.4 Address Mode Wakeup

The purpose and basic operational procedure for address mode wakeup is the same as idle line wakeup. The difference is that address mode wakeup re-enables the SCI when the ninth bit in a character is set to one (if cleared, this bit marks a character as data; if set, an address). As a result, an idle line is not needed, which eliminates the dead time between messages. If the protocol is such that the address byte is not needed or is not wanted in the first byte of the message, a data byte can be written to STXA at the beginning of each message. It is not essential that the first byte of the message contain an address; it is essential that the start of a new message is indicated by setting the ninth bit to one using STXA.

Figure 6-33 shows how to configure the SCI to detect and respond to an address character. The word format chosen (WDS2, WDS1, and WDS0 in the SCR) must be an asynchronous word format. The WAKE bit must be set to select address mode wakeup and RWU must be set to put the SCI to “sleep” and enable the wakeup function. RIE should be set if interrupts are to be used to receive data. (1) When an address character (ninth bit=1) is received, then R8 is set to one in the SSR, and RWU is cleared. Clearing RWU re-enables the SCI receiver. Since (2) RIE was set in this example, when the first character is received, an SCI receive data interrupt (or SCI receive data with exception status interrupt if an error is detected) will be recognized as pending. When the receiver is ready to wait for another address character, RWU must be set to one again.

6.3.8.5 Multidrop Example

The program shown in Figure 6-34 configures the SCI as a multidrop master transmitter and slave receiver (using wakeup on address bit) that uses interrupts to transmit data from a circular buffer and to receive data into a different circular buffer. This program can be run with the I/O pins (RXD and TXD) connected and with a pullup resistor for test purposes.

The program starts by setting equates for convenience and clarity and then points the reset vector to the start of the program. The receive and transmit interrupt vector locations have JSRs forming long interrupts because the multidrop protocol and circular buffers require more than two instructions for maintenance. Byte packing and unpacking are not used in this example. The SRX and STX registers are equated to $FFF4, causing only the LSB of the 24-bit DSP word to be used for SCI data. The SCI is then initialized as wired-OR, multidrop, and using interrupts. The SCI

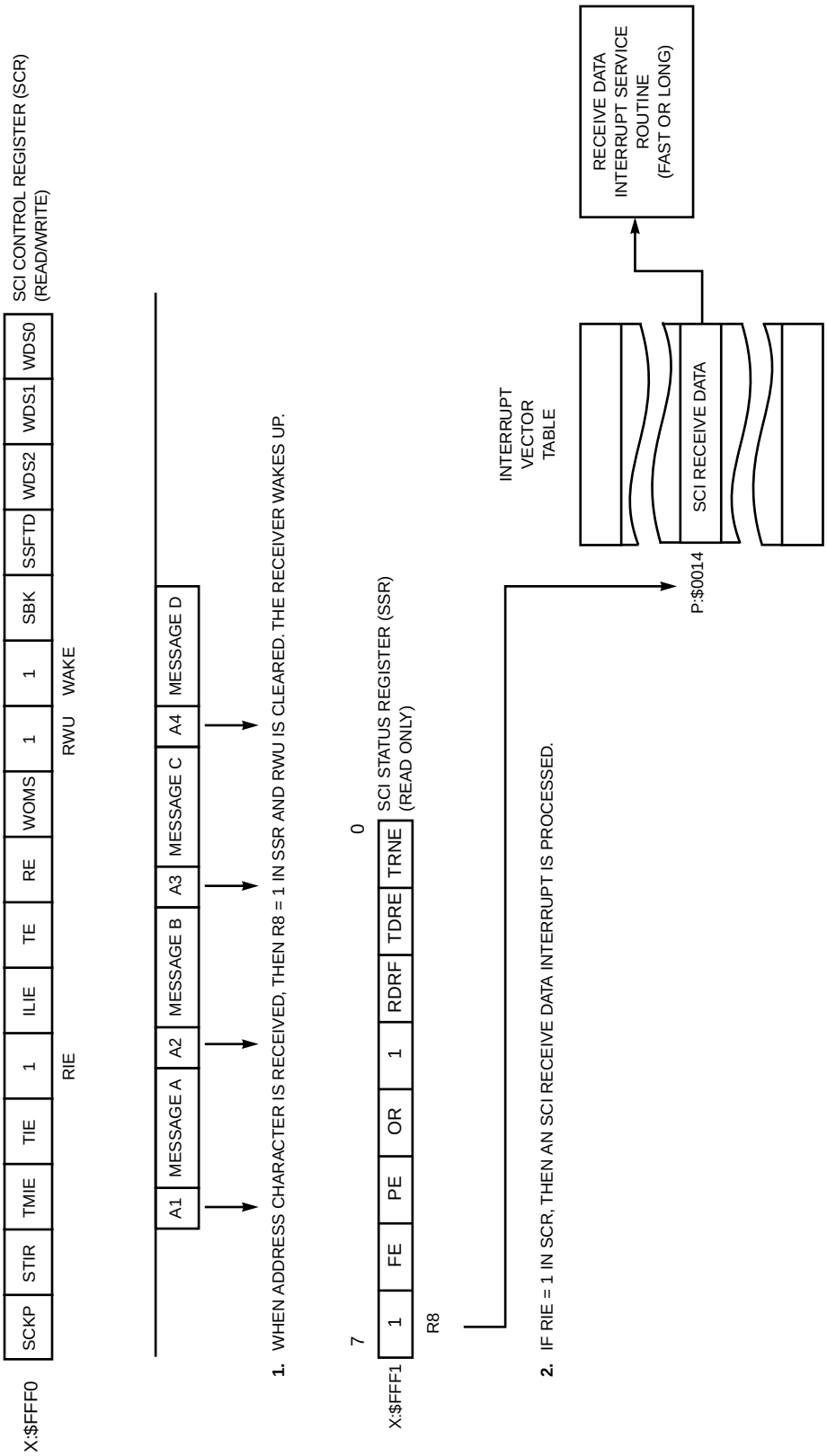


Figure 6-33 Address Mode Wakeup

is enabled but the interrupts are masked, which prevents the SCI from transmitting or receiving data at this time.

The circular buffers used have two pointers. The first points to the first data byte; the second points to the last data byte. This configuration allows the transmit buffer to act as a first-in first-out (FIFO) memory. The FIFO can be loaded by a program and emptied by the SCI in real time. As long as the number of data bytes never exceeds the buffer size, there will be no overflow or underflow of the buffer. Registers M0-M3 must be loaded with the buffer size minus one to make pointer registers R0-R3 work as circular pointers. Register N2 is used as a constant to clear the receive buffer empty flag.

The main program starts by filling the transmit buffer with a data packet. When the transmit buffer is full, it calls the subroutine that transmits the slave's address and then jumps to self (SEND jmp SEND), allowing interrupts to transmit and receive the data.

The receive subroutine first checks each byte to see if it is address or data. If it is an address, it compares the address with its own. If the addresses do not match, the SCI is put back to sleep. If the addresses match, the SCI is left awake, and control is returned to the main program. If the byte is data, it is placed in the receive buffer, and the receive buffer empty flag is cleared. Although this flag is not used in this program, it can be used by another program as a simple test to see if data is available. Using N2 as the constant $0 allows the flag to be cleared with a single-word instruction, which can be part of a fast interrupt.

The transmit subroutine transmits a byte and then checks to see if the transmit buffer is empty. If the buffer is not empty, control is returned to the main program, and interrupts are allowed to continue emptying the buffer. If the buffer is empty, the transmit buffer empty flag is set, the transmit interrupt is disabled, and control is returned to the main program.

The wakeup subroutine transmits the slave's address by writing the address to the STXA register and by enabling the transmit interrupt to allow interrupts to empty the transmit buffer. Control is then returned to the main program.

```
;*********************************************************************************************
;       MULTIDROP MASTER/SLAVE WITH INTERRUPTS AND CIRCULAR BUFFERS*
;*********************************************************************************************
;*************************************************
;       SCI and other EQUATES*
;*************************************************
START       EQU     $0040           ;Start of program
TX_BUFF     EQU     $0010           ;Transmit buffer location
RX_BUFF     EQU     $0020           ;Receive buffer location
B_SIZE      EQU     $000E           ;Transmit and receive buffer size
                                    ;(don't allow the TX buffer and RX
                                    ;buffers to overlap).
TX_MTY      EQU     $0000           ;Transmit buffer empty
RX_MTY      EQU     $0001           ;Receive buffer empty
PCC         EQU     $FFE1           ;Port C control register
SCR         EQU     $FFF0           ;SCI interface control register
SCCR        EQU     $FFF2           ;SCI clock control register
STXA        EQU     $FFF3           ;SCI transmit address register
SRX         EQU     $FFF4           ;SCI receive register
STX         EQU     $FFF4           ;SCI transmit register
BCR         EQU     $FFFE           ;Bus control register
IPR         EQU     $FFFF           ;Interrupt priority register
;*************************************************
;       RESET VECTOR*
;*************************************************
            ORG     P:$0000
            JMP     START
;*************************************************
;       SCI RECEIVE INTERRUPT VECTOR*
;*************************************************
            ORG     P:$0014         ;Load the SCI RX interrupt vectors
            JSR     RX              ;Jump to the receive routine that puts
                                    ;data packet in a circular buffer if it is for
                                    ;this address.
            NOP                     ;Second word of fast interrupt not needed
```

Figure 6-34 Multidrop Transmit Receive Example (Sheet 1 of 4)

```
                ORG     P:$0016         ;This interrupt occurs when data is
                                        ;received with errors. This example
                NOP                     ;does not trap errors so this
                NOP                     ;interrupt is not used.
;************************************************
;       SCI TRANSMIT INTERRUPT VECTOR*
;************************************************
                ORG     P:$0018         ;Load the SCI TX interrupt vectors
                JSR     TX              ;Transmit next byte in buffer
                NOP
;************************************************
;       INITIALIZE THE SCI PORT*
;************************************************
                ORG     P:START         ;Start the program at location $40
                ORI     #$03,MR         ;Mask interrupts temporarily
                MOVEP   #$C000,X:IPR    ;Set interrupt priority to 2
                MOVEP   #$0BE6,X:SCR    ;Disable TX, enable RX interrupts
                                        ;Enable transmitter and receiver,
                                        ;Wired-OR mode, Rec. wakeup
                                        ;mode,11-bit multidrop (1 start,
                                        ;8 data,1 data type, 1 stop)
                MOVEP   #$0000,X:SCCR   ;Use internal TX, RX clocks
                                        ;625K BPS at 40 MHz
                MOVEP   #>$03,X:PCC     ;Select pins TXD and RXD for SCI
;************************************************
;INITIALIZE INTERRUPTS, REGISTERS, ETC.*
;************************************************
                MOVEP   #$0,X:BCR       ;No wait states
                MOVE    #TX_BUFF,R0     ;Load start pointer of transmit buffer
                MOVE    #TX_BUFF,R1     ;Load end   pointer of transmit buffer
                MOVE    #RX_BUFF,R2     ;Load start pointer of receive buffer
                MOVE    #RX_BUFF,R3     ;Load end pointer of receive buffer
                MOVE    #>$41,R5        ;Init data register... R5 contains
                                        ;the data that will be sent in this
                                        ;example; it is initialized to an ASCII A.
```

Figure 6-34 Multidrop Transmit Receive Example (Sheet 2 of 4)

```
                MOVE    #B_SIZE,M0      ;Load transmit buffer size
                MOVE    #B_SIZE,M1      ;Load transmit buffer size
                MOVE    #B_SIZE,M2      ;Load receive buffer size
                MOVE    #B_SIZE,M3      ;Load receive buffer size
                MOVE    #>$1,N0         ;Load receive address
                MOVE    #>$1,N1         ;Load first slave address
                MOVE    #0,N2           ;Load a constant (0) into N2
                MOVEP   X:SRX,X:(R0)    ;Clear receive register
;***********************************************
;       MAIN PROGRAM*
;***********************************************
                ANDI    #$FC,MR         ;Re-enable interrupts
                MOVE    (R1)+           ;Temporarily increment the tail pointer
                                        ;Build a packet
LOOP            MOVE    R1,A            ;Check to see if the TX buffer is full
                MOVE    (R1)-           ;(fix tail pointer now that we've used it)
                MOVE    R0,B            ;by comparing the head and tail pointers
                CMP     A,B             ;of the circular transmit buffer.
                JEQ     SND_BUF         ;if equal, transmit completed packet
                MOVE    R5,X:(R1)+      ;if not, put next character in
                                        ;transmit buffer and
                MOVE    (R5)+           ;increment the pointers.
                MOVE    (R1)+           ;Temporarily increment the tail
                                        ;pointer to test buffer again
                JMP     LOOP
SND_BUF         JSR     WAKE_UP         ;Wake up proper slave and send packet
SEND            JMP     SEND            ;and allow interrupts to drain
                                        ;the transmit buffer.
```

Figure 6-34 Multidrop Transmit Receive Example (Sheet 3 of 4)

```
;*****************************************************************************
;  SUBROUTINE TO READ SCI AND STORE IN BUFFER USING A LONG INTERRUPT*
;*****************************************************************************
RX          JCLR    #7,X:$FFF1,RX_DATA      ;Check if this is address or data.
            MOVEP   X:SRX,A                 ;Compare the received address
            MOVE    N1,B                    ;with the slave address.
            CMP     A,B
            JEQ     END_RX                  ;If address OK, use interrupts to Rx
                                            ;packet
            BSET    #6,X:$FFF0              ;if not, go back to sleep
            JMP     END_RX                  ;and return to previous program.
RX_DATA     MOVEP   X:SRX,X:(R3)+           ;Put data in buffer,
            MOVE    N2,X:RX_MTY             ;and clear the Rx buffer empty flag
END_RX      RTI                             ;Return to previous program
;*****************************************************************************
;       SUBROUTINE TO WRITE BUFFER TO SCI USING A LONG INTERRUPT*
;*****************************************************************************
TX          MOVEP   X:(R0)+,X:STX           ;Transmit a byte and increment the
                                            ;pointer
            MOVE    R0,A                    ;Check to see if the TX buffer is
                                            ;empty
            MOVE    R1,B
            CMP     A,B
            JNE     END_TX                  ;If not, return to main
            MOVE    #$000001,X0             ;If it is, set the TX buffer empty flag
            MOVE    X0,X:TX_MTY
            BCLR    #12,X:SCR               ;disable transmit interrupts, and
END_TX      RTI                             ;return to main
;*****************************************************************************
;       SUBROUTINE TO WAKE UP THE ADDRESSED SLAVE*
;*****************************************************************************
WAKE_UP     MOVEP   N1,X:STXA               ;Transmit slave address using STXA
                                            ;not STX
            BSET    #12,X:SCR               ;Enable transmit interrupts to send
                                            ;packet
AWAKE       RTI
            END                             ;End of example.
```

Figure 6-34 Multidrop Transmit/Receive Example (Sheet 4 of 4)

6.3.9 SCI Timer

The SCI clock determines the data transmission rate and can also be used to establish a periodic interrupt that can act as an event timer or be used in any other timing function. Figure 6-35 illustrates how the SCI timer is programmed. Bits CD11–CD0, SCP, and STIR in the SCCR work together to determine the time base. The crystal oscillator f_{osc} is first divided by 2 and then divided by the number CD11–CD0 in the SCCR. The oscillator is then divided by 1 (if SCP=0) or eight (if SCP=1). This output is used as is if STIR = 1 or, if STIR = 0, it is divided by 2 and then by 16 before being used. If TMIE in the SCR = 1 when the periodic timeout occurs, the SCI timer interrupt is recognized and pending. The SCI timer interrupt is automatically cleared when the interrupt is serviced. This interrupt will occur every time the periodic timer times out. If only the timer function is being used (i.e., PC0, PC1, and PC2 pins have been programmed as GPIO pins), the transmit interrupts should be turned off (TIE=0). Under individual reset, TDRE will remain set and the timer will continuously generate interrupts.

Figure 6-35 shows that an external clock can be used for SCI receive and/or transmit, which frees the SCI timer to be programmed for a different interrupt rate. In addition, both the SCI timer interrupt and the SCI can use the internal time base if the SCI receiver and/or transmitter require the same clock period as the SCI timer.

The program in Figure 6-36 configures the SCI to interrupt the DSP at fixed intervals. The program starts by setting equates for convenience and clarity and then points the reset vector to the start of the program. The SCI timer interrupt vector location contains “move (R0)+”, incrementing the contents of R0, which serves as an elapsed time counter.

The timer initialization consists of enabling the SCI timer interrupt, setting the SCI baud rate counters for the desired interrupt rate, setting the interrupt mask, enabling the interrupt, and then enabling the SCI state machine.

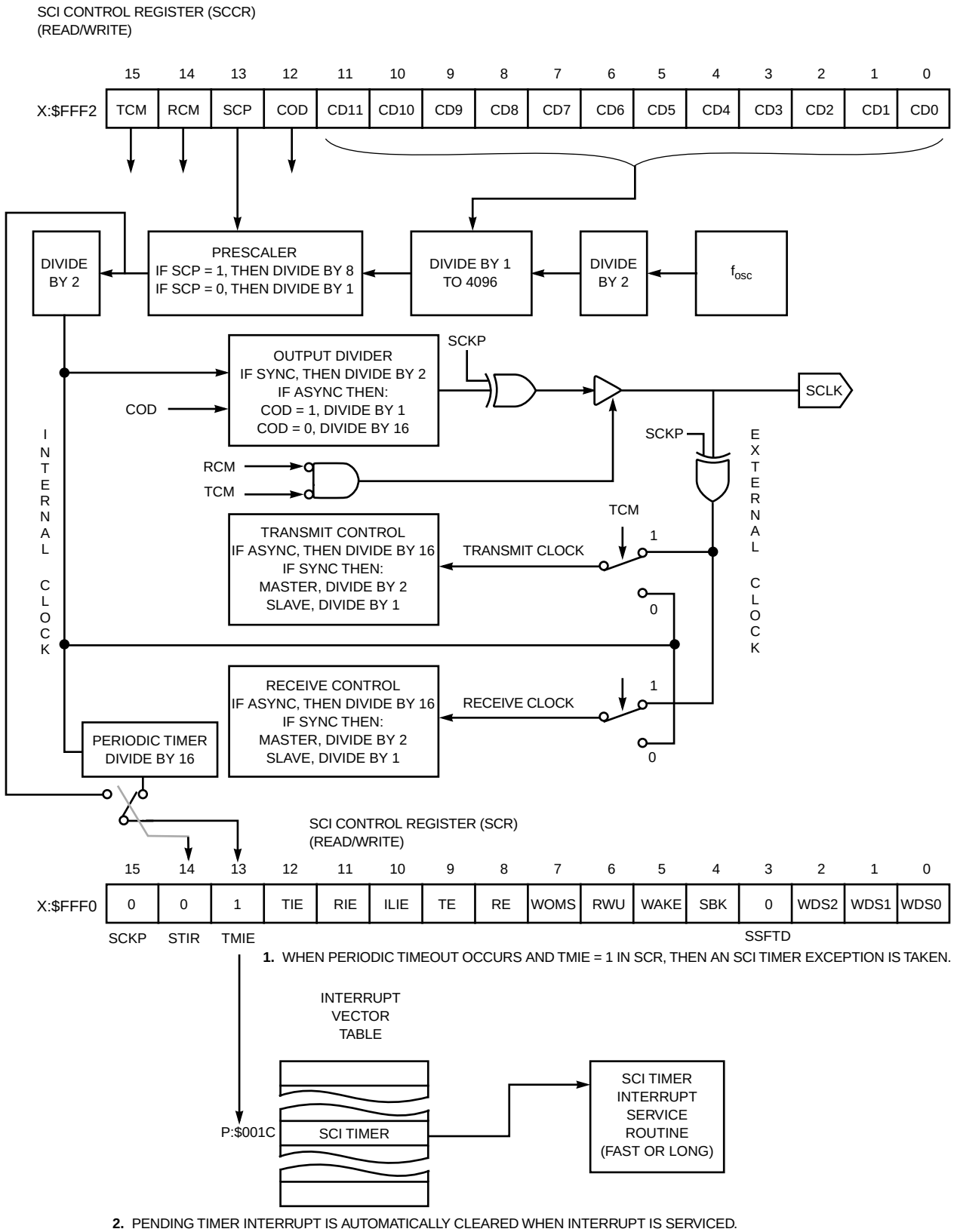


Figure 6-35 SCI Timer Operation

```
;*********************************************************************************************
;       TIMER USING SCI TIMER INTERRUPT*
;*********************************************************************************************

;*************************************************
;       SCI and other EQUATES*
;*************************************************
START       EQU         $0040           ;Start of program
SCR         EQU         $FFF0           ;SCI control register
SCCR        EQU         $FFF2           ;SCI clock control register
IPR         EQU         $FFFF           ;Interrupt priority register

;*************************************************
;       RESET VECTOR*
;*************************************************
            ORG         P:$0000
            JMP         START
;*************************************************
;       SCI TIMER INTERRUPT VECTOR*
;*************************************************
            ORG         P:$001C         ;Load the SCI timer interrupt vectors
            MOVE        (R0)+           ;Increment the timer interrupt counter
            NOP                         ;This timer routine is implemented
                                        ;as a fast interrupt
;*************************************************
;       INITIALIZE THE SCI PORT*
;*************************************************
            ORG         P:START         ;Start the program at location $40
            MOVE        #0,R0           ;Initialize the timer interrupt counter
            MOVEP       #$2000,X:SCR    ;Select the timer interrupt
            MOVEP       #$013F,X:SCCR   ;Set the interrupt rate at 1 ms
                                        ;(arbitrarily chosen)
                                        ;Interrupts/second =
                                        ;fosc/(64×(7(SCP)-+1)×(CD+1))
                                        ;Note that this is the same equation
                                        ;as for SCI async baud rate
```

Figure 6-36 SCI Timer Example (Sheet 1 of 2)

```
                                        ;For 1 ms, SCP=0,
                                        ;CD=0001 0011 1111.
            MOVEP       #$C000,X:IPR    ;Set the interrupt priority level–
                                        ;application specific.
            ANDI        #$FC,MR         ;Enable interrupts, set MR bits I1 and
                                        ;I0=0
END         JMP         END             ;Normally something more useful
                                        ;would be put here.
            END                         ;End of example.
```

Figure 6-36 SCI Timer Example (Sheet 2 of 2)

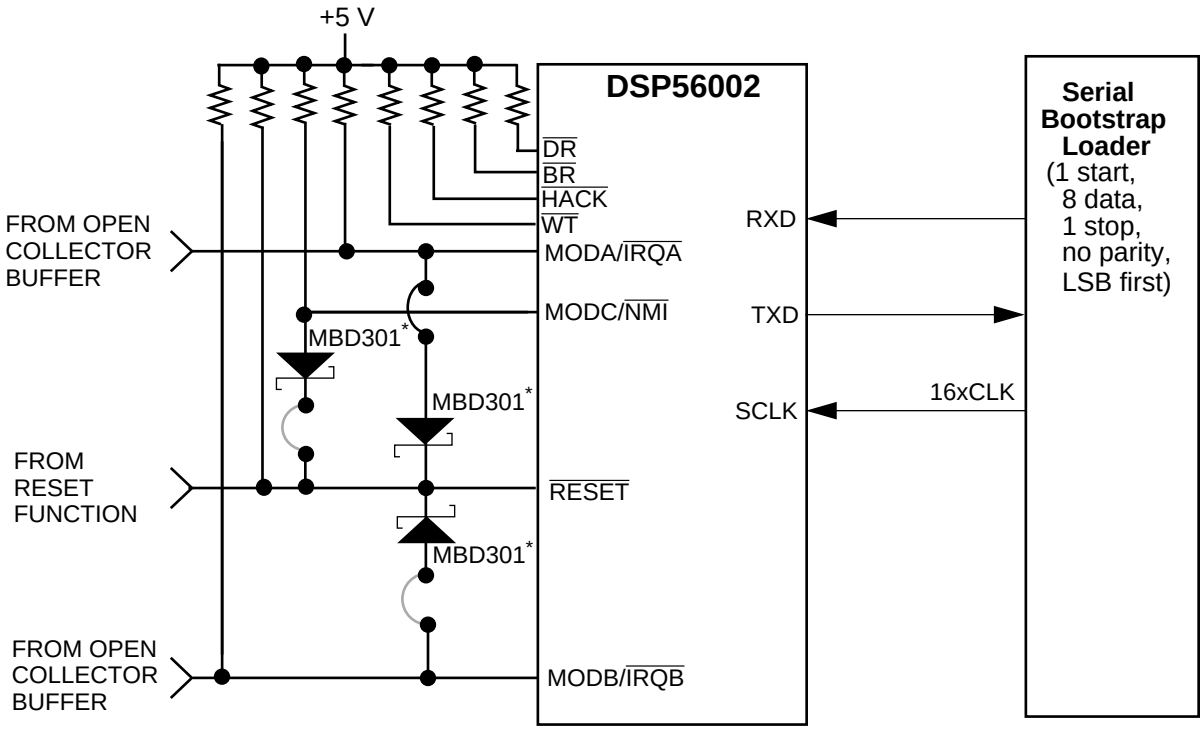
6.3.10 Bootstrap Loading Through the SCI (Operating Mode 6)

When the DSP comes out of reset, it looks at the MODC, MODB, and MODA pins and sets the corresponding mode bits in the OMR. If the mode bits are set to 110 respectively, the DSP will load the program RAM from the SCI. Figure 6-37 shows how the SCI is configured for receiving this code and Figure 6-37 shows the segment of bootstrap code that is used to load from the SCI. The complete code used in the bootstrap program is given in **APPENDIX A.** This program (1) configures the SCI, (2) loads the program size, (3) loads the location where the program will begin loading in program memory, and (4) loads the program.

First, the SCI Control Register is set to $0302 (see Figure 5-2) which enables the transmitter and receiver and configures the SCI for 10 bits **asynchronous** with **one start bit, 8 data bits, one stop bit, and no parity.** Next, the SCI Clock Control Register is set to $C000 which configures the SCI to use external receive and transmit clocks on the SCLK pin. This **clock** must be **16 times the serial data rate.**

The next step is to receive the program size and then the starting address to load the program. These two numbers are three bytes each loaded least significant byte first. Each byte will be echoed back as it is received. After both numbers are loaded, the program size is in A0 and the starting address is in A1.

The program is then loaded one byte at a time, least significant byte first. After loading the program, the operating mode is set to zero, the CCR is cleared, and the DSP begins execution with the first instruction that was loaded.



Notes: 1. ***These diodes **must** be Schottky diodes.
2. All resistors are 15KΩ unless noted otherwise.
3. When in RESET, IRQA, IRQB and NMI must be deasserted by external peripherals.

Figure 6-37 DSP56002 Bootstrap Example - Mode 6

```
; This routine loads from the SCI.
; MC:MB:MA=110 - external SCI clock
; MC:MB:MA=111 - reserved
SCILD     MOVEP     #$0302,X:SCR        ; Configure SCI Control Reg
          JMP       <EXTC               ; go to next boot rom segment
          NOP                           ; just to fill the last space

          ORG       PL:$100,PL:$100     ; starting address of 2nd ROM

EXTC      MOVEP     #$C000,X:SCCR       ; Configure SCI Clock Control Reg
          MOVEP     #7,X:PCC            ; Configure SCLK, TXD and RXD

_SCI1     DO        #6,_LOOP6           ; get 3 bytes for number of
                                        ; program words and 3 bytes
                                        ; for the starting address
          JCLR      #2,X:SSR,*          ; Wait for RDRF to go high
          MOVEP     X:SRXL,A2           ; Put 8 bits in A2
          JCLR      #1,X:SSR,*          ; Wait for TDRE to go high
          MOVEP     A2,X:STXL           ; echo the received byte
          REP       #8
          ASR       A
_LOOP6
          MOVE      A1,R0               ; starting address for load
          MOVE      A1,R1               ; save starting address
          DO        A0,_LOOP4           ; Receive program words

          DO        #3,_LOOP5
          JCLR      #2,X:SSR,*          ; Wait for RDRF to go high
          MOVEP     X:SRXL,A2           ; Put 8 bits in A2
          JCLR      #1,X:SSR,*          ; Wait for TDRE to go high
          MOVEP     A2,X:STXL           ; echo the received byte
          REP       #8
          ASR       A
_LOOP5
          MOVEM     A1,P:(R0)+          ; Store 24-bit result in P memory
_LOOP4
```

Figure 6-38 Bootstrap Code Fragment

6.3.11 Example Circuits

The SCI can be used in a number of configurations to connect multiple processors. The synchronous mode shown in Figure 6-39 shows the DSP acting as a slave. The 8051 provides the clock that clocks data in and out of the SCI, which is possible because the SCI shift register mode timing is compatible with the timing for 8051/8096 processors. Transmit data is changed on the negative edge of the clock, and receive data is latched on the positive edge of the clock. A protocol must be used to prevent both processors from transmitting simultaneously. The DSP is also capable of being the master device.

A multimaster system can be configured (see Figure 6-41) using a single transmit/receive line, multidrop word format, and wired-OR. The use of wired-OR requires a pullup resistor as shown. A protocol must be used to prevent collisions. This scheme is physically the simplest multiple DSP interconnection because it uses only one wire and one resistor.

The master-slave system shown in Figure 6-40 is different in that it is full duplex. The clock pin is not required; thus, it is configured as a GPIO pin. Communication is asynchronous. The slave's transmitters must be wire-ORed because more than one transmitter is on one line. The master's transmitter does not need to be wire-ORed.

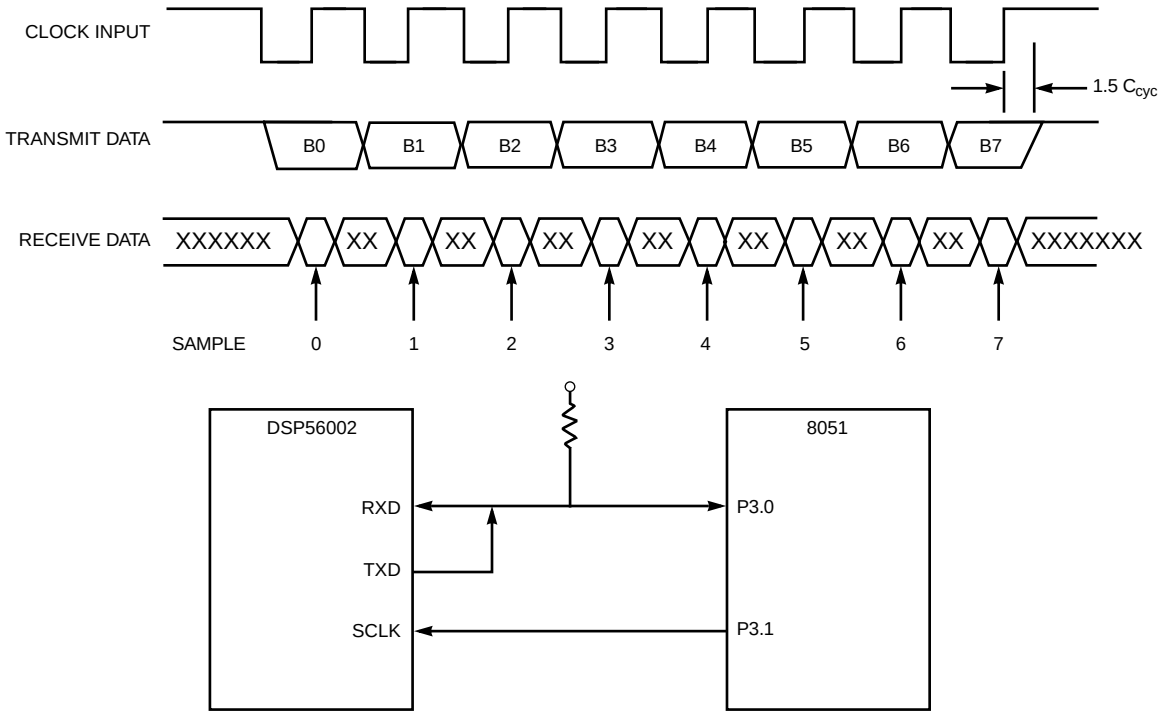


Figure 6-39 Synchronous Mode Example

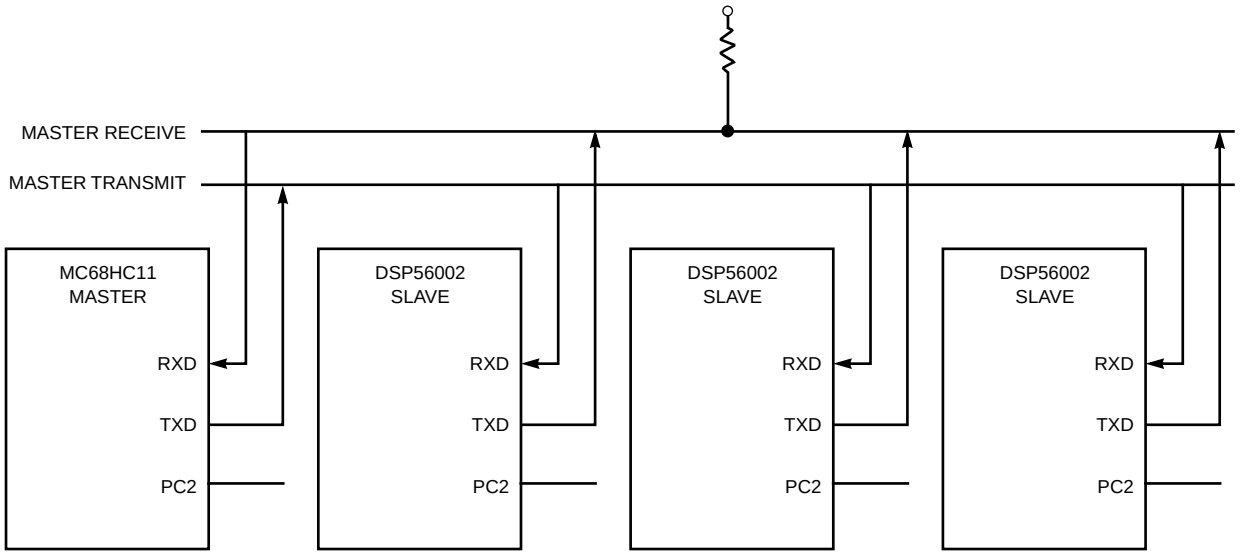


Figure 6-40 Master-Slave System Example

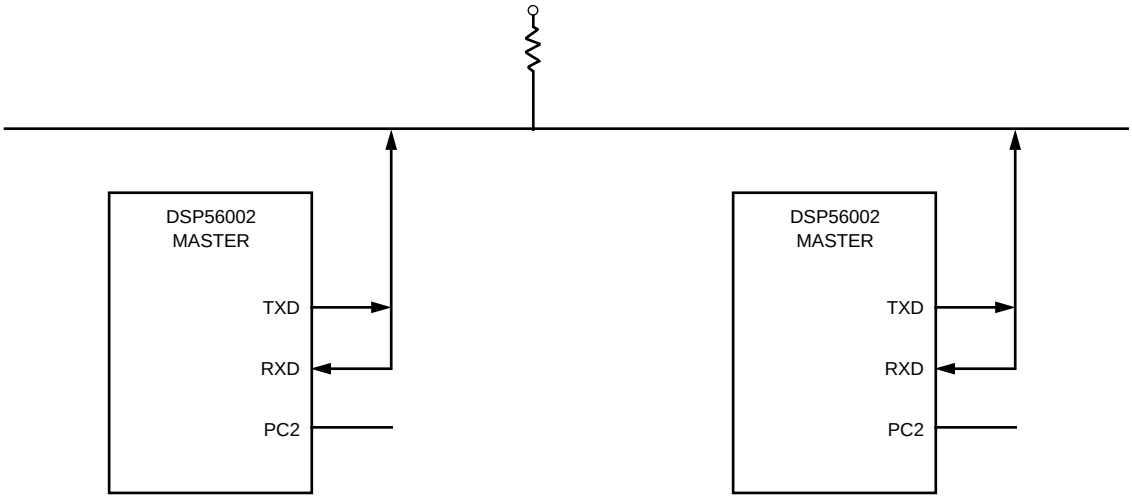


Figure 6-41 Multimaster System Example

6.4 SYNCHRONOUS SERIAL INTERFACE (SSI)

The synchronous serial interface (SSI) provides a full-duplex serial port for serial communication with a variety of serial devices including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals which implement the Motorola SPI.

The user can independently define the following characteristics of the SSI: the number of bits per word, the protocol, the clock, and the transmit/receive synchronization.

The user can select among three modes: normal, on-demand, and network. The normal mode is typically used to interface with devices on a regular or periodic basis. The data-driven on-demand mode is intended to be used to communicate with devices on a non-periodic basis. The network mode provides time slots in addition to a bit clock and frame synchronization pulse.

The SSI functions with a range of 2 to 32 words of I/O per frame in the network mode. This mode is typically used in star or ring time division multiplex networks with other DSP56K processors and/or codecs. The clock can be programmed to be continuous or gated. Since the transmitter and receiver sections of the SSI are independent, they can be programmed to be synchronous (using a common clock) or asynchronous with respect to each other.

The SSI requires up to six pins, depending on its operating mode. The most common minimum configuration is three pins: transmit data (STD), receive data (SRD) and clock (SCK).

The SSI consists of independent transmitter and receiver sections and a common SSI clock generator. Three to six pins are required for operation, depending on the operating mode selected.

The following is a short list of SSI features:

- Three-Pin Interface:
  TXD – Transmit Data
  RXD – Receive Data
  SCLK – Serial Clock
- A 10 Mbps at 40 MHz ($f_{osc}/4$) serial interface
- Double Buffered
- User Programmable
- Separate Transmit and Receive Sections
- Control and Status Bits

- Interface to a Variety of Serial Devices, Including:
  - Codecs (usually without additional logic)
    - MC145502
    - MC145503
    - MC145505
    - MC145402 (13-bit linear codec)
    - MC145554 Family of Codecs
    - MC145532
  - Serial Peripherals (A/D, D/A)
    - Most Industry-Standard A/D, D/A
    - DSP56ADC16 (16-bit linear A/D)
  - DSP56K to DSP56K Networks
  - Motorola SPI Peripherals and Processors
  - Shift Registers
- Interface to Time Division Multiplexed Networks without Additional Logic
- Six Pins:
  - STD SSI Transmit Data
  - SRD SSI Receive Data
  - SCK SSI Serial Clock
  - SC0 Serial Control 0 (defined by SSI mode)
  - SC1 Serial Control 1 (defined by SSI mode)
  - SC2 Serial Control 2 (defined by SSI mode)
- On-chip Programmable Functions Include:
  - Clock – Continuous, Gated, Internal, External
  - Synchronization Signals – Bit Length and Word Length
  - TX/RX Timing – Synchronous, Asynchronous
  - Operating Modes – Normal, Network, On-Demand
  - Word Length – 8, 12, 16, 24 Bits
  - Serial Clock and Frame Sync Generator
- Four Interrupt Vectors:
  - Receive
  - Receive with Exception
  - Transmit
  - Transmit with Exception

This interface is descriptively named “synchronous” because all serial transfers are synchronized to a clock. Additional synchronization signals are used to delineate the word frames. The normal mode of operation is used to transfer data at a periodic rate, but only one word per period. The network mode is similar in that it is also intended for periodic transfers; however, it will support up to 32 words (time slots) per period. This mode can be used to build time division multiplexed (TDM) networks. In contrast, the on-demand mode is intended for nonperiodic transfers of data. This mode can be used to transfer data serially at high speed when the data becomes available. This mode offers a subset of the SPI protocol.

6.4.1 SSI Data and Control Pins

The SSI has three dedicated I/O pins (see Figure 6-1), which are used for transmit data (STD), receive data (SRD), and serial clock (SCK), where SCK may be used by both the transmitter and the receiver for synchronous data transfers or by the transmitter only for asynchronous data transfers. Three other pins may also be used, depending on the mode selected; they are serial control pins SC0, SC1, and SC2. They may be programmed as SSI control pins in the Port C control register. Table 6-5 shows the definition of SC0, SC1, SC2, and SCK in the various configurations. The following paragraphs describe the uses of these pins for each of the SSI operating modes. Figure 6-42 and Figure 6-43 show the internal clock path connections in block diagram form. The receiver and transmitter clocks can be internal or external depending on the SYN, SCD0, and SCKD bits in CRB.

6.4.1.1 Serial Transmit Data Pin (STD)

STD is used for transmitting data from the serial transmit shift register. STD is an output when data is being transmitted. Data changes on the positive edge of the bit clock. STD goes to high impedance on the negative edge of the bit clock of the last data bit of the word (i.e., during the second half of the last data bit period) with external gated clock, regardless of the mode. With an internally generated bit clock, the STD pin becomes high impedance after the last data bit has been transmitted for a full clock period, assuming another data word does not follow immediately. If a data word follows immediately, there will not be a high-impedance interval.

Codecs label the MSB as bit 0; whereas, the DSP labels the LSB as bit 0. Therefore, when using a standard codec, the DSP MSB (or codec bit 0) is shifted out first when SHFD=0, and the DSP LSB (or codec bit 7) is shifted out first when SHFD=1. STD may be programmed as a general-purpose pin called PC8 when the SSI STD function is not being used.

Table 6-5 Definition of SC0, SC1, SC2, and SCK

SSI Pin Name (Control Bit Name)	Asynchronous (SYN=0)		Synchronous (SYN=1)	
	Continuous Clock (GCK=0)	Gated Clock (GCK=1)	Continuous Clock (GCK=0)	Gated Clock (GCK=1)
SC0=0 (in) SC0=1 (out) (SCD0)	RXC External RXC Internal	RXC External RXC Internal	Input F0 Output F0	Input F0 Output F0
SC1=0 (in) SC1=1 (out) (SCD1)	FSR External FSR Internal	Not Used FSR Internal	Input F1 Output F1	Input F1 Output F1
SC2=0 (in) SC2=1 (out) (SCD2)	FST External FST Internal	Not Used FST Internal	FS* External FS* Internal	Not Used FS* Internal
SCK=0 (in) SCK=1 (out (SCKD)	TXC External TXC Internal	TXC External TXC Internal)	*XC External *XC Internal	*XC External *XC Internal

TXC – Transmitter Clock
RXC – Receiver Clock
*XC – Transmitter/Receiver Clock (synchronous operation)
FST – Transmitter Frame Sync
FSR – Receiver Frame Sync
FS* – Transmitter/Receiver Frame Sync (synchronous operation)
F0 – Flag 0
F1 – Flag 1

Table 6-6 SSI Clock Sources, Inputs, and Outputs

SYN	SCKD	SCD0	R Clock Source	RX Clock Out	T Clock Source	TX Clock Out
Asynchronous						
0	0	0	EXT, SC0	–	EXT, SCK	–
0	0	1	INT	SC0	EXT, SCK	–
0	1	0	EXT, SC0	–	INT	SCK
0	1	1	INT	SC0	INT	SCK
Synchronous						
1	0	0	EXT, SCK	–	EXT, SCK	–
1	0	1	EXT, SCK	–	EXT, SCK	–
1	1	0	INT	SCK	INT	SCK
1	1	1	INT	SCK	INT	SCK

EXT – External Pin Name
INT – Internal Bit Clock

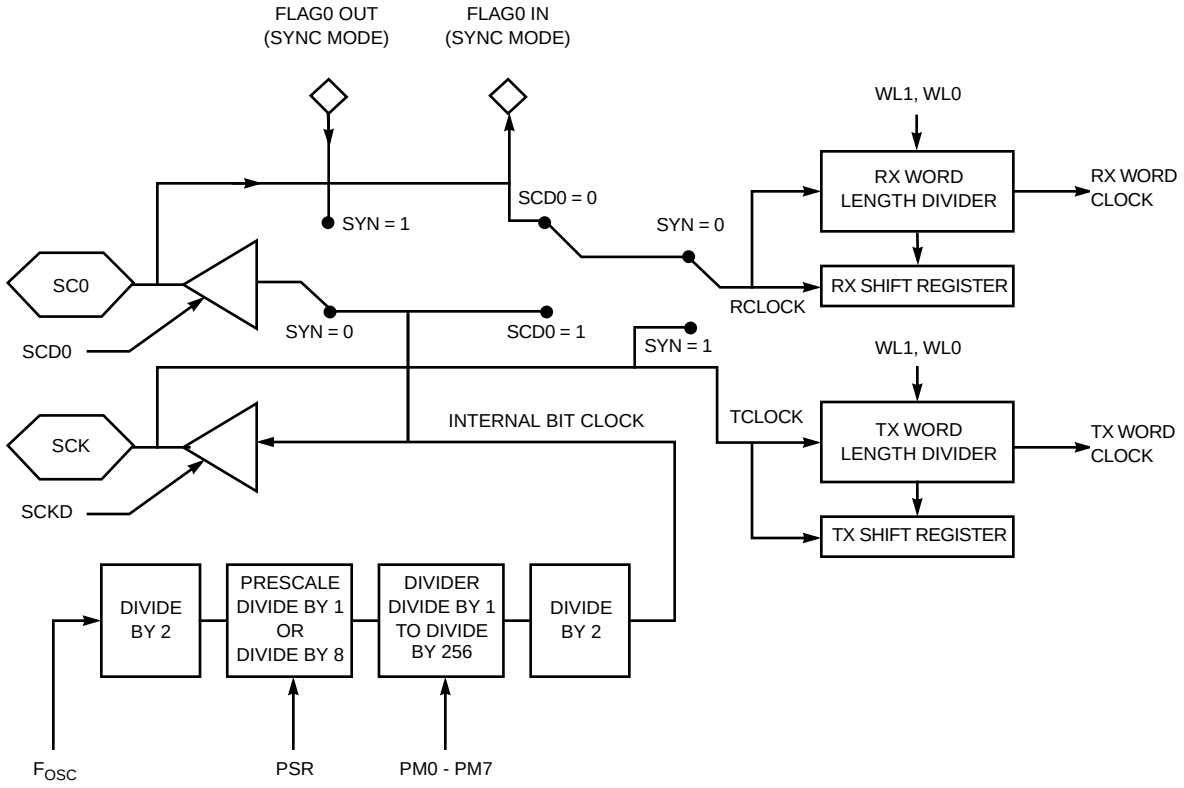


Figure 6-42 SSI Clock Generator Functional Block Diagram

6.4.1.2 Serial Receive Data Pin (SRD)

SRD receives serial data and transfers the data to the SSI receive shift register. SRD may be programmed as a general-purpose I/O pin called PC7 when the SSI SRD function is not being used. Data is sampled on the negative edge of the bit clock.

6.4.1.3 Serial Clock (SCK)

SCK is a bidirectional pin providing the serial bit rate clock for the SSI interface. The SCK is a clock input or output used by both the transmitter and receiver in synchronous modes or by the transmitter in asynchronous modes (see Table 6-6).

Note: Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 8T (i.e., the system clock frequency must be at least four times the external SSI clock frequency). The SSI needs at least four DSP phases (DSP phase=T) inside each half of the serial clock.

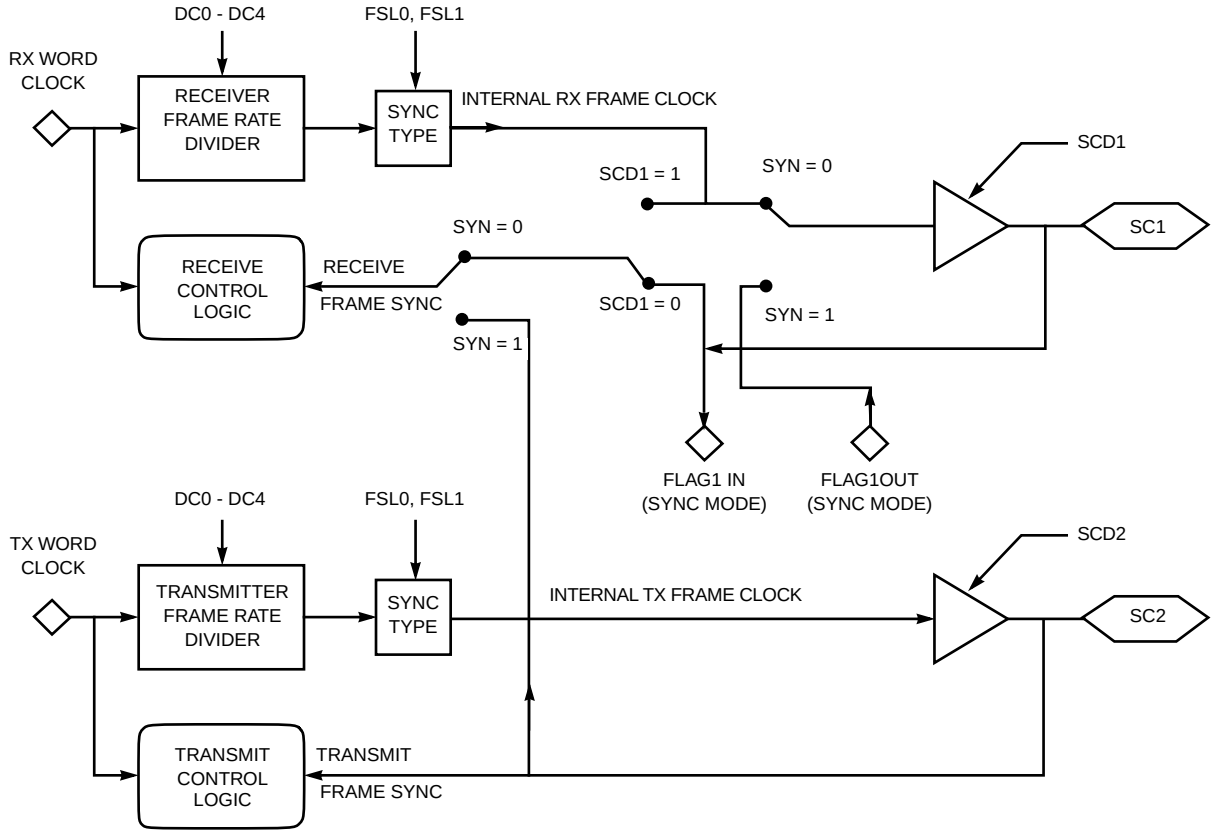


Figure 6-43 SSI Frame Sync Generator Functional Block Diagram

6.4.1.4 Serial Control Pin (SC0)

The function of this pin is determined solely on the selection of either synchronous or asynchronous mode (see Table 6-5 and Table 6-6). For asynchronous mode, this pin will be used for the receive clock I/O. For synchronous mode, this pin is used for serial flag I/O. A typical application of flag I/O would be multiple device selection for addressing in codec systems. The direction of this pin is determined by the SCD0 bit in the CRB as described in Table 6-7. When configured as an output, this pin will be either serial output flag 0, based on control bit OF0 in CRB, or a receive shift register clock output. When configured as an input, this pin may be used either as serial input flag 0, which will control status bit IF0 in the SSISR, or as a receive shift register clock input.

Table 6-7 SSI Operation: Flag 0 and Rx Clock

SYN	GCK	SCD0	Operation
Synchronous	Continuous	Input	Flag 0 Input
Synchronous	Continuous	Output	Flag 0 Output
Synchronous	Gated	Input	Flag 0 Input
Synchronous	Gated	Output	Flag 0 Output
Asynchronous	Continuous	Input	Rx Clock – External
Asynchronous	Continuous	Output	Rx Clock – Internal
Asynchronous	Gated	Input	Rx Clock – External
Asynchronous	Gated	Output	Rx Clock – Internal

6.4.1.5 Serial Control Pin (SC1)

The function of this pin is determined solely on the selection of either synchronous or asynchronous mode (see Table 6-5 and Table 6-8). In asynchronous mode (such as a single codec with asynchronous transmit and receive), this pin is the receiver frame sync I/O. For synchronous mode with continuous clock, this pin is serial flag SC1 and operates like the previously described SC0. SC0 and SC1 are independent serial I/O flags but may be used together for multiple serial device selection. SC0 and SC1 can be used unencoded to select up to two codecs or may be decoded externally to select up to four codecs. The direction of this pin is determined by the SCD1 bit in the CRB. When configured as an output, this pin will be either a serial output flag, based on control bit OF1, or it will make the receive frame sync signal available. When configured as an input, this pin may be used as a serial input flag, which will control status bit IF1 in the SSI status register, or as a receive frame sync from an external source for continuous clock mode. In the gated clock mode, external frame sync signals are not used.

Table 6-8 SSI Operation: Flag 1 and Rx Frame Sync

SYN	GCK	SCD1	Operation
Synchronous	Continuous	Input	Flag 1 Input
Synchronous	Continuous	Output	Flag 1 Output
Synchronous	Gated	Input	Flag 1 Input
Synchronous	Gated	Output	Flag 1 Output
Asynchronous	Continuous	Input	RX Frame Sync – External
Asynchronous	Continuous	Output	RX Frame Sync – Internal
Asynchronous	Gated	Input	–
Asynchronous	Gated	Output	RX Frame Sync – Internal

6.4.1.6 Serial Control Pin (SC2)

This pin is used for frame sync I/O (see Table 6-5 and Table 6-9). SC2 is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode. The direction of this pin is determined by the SCD2 bit in CRB. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitter (and the receiver in synchronous operation). In the gated clock mode, external frame sync signals are not used.

Table 6-9 SSI Operation: Tx and Rx Frame Sync

SYN	GCK	SCD2	Operation
Synchronous	Continuous	Input	TX and RX Frame Sync
Synchronous	Continuous	Output	TX and RX Frame Sync
Synchronous	Gated	Input	–
Synchronous	Gated	Output	TX and RX Frame Sync
Asynchronous	Continuous	Input	TX Frame Sync – External
Asynchronous	Continuous	Output	TX Frame Sync – Internal
Asynchronous	Gated	Input	–
Asynchronous	Gated	Output	TX Frame Sync – Internal

6.4.2 SSI Programming Model

The SSI can be viewed as two control registers, one status register, a transmit register, a receive register, and special-purpose time slot register. These registers are illustrated in Figure 6-44 and Figure 6-45. The following paragraphs give detailed descriptions and operations of each of the bits in the SSI registers. The SSI registers are not prefaced with an “S” (for serial) as are the SCI registers.

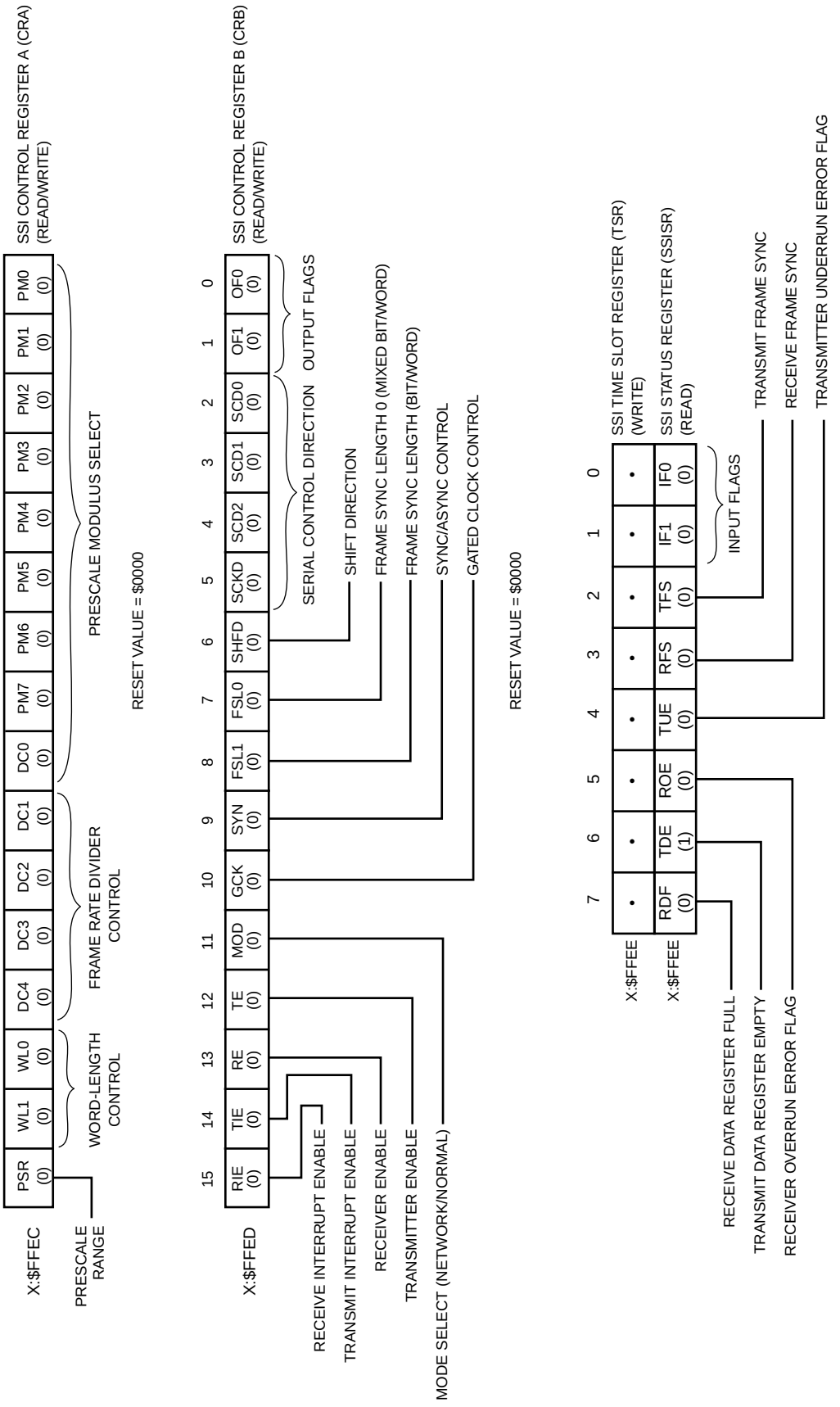
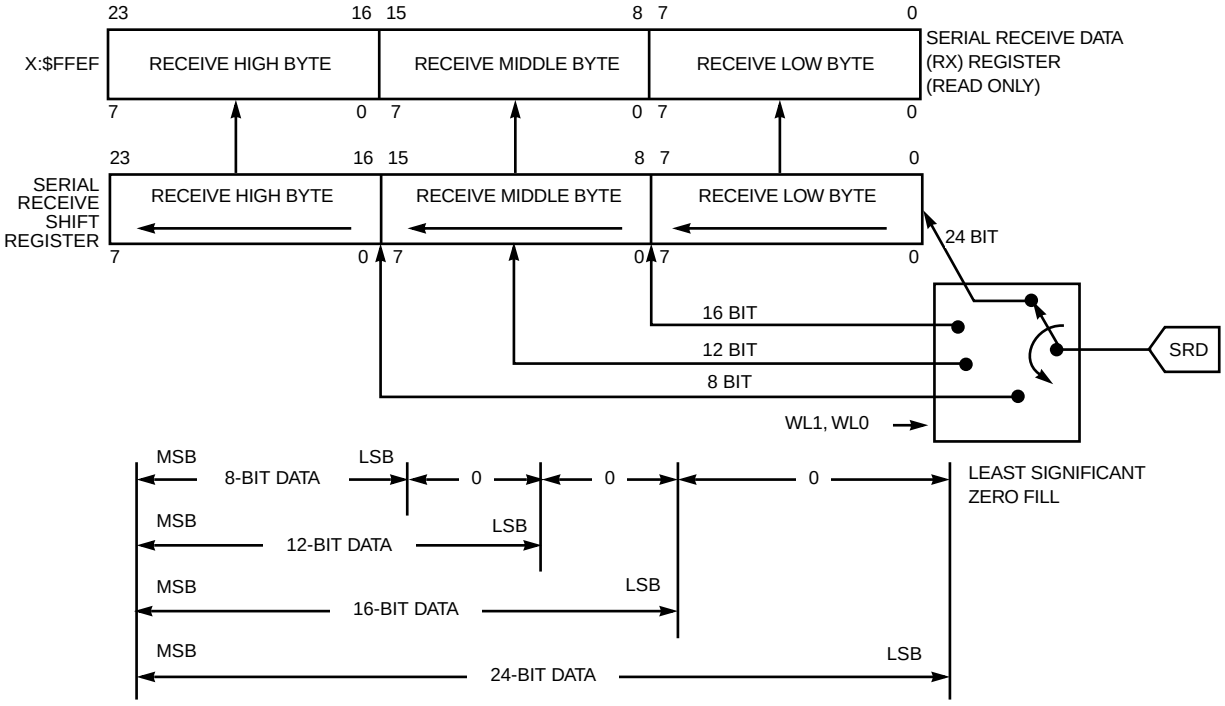
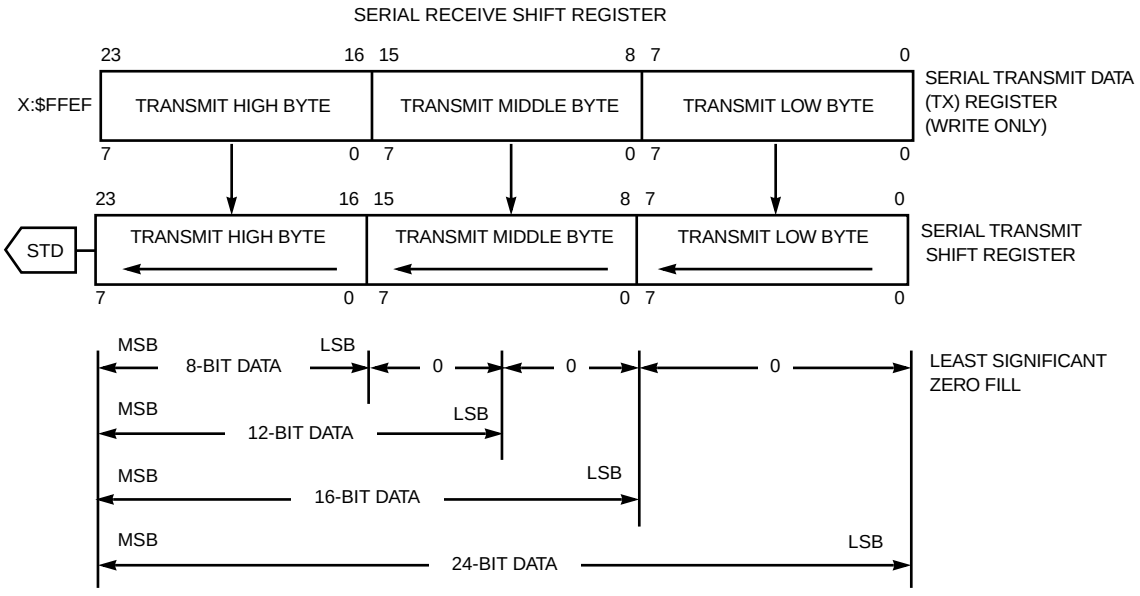


Figure 6-44 SSI Programming Model — Control and Status Registers



NOTES:
1. Data is received MSB first if SHFD = 0.
2. Compatible with fractional format.

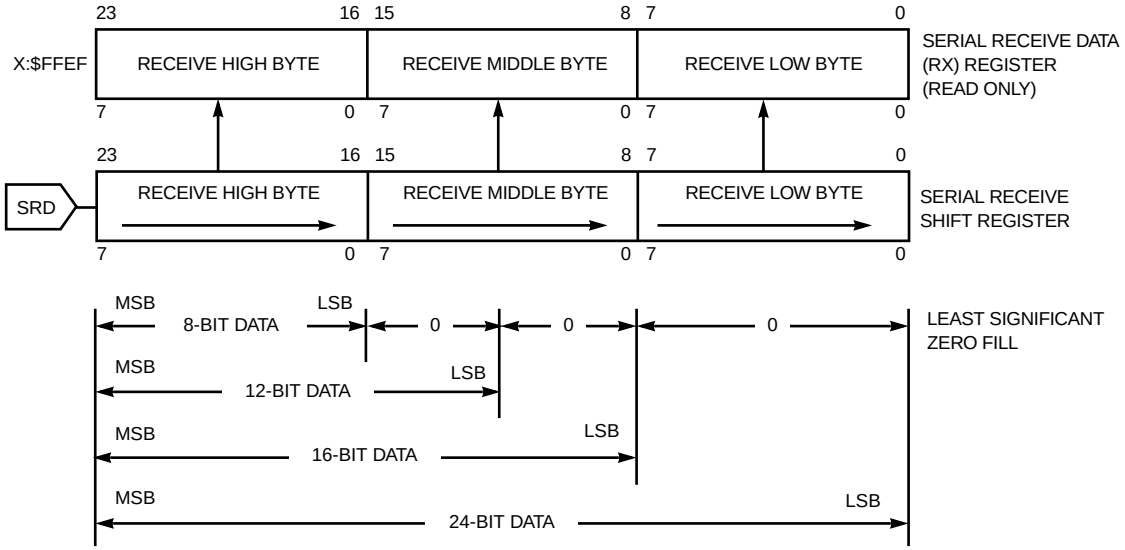
(a) Receive Registers for SHFD = 0



NOTES:
1. Data is sent MSB first if SHFD = 0.
2. Compatible with fractional format.

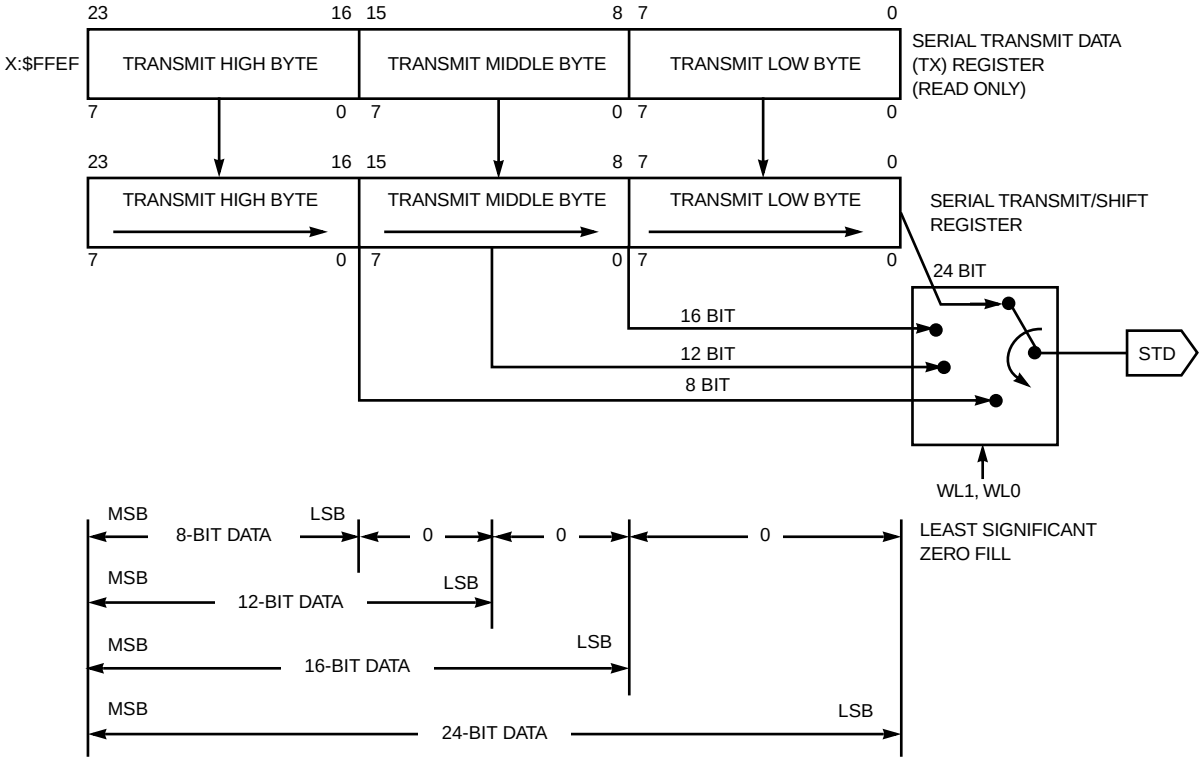
(b) Transmit Registers for SHFD = 0

Figure 6-45 SSI Programming Model (Sheet 1 of 2)



NOTES:
1. Data is received LSB first if SHFD = 1.
2. Compatible with fractional format.

(c) Receive Registers for SHFD = 1



NOTES:
1. Data is received LSB first if SHFD = 1.
2. Compatible with fractional format.

(d) Transmit Registers for SHFD = 1

Figure 6-45 SSI Programming Model (Sheet 2 of 2)

6.4.2.1 SSI Control Register A (CRA)

CRA is one of two 16-bit read/write control registers used to direct the operation of the SSI. The CRA controls the SSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The high-order bits of CRA are read as zeros by the DSP CPU. The CRA control bits are described in the following paragraphs.

6.4.2.1.1 CRA Prescale Modulus Select (PM7–PM0) Bits 0–7

The PM0–PM7 bits specify the divide ratio of the prescale divider in the SSI clock generator. A divide ratio from 1 to 256 (PM=0 to $FF) may be selected. The bit clock output is available at the transmit clock (SCK) and/or the receive clock (SC0) pins of the DSP. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus will allow the industry-standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. Hardware and software reset clear PM0–PM7.

6.4.2.1.2 CRA Frame Rate Divider Control (DC4–DC0) Bits 8–12

The DC4–DC0 bits control the divide ratio for the programmable frame rate dividers used to generate the frame clocks (see Figure 6-43). In network mode, this ratio may be interpreted as the number of words per frame minus one. In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC=00000 to 11111) for normal mode and 2 to 32 (DC=00001 to 11111) for network mode.

A divide ratio of one (DC=00000) in network mode is a special case (see **6.4.7.4**). In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data word transfers. A bit-length sync (FSL1=1, FSL0=0) must be used in this case. Hardware and software reset clear DC4–DC0.

6.4.2.1.3 CRA Word Length Control (WL0, WL1) Bits 13 and 14

The WL1 and WL0 bits are used to select the length of the data words being transferred via the SSI. Word lengths of 8, 12, 16, or 24 bits may be selected according to Table 6-10.

Table 6-10 Number of Bits/Word

WL1	WL0	Number of Bits/Word
0	0	8
0	1	12
1	0	16
1	1	24

These bits control the number of active clock transitions in the gated clock modes and control the word length divider (see Figure 6-42 and Figure 6-43), which is part of the frame rate signal generator for continuous clock modes. The WL control bits also control the frame sync pulse length when FSL0 and FSL1 select a WL bit clock (see Figure 6-42). Hardware and software reset clear WL0 and WL1.

6.4.2.1.4 CRA Prescaler Range (PSR) Bit 15

The PSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired (see Figure 6-42). When PSR is cleared, the fixed prescaler is bypassed. When PSR is set, the fixed divide-by-eight prescaler is operational. This allows a 128-kHz master clock to be generated for MC14550x series codecs.

The maximum internally generated bit clock frequency is fosc/4, the minimum internally generated bit clock frequency is fosc/4/8/256=fosc/8192. Hardware and software reset clear PSR.

6.4.2.2 SSI Control Register B (CRB)

The CRB is one of two 16-bit read/write control registers used to direct the operation of the SSI. CRB controls the SSI multifunction pins, SC2, SC1, and SC0, which can be used as clock inputs or outputs, frame synchronization pins, or serial I/O flag pins. The serial output flag control bits and the direction control bits for the serial control pins are in the SSI CRB. Interrupt enable bits for each data register interrupt are provided in this control register. When read by the DSP, CRB appears on the two low-order bytes of the 24-bit word, and the high-order byte reads as zeros. Operating modes are also selected in this register. Hardware and software reset clear all the bits in the CRB. The relationships between the SSI pins (SC0, SC1, SC2, and SCK) and some of the CRB bits are summarized in Tables Table 6-5, Table 6-12, and Table 6-13. The SSI CRB bits are described in the following paragraphs.

6.4.2.2.1 CRB Serial Output Flag 0 (OF0) Bit 0

When the SSI is in the synchronous clock mode and the serial control direction zero bit (SCD0) is set, indicating that the SC0 pin is an output, then data present in OF0 will be written to SC0 at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF0.

6.4.2.2.2 CRB Serial Output Flag 1 (OF1) Bit 1

When the SSI is in the synchronous clock mode and the serial control direction one

(SCD1) bit is set, indicating that the SC1 pin is an output, then data present in OF1 will be written to the SC1 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode (see **6.4.7**).

The normal sequence for setting output flags when transmitting data is to poll TDE (TX empty), to first write the flags, and then write the transmit data to the TX register. OF0 and OF1 are double buffered so that the flag states appear on the pins when the TX data is transferred to the transmit shift register (i.e., the flags are synchronous with the data). Hardware and software reset clear OF1.

Note: The optional serial output pins (SC0, SC1, and SC2) are controlled by the frame timing and are not affected by TE or RE.

6.4.2.2.3 CRB Serial Control 0 Direction (SCD0) Bit 2

SCD0 controls the direction of the SC0 I/O line. When SCD0 is cleared, SC0 is an input; when SCD0 is set, SC0 is an output (see Tables Table 6-5 and Table 6-6, and Figure 6-46). Hardware and software reset clear SCD0.

6.4.2.2.4 CRB Serial Control 1 Direction (SCD1) Bit 3

SCD1 controls the direction of the SC1 I/O line. When SCD1 is cleared, SC1 is an input; when SCD1 is set, SC1 is an output (see Tables Table 6-5 and Table 6-6 and Figure 6-46). Hardware and software reset clear SCD1.

6.4.2.2.5 CRB Serial Control 2 Direction (SCD2) Bit 4

SCD2 controls the direction of the SC2 I/O line. When SCD2 is cleared, SC2 is an input; when SCD2 is set, SC2 is an output (see Tables Table 6-5 and Table 6-6, and Figure 6-46). Hardware and software reset clear SCD2.

6.4.2.2.6 CRB Clock Source Direction (SCKD) Bit 5

SCKD selects the source of the clock signal used to clock the transmit shift register in the asynchronous mode and both the transmit shift register and the receive shift register in the synchronous mode. When SCKD is set, the internal clock source becomes the bit clock for the transmit shift register and word length divider and is the output on the SCK pin. When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCK pin, and an external clock source may drive this pin. Hardware and software reset clear SCKD.

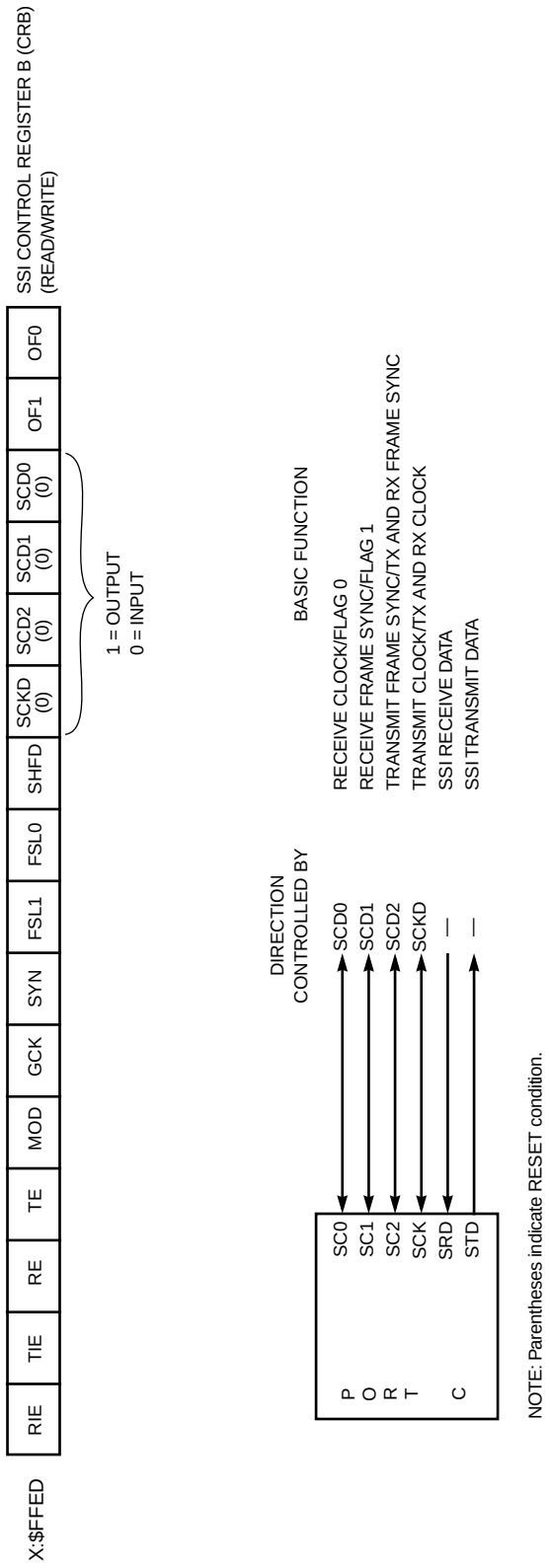


Figure 6-46 Serial Control, Direction Bits

6.4.2.2.7 CRB Shift Direction (SHFD) Bit 6

This bit causes the transmit shift register to shift data out MSB first when SHFD equals zero or LSB first when SHFD equals one. Receive data is shifted in MSB first when SHFD equals zero or LSB first when SHFD equals one. Hardware reset and software reset clear SHFD.

6.4.2.2.8 CRB Frame Sync Length (FSL0 and FSL1) Bits 7 and 8

These bits select the type of frame sync to be generated or recognized (see Table 6-11). If FSL1 equals zero and FSL0 equals zero, a word-length frame sync is selected for both TX and RX that is the length of the data word defined by bits WL1 and WL0. If FSL1 equals one and FSL0 equals zero, a 1-bit clock period frame sync is selected for both TX and RX. When FSL0 equals one, the TX and RX frame syncs are different lengths. Hardware reset and software reset clear FSL0 and FSL1.

Table 6-11 Frame Sync Length

FSL1	FSL0	Frame Sync Length
0	0	WL bit clock for both TX/RX
0	1	One-bit clock for TX and WL bit clock for RX
1	0	One-bit clock for both TX/RX
1	1	One-bit clock for RX and WL bit clock for TX

6.4.2.2.9 CRB Sync/Async (SYN) Bit 9

SYN controls whether the receive and transmit functions of the SSI occur synchronously or asynchronously with respect to each other. When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. Hardware reset and software reset clear SYN.

6.4.2.2.10 CRB Gated Clock Control (GCK) Bit 10

GCK is used to select between a continuously running data clock or a clock that runs only when there is data to be sent in the transmit shift register. When GCK is cleared, a continuous clock is selected; when GCK is set, the clock will be gated. Hardware reset and software reset clear GCK.

Note: For gated clock mode with externally generated bit clock, internally generated frame sync is not defined.

6.4.2.2.11 CRB SSI Mode Select (MOD) Bit 11

MOD selects the operational mode of the SSI. When MOD is cleared, the normal mode is selected; when MOD is set, the network mode is selected. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot. In network mode, a word is (possibly) transferred every time slot. For more details, see **6.4.3**. Hardware and software reset clear MOD.

6.4.2.2.12 CRB SSI Transmit Enable (TE) Bit 12

TE enables the transfer of data from TX to the transmit shift register. When TE is set and a frame sync is detected, the transmit portion of the SSI is enabled for that frame. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the SSI transmit shift register. The serial output is three-stated, and any data present in TX will not be transmitted (i.e., data can be written to TX with TE cleared; TDE will be cleared, but data will not be transferred to the transmit shift register).

The normal mode transmit enable sequence is to write data to TX or TSR before setting TE. The normal transmit disable sequence is to clear TE and TIE after TDE equals one.

In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completing transmission of the current data word until the beginning of the next frame. During that time period, the STD pin will remain in the high-impedance state. Hardware reset and software reset clear TE.

The on-demand mode transmit enable sequence can be the same as the normal mode, or TE can be left enabled.

Note: TE does not inhibit TDE or transmitter interrupts. TE does not affect the generation of frame sync or output flags.

6.4.2.2.13 CRB SSI Receive Enable (RE) Bit 13

When RE is set, the receive portion of the SSI is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RX. If data is being received while this bit is cleared, the remainder of the word will be shifted in and transferred to the SSI receive data register.

RE must be set in the normal mode and on-demand mode to receive data. In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the next data frame. Hardware and software reset clear RE.

Note: RE does not inhibit RDF or receiver interrupts. RE does not affect the generation

of a frame sync.

6.4.2.2.14 CRB SSI Transmit Interrupt Enable (TIE) Bit 14

The DSP will be interrupted when TIE and the TDE flag in the SSI status register is set. (In network mode, the interrupt takes effect in the next frame synch, not in the next time slot.) When TIE is cleared, this interrupt is disabled. However, the TDE bit will always indicate the transmit data register empty condition even when the transmitter is disabled with the TE bit. Writing data to TX or TSR will clear TDE, thus clearing the interrupt. Hardware and software reset clear RE.

There are two transmit data interrupts that have separate interrupt vectors:

1. Transmit data with exceptions – This interrupt is generated on the following condition:
   TIE=1, TDE=1, and TUE=1
2. Transmit data without exceptions – This interrupt is generated on the following condition:
   TIE=1, TDE=1, and TUE=0

See **SECTION 7 PROCESSING STATES** in the DSP56000 Family Manual for more information on exceptions.

6.4.2.2.15 CRB SSI Receive Interrupt Enable (RIE) Bit 15

When RIE is set, the DSP will be interrupted when RDF in the SSI status register is set. (In network mode, the interrupt takes effect in the next frame synch, not in the next time slot.) When RIE is cleared, this interrupt is disabled. However, the RDF bit still indicates the receive data register full condition. Reading the receive data register will clear RDF, thus clearing the pending interrupt. Hardware and software reset clear RIE.

There are two receive data interrupts that have separate interrupt vectors:

1. Receive data with exceptions – This interrupt is generated on the following condition:
   RIE=1, RDF=1, and ROE=1
2. Receive data without exceptions – This interrupt is generated on the following condition:
   RIE=1, RDF=1, and ROE=0

See **SECTION 7 PROCESSING STATES** in the DSP56000 Family Manual for more information on exceptions.

6.4.2.3 SSI Status Register (SSISR)

The SSISR is an 8-bit read-only status register used by the DSP to interrogate the status and serial input flags of the SSI. When the SSISR is read to the internal data bus, the register contents occupy the low-order byte of the data bus, and the high-order portion is zero filled. The status bits are described in the following paragraphs.

6.4.2.3.1 SSISR Serial Input Flag 0 (IF0) Bit 0

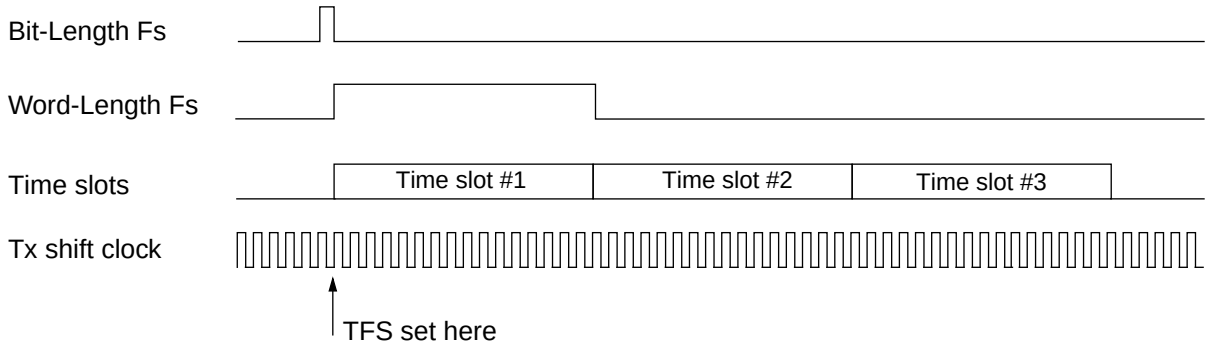
The SSI latches data present on the SC0 pin during reception of the first received bit after frame sync is detected. IF0 is updated with this data when the receive shift register is transferred into the receive data register. The IF0 bit is enabled only when SCD0 is cleared and SYN is set, indicating that SC0 is an input and the synchronous mode is selected (see Table 6-5); otherwise, IF0 reads as a zero when it is not enabled. Hardware, software, SSI individual, and STOP reset clear IF0.

6.4.2.3.2 SSISR Serial Input Flag 1 (IF1) Bit 1

The SSI latches data present on the SC1 pin during reception of the first received bit after frame sync is detected. The IF1 flag is updated with the data when the receiver shift register is transferred into the receive data register. The IF1 bit is enabled only when SCD1 is cleared and SYN is set, indicating that SC1 is an input and the synchronous mode is selected (see Table 6-5); otherwise, IF1 reads as a zero when it is not enabled. Hardware, software, SSI individual, and STOP reset clear IF1.

6.4.2.3.3 SSISR Transmit Frame Sync Flag (TFS) Bit 2

When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. If word-wide transmit frame sync is selected (FSL0=FSL1), this indicates that the frame sync was high at least at the beginning of the time slot if external frame sync is selected, or high throughout the time slot if internal frame sync was selected. If bit-wide transmit frame sync is selected (FSL0≠FSL1), this indicates that the frame sync (either internal or external) was high during the last Tx clock bit period prior to the current time slot, and that the frame sync falling edge corresponds to the assertion of the first output data bit, as shown below.



Data written to the transmit data register during the time slot when TFS is set will be transmitted (in network mode) during the second time slot in the frame. TFS is useful in network mode to identify the start of the frame. This is illustrated in a typical transmit interrupt handler:

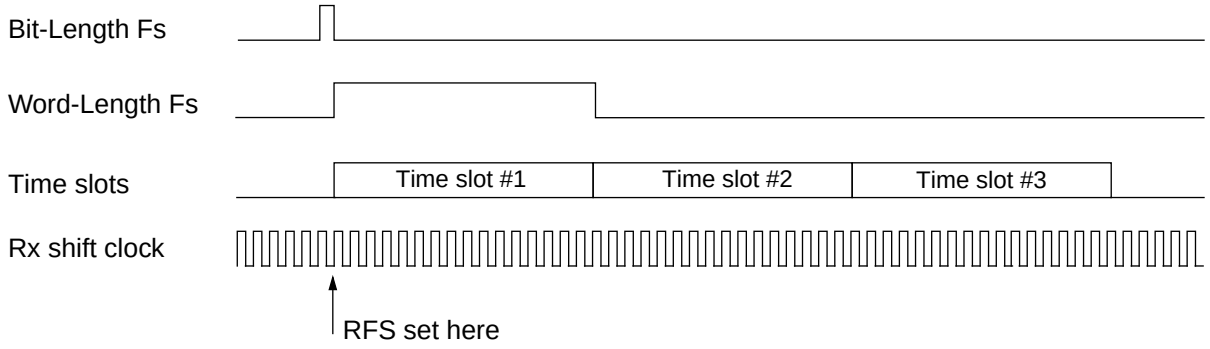
```
                MOVEP       X:(R4)+,X:SSITx
                JCLR        #2,X:SSISR,_NoTFS;1 = FIRST TIMESLOT
                            ;Do something
                JMP         _DONE
_NoTFS
                            ;Do something else
_DONE
```

Note: In normal mode, TFS will always read as a one when transmitting data because there is only one time slot per frame – the “frame sync” time slot.

TFS, which is cleared by hardware, software, SSI individual, or STOP reset, is not affected by TE.

6.4.2.3.4 SSISR Receive Frame Sync Flag (RFS) Bit 3

When set, RFS indicates that a receive frame sync occurred during reception of the word in the serial receive data register. This indicates that the data word is from the first time slot in the frame. If word-wide receive frame sync is selected (FSL1=0), this indicates that the frame sync was high at least at the beginning of the timeslot. If bit-wide receive frame sync is selected (FSL1=1), this indicates that the frame sync (either internal or external) was high during the last bit period prior to the current timeslot, and that the frame sync falling edge corresponds to the assertion of the first output data bit, as shown below.



When RFS is clear and a word is received, it indicates (only in network mode) that the frame sync did not occur during reception of that word. RFS is useful in network mode to identify the start of the frame. This feature is illustrated in a typical receive interrupt handler:

```
                MOVEP       X:SSIRx,X:(R4)+
                JCLR        #3,X:SSISR,_NoRFS ;1 = FIRST TIMESLOT
                            ;Do something
                JMP         _DONE
_NoRFS
                            ;Do something else
_DONE
```

Note: In normal mode, RFS will always read as a one when reading data because there is only one time slot per frame – the “frame sync” time slot.

RFS, which is cleared by hardware, software, SSI individual, or STOP reset, is not affected by RE.

6.4.2.3.5 SSISR Transmitter Underrun Error Flag (TUE) Bit 4

TUE is set when the serial transmit shift register is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TX) will be retransmitted.

In the normal mode, there is only one transmit time slot per frame. In the network mode, there can be up to 32 transmit time slots per frame.

TUE does not cause any interrupts; however, TUE does cause a change in the interrupt vector used for transmit interrupts so that a different interrupt handler may be used for a transmit underrun condition. If a transmit interrupt occurs with TUE set, the transmit data with exception status interrupt will be generated; if a transmit interrupt occurs with TUE clear, the transmit data without errors interrupt will be generated.

Hardware, software, SSI individual, and STOP reset clear TUE. TUE is also cleared by reading the SSISR with TUE set, followed by writing TX or TSR.

6.4.2.3.6 SSISR Receiver Overrun Error Flag (ROE) Bit 5

This flag is set when the serial receive shift register is filled and ready to transfer to the receiver data register (RX) and RX is already full (i.e., RDF=1). The receiver shift register is not transferred to RX. ROE does not cause any interrupts; however, ROE does cause a change in the interrupt vector used for receive interrupts so that a different interrupt handler may be used for a receive error condition. If a receive interrupt occurs with ROE set, the receive data with exception status interrupt will be generated; if a receive interrupt occurs with ROE clear, the receive data without errors interrupt will be generated.

Hardware, software, SSI individual, and STOP reset clear ROE. ROE is also cleared by reading the SSISR with ROE set, followed by reading the RX. Clearing RE does not affect ROE.

6.4.2.3.7 SSISR SSI Transmit Data Register Empty (TDE) Bit 6

This flag is set when the contents of the transmit data register are transferred to the transmit shift register; it is also set for a disabled time slot period in network mode (as if data were being transmitted after the TSR was written). Thirdly, it can be set by the hardware, software, SSI individual, or STOP reset. When set, TDE indicates that data should be written to the TX or to the time slot register (TSR). TDE is cleared when the DSP writes to the transmit data register or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, a DSP transmit data interrupt request will be issued when TDE is set. The vector of the interrupt will depend on the state of the transmitter underrun bit.

6.4.2.3.8 SSISR SSI Receive Data Register Full (RDF) Bit 7

RDF is set when the contents of the receive shift register are transferred to the receive data register. RDF is cleared when the DSP reads the receive data register or cleared by hardware, software, SSI individual, or STOP reset. If RIE is set, a DSP receive data interrupt request will be issued when RDF is set. The vector of the interrupt request will depend on the state of the receiver overrun bit.

6.4.2.3.9 SSI Receive Shift Register

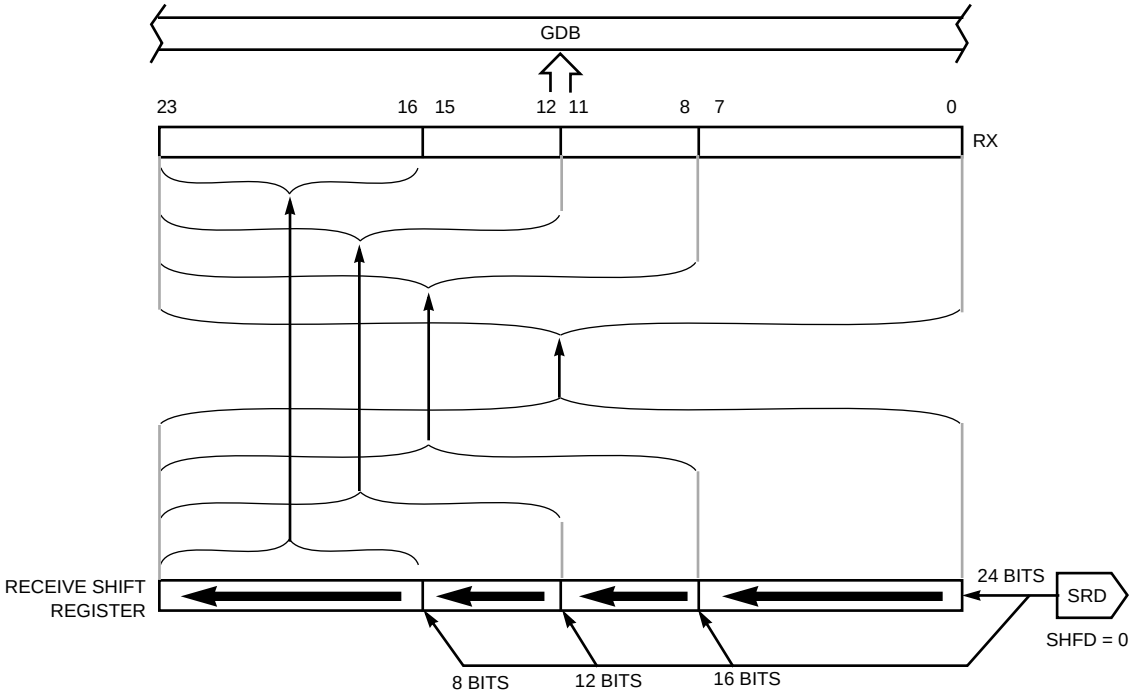
This 24-bit shift register receives the incoming data from the serial receive data pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O (or gated clock) is asserted. Data is assumed to be received MSB first if SHFD equals zero and LSB first if SHFD equals one. Data is transferred to the SSI receive data register after 8, 12, 16, or 24 bits have been shifted in, depending on the word-length control bits in the CRA (see Figure 6-47).

6.4.2.3.10 SSI Receive Data Register (RX)

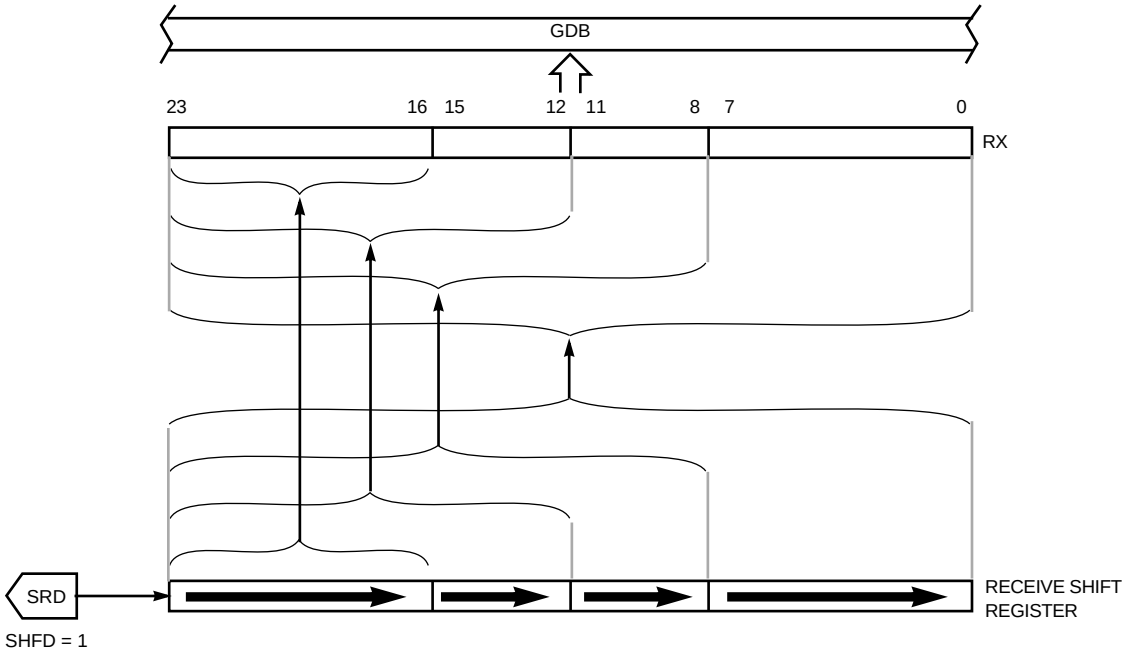
RX is a 24-bit read-only register that accepts data from the receive shift register as it becomes full. The data read will occupy the most significant portion of the receive data register (see Figure 6-47). The unused bits (least significant portion) will read as zeros. The DSP is interrupted whenever RX becomes full if the associated interrupt is enabled.

6.4.2.3.11 SSI Transmit Shift Register

This 24-bit shift register contains the data being transmitted. Data is shifted out to the serial transmit data pin by the selected (internal/external) bit clock when the associated frame sync I/O (or gated clock) is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12, 16, or 24 bits (determined by the word-length control bits in CRA). The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register MSB first if SHFD equals zero and LSB first if SHFD equals one (see Figure 6-48).

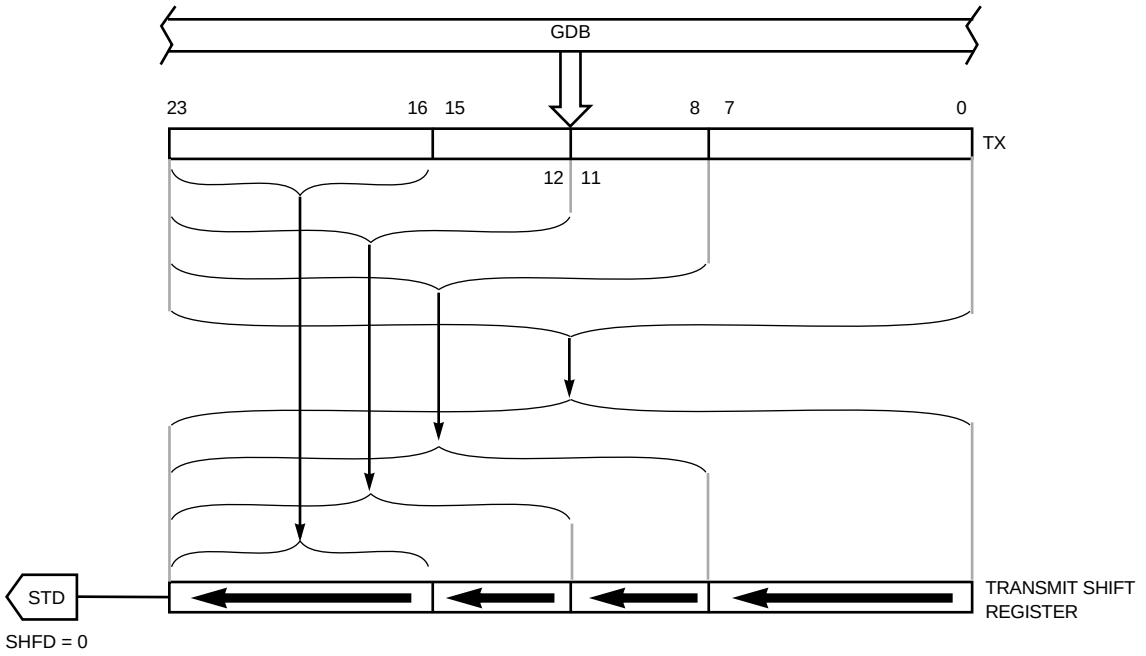


(a) SHFD = 0

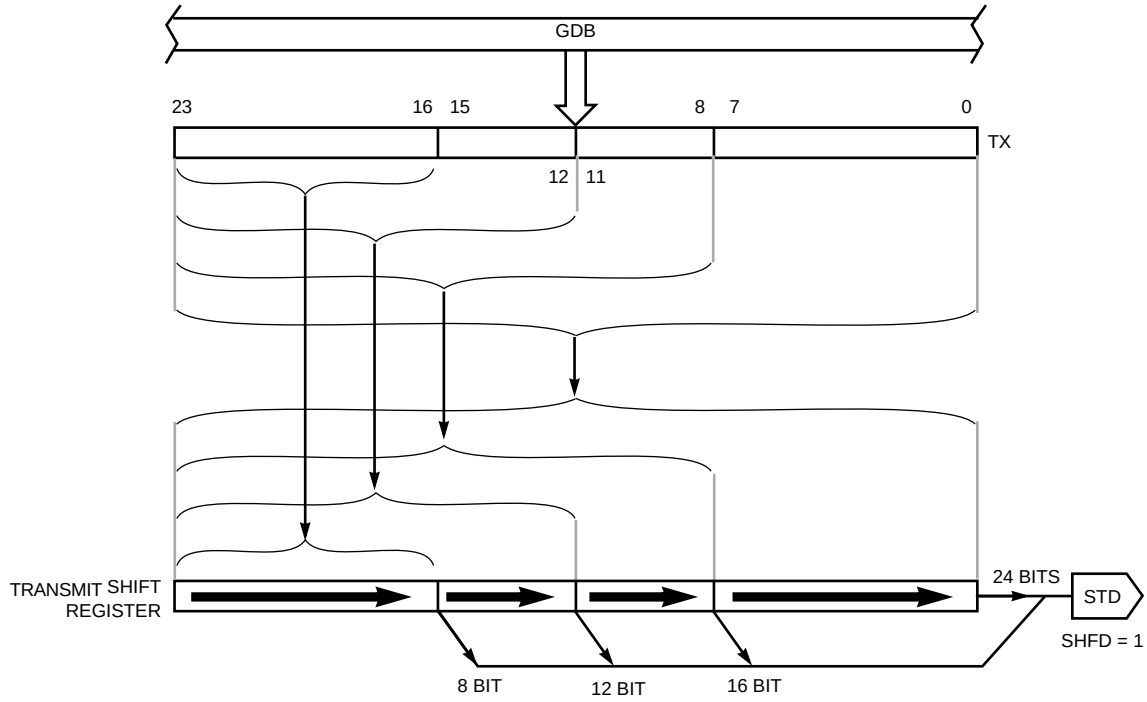


(b) SHFD = 1

Figure 6-47 Receive Data Path



(a) SHFD = 0



(b) SHFD = 1

Figure 6-48 Transmit Data Path

6.4.2.3.12 SSI Transmit Data Register (TX)

TX is a 24-bit write-only register. Data to be transmitted is written into this register and is automatically transferred to the transmit shift register. The data written (8, 12, 16, or 24 bits) should occupy the most significant portion of TX (see Figure 6-48). The unused bits (least significant portion) of TX are don't care bits. The DSP is interrupted whenever TX becomes empty if the transmit data register empty interrupt has been enabled.

6.4.2.3.13 Time Slot Register (TSR)

TSR is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. For the purposes of timing, TSR is a write-only register that behaves like an alternative transmit data register, except that, rather than transmitting data, the transmit data pin is in the high-impedance state for that time slot.

6.4.3 Operational Modes and Pin Definitions

Table 6-12 and Table 6-13 completely describe the SSI operational modes and pin definitions (Table 6-5 is a simplified version of these tables). The operational modes are as follows:

1. Continuous Clock
   - Mode 1 – Normal with Internal Frame Sync
   - Mode 2 – Network with Internal Frame Sync
   - Mode 3 – Normal with External Frame Sync
   - Mode 4 – Network with External Frame Sync
2. Gated Clock
   - Mode 5 – External Gated Clock
   - Mode 6 – Normal with Internal Gated Clock
   - Mode 7 – Network with Internal Gated Clock
3. Special Case (Both Gated and Continuous Clock)
   - Mode 8 – On-Demand Mode (Transmitter Only)
   - Mode 9 – Receiver Follows Transmitter Clocking

6.4.4 Registers After Reset

Hardware or software reset clears the port control register bits, which configure all I/O as general-purpose input. The SSI will remain in reset while all SSI pins are programmed as general-purpose I/O (CC8–CC3=0) and will become active only when at least one of the SSI I/O pins is programmed as not general-purpose I/O. Table 6-14 shows how each type of reset affects each SSI register bit.

Table 6-12 Mode and Pin Definition Table – Continuous Clock

Control Bits								Mode		SC0		SC1		SC2		SCK	
MOD	GCLK	SYN	SCD2	SCD1	SCD0	SCKD	DC4-DC0	TX	RX	In	Out	In	Out	In	Out	In	Out
0	0	0	1	1	X	X	X	1	1	RXC	RXC	—	FSR	—	FST	TXC	TXC
0	0	1	1	X	X	X	X	1	1	F0	F0	F1	F1	—	FS*	*XC	*XC
1	0	0	1	1	X	X	1	2	2	RXC	RXC	—	FSR	—	FST	TXC	TXC
1	0	1	1	X	X	X	1	2	2	F0	F0	F1	F1	—	FS*	*XC	*XC
0	0	0	0	1	X	X	X	3	1	RXC	RXC	—	FSR	FST	—	TXC	TXC
0	0	0	1	0	X	X	X	1	3	RXC	RXC	FSR	—	—	FST	TXC	TXC
0	0	0	0	0	X	X	X	3	3	RXC	RXC	FSR	—	FST	—	TXC	TXC
0	0	1	0	X	X	X	X	3	3	F0	F0	F1	F1	FS*	—	*XC	*XC
1	0	0	0	1	X	X	X	4	2	RXC	RXC	—	FSR	FST	—	TXC	TXC
1	0	0	1	0	X	X	1	2	4	RXC	RXC	FSR	—	—	FST	TXC	TXC
1	0	0	0	0	X	X	X	4	4	RXC	RXC	FSR	—	FST	—	TXC	TXC
1	0	1	0	X	X	X	X	4	4	F0	F0	F1	F1	FS*	—	*XC	*XC
1	0	0	1	1	X	X	0	8	2	RXC	RXC	—	FSR	—	FST	TXC	TXC
1	0	1	1	X	X	X	0	8	9	F0	F0	F1	F1	—	FS*	*XC	*XC
1	0	0	1	0	X	X	0	8	4	RXC	RXC	FSR	—	—	FST	TXC	TXC

DC4-DC0 = 0 means that bits DC4 = 0, DC3 = 0, DC2 = 0, DC1 = 0, and DC0 = 0
DC4-DC0 = 1 means that bits DC4-DC0≠0
TXC — Transmitter Clock
RXC — Receiver Clock
*XC — Transmitter/Receiver Clock (Synchronous Operation)
FST — Transmitter Frame Sync
FSR — Receiver Frame Sync
FS* — Transmitter/Receiver Frame Sync (Synchronous Operation)
F0 — Flag 0
F1 — Flag 1

Table 6-13 Mode and Pin Definition Table – Gated Clock

Control Bits								Mode		SC0		SC1		SC2		SCK	
MOD	GCLK	SYN	SCD2	SCD1	SCD0	SCKD	DC4-DC0	TX	RX	In	Out	In	Out	In	Out	In	Out
0	1	0	X	X	1	1	X	6	6	—	RXC	?	FSR	?	FST	—	TXC
0	1	1	X	X	X	1	X	6	6	F0	F0	F0	F1	?	FS*	—	*XC
0	1	0	X	X	1	0	X	5	6	—	RXC	?	FSR	?	?	TXC	—
0	1	0	X	X	0	0	X	5	5	RXC	—	?	?	?	?	TXC	—
0	1	1	X	X	X	0	X	5	5	F0	F0	F1	F1	?	?	*XC	—
1	1	0	X	X	1	1	0	8	7	—	RXC	?	FSR	?	FST	—	TXC
1	1	0	X	X	0	1	0	8	5	RXC	—	?	?	?	FST	—	TXC
1	1	1	X	X	X	1	0	8	9	F0	F0	F1	F1	?	FS*	—	*XC
0	1	0	X	X	0	1	X	6	5	RXC	—	?	?	?	FST	—	TXC

DC4–DC0=0 means that bits DC4=0, DC3=0, DC2=0, DC1=0, and DC0=0.
TXC – Transmitter Clock
RXC – Receiver Clock
*XC – Transmitter/Receiver Clock (Synchronous Operation)
FST – Transmitter Frame Sync
FSR – Receiver Frame Sync
FS* – Transmitter/Receiver Frame Sync (Synchronous Operation)
F0 – Flag 0
F1 – Flag 1
? – Undefined

Table 6-14 SSI Registers After Reset

Register Name	Register Data	Bit Number	Reset			
			HW Reset	SW Reset	Individual Reset	ST Reset
CRA	PSR	15	0	0	–	–
	WL(2–0)	13,14	0	0	–	–
	DC(4–0)	8–12	0	0	–	–
	PM(7–0)	0–7	0	0	–	–
CRB	RIE	15	0	0	–	–
	TIE	14	0	0	–	–
	RE	13	0	0	–	–
	TE	12	0	0	–	–
	MOD	11	0	0	–	–
	GCK	10	0	0	–	–
	SYN	9	0	0	–	–
	FSL1	8	0	0	–	–
	FSL0	7	0	0	–	–
	SHFD	6	0	0	–	–
	SCKD	5	0	0	–	–
	SCD(2–0)	2–4	0	0	–	–
	OF(1–0)	0,1	0	0	–	–
SSISR	RDF	7	0	0	0	0
	TDE	6	1	1	1	1
	ROE	5	0	0	0	0
	TUE	4	0	0	0	0
	RFS	3	0	0	0	0
	TFS	2	0	0	0	0
	IF(1–0)	0,1	0	0	0	0
RDR	RDR (23–0)	23–0	–	–	–	–
TDR	TDR (23–0)	23–0	–	–	–	–
RSR	RDR (23–0)	23–0	–	–	–	–
TSR	RDR (23–0)	23–0	–	–	–	–

NOTES:
1. RSR – SSI receive shift register
2. TSR – SSI transmit shift register
3. HW – Hardware reset is caused by asserting the external pin $\overline{\text{RESET}}$.
4. SW – Software reset is caused by executing the RESET instruction.
5. IR – Individual reset is caused by SSI peripheral pins (i.e., PCC(3–8)) being configured as general-purpose I/O.
6. ST – Stop reset is caused by executing the STOP instruction.

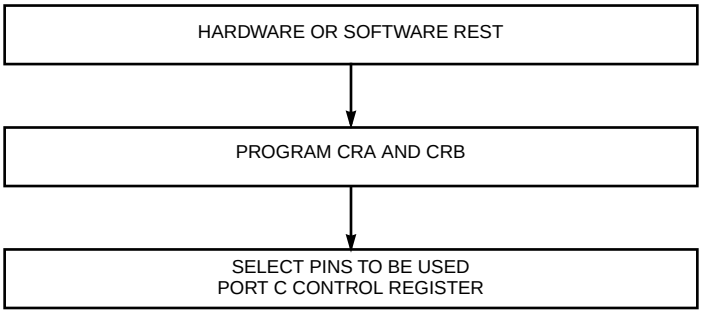


Figure 6-49 SSI Initialization Block Diagram

6.4.5 SSI Initialization

The correct way to initialize the SSI is as follows:

1. Hardware, software, SSI individual, or STOP reset
2. Program SSI control registers
3. Configure SSI pins (at least one) as not general-purpose I/O

During program execution, CC8–CC3 may be cleared, causing the SSI to stop serial activity and enter the individual reset state. All status bits of the interface will be set to their reset state; however, the contents of CRA and CRB are not affected. This procedure allows the DSP program to reset each interface separately from the other internal peripherals.

The DSP program must use an SSI reset when changing the MOD, GCK, SYN, SCKD, SCD2, SCD1, or SCD0 bits to ensure proper operation of the interface. Figure 6-49 is a flowchart illustrating the three initialization steps previously listed. Figure 6-50, Figure 6-51, and Figure 6-52 provide additional detail to the flowchart.

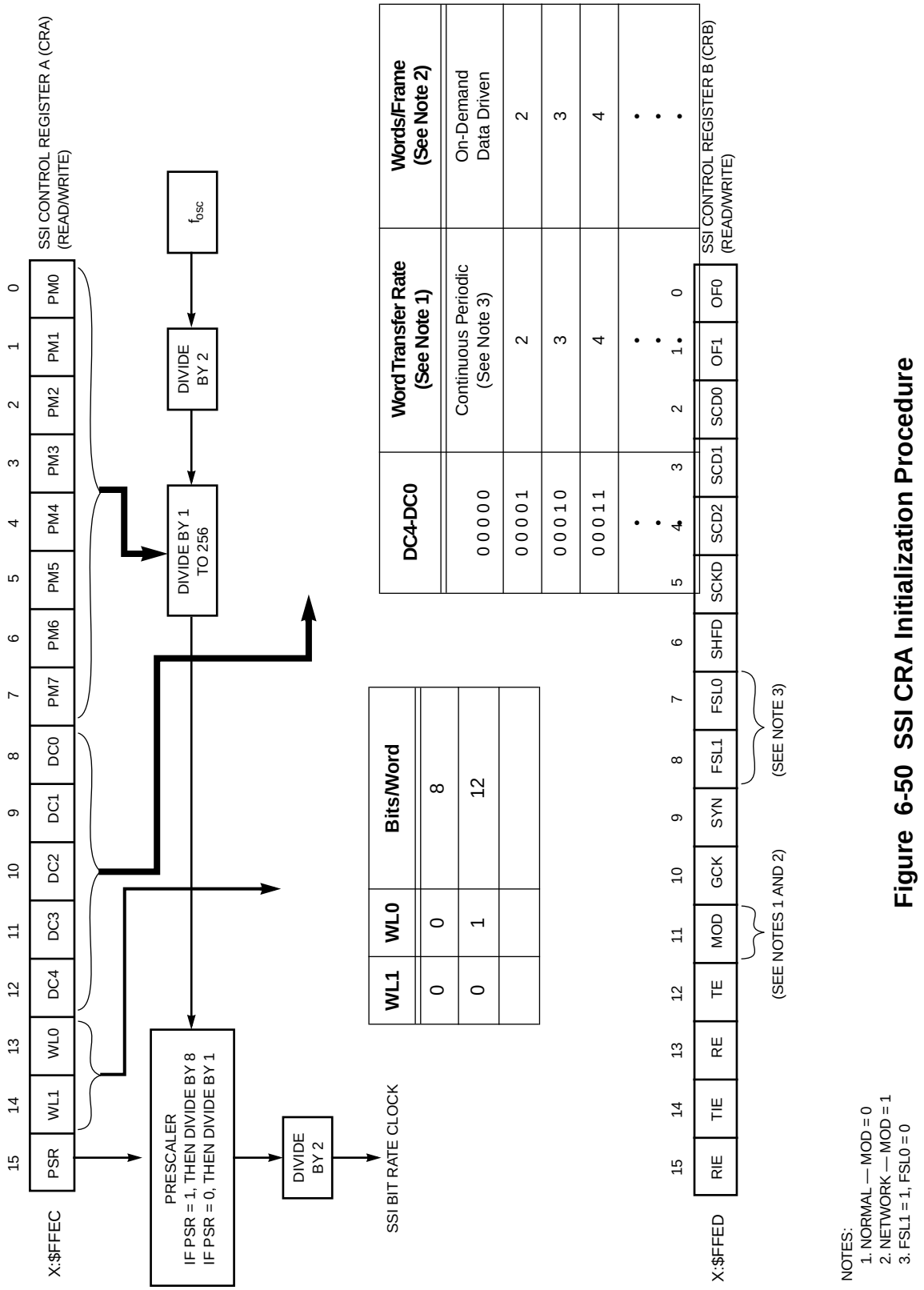


Figure 6-50 SSI CRA Initialization Procedure

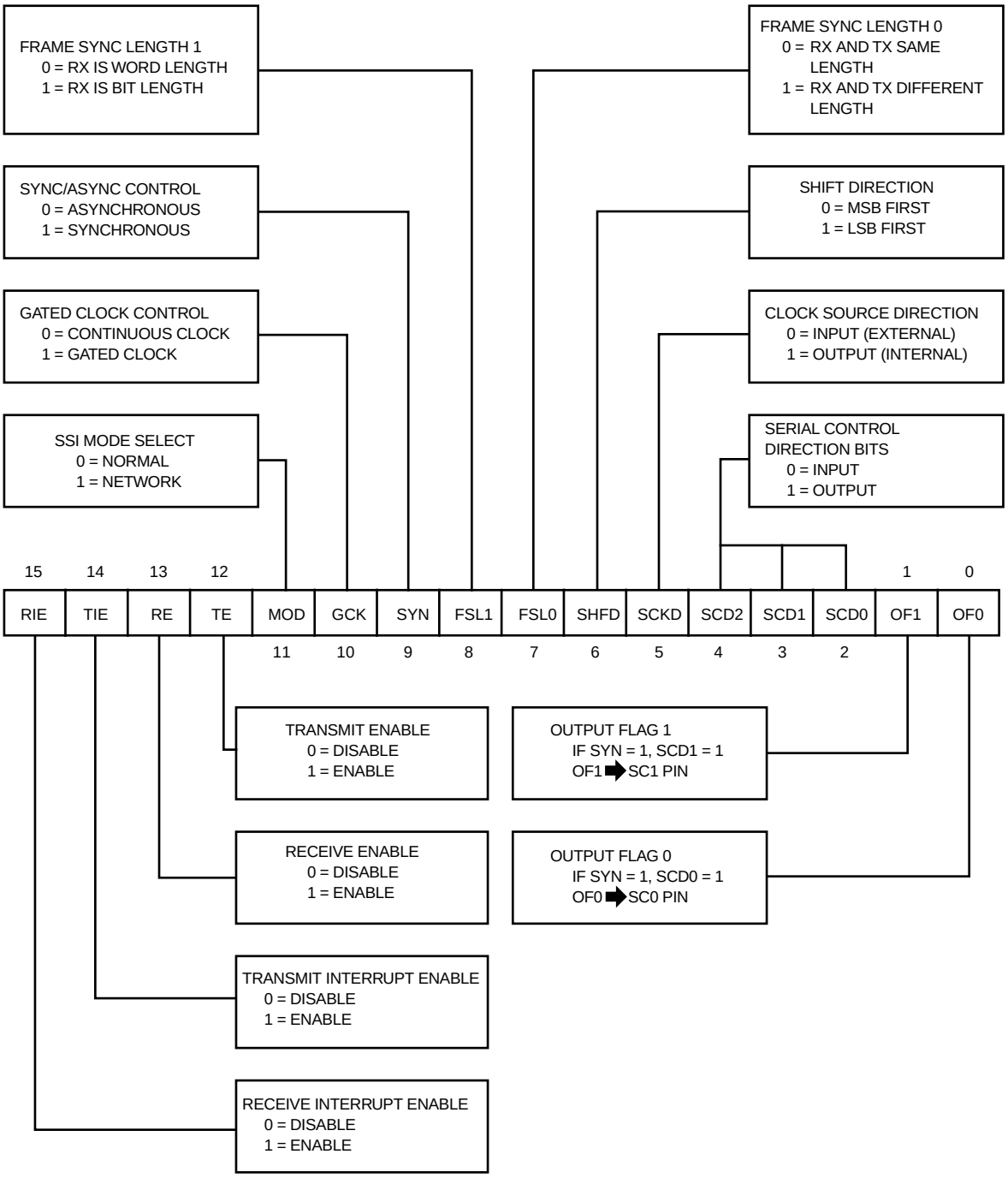
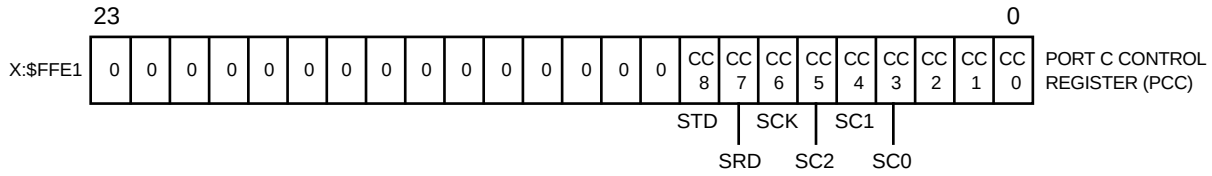


Figure 6-51 SSI CRB Initialization Procedure



CCx	Function
0	GPIO
1	Serial Interface

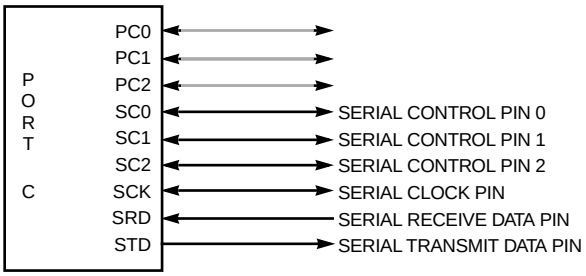


Figure 6-52 SSI Initialization Procedure

Figure 6-52 shows the six control bits in the PCC, which select the six SSI pins as either general-purpose I/O or as SSI pins. The STD pin can only transmit data; the SRD pin can only receive data. The other four pins can be inputs or outputs, depending on how they are programmed. This programming is accomplished by setting bits in CRA and CRB as shown in Figure 6-46. The CRA (see Figure 6-50) sets the SSI bit rate clock with PSR and PM0–PM7, sets the word length with WL1 and WL0, and sets the number of words in a frame with DC0–DC4. There is a special case where DC4–DC0 equals zero (one word per frame). Depending on whether the normal or network mode is selected (MOD=0 or MOD=1, respectively), either the continuous periodic data mode is selected, or the on-demand data driven mode is selected. The continuous periodic mode requires that FSL1 equals one and FSL0 equals zero. Figure 6-51 shows the meaning of each individual bit in the CRB. These bits should be set according to the application requirements.

Table 6-15 (a) and Table 6-15 (b) provide a convenient listing of PSR and PM0–PM7 settings for the common data communication rates and the highest rate possible for the SSI for the chosen crystal frequencies. The crystal frequency selected for Table 6-15 (a) is the one used by the DSP56002ADS board; the one selected for Table 6-15 (b) is the closest one to 40 MHz that divides down to exactly 128 kHz. If an exact baud rate is required, the crystal frequency may have to be selected. Table 6-16 gives the PSR and PM0–PM7 settings in addition to the required crystal frequency for three common telecommunication frequencies.

Table 6-15 (a) SSI Bit Rates for a 40-MHz Crystal

Bit Rate (BPS)	PSR	PM
1000	1	$4E1
2000	1	$270
4000	1	$138
8000	1	$9B
16K	1	$4D
32K	1	$26
64K	0	$9B
128K	0	$4D
10M	0	$00

BPS = $f_{osc} \div (4 \times (7(PSR) +1) \times (PM + 1))$ where
f_{osc}=40 MHz
PSR = 0 or 1
PM = 0 to $FFF

Table 6-15 (b) SSI Bit Rates for a 39.936-MHz Crystal

Bit Rate (BPS)	PSR	PM
1000	1	$4DF
2000	1	$26F
4000	1	$137
8000	1	$9B
16K	1	$4D
32K	1	$26
64K	0	$9B
128K	0	$4D
9.984M	0	$00

BPS = $f_{osc} \div (4 \times (7(PSR) +1) \times (PM + 1))$ where
f_{osc}=39.936 MHz
PSR = 0 or 1
PM = 0 to $FFF

Table 6-16 Crystal Frequencies Required for Codecs

Bit Rate (BPS)	PSR	PM	Crystal Frequency
1.536M	0	$05	36.864 MHz
1.544M	0	$05	37.056 MHz
2.048M	0	$03	32.678 MHz

BPS = $f_{osc} \div (4 \times (7(PSR) +1) \times (PM + 1))$
PSR = 0 or 1
PM = 0 to $FFF

6.4.6 SSI Exceptions

The SSI can generate four different exceptions (see Figure 6-53 and Figure 6-54):

1. SSI Receive Data – occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. Reading RX clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead.
2. SSI Receive Data with Exception Status – occurs when the receive interrupt is enabled, the receive data register is full, and a receiver overrun error has occurred. ROE is cleared by first reading the SSISR and then reading RX.
3. SSI Transmit Data – occurs when the transmit interrupt is enabled, the transmit data register is empty, and no transmitter error conditions exist. Writing to TX or the TSR will clear this interrupt. This error-free interrupt may use a fast interrupt service routine for minimum overhead.
4. SSI Transmit Data with Exception Status – occurs when the transmit interrupt is enabled, the transmit data register is empty, and a transmitter underrun error has occurred. TUE is cleared by first reading the SSISR and then writing to TX or the TSR to clear the pending interrupt.

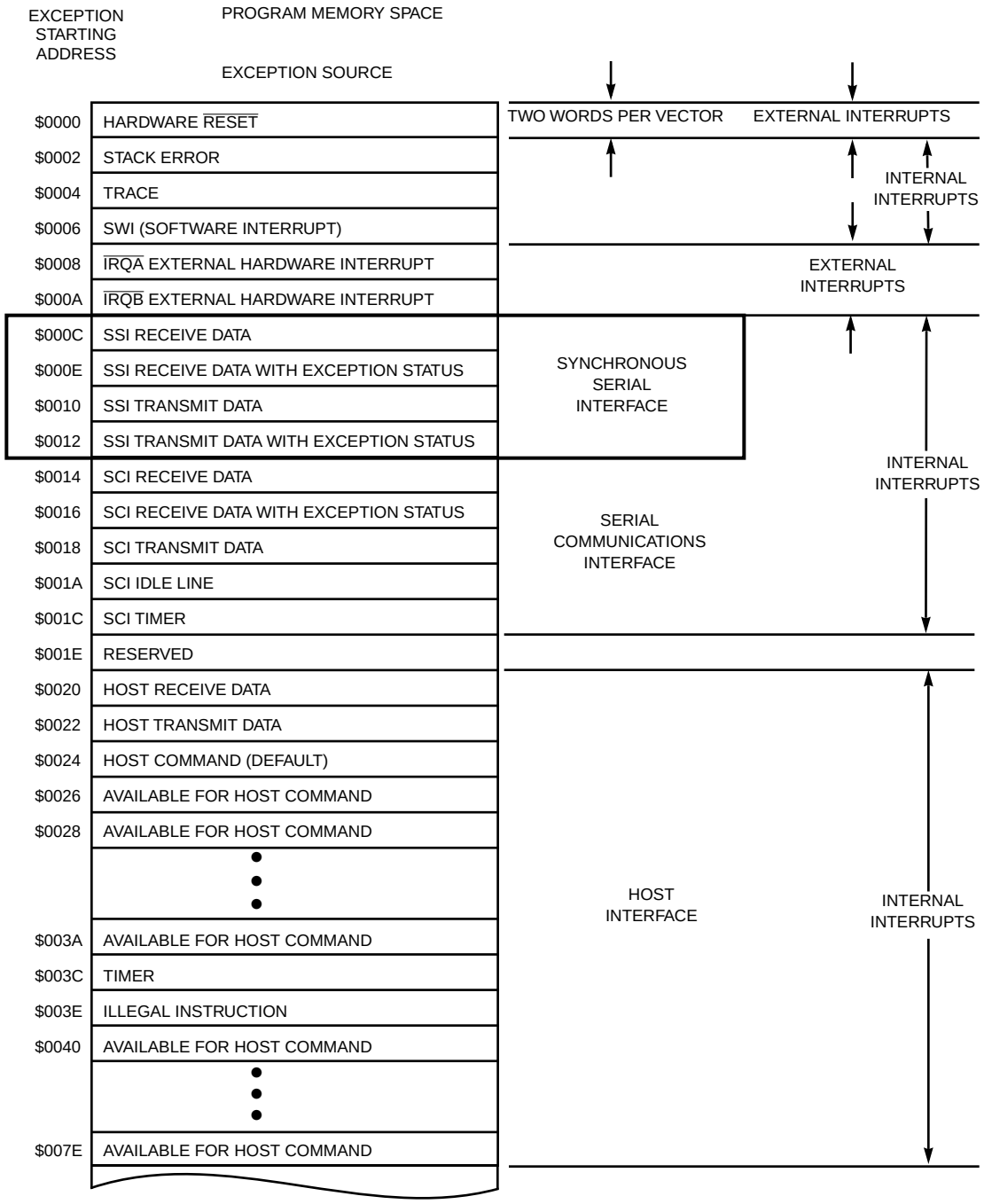


Figure 6-53 SSI Exception Vector Locations

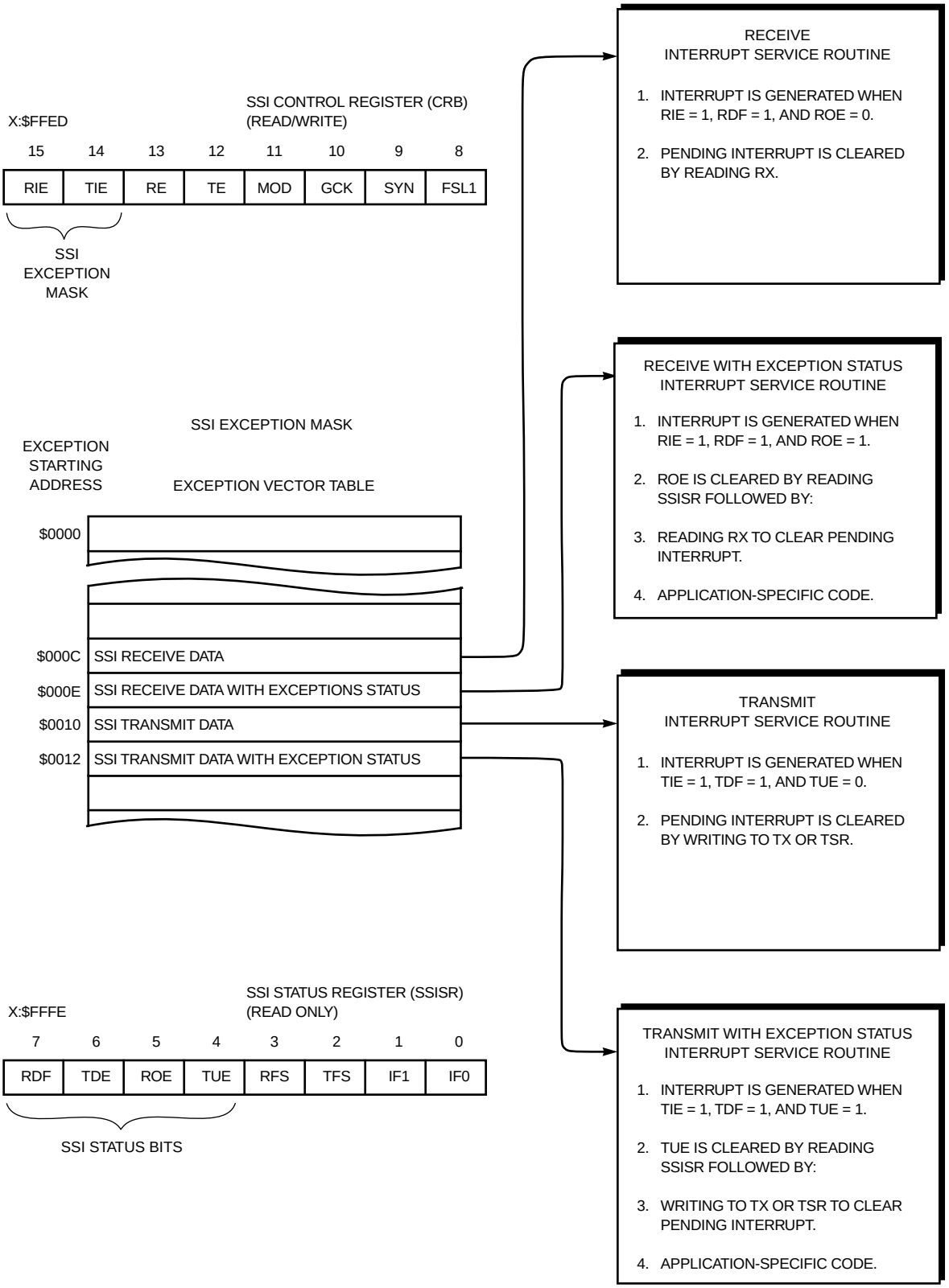


Figure 6-54 SSI Exceptions

Table 6-17 SSI Operating Modes

Operating Format	Serial Clock	TX, RX Sections	Typical Applications
Normal	Continuous	Asynchronous	Single Asynchronous Codec; Stream-Mode Channel Interface
Normal	Continuous	Synchronous	Multiple Synchronous Codecs
Normal	Gated	Asynchronous	DSP-to-DSP; Serial Peripherals (A/D,D/A)
Normal	Gated	Synchronous	SPI-Type Devices; DSP to MCU
Network	Continuous	Asynchronous	TDM Networks
Network	Continuous	Synchronous	TDM Codec Networks, TDM DSP Networks
On Demand	Gated	Asynchronous	Parallel-to-Serial and Serial-to-Parallel Conversion
On Demand	Gated	Synchronous	DSP to SPI Peripherals

6.4.7 Operating Modes – Normal, Network, and On-Demand

The SSI has three basic operating modes and many data/operation formats. These modes can be programmed by several bits in the SSI control registers. Table 6-17 lists the SSI operating modes and some of the typical applications in which they may be used.

The data/operation formats are selected by choosing between gated and continuous clocks, synchronization of transmitter and receiver, selection of word or bit frame sync, and whether the LSB is transferred first or last. The following paragraphs describe how to select a particular data/operation format and describe examples of normal-mode and network-mode applications. The on-demand mode is selected as a special case of the network mode.

The SSI can function as an SPI master or SPI slave, using additional logic for arbitration, which is required because the SSI interface does not perform SPI master/slave arbitration. An SPI master device always uses an internally generated clock; whereas, an SPI slave device always uses an external clock.

6.4.7.1 Data/Operation Formats

The data/operation formats available to the SSI are selected by setting or clearing control bits in the CRB. These control bits are MOD, GCK, SYN, FSL1, FSL0, and SHFD.

6.4.7.1.1 Normal/Network Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the MOD bit in the CRB (see Figure 6-55). For normal mode, the SSI functions with one data word of I/O per frame (see Figure 6-56). For the network mode, 2 to 32 data words of

I/O may be used per frame. In either case, the transfers are periodic. The normal mode is typically used to transfer data to/from a single device. Network mode is typically used in time division multiplexed (TDM) networks of codecs or DSPs with multiple words per frame (see Figure 6-57, which shows two words in a frame with either word-length or bit-length frame sync). The frame sync shown in Figure 6-55 is the word-length frame sync. A bit-length frame sync can be chosen by setting FSL1 and FSL0 for the configuration desired.

6.4.7.1.2 Continuous/Gated Clock Selection

The TX and RX clocks may be programmed as either continuous or gated clock signals by the GCK bit in the CRB. A continuous TX and RX clock is required in applications such as communicating with some codecs where the clock is used for more than just data transfer. A gated clock, in which the clock only toggles while data is being transferred, is useful for many applications and is required for SPI compatibility. The frame sync outputs may be used as a start conversion signal by some A/D and D/A devices.

Figure 6-58 illustrates the difference between continuous clock and gated clock systems. A separate frame-sync signal is required in continuous clock systems to delimit the active clock transitions. Although the word-length frame sync is shown in Figure 6-58, a bit-length frame sync can be used (see Figure 6-59). In gated clock systems, frame synchronization is inherent in the clock signal; thus a separate sync signal is not required (see Figure 6-60 and Figure 6-61). The SSI can be programmed to generate frame sync outputs in gated clock mode but does not use frame sync inputs.

Input flags (see Figure 6-60 and Figure 6-61) are latched on the negative edge of the first data bit of a frame. Output flags are valid during the entire frame.

6.4.7.1.3 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of this interface may be synchronous or asynchronous – i.e., the transmitter and receiver may use common clock and synchronization signals (synchronous operating mode, see Figure 6-62) or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in CRB selects synchronous or asynchronous operation. Since the SSI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

Figure 6-63 illustrates the operation of the SYN bit in the CRB. When SYN equals zero, the SSI TX and RX clocks and frame sync sources are independent. If SYN equals one, the SSI TX and RX clocks and frame sync come from the same source (either external or internal).

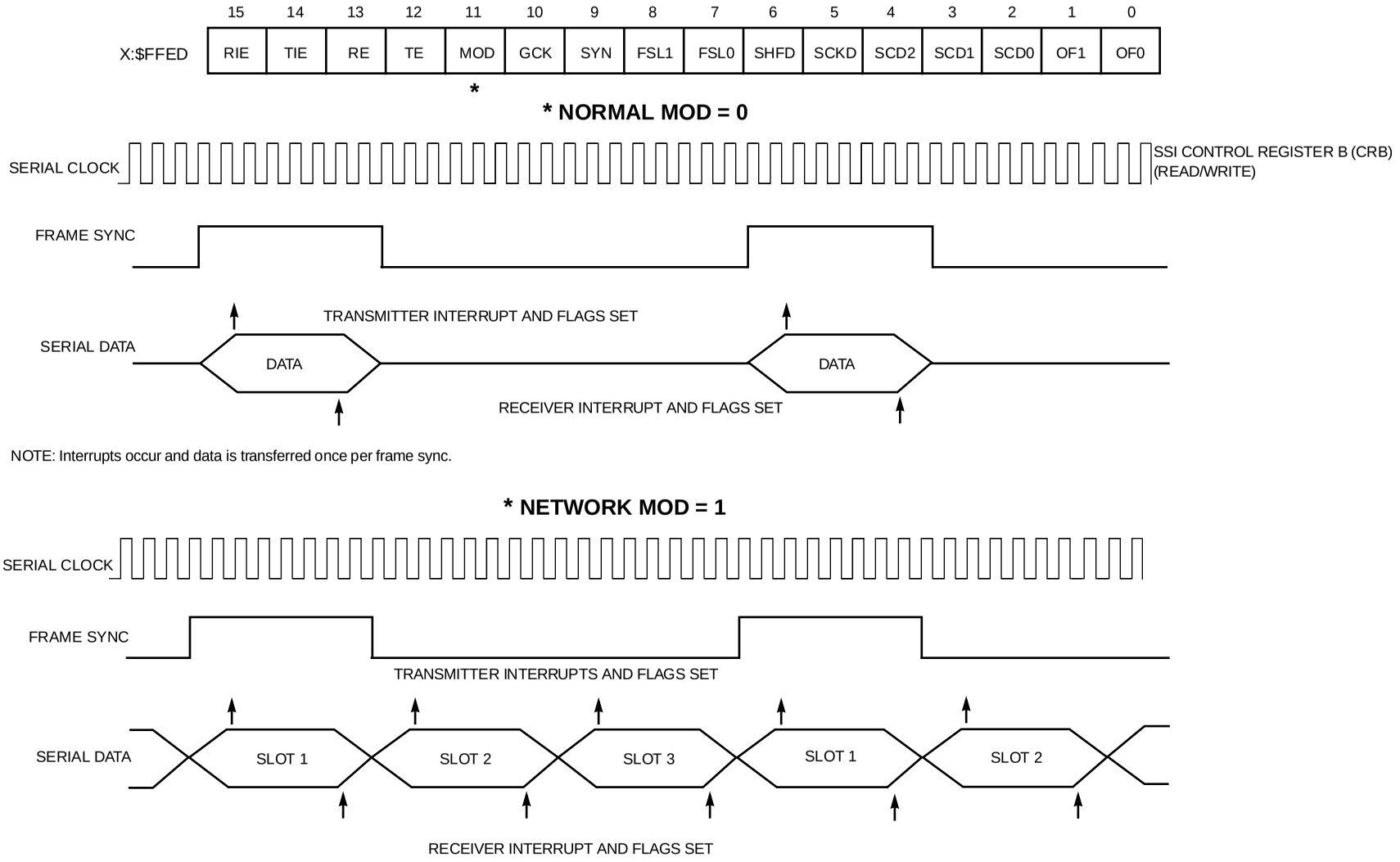


Figure 6-55 CRB MOD Bit Operation

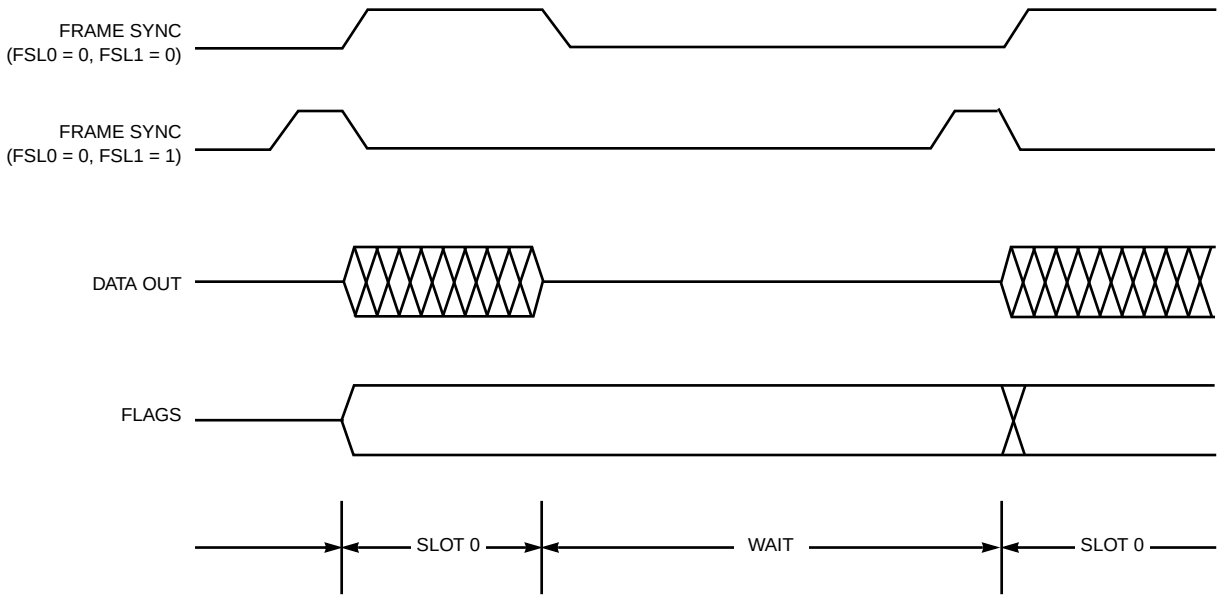


Figure 6-56 Normal Mode, External Frame Sync (8 Bit, 1 Word in Frame)

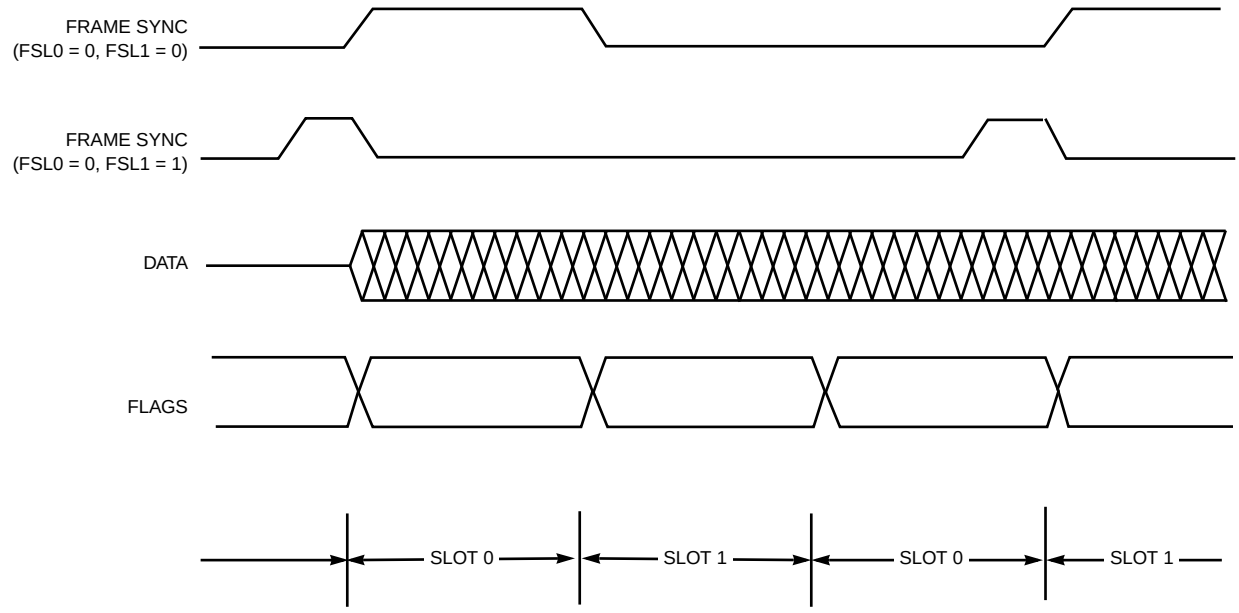


Figure 6-57 Network Mode, External Frame Sync (8 Bit, 2 Words in Frame)

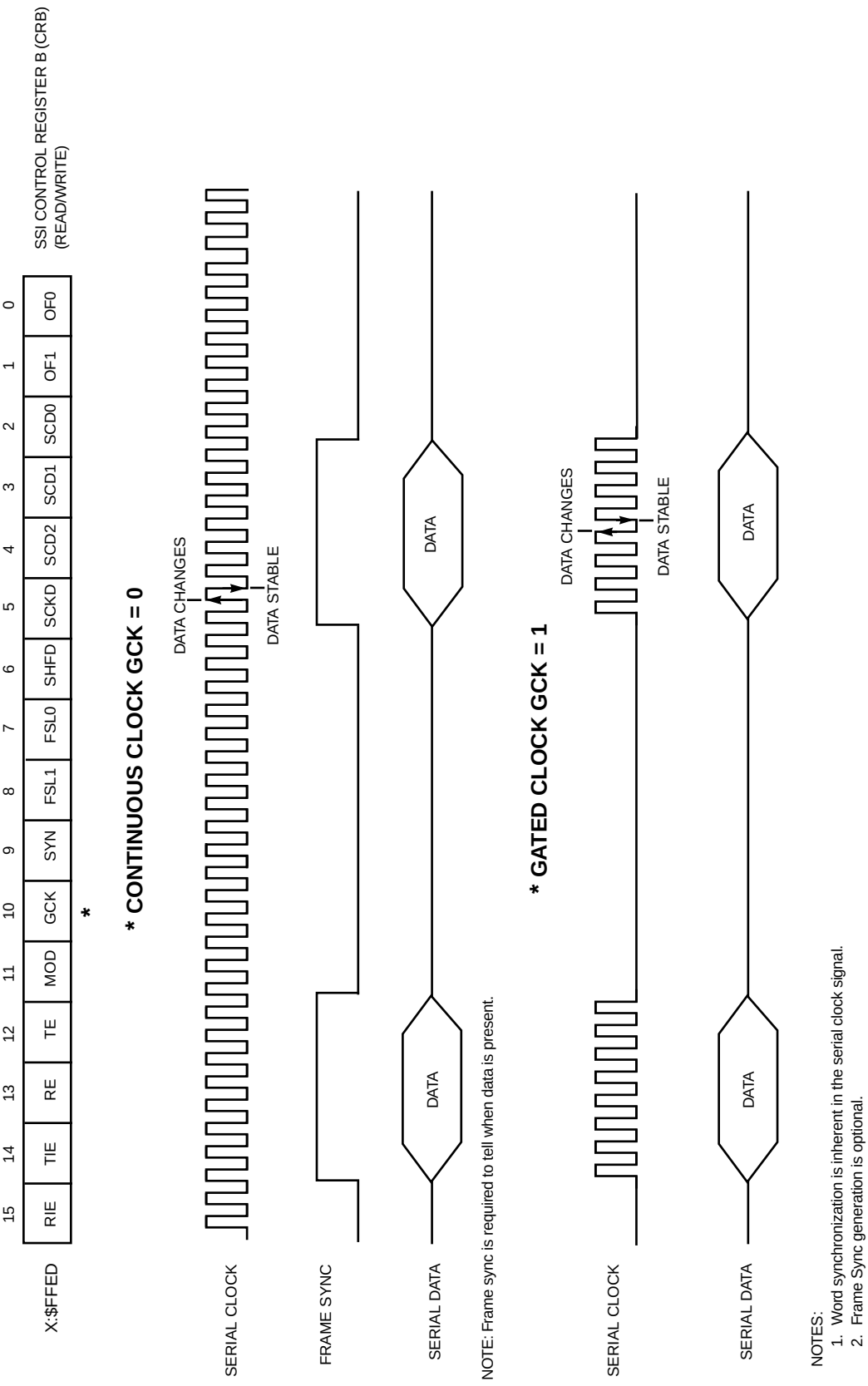
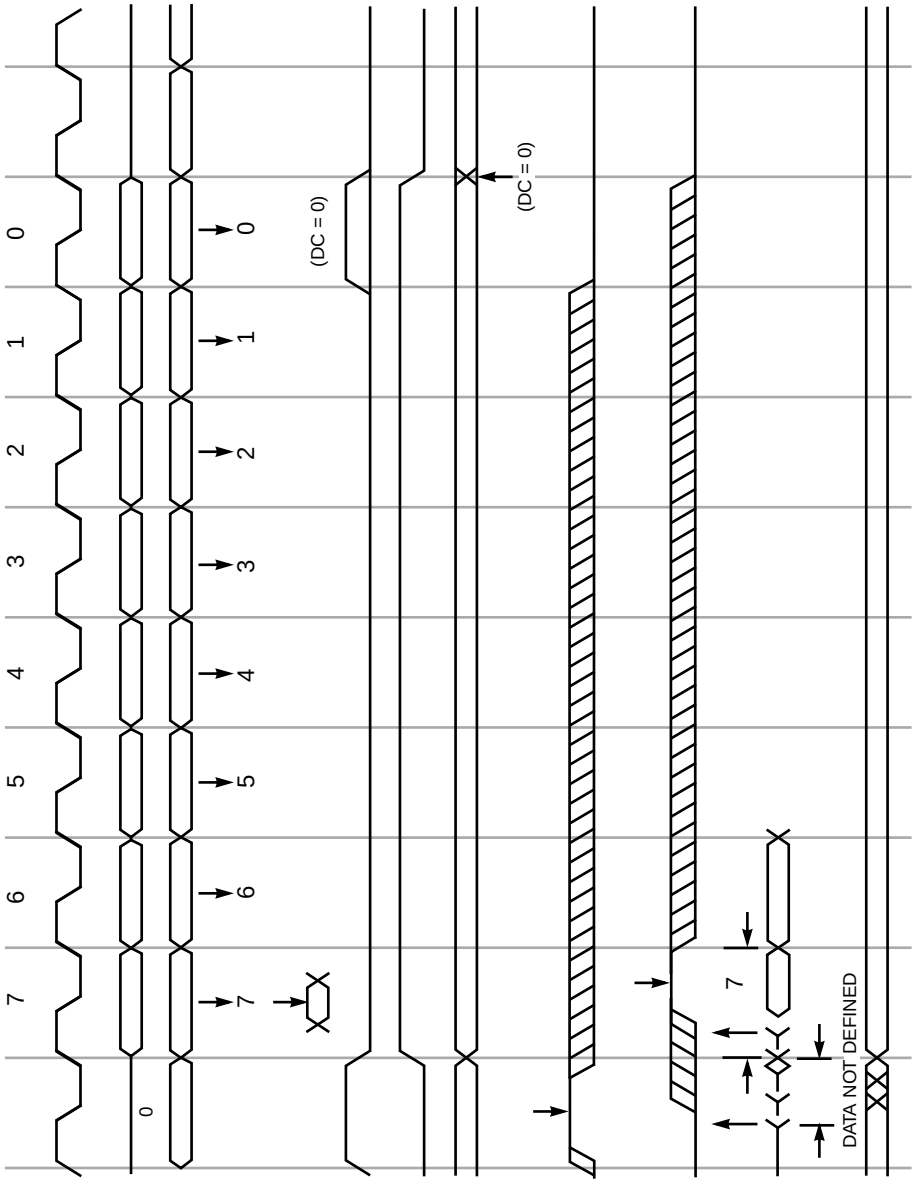


Figure 6-58 CRB GCK Bit Operation



NOTES:
1. For FSL1 = 0 the frame sync is latched and enables the STD output buffer, but data may not be valid until the rising edge of the bit clock.
2. WL bit frame sync (FSL0 = 0, FSL1 = 0) is not defined for DC = 0 in continuous clock mode.
3. Data and flags transition after external frame sync but not before the rising edge of the clock.

Figure 6-59 Continuous Clock Timing Diagram (8-Bit Example)

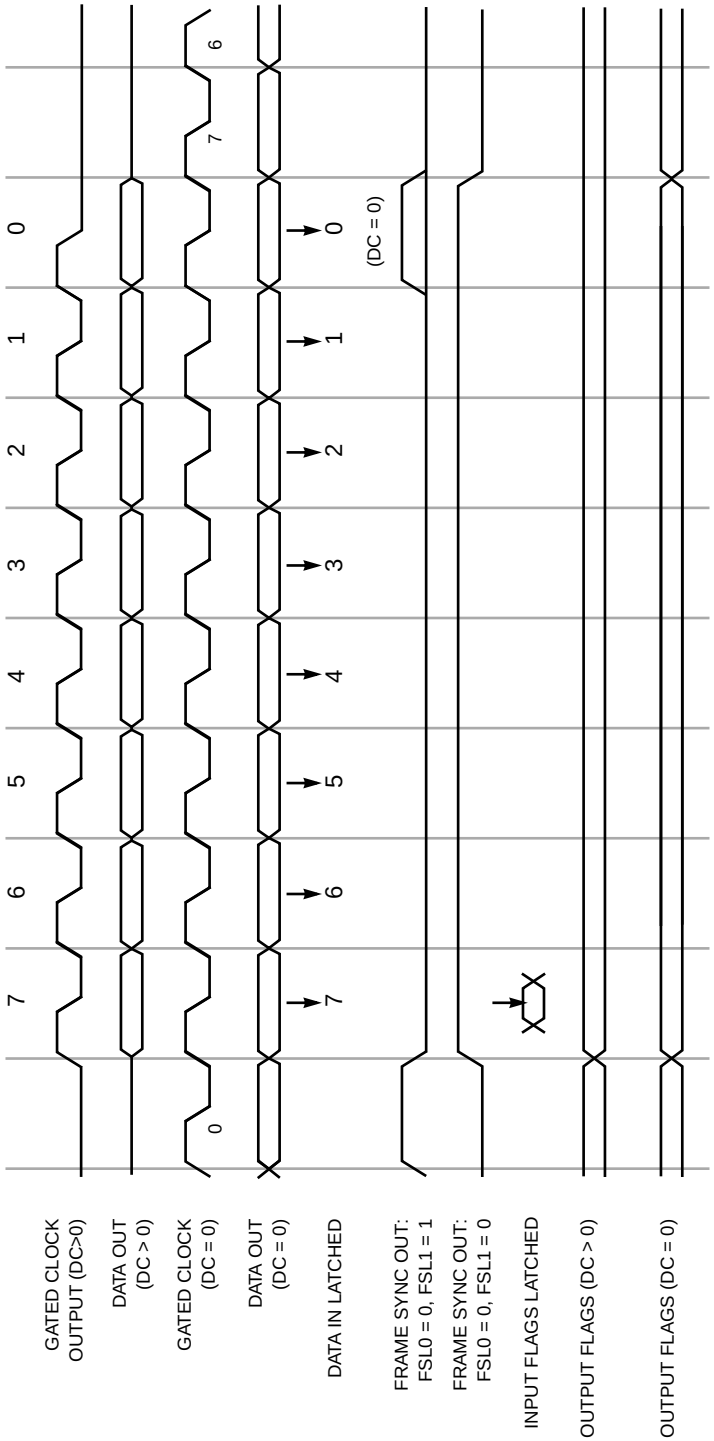
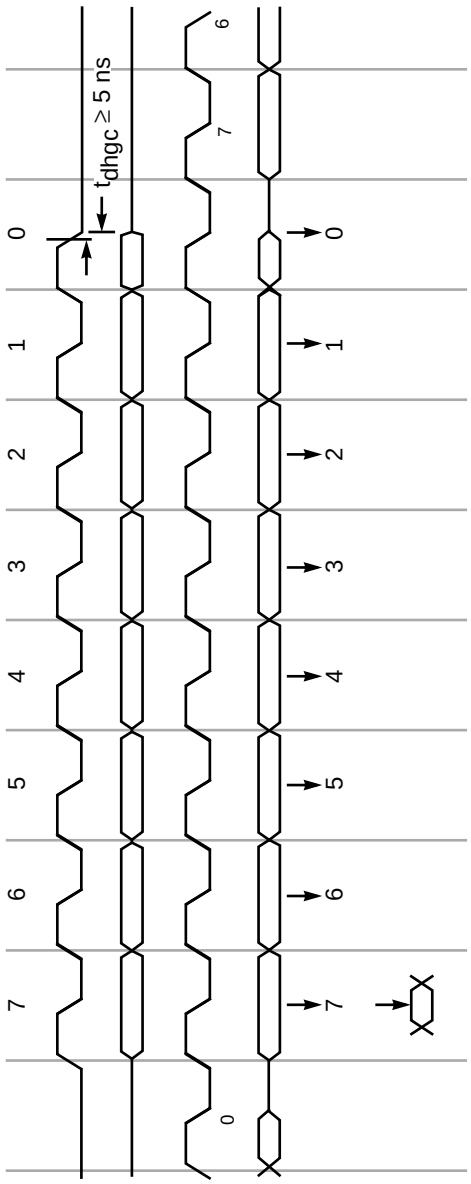


Figure 6-60 Internally Generated Clock Timing (8-Bit Example)



NOTES:
1. Output enabled on rising edge of first clock input.
2. Output disabled on falling edge of last clock pulse.
3. t_{dhgc} is guaranteed by circuit design.
4. Frame syncs (in or out) are not defined for external gated clock mode.

Figure 6-61 Externally Generated Gated Clock Timing (8-Bit

Data clock and frame sync signals can be generated internally by the DSP or may be ob-

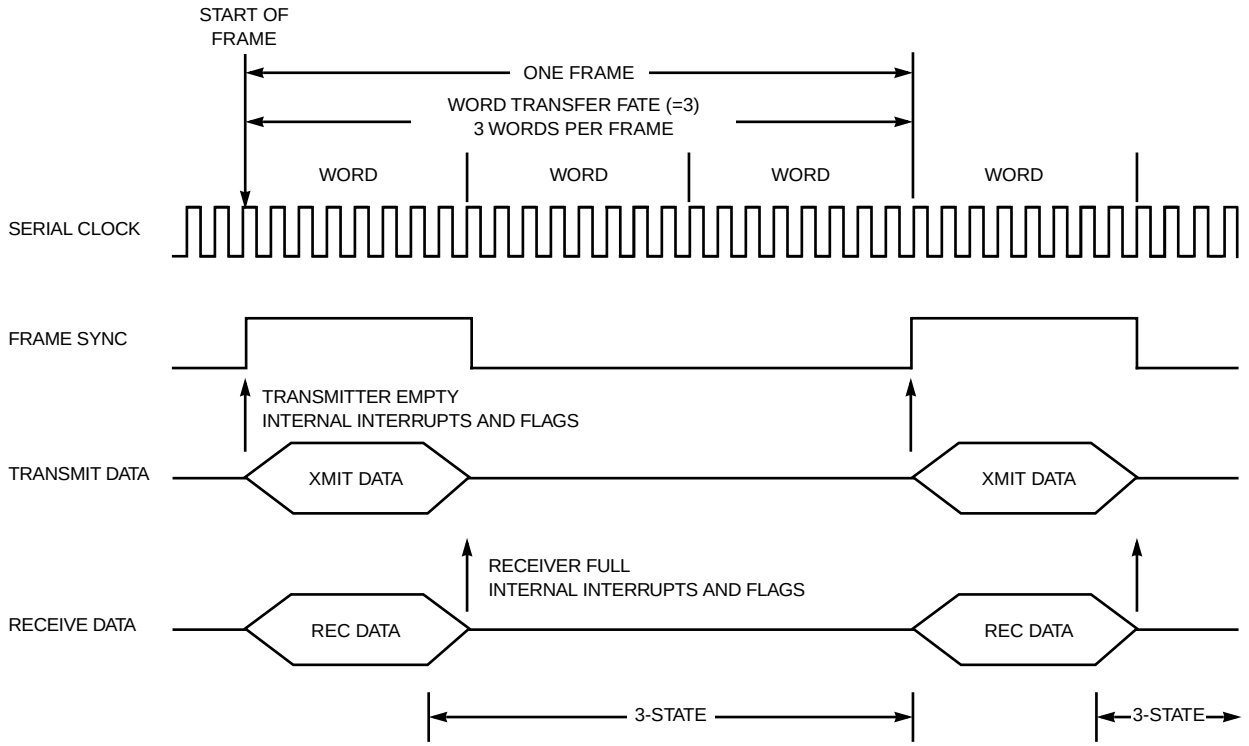


Figure 6-62 Synchronous Communication

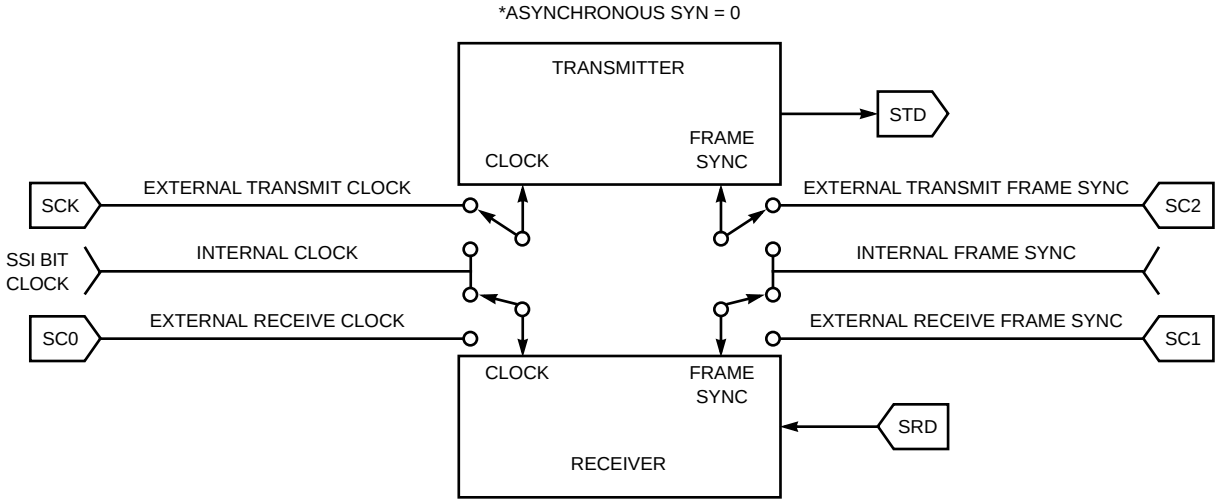
tained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The SSI clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame-rate divider and a word-length divider for frame-rate sync-signal generation.

Figures Figure 6-64 through Figure 6-67 show the definitions of the SSI pins during each of the four main operating modes of the SSI I/O interface. Figure 6-64 uses a gated clock (from either an external source or the internal clock), which means that frame sync is inherent in the clock. Since both the transmitter and receiver use the same clock (synchronous configuration), both use the SCK pin. SC0 and SC1 are designated as flags or can be used as general purpose-parallel I/O. SC2 is not defined if it is an input; SC2 is the transmit and receive frame sync if it is an output.

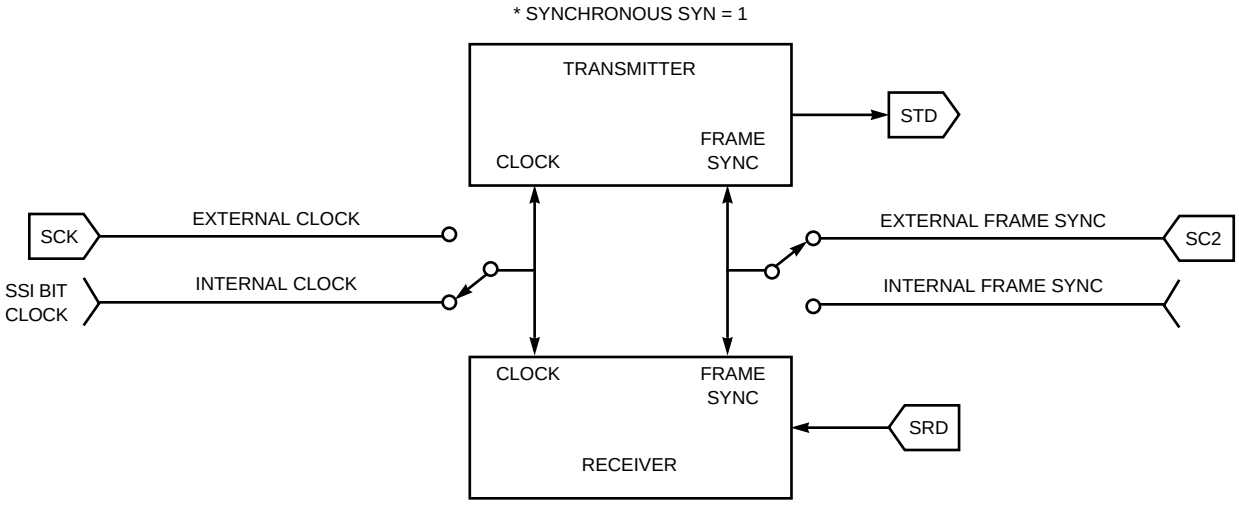
Figure 6-65 shows a gated clock (from either an external source or the internal clock), which means that frame sync is inherent in the clock. Since this configuration is asynchronous, SCK is the transmitter clock pin (input or output) and SC0 is the receiver clock pin (input or output). SC1 and SC2 are designated as receive or transmit frame sync, respectively, if they are se-

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FF	RIE	TIE	RE	TE	MOD	GCK	SYN *	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0



NOTE: Transmitter and receiver may have different clocks and frame syncs.



NOTE: Transmitter and receiver may have the same clock frame syncs.

Figure 6-63 CRB SYN Bit Operation

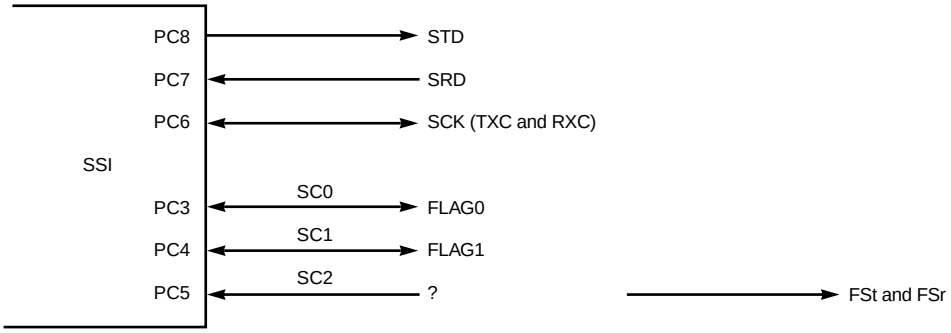


Figure 6-64 Gated Clock — Synchronous Operation

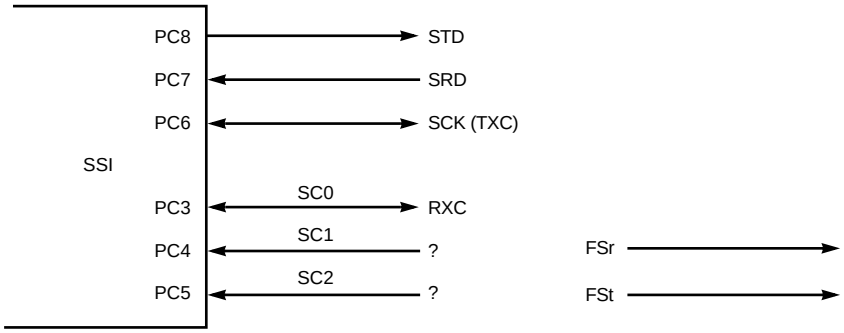


Figure 6-65 Gated Clock — Asynchronous Operation

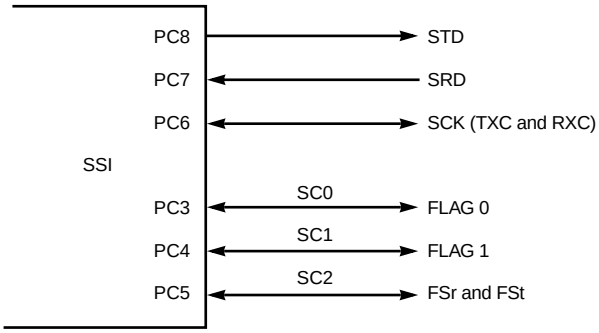


Figure 6-66 Continuous Clock — Synchronous Operation

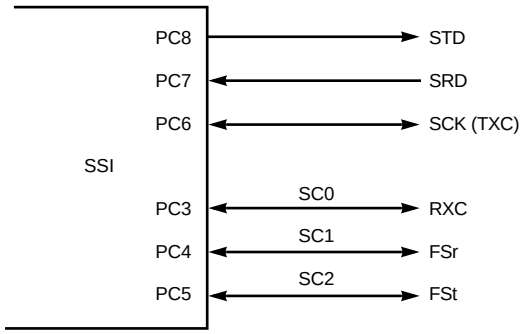


Figure 6-67 Continuous Clock — Asynchronous Operation

lected to be outputs; these bits are undefined if they are selected to be inputs. SC1 and SC2 can also be used as general-purpose parallel I/O.

Figure 6-66 shows a continuous clock (from either an external source or the internal clock), which means that frame sync must be a separate signal. SC2 is used for frame sync, which can come from an internal or external source. Since both the transmitter and receiver use the same clock (synchronous configuration), both use the SCK pin. SC0 and SC1 are designated as flags or can be used as general-purpose parallel I/O.

Figure 6-67 shows a continuous clock (from either an external source or the internal clock), which means that frame sync must be a separate signal. SC1 is used for the receive frame sync, and SC2 is used for the transmit frame sync. Either frame sync can come from an internal or external source. Since the transmitter and receiver use different clocks (asynchronous configuration), SCK is used for the transmit clock, and SC0 is used for the receive clock.

6.4.7.1.4 Frame Sync Selection

The transmitter and receiver can operate totally independent of each other. The transmitter can have either a bit-long or word-long frame-sync signal format, and the receiver can have the same or opposite format. The selection is made by programming FSL0 and FSL1 in the CRB as shown in Figure 6-68.

1. If FSL1 equals zero (see Figure 6-69), the RX frame sync is asserted during the entire data transfer period. This frame sync length is compatible with Motorola codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers, and telecommunication PCM serial I/O.
2. If FSL1 equals one (see Figure 6-70), the RX frame sync pulses active for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs, and telecommunication PCM serial I/O.

The ability to mix frame sync lengths is useful in configuring systems in which data is received from one type device (e.g., codec) and transmitted to a different type device.

FSL0 controls whether RX and TX have the same frame sync length (see Figure 6-68). If FSL0 equals zero, RX and TX have the same frame sync length, which is selected by FSL1. If FSL0 equals one, RX and TX have different frame sync lengths, which are selected by FSL1.

The SSI receiver looks for a receive frame sync leading edge only when the previous

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFED	RIE	TIE	RE	TE	MOD	GCK	SYN	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0
								*	*							

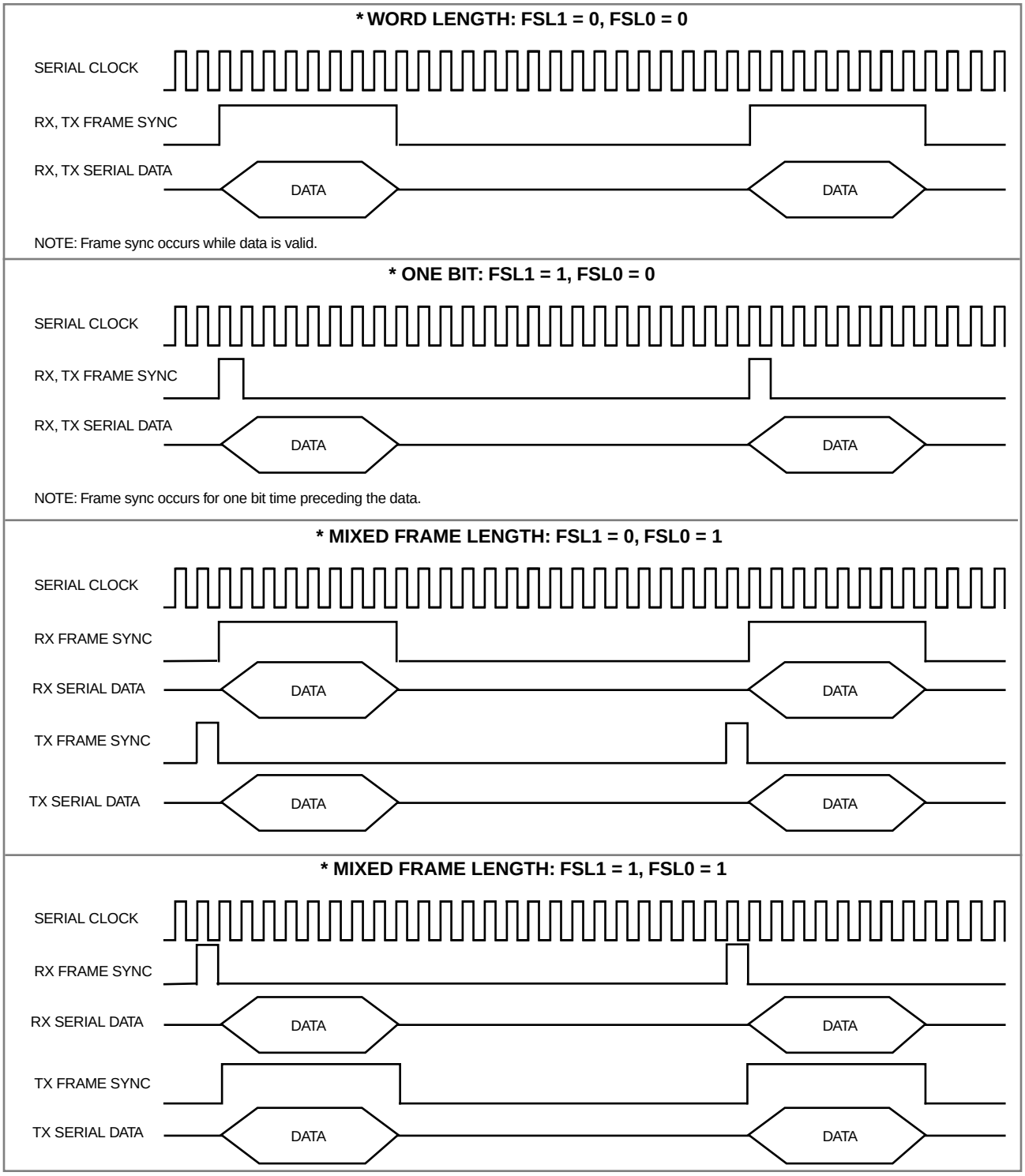


Figure 6-68 CRB FSL0 and FSL1 Bit Operation

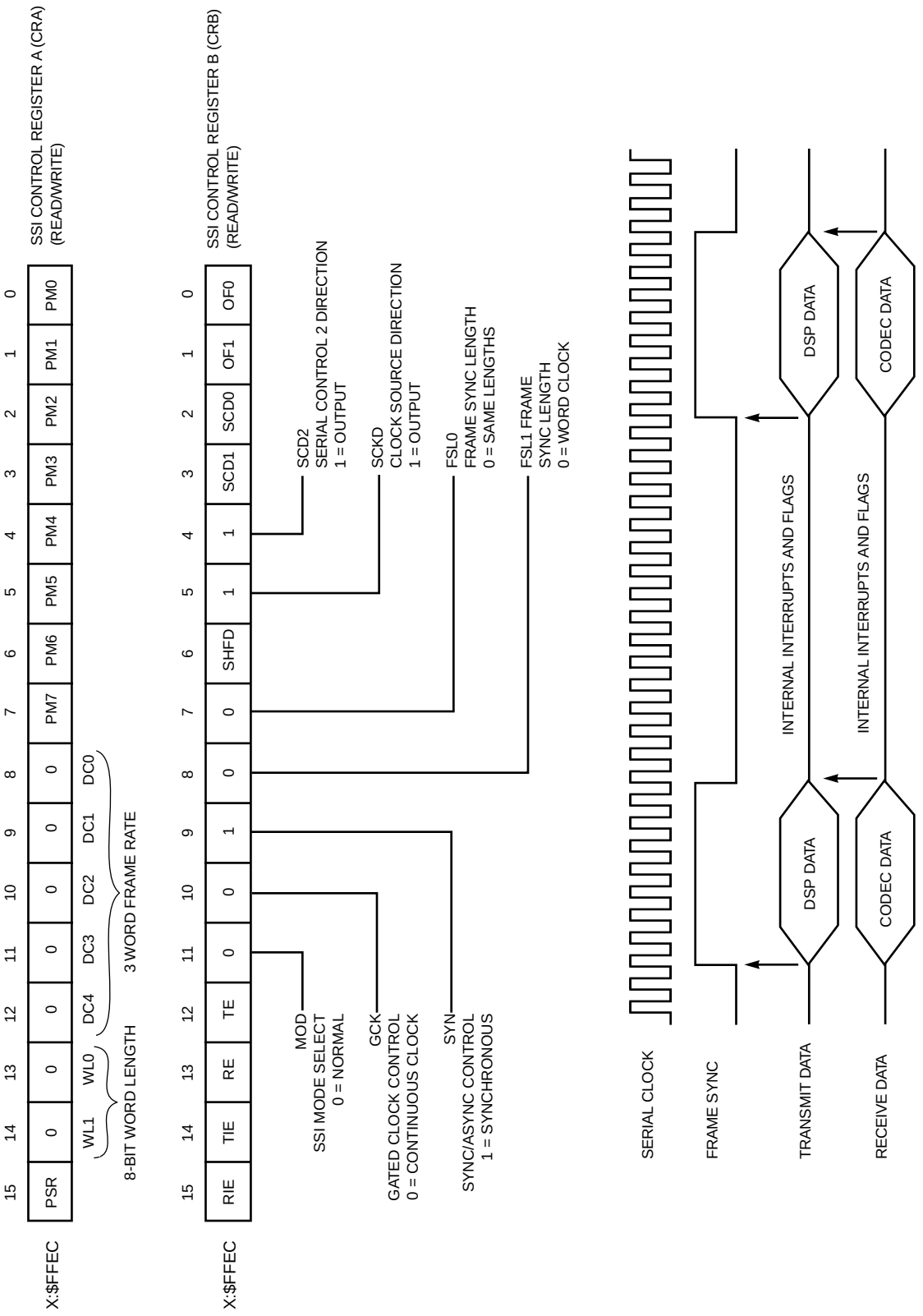


Figure 6-69 Normal Mode Initialization for FLS1=0 and FSL0=0

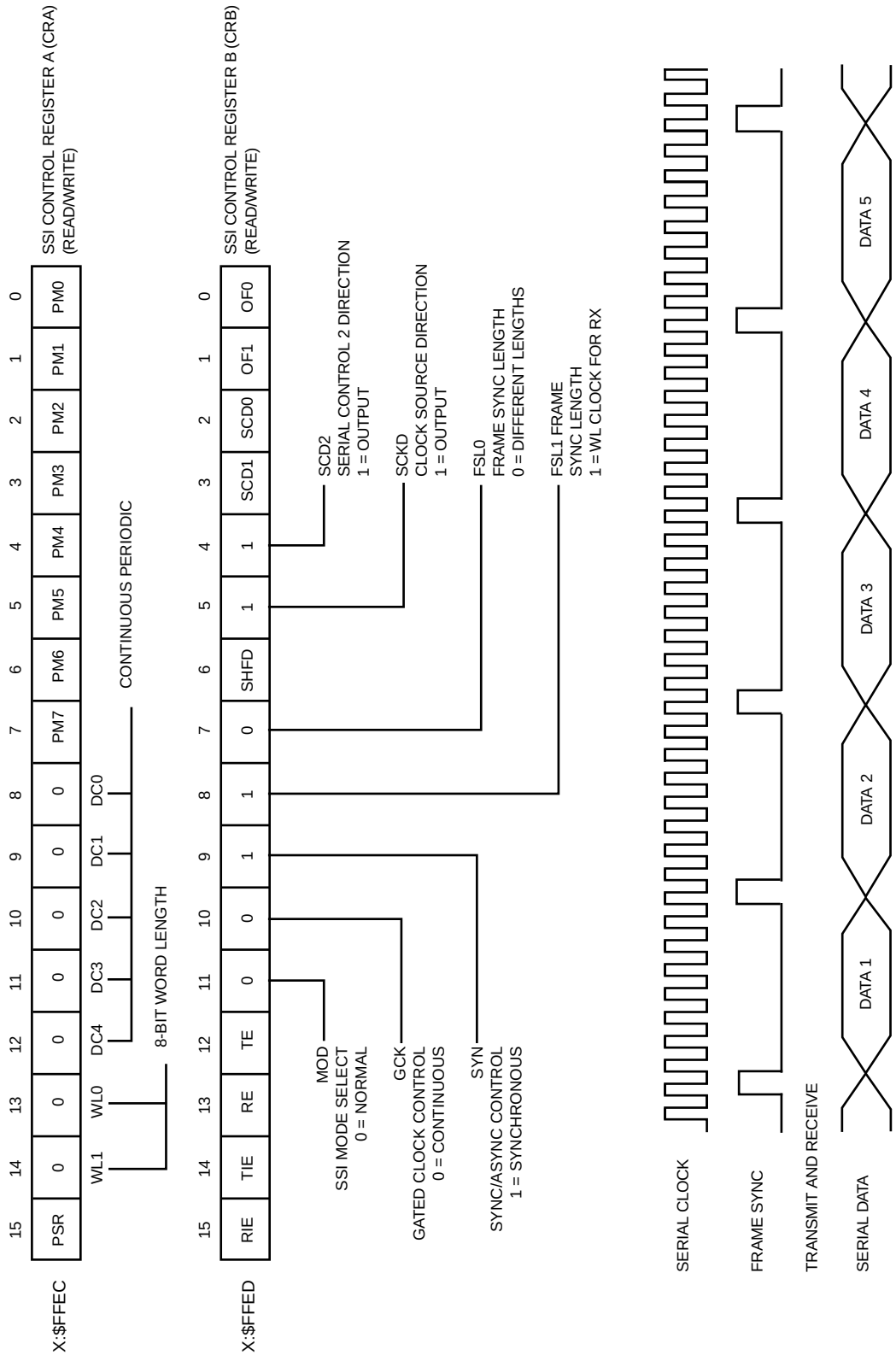


Figure 6-70 Normal Mode Initialization for FSL1=1 and FSL0=0

frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync), the current frame sync will not be recognized, and the receiver will be internally disabled until the next frame sync. Frames do not have to be adjacent – i.e., a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. The transmitter will be three-stated during these gaps.

6.4.7.1.5 Shift Direction Selection

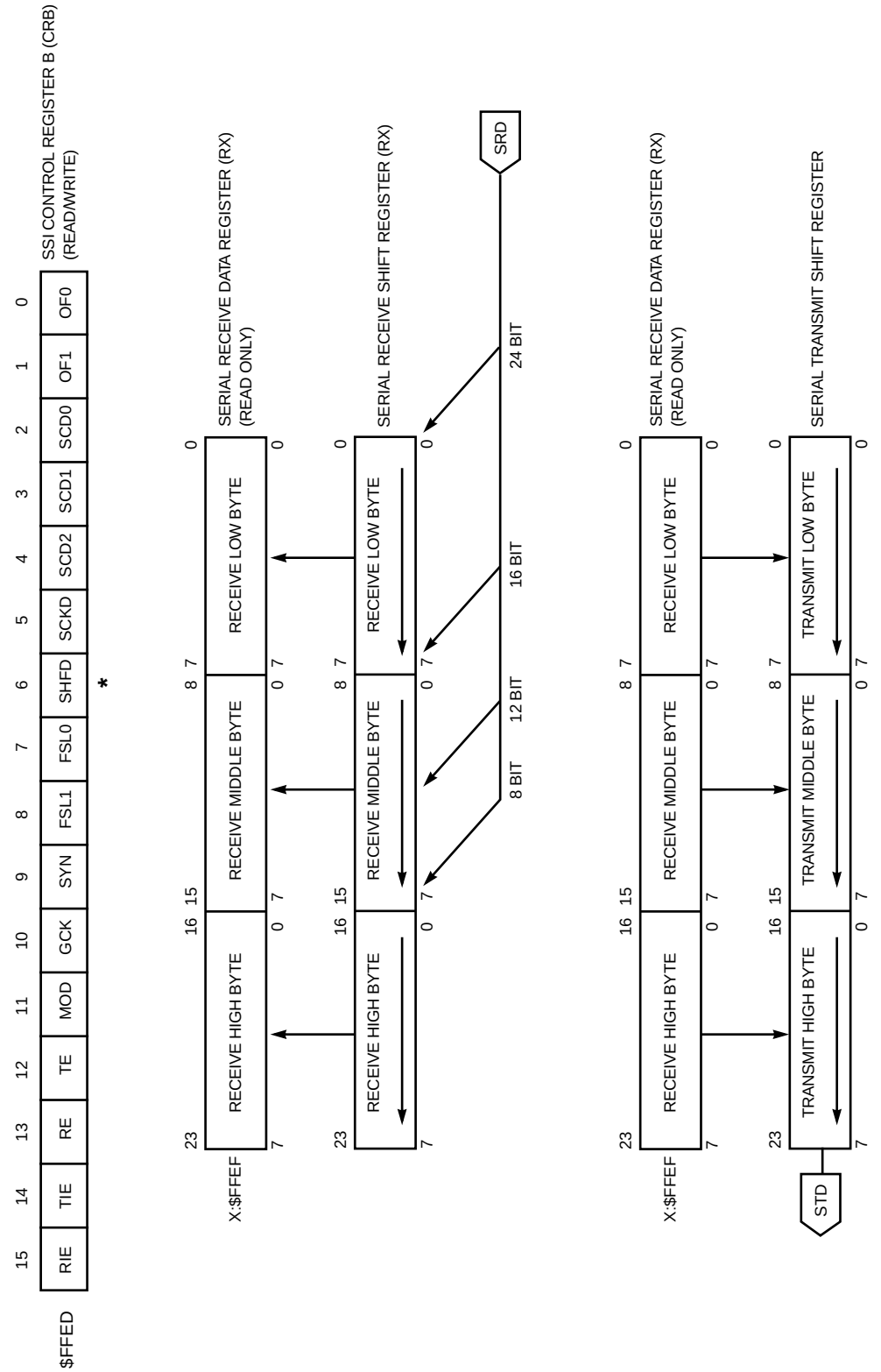
Some data formats, such as those used by codecs, specify MSB first other data formats, such as the AES-EBU digital audio, specify LSB first. To interface with devices from both systems, the shift registers in the SSI are bidirectional. The MSB/LSB selection is made by programming SHFD in the CRB.

Figure 6-71 illustrates the operation of the SHFD bit in the CRB. If SHFD equals zero (see Figure 6-71(a)), data is shifted into the receive shift register MSB first and shifted out of the transmit shift register MSB first. If SHFD equals one (see Figure 6-71(b)), data is shifted into the receive shift register LSB first and shifted out of the transmit shift register LSB first.

6.4.7.2 Normal Mode Examples

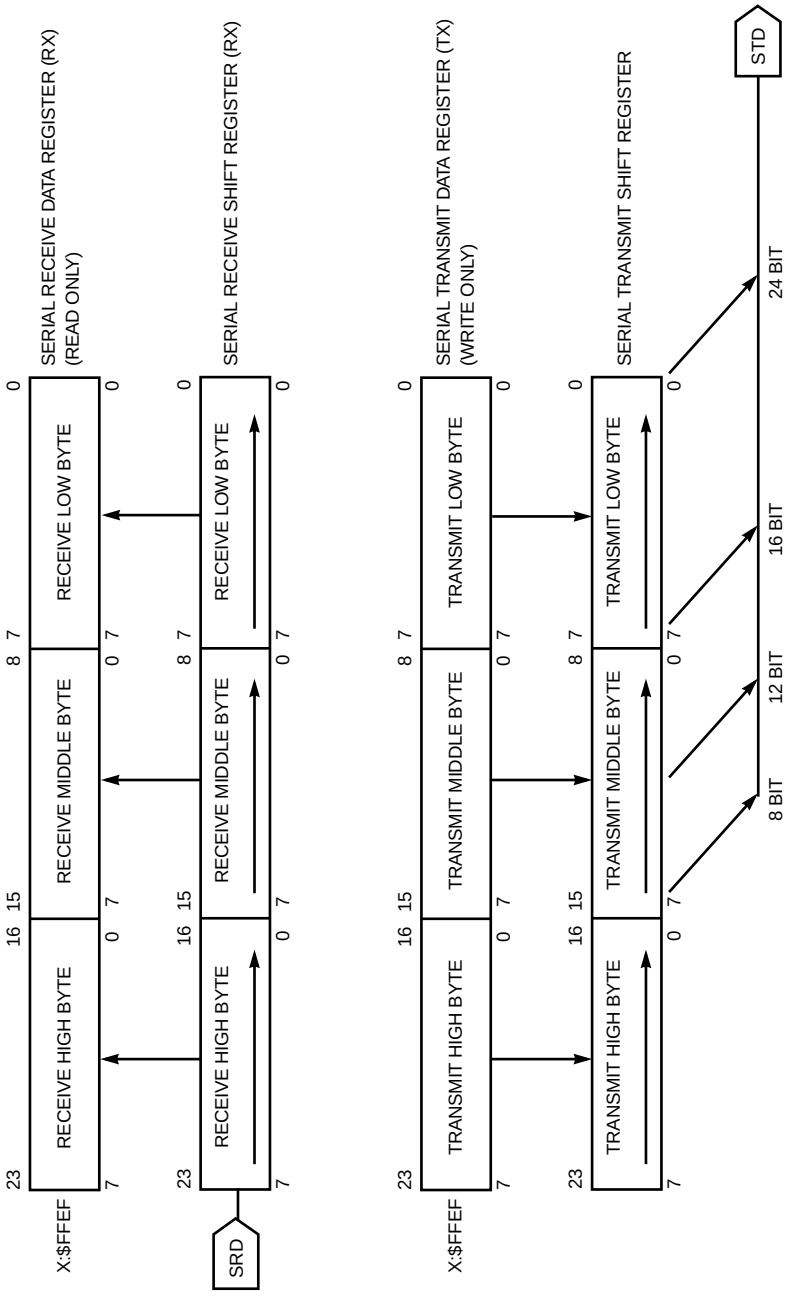
The normal SSI operating mode characteristically has one time slot per serial frame, and data is transferred every frame sync. When the SSI is not in the normal mode, it is in the network mode. The MSB is transmitted first (SHFD=0), with overrun and underrun errors detected by the SSI hardware. Transmit flags are set when data is transferred from the transmit data register to the transmit shift register. The receive flags are set when data is transferred from the receive shift register to the receive data register.

Figure 6-72 shows an example of using the SSI to connect an MC15500 codec with a DSP56002. No glue logic is needed. The serial clock, which is generated internally by the DSP, provides the transmit and receive clocks (synchronous operation) for the codec. SC2 provides all the necessary handshaking. Data transfer begins when the frame sync is asserted. Transmit data is clocked out and receive data is clocked in with the serial clock while the frame sync is asserted (word-length frame sync). At the end of the data transfer, DSP internal interrupts programmed to transfer data to/from will occur, and the SSISR will be updated.



(a) SHFD = 0

Figure 6-71 CRB SHFD Bit Operation (Sheet 1 of 2)



(b) SHFD=1

Figure 6-71 CRB SHFD Bit Operation (Sheet 2 of 2)

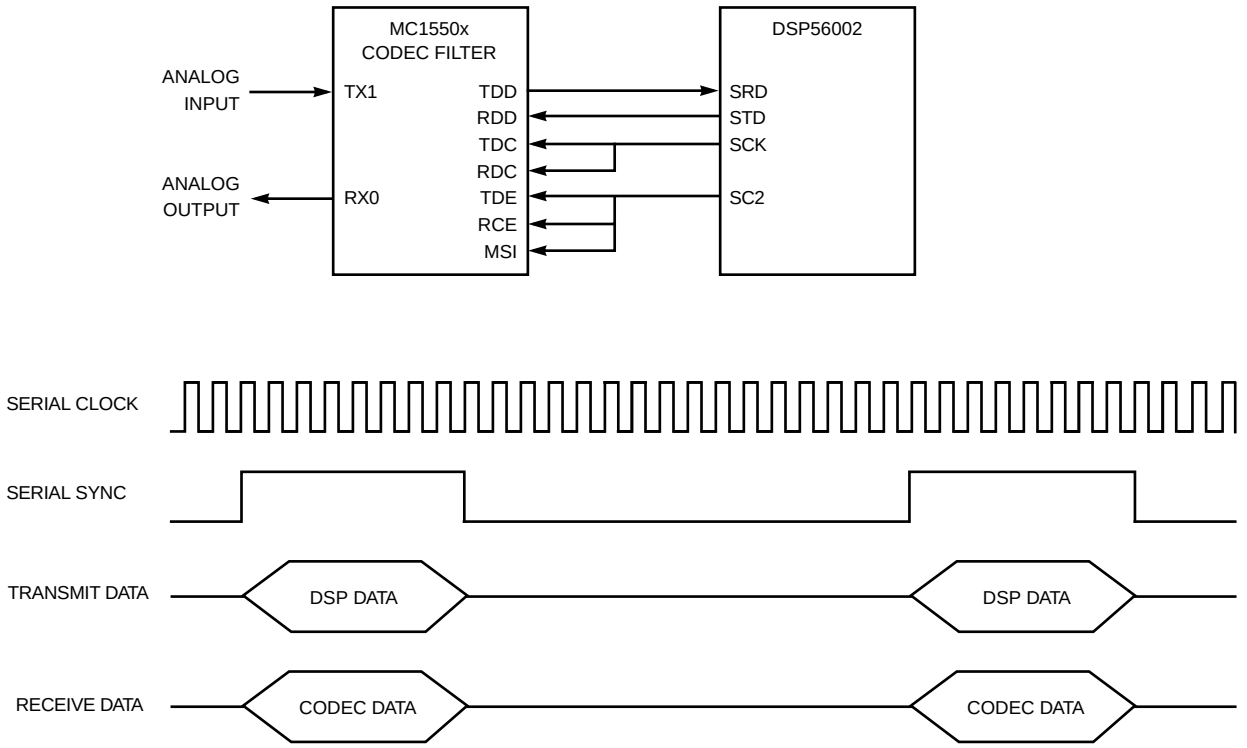


Figure 6-72 Normal Mode Example

6.4.7.2.1 Normal Mode Transmit

The conditions for data transmission from the SSI are as follows:

1. Transmitter is Enabled (TE=1)
2. Frame sync (or clock in gated clock mode) is active

When these conditions occur in normal mode, the next data word will be transferred from TX to the transmit shift register, the TDE flag will be set (transmitter empty), and the transmit interrupt will occur if TIE equals one (transmit interrupt enabled.) The new data word will be transmitted immediately.

The transmit data output (STD) is three-stated, except during the data transmission period. The optional frame sync output, flag outputs, and clock outputs are not three-stated even if both receiver and transmitter are disabled.

The optional output flags are always updated at the beginning of the frame, regardless of TE. The state of the flag does not change for the entire frame.

Figure 6-73 is an example of transmitting data using the SSI in the normal mode with a continuous clock, a bit-length frame sync, and 16-bit data words. The purpose of the program is to interleave and transmit right and left channels in a compact disk player. Four SSI pins are used:

1. SC0 is used as an output flag to indicate right-channel data (OF0=1) or left-channel data (OF0=0)
2. SC2 is TX and RX frame sync out
3. STD is transmit data out
4. SCK clocks the transmit data out

Equates are set for convenience and readability. Test data is then put in the low X: memory locations. The transmit interrupt vector contains a JSR instruction (which forms a long interrupt). The data pointer and channel flag are initialized before initializing CRA and CRB. It is assumed that the DSP CPU and SSI have been previously reset.

At this point, the SSI is ready to transmit except that the interrupt is masked because the MR was cleared on reset and Port C is still configured as general-purpose I/O. Unmasking the interrupt and enabling the SSI pins allows transmission to begin. A “jump to self” instruction causes the DSP to hang and wait for interrupts to transmit the data. When an interrupt occurs, a JSR instruction at the interrupt vector location causes the XMT routine to be executed. Data is then moved to the TX register, and the data pointer is incremented. The flag is tested by the JSET instruction and, if it is set, a jump to left occurs, and the code for the left channel is executed. If the flag is not set, the code for the right channel is executed. In either case, the channel flag in X0 and then the output flag are set to reflect the channel being transmitted. Control is then returned to the main program, which will wait for the next interrupt.

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************
IPR     EQU     $FFFF
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
TX      EQU     $FFEF
FLG     EQU     $0010
        ORG     X:0
        DC      $AAAA00             ;Data to transmit.
        DC      $333300
        DC      $CCCC00
        DC      $F0F000
;************************************************
;       INTERRUPT VECTOR*
;************************************************
        ORG     P:$0010
        JSR     XMT
;************************************************
;       MAIN PROGRAM*
;************************************************
        ORG     P:$40
        MOVE    #0,R0               ;Pointer to data buffer.
        MOVE    #3,M0               ;Set modulus to 4.
        MOVE    #0,X0               ;Initialize channel flag for SSI flag.
        MOVE    X0,X:FLG            ;Start with right channel first.
;************************************************
;       Initialize SSI Port*
;************************************************
        MOVEP   #$3000,X:IPR        ;Set interrupt priority register for SSI.
        MOVEP   #$401F,X:CRA        ;Set continuous clock=5.12/32 MHz
                                    ;word length=16.
        MOVEP   #$5334,X:CRB        ;Enable TIE and TE; make clock and
                                    ;frame sync outputs; frame
                                    ;sync=bit mode; synchronous mode;
                                    ;make SC0 an output.
```

Figure 6-73 Normal Mode Transmit Example (Sheet 1 of 2)

```
;************************************************
;       Init SSI Interrupt*
;************************************************
            ANDI        #$FC,MR             ;Unmask interrupts.
            MOVEP       #$01F8,X:PCC        ;Turn on SSI port.
            JMP         *                   ;Wait for interrupt.

;************************************************
;       MAIN INTERRUPT ROUTINE*
;************************************************
XMT         MOVEP       X:(R0);pl,X:TX      ;Move data to TX register.
            JSET        #0,X:FLG,LEFT       ;Check channel flag.
RIGHT       BCLR        #0,X:CRB            ;Clear SC0 indicating right channel data
            MOVE        #>$01,X0            ;Set channel flag to 1 for next data.
            MOVE        X0,X:FLG
            RTI
LEFT        BSET        #0,X:CRB            ;Set SC0 indicating left channel data.
            MOVE        #>$00,X0            ;Clear channel flag for next data.
            MOVE        X0,X:FLG
            RTI
            END
```

Figure 6-73 Normal Mode Transmit Example (Sheet 2 of 2)

6.4.7.2.2 Normal Mode Receive

If the receiver is enabled, a data word will be clocked in each time the frame sync signal is generated (internal) or detected (external). After receiving the data word, it will be transferred from the SSI receive shift register to the receive data register (RX), RDF will be set (receiver full), and the receive interrupt will occur if it is enabled (RIE=1).

The DSP program has to read the data from RX before a new data word is transferred from the receive shift register; otherwise, the receiver overrun error will be set (ROE=1).

Figure 6-74 illustrates the program that receives the data transmitted by the program shown in Figure 6-73. Using the flag to identify the channel, the receive program receives the right- and left-channel data and separates the data into a right data buffer and a left data buffer. The program shown in Figure 6-74 begins by setting equates and then using a JSR instruction at the receive interrupt vector location to form a long interrupt. The main program starts by initializing pointers to the right and left data buffers. The IPR, CRA, and CRB are then initialized. The clock divider bits in the CRA do not have to be set since an external receive clock is specified (SCKD=0). Pin SC0 is specified as an input flag (SYN=1, SCD0=0); pin SC2 is specified as TX and RX frame sync (SYN=1, SCD2=0). The SSI port is then enabled and interrupts are unmasked, which allows the

SSI port to begin data reception. A jump-to-self instruction is then used to hang the processor and allow interrupts to receive the data. Normally, the processor would execute useful instructions while waiting for the receive interrupts. When an interrupt occurs, the JSR instruction at the interrupt vector location transfers control to the RCV subroutine. The input flag is tested, and data is put in the left or right data buffer depending on the results of the test. The RTI instruction then returns control to the main program, which will wait for the next interrupt.

```
;*************************************************
;       SSI and other I/O EQUATES*
;*************************************************
IPR     EQU     $FFFF
SSISR   EQU     $FFEE
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
RX      EQU     $FFEF
FLG     EQU     $0010
;*************************************************
;       INTERRUPT VECTOR*
;*************************************************
        ORG     P:$000C
        JSR     RCV
;*************************************************
;       MAIN PROGRAM*
;*************************************************
        ORG     P:$40
        MOVE    #0,R0           ;Pointer to memory buffer for
        MOVE    #$08,R1         ;received data. Note data will be
        MOVE    #1,M0           ;split between two buffers which are
        MOVE    #1,M1           ;modulus 2.
```

Figure 6-74 Normal Mode Receive Example (Sheet 1 of 2)

```
;*************************************************
;       Initialize SSI Port*
;*************************************************
            MOVEP     #$3000,X:IPR      ;Set interrupt priority register for SSI.
            MOVEP     #$4000,X:CRA      ;Set word length = 16 bits.
            MOVEP     #$A300,X:CRB      ;Enable RIE and RE; synchronous
                                        ;mode with bit frame sync;
                                        ;clock and frame sync are
                                        ;external; SC0 is an output.
;*************************************************
;       Init SSI Interrupt*
;*************************************************
            ANDI      #$FC,MR           ;Unmask interrupts.
            MOVEP     #$01F8,X:PCC      ;Turn on SSI port.
            JMP       *                 ;Wait for interrupt.
;*************************************************
;       MAIN INTERRUPT ROUTINE*
;*************************************************
RCV         JSET      #0,X:SSISR, RIGHT ;Test SCO flag.
LEFT        MOVEP     X:RX,X:(RO)+      ;If SCO clear, receive data
            RTI                         ;into left buffer (R0).
RIGHT       MOVEP     X:RX,X:(R1)+      ;If SCO set, receive data
            RTI                         ;into right buffer (R1).
            END
```

Figure 6-74 Normal Mode Receive Example (Sheet 2 of 2)

6.4.7.3 Network Mode Examples

The network mode, the typical mode in which the DSP would interface to a TDM codec network or a network of DSPs, is compatible with Bell and CCITT PCM data/operation formats. The DSP may be a master device (see Figure 6-75) that controls its own private network or a slave device that is connected to an existing TDM network, occupying one or more time slots. The key characteristic of the network mode is that each time slot (data word time) is identified by an interrupt or by polling status bits, which allows the option of ignoring the time slot or transmitting data during the time slot. The receiver operates in the same manner except that data is always being shifted into the receive shift register and transferred to the RX. The DSP reads the receive data register and uses or discards the contents. Overrun and underrun errors are detected.

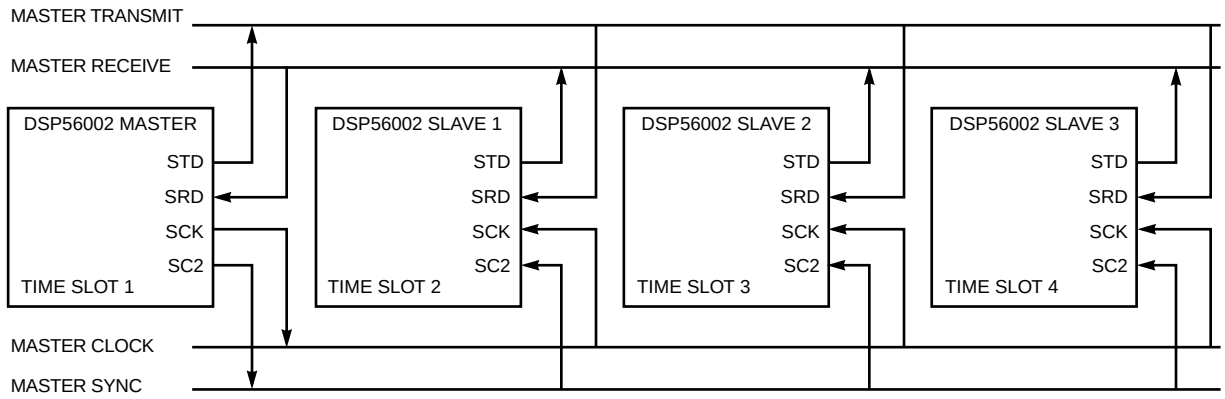


Figure 6-75 Network Mode Example

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots; transmission or reception can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate dividers (controlled by DC4, DC3, DC2, DC1, and DC0) control the number of time slots per frame from 2 to 32. Time-slot assignment is totally under software control. Devices can transmit on multiple time slots, receive multiple time slots, and the time-slot assignment can be changed dynamically.

A simplified flowchart showing operation of the network mode is shown in Figure 6-76. Two counters are used to track the current transmit and receive time slots. Slot counter one (SLOTCT1) is used to track the transmit time slot; slot counter two (SLOTCT2) is used for receive. When the transmitter is empty, it generates an interrupt; a test is then made to see if it is the beginning of a frame. If it is the beginning of a frame, SLOTCT1 is cleared to start counting the time slots. If it is not the beginning of a frame, SLOTCT1 is incremented. The next test checks to see if the SSI should transmit during this time slot. If it is time to transmit, data is written to the TX; otherwise, dummy data is written to the TSR, which prevents a transmit underrun error from occurring and three-states the STD pin. The DSP can then return to what it was doing before the interrupt and wait for the next interrupt to occur. SLOTCT1 should reflect the data in the shift registers to coincide with TFS. Software must recognize that the data being written to TX will be transmitted in time slot SLOTCT1 plus one.

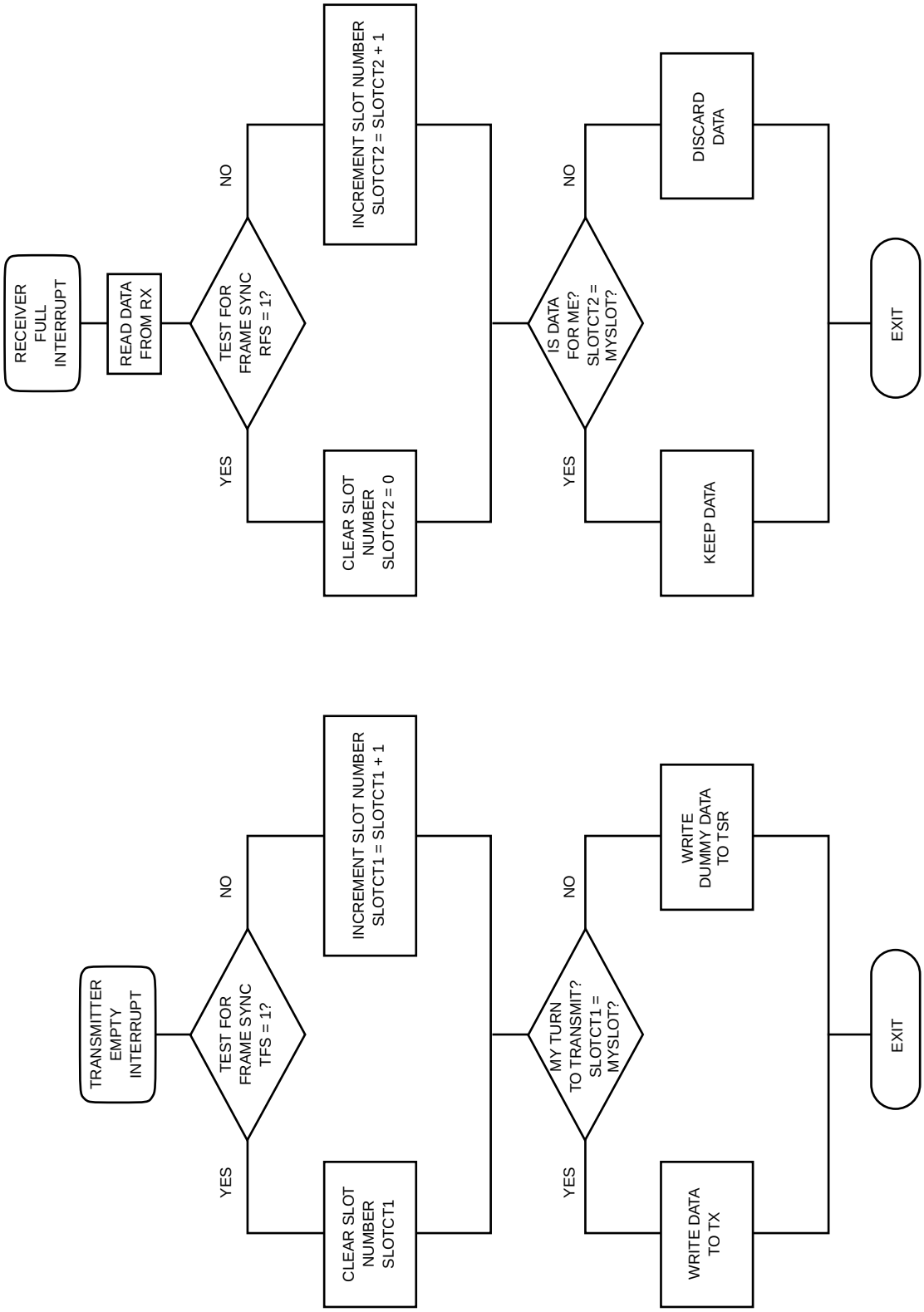


Figure 6-76 TDM Network Software Flowchart

The receiver operates in a similar manner. When the receiver is full, an interrupt is generated, and a test is made to see if this is the beginning of a frame. If it is the beginning of a frame, SLOTCT2 is cleared to start counting the time slots. If it is not the beginning of a frame, SLOTCT2 is incremented. The next test checks to see if the data received is intended for this DSP. If the current time slot is the one assigned to the DSP receiver, the data is kept; otherwise, the data is discarded, and the DSP can then return to what it was doing before the interrupt. SLOTCT2 should reflect the data in the receive shift register to coincide with the RFS flag. Software must recognize that the data being read from RX is for time slot SLOTCT2 minus two.

Initializing the network mode is accomplished by setting the bits in CRA and CRB as follows (see Figure 6-77):

1. The word length must be selected by setting WL1 and WL0. In this example, an 8-bit word length was chosen (WL1=0 and WL0=0).
2. The number of time slots is selected by setting DC4–DC0. Four time slots were chosen for this example (DC4–DC0=$03).
3. The serial clock rate must be selected by setting PSR and PM7–PM0 (see Table 6-15 (a), Table 6-15 (b), and Table 6-16).
4. RE and TE must be set to activate the transmitter and receiver. If interrupts are to be used, RIE and TIE should be set. RIE and TIE are usually set after everything else is configured and the DSP is ready to receive interrupts.
5. The network mode must be selected (MOD=1).
6. A continuous clock is selected in this example by setting GCK=0.
7. Although it is not required for the network mode, synchronous clock control was selected (SYN=1).
8. The frame sync length was chosen in this example as word length (FSL1=0) for both transmit and receive frame sync (FSL0=0). Any other combinations could have been selected, depending on the application.
9. Control bits SHFD, SCKD, SCD2, SCD1, SCD0, and the flag bits (OF1 and OF0) should be set as needed for the application.

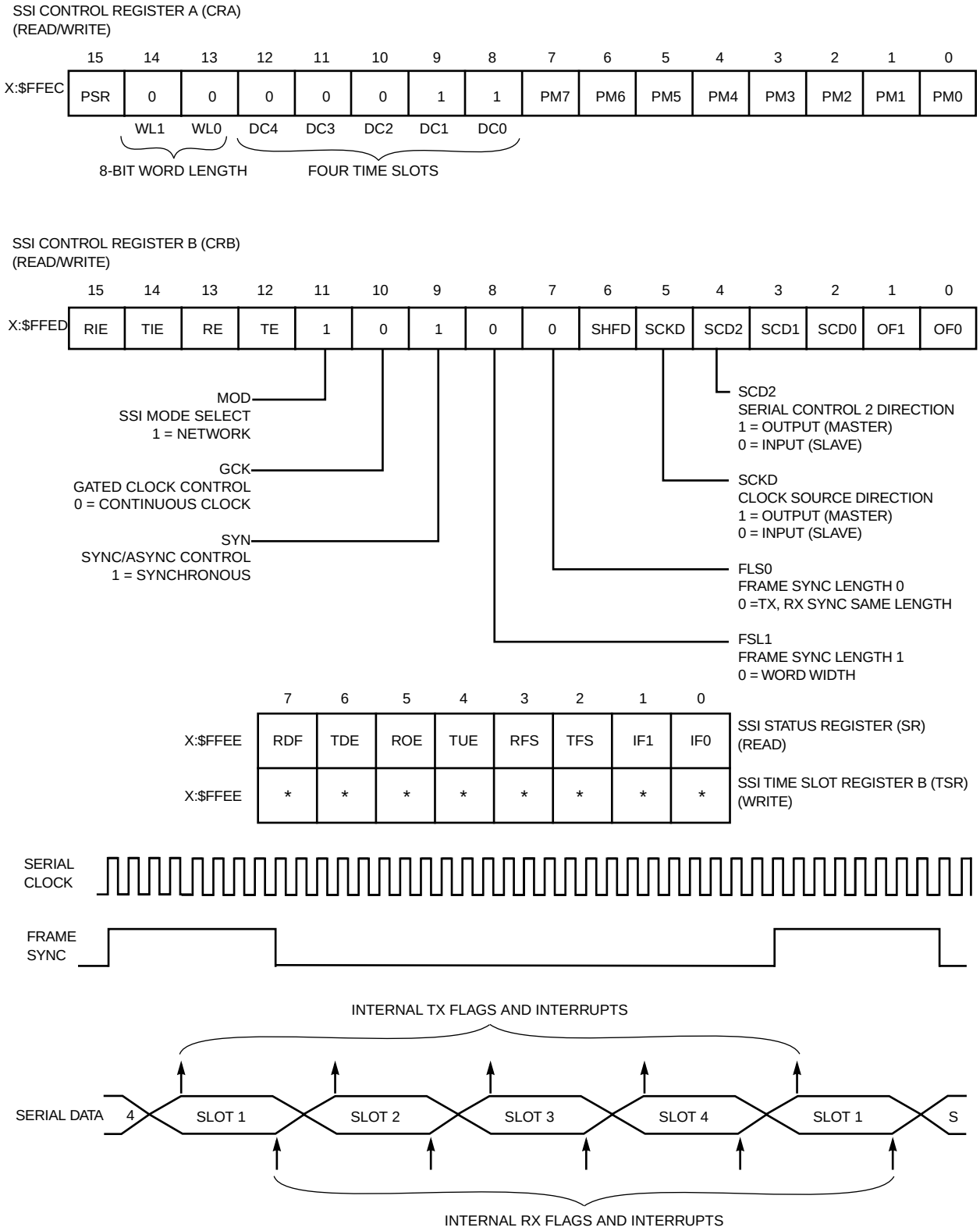


Figure 6-77 Network Mode Initialization

6.4.7.3.1 Network Mode Transmit

When TE is set, the transmitter will be enabled only after detection of a new data frame sync. This procedure allows the SSI to synchronize to the network timing.

Normal startup sequence for transmission in the first time slot is to write the data to be transmitted to TX, which clears the TDE flag. Then set TE and TIE to enable the transmitter on the next frame sync and to enable transmit interrupts.

Alternatively, the DSP programmer may decide not to transmit in the first time slot by writing any data to the time slot register (TSR). This will clear the TDE flag just as if data were going to be transmitted, but the STD pin will remain in the high-impedance state for the first time slot. The programmer then sets TE and TIE.

When the frame sync is detected (or generated), the first data word will be transferred from TX to the transmit shift register and will be shifted out (transmitted). TX being empty will cause TDE to be set, which will cause a transmitter interrupt. Software can poll TDE or use interrupts to reload the TX register with new data for the next time slot. Software can also write to TSR to prevent transmitting in the next time slot. Failing to reload TX (or writing to the TSR) before the transmit shift register is finished shifting (empty) will cause a transmitter underrun. The TUE error bit will be set, causing the previous data to be retransmitted.

The operation of clearing TE and setting it again will disable the transmitter after completion of transmission of the current data word until the beginning of the next frame sync period. During that time, the STD pin will be three-stated. When it is time to disable the transmitter, TE should be cleared after TDE is set to ensure that all pending data is transmitted.

The optional output flags are updated every time slot regardless of TE.

To summarize, the network mode transmitter generates interrupts every time slot and requires the DSP program to respond to each time slot. These responses can be:

1. Write data register with data to enable transmission in the next time slot
2. Write the time slot register to disable transmission in the next time slot
3. Do nothing – transmit underrun will occur the at beginning of the next time slot, and the previous data will be transmitted

Figure 6-78 differs from the program shown in Figure 6-73 only in that it uses the network mode to transmit only right-channel data. A time slot is assigned for the left-channel data, which could be inserted by another DSP using the network mode. In the “Initialize SSI Port” section of the program, two words per frame are selected using CRA, and the network mode is selected by setting MOD to one in the CRB. The main interrupt routine, which waits to move the data to TX, only transmits data if the current time slot is for the right channel. If the current time slot is for the left channel, the TSR is written, which three-states the output to allow another DSP to transmit the left channel during the time slot.

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************

IPR     EQU     $FFFF
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
TX      EQU     $FFEF
TSR     EQU     $FFEE
FLG     EQU     $0010
        ORG     X:0
        DC      $AAAA00             ;Data to transmit.
        DC      $333300
        DC      $CCCC00
        DC      $F0F000

;************************************************
;       INTERRUPT VECTOR*
;************************************************

        ORG     P:$0010
        JSR     XMT

;************************************************
;       MAIN PROGRAM*
;************************************************
        ORG     P:$40
        MOVE    #0,R0               ;Pointer to data buffer.
        MOVE    #3,M0               ;Set modulus to 4.
        MOVE    #0,X0               ;Initialize user flag for SSI flag.
        MOVE    X0,X:FLG            ;Start with the right channel.
```

Figure 6-78 Network Mode Transmit Example Program (Sheet 1 of 2)

```
;************************************************
;       Initialize SSI Port*
;************************************************
            MOVEP   #$3000,X:IPR        ;Set interrupt priority register for SSI.
            MOVEP   #$411F,X:CRA        ;Set continuous clock=5.12/32 MHz
                                        ;word length=16.
            MOVEP   #$5B34,X:CRB        ;Enable TIE and TE; make clock and
                                        ;frame sync outputs; frame
                                        ;sync=bit mode; synchronous mode;
                                        ;make SC0 an output.
;************************************************
;       Init SSI Interrupt*
;************************************************
            ANDI    #$FC,MR             ;Unmask interrupts.
            MOVEP   #$01F8,X:PCC        ;Turn on SSI port.
            JMP     *                   ;Wait for interrupt.
;************************************************
;       MAIN INTERRUPT ROUTINE*
;************************************************
XMT
            JSET    #0,X:FLG,LEFT       ;Check user flag.
RIGHT       BCLR    #0,X:CRB            ;Clear SC0 indicating right channel data
            MOVEP   X:(R0)+,X:TX        Move data to TX register.
            MOVE    #>$01,X0            ;Set user flag to 1
            MOVE    X0,X:FLG            ;for next data.
            RTI
LEFT        BSET    #0,X:CRB            ;Set SC0 indicating left channel data.
            MOVEP   X0,X:TSR            ;Write to TSR register.
            MOVE    #>$00,X0            ;Clear user flag
            MOVE    X0,X:FLG            ;for next data.
            RTI
            END
```

Figure 6-78 Network Mode Transmit Example Program (Sheet 2 of 2)

```
;*************************************************
;       SSI and other I/O EQUATES*
;*************************************************
IPR     EQU     $FFFF
SSISR   EQU     $FFEE
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
RX      EQU     $FFEF

;*************************************************
;       INTERRUPT VECTOR*
;*************************************************

        ORG     P:$000C
        JSR     RCV

;*************************************************
;       MAIN PROGRAM*
;*************************************************

        ORG     P:$40
        MOVE    #0,R0           ;Pointer to memory buffer for
        MOVE    #$08,R1         ;received data. Note data will be
        MOVE    #3,M0           ;split between two buffers which are
        MOVE    #3,M1           ;modulus 4.

;*************************************************
;       Initialize SSI Port*
;*************************************************

        MOVEP   #$3000,X:IPR    ;Set interrupt priority register for SSI.
        MOVEP   #$4100,X:CRA    ;Set word length = 16 bits.
        MOVEP   #$AB00,X:CRB    ;Enable RIE and RE; synchronous
                                ;mode with bit frame sync;
                                ;clock and frame sync are
                                ;external; SC0 is an input.
```

Figure 6-79 Network Mode Receive Example Program (Sheet 1 of 2)

```
;*************************************************
;       Init SSI Interrupt*
;*************************************************

            ANDI    #$FC,MR             ;Unmask interrupts.
            MOVEP   #$01F8,X:PCC        ;Turn on SSI port.
            JMP     *                   ;Wait for interrupt.

;*************************************************

;       MAIN INTERRUPT ROUTINE*

;*************************************************

RCV     JSET    #0,X:SSISR, RIGHT   ;Test SCO flag.
LEFT    MOVEP   X:RX,X:(RO)+        ;If SCO clear, receive data
        RTI                         ;into left buffer (R0).
RIGHT   MOVEP   X:RX,X:(R1)+        ;If SCO set, receive data
        RTI                         ;into right buffer (R1).
        END
```

Figure 6-79 Network Mode Receive Example Program (Sheet 2 of 2)

6.4.7.3.2 Network Mode Receive

The receive enable will occur only after detection of a new data frame with RE set. The first data word is shifted into the receive shift register and is transferred to the RX, which sets RDF if a frame sync was received (i.e., this is the start of a new frame). Setting RDF will cause a receive interrupt to occur if the receiver interrupt is enabled (RIE=1).

The second data word (second time slot in the frame) begins shifting in immediately after the transfer of the first data word to the RX. The DSP program has to read the data from RX (which clears RDF) before the second data word is completely received (ready to transfer to RX), or a receive overrun error will occur (ROE=1), and the data in the receiver shift register will not be transferred and will be lost.

If RE is cleared and set again by the DSP program, the receiver will be disabled after receiving the current time slot in progress until the next frame sync (first time slot). This mechanism allows the DSP programmer to ignore data in the last portion of a data frame.

Note: The optional frame sync output and clock output signals are not affected, even if the transmitter and/or receiver are disabled. TE and RE do not disable bit clock and frame sync generation.

To summarize, the network mode receiver receives every time slot data word unless the receiver is disabled. An interrupt can occur after the reception of each data word, or the programmer can poll RDF. The DSP program response can be

1. Read RX and use the data
2. Read RX and ignore the data
3. Do nothing – the receiver overrun exception will occur at the end of the current time slot
4. Toggle RE to disable the receiver until the next frame, and read RX to clear RDF

Figure 6-79 is essentially the same program shown in Figure 6-74 except that this program uses the network mode to receive only right-channel data. In the “Initialize SSI Port” section of the program, two words per frame are selected using the DC bits in the CRA, and the network mode is selected by setting MOD to one in the CRB. If the program in Figure 6-78 is used to transmit to the program in Figure 6-79, the correct data will appear in the data buffer for the right channel, but the buffer for the left channel will probably contain $000000 or $FFFFFF, depending on whether the transmitter output was high or low when TSR was written and whether the output was three-stated.

6.4.7.4 On-Demand Mode Examples

A divide ratio of one (DC=00000) in the network mode is defined as the on-demand mode of the SSI because it is the only data-driven mode of the SSI – i.e., data is transferred whenever data is present (see Figure 6-80 and Figure 6-81). STD and SCK from DSP1 are connected to DSP2 – SRD and SC0, respectively. SC0 is used as an input clock pin in this application. Receive data and receive data clock are separate from the transmit signals. On-demand data transfers are nonperiodic, and no time slots are defined. When there is a clock in the gated clock mode, data is transferred. Although they are not necessarily needed, frame sync and flags are generated when data is transferred. Transmitter underruns (TUE) are impossible in this mode and are therefore disabled. In the on-demand transmit mode, two additional SSI clock cycles are automatically inserted between each data word transmitted. This procedure guarantees that frame sync will be low between every transmitted data word or that the clock will not be continuous between two consecutive words in the gated clock mode. The on-demand mode is similar to the SCI shift register mode with SSFTD equals one and SCKP equals one. The receiver should be configured to receive the bit clock and, if continuous clock is used, to receive an external frame sync. Therefore, for all full-duplex communication in on-demand mode, the asynchronous mode should be used. The on-demand mode is SPI compatible.

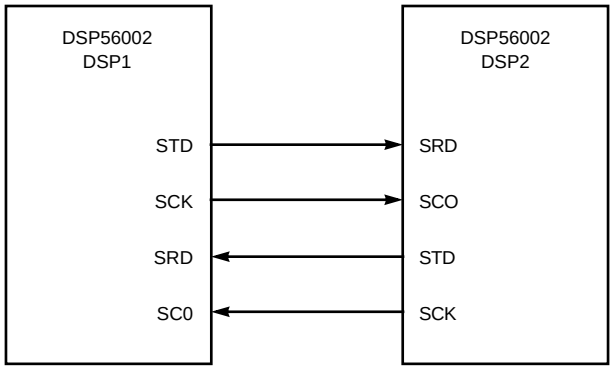


Figure 6-80 On Demand Example

Initializing the on-demand mode for the example illustrated in Figure 6-81 is accomplished by setting the bits in CRA and CRB as follows:

1. The word length must be selected by setting WL1 and WL0. In this example, a 24-bit word length was chosen (WL1=1 and WL0=1).
2. The on-demand mode is selected by clearing DC4–DC0.
3. The serial clock rate must be selected by setting PSR and PM7–PM0 (see Table 6-15 (a), Table 6-15 (b), and Table 6-16).
4. RE and TE must be set to activate the transmitter and receiver. If interrupts are to be used, RIE and TIE should be set. RIE and TIE are usually set after everything else is configured and the DSP is ready to receive interrupts.
5. The network mode must be selected (MOD=1).
6. A gated clock (GCK=1) is selected in this example. A continuous clock example is shown in Figure 6-78.
7. Asynchronous clock control was selected (SYN=0) in this example.
8. Since gated clock is used, the frame sync is not necessary. FSL1 and FSL0 can be ignored.
9. SCKD must be an output (SCKD=1).
10. SCD0 must be an input (SCD0=0).
11. Control bit SHFD should be set as needed for the application. Pins SC1 and SC2 are undefined in this mode (see Table 6-13) and should be programmed as general-purpose I/O pins.

SSI CONTROL REGISTER A (CRA)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFEC	PSR	1	1	0	0	0	0	0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
		WL1	WL0	DC4	DC3	DC2	DC1	DC0								

24-BIT WORD LENGTH

ON-DEMAND

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFED	RIE	TIE	RE	TE	1	1	0	FSL1	FSL0	SHFD	1	SCD2	SCD1	0	OF1	OF0

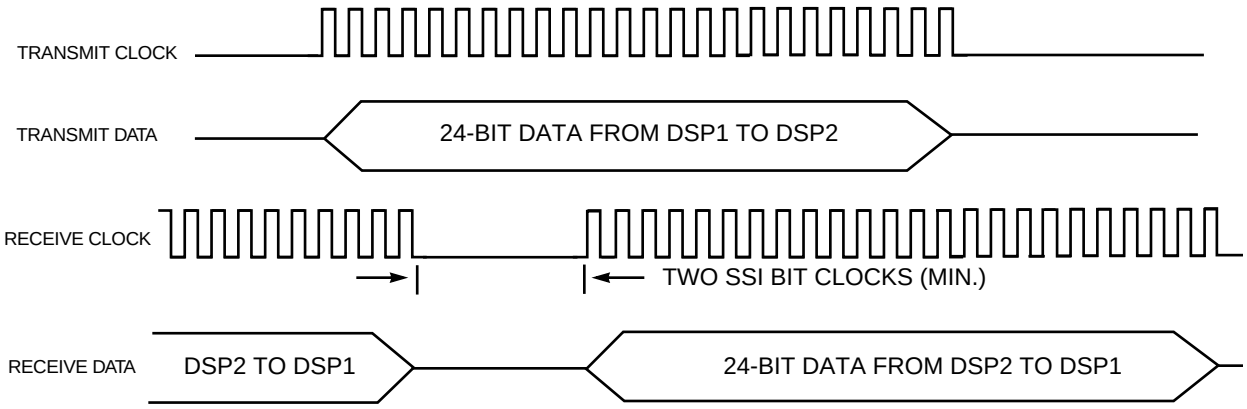
MOD
SSI MODE SELECT
1 = NETWORK

GCK
GATED CLOCK CONTROL
1=GATED CL0CK

SYN
SYNC/ASYNC CONTROL
0 = ASYNCHRONOUS

SCD0
SERIAL CONTROL 2
DIRECTION
0 = INPUT

SCKD
CLOCK SOURCE
DIRECTION
1 = OUTPUT



NOTE: Two SSI bit clock times are automatically inserted between each data word. This guarantees frame sync will be low between every data word transmitted and the clock will not be continuous for two consecutive data words.

Figure 6-81 On-Demand Data-Driven Network Mode

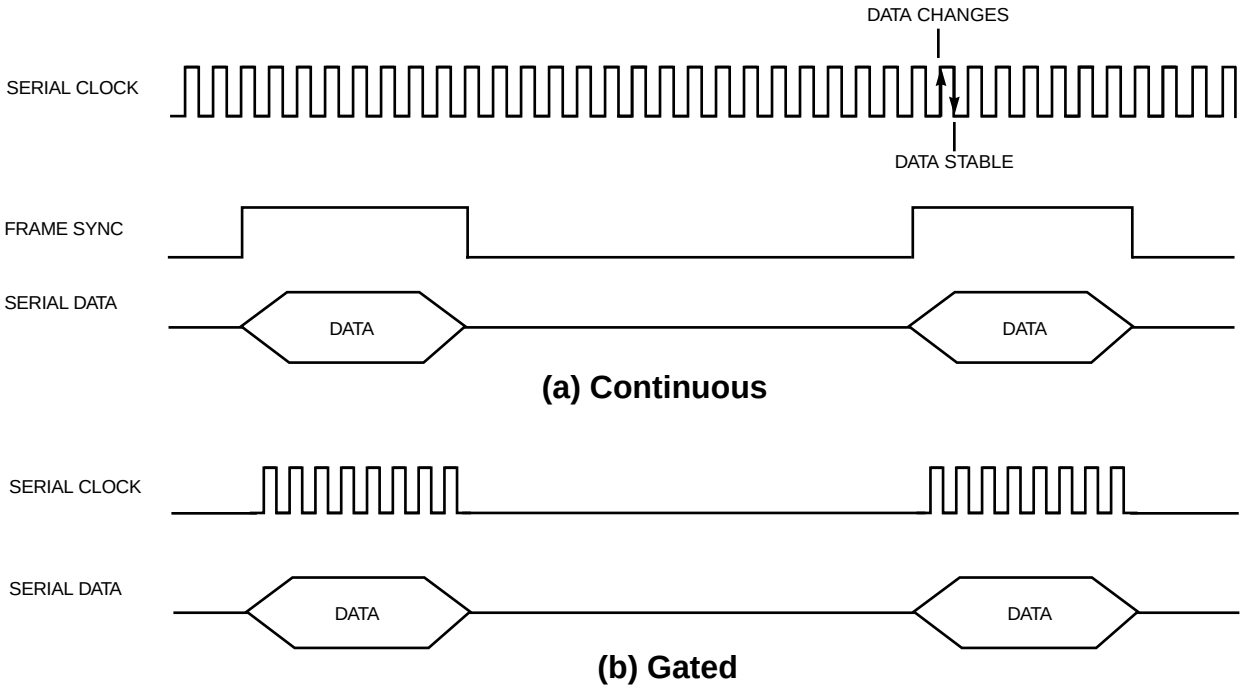


Figure 6-82 Clock Modes

6.4.7.4.1 On-Demand Mode – Continuous Clock

This special case will not generate a periodic frame sync. A frame sync pulse will be generated only when data is available to transmit (see Figure 6-82(a)). The frame sync signal indicates the first time slot in the frame. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore, for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into TX. Although the SSI is double buffered, only one word can be written to TX, even if the transmit shift register is empty. The receive and transmit interrupts function as usual using TDE and RDF; however, transmit and receive underruns are impossible for on-demand transmission and are disabled. This mode is useful for interfacing to codecs requiring a continuous clock.

6.4.7.4.2 On-Demand Mode – Gated Clock

Gated clock mode (see Figure 6-82(b)) is defined for on-demand mode, but the gated clock mode is considered a frame sync source; therefore, in gated clock mode, the transmit clock must be internal (output) and the receive clock must be external (input). For on-demand mode, with internal (output) synchronous gated clock, output clock is enabled for

the transmitter and receiver when TX data is transferred to the transmit data shift register. This SPI master operating mode is shown in Figure 6-83. Word sync is inherent in the clock signal, and the operation format must provide frame synchronization.

Figure 6-84 is the block diagram for the program presented in Figure 6-85. This program contains a transmit test program that was written as a scoping loop (providing a repetitive sync) using the on-demand, gated, synchronous mode with no interrupts (polling) to transmit data to the program shown in Figure 6-86. The program also demonstrates using GPIO pins as general-purpose control lines. PC3 is used as an external strobe or enable for hardware such as an A/D converter.

The transmit program sets equates for convenience and readability. Test data is then written to X: memory, and the data pointer is initialized. Setting M0 to two makes the buffer circular (modulo 3), which saves the step of resetting the pointer each loop. PC3 is configured as a general-purpose output for use as a scope sync, and CRA and CRB are then initialized. Setting the PCC bits begins SSI operation; however, no data will be transmitted until data is written to TX. PC3 is set high at the beginning of data transmission; data is then moved to TX to begin transmission. A JCLR instruction is then used to form a wait loop until TDE equals one and the SSI is ready for another data word to be transmitted. Two more data words are transmitted in this fashion (this is an arbitrary number chosen for this test loop). An additional wait is included to make sure that the frame sync has gone low before PC3 is cleared, indicating on the scope that transmission is complete. A wait of 100 NOPs is implemented by using the REP instruction before starting the loop again.

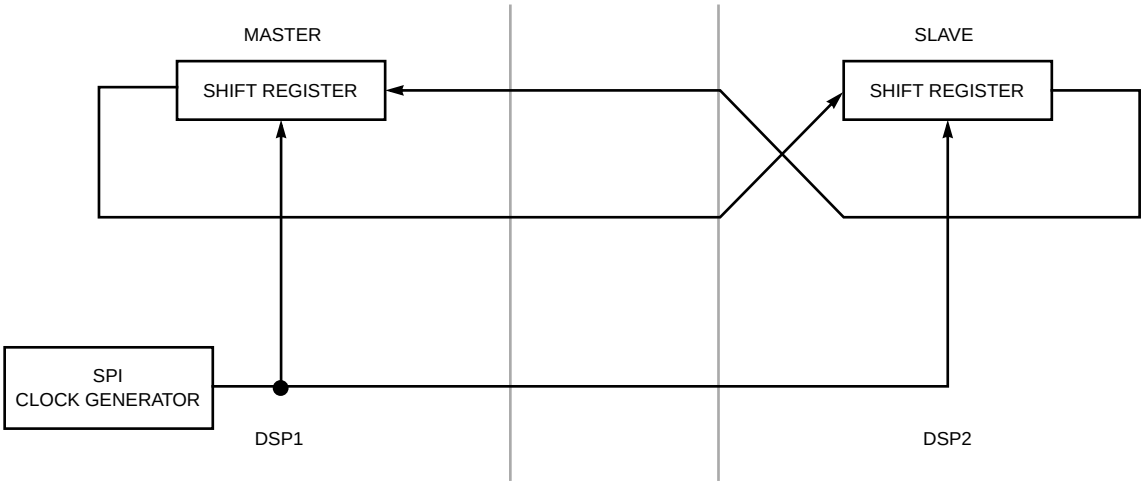


Figure 6-83 SPI Configuration

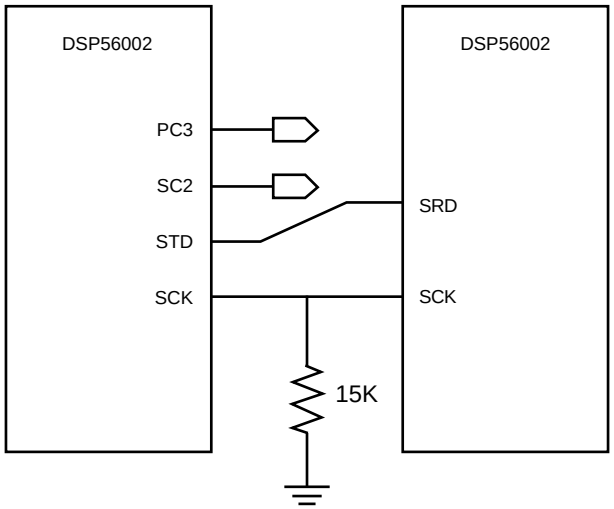


Figure 6-84 On-Demand Mode Example — Hardware Configuration

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
PCD     EQU     $FFE5
SSISR   EQU     $FFEE
TX      EQU     $FFEF
PCDDR   EQU     $FFE3

        ORG     X:0
        DC      $AA0000         ;Data to transmit.
        DC      $330000
        DC      $F00000

;************************************************
;       MAIN PROGRAM*
;************************************************
        ORG     P:$40

        MOVE    #0,R0           ;Pointer to data buffer
        MOVE    #2,M0           ;Length off buffer is 3
```

Figure 6-85 On-Demand Mode Transmit Example Program (Sheet 1 of 2)

```
        MOVEP   #$08,X:PCDDR    ;SC0 (PC3) as general purpose output.
        MOVEP   #$001F,X:CRA    ;Set Word Length=8, CLK=5.12/32 MHz.
        MOVEP   #$1E30,X:CRB    ;Enable transmitter, Mode=On- Demand,
                                ;Gated clock on, synchronous mode,
                                ;Word frame sync selected, frame
                                ;sync and clock are internal and
                                ;output to port pins.
        MOVEP   #$1F0,X:PCC     ;Set PCC for SSI and
LOOP0   BSET    #3,X:PCD        ;Set PC3 high (this is example enable
                                ;or strobe for an external device
                                :such as an ADC).
        MOVEP   X:(R0);pl,X:TX  ;Move data to TX register
TDE1    JCLR    #6,X:SSISR,TDE1 ;Wait for TDE (transmit data register
                                ;empty) to go high.
        MOVEP   X:(R0);pl,X:TX  ;Move next data to TX.
TDE2    JCLR    #6,X:SSISR,TDE2 ;Wait for TDE to go high.
        MOVEP   X:(R0);pl,X:TX  ;Move data to TX.
TDE3    JCLR    #6,X:SSISR,TDE3 ;Wait for TDE=1.
FSC     JSET    #5,X:PCD,FSC    ;Wait for frame sync to go low. NOTE:
                                ;State of frame sync is directly
                                ;determined by reading PC5.
        BCLR    #3,X:PCD        ;Set PC3 lo (example external enable).
;anything goes here (i.e., any processing)
        REP     #100
        NOP
        JMP     LOOP0           ;Continue sequence forever.
        END
```

Figure 6-85 On-Demand Mode Transmit Example Program (Sheet 2 of 2)

Figure 6-86 is the receive program for the scoping loop program presented in Figure 6-85. The receive program also uses the on-demand, gated, synchronous mode with no interrupts (polling). Initialization for the receiver is slightly different than for the transmitter. In CRB, RE is set rather than TE, and SCKD and SCD2 are inputs rather than outputs. After initialization, a JCLR instruction is used to wait for a data word to be received (RDF=1).

When a word is received, it is put into the circular buffer and loops to wait for another data word. The data in the circular buffer will be overwritten after three words are received (does not matter in this application).

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
PCD     EQU     $FFE5
SSISR   EQU     $FFEE
RX      EQU     $FFEF
PCDDR   EQU     $FFE3

;************************************************
;       MAIN PROGRAM*
;************************************************
        ORG     P:$40

        MOVE    #0,R0               ;Pointer to data buffer
        MOVE    #2,M0               ;Length of buffer is 3

        MOVEP   #$001F,X:CRA        ;Set Word Length=8, CLK=5.12/32 MHz.
        MOVEP   #$1E30,X:CRB        ;Enable receiver, Mode=On-Demand,
                                    ;gated clock on, synchronous mode,
                                    ;Word frame sync selected, frame
                                    ;sync and clock are external.
        MOVEP   #$1F0,X:PCC         ;Set PCC for SSI
LOOP
RDF1    JCLR    #7,X:SSISR,RDF1     ;Wait for RDF (receive data register
                                    ;Full) go to high.
        MOVEP   X:RX,X:(R0)+        ;Read data from RX into memory.

        JMP     LOOP                ;Continue sequence forever.

        END
```

Figure 6-86 On-Demand Mode Receive Example Program

6.4.8 Flags

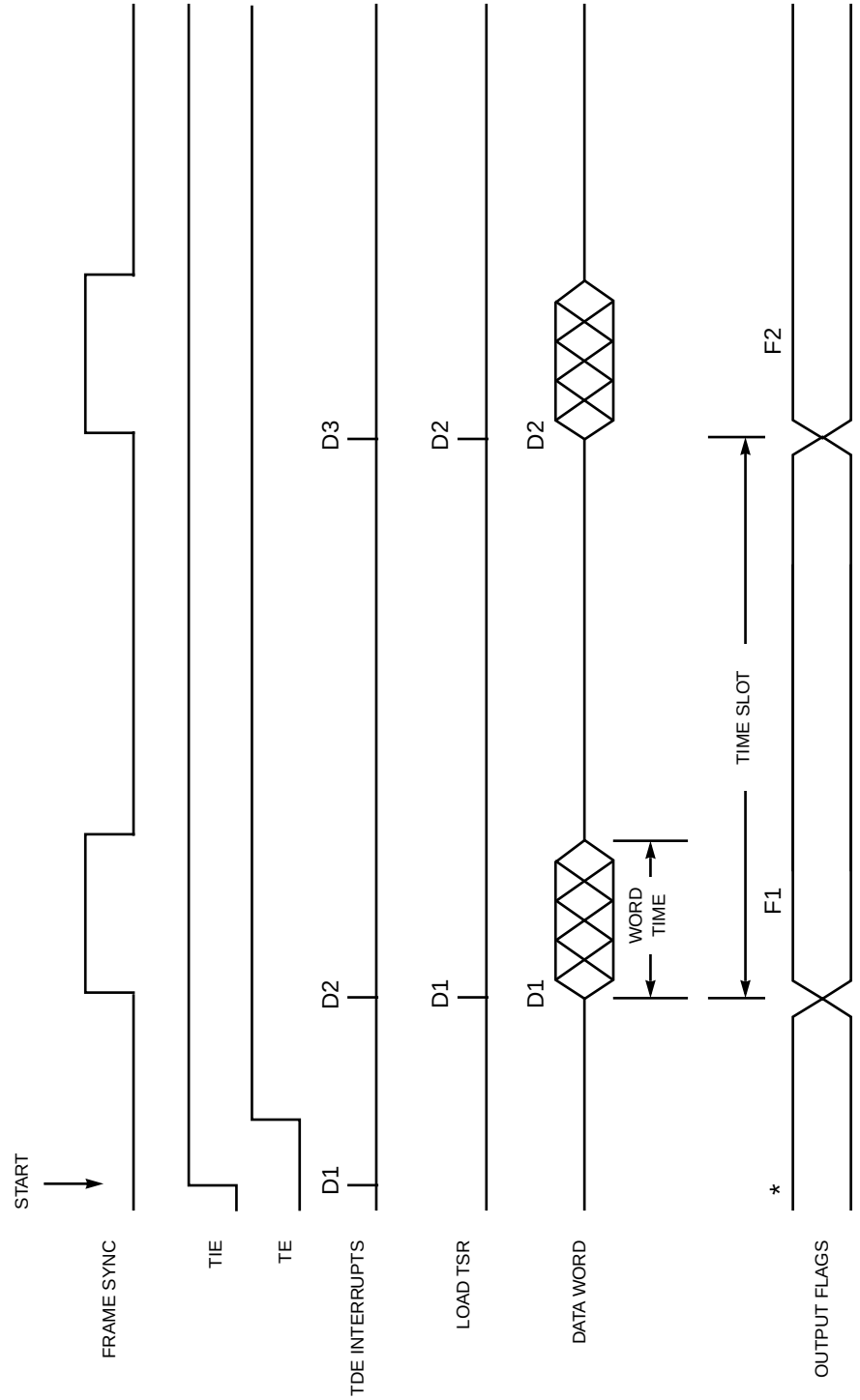
Two SSI pins (SC1 and SC0) are available in the synchronous mode for use as serial I/O flags. The control bits (OF1 and OF0) and status bits (IF1 and IF0) are double buffered to/from SC1 and SC0. Double buffering the flags keeps them in sync with TX and RX. The direction of SC1 and SC0 is controlled by SCD1 and SCD0 in CRB.

Figure 6-87 shows the flag timing for a network mode example. Initially, neither TIE nor TE is set, and the flag outputs are the last flag output value. When TIE is set, a TDE interrupt occurs (the transmitter does not have to be enabled for this interrupt to occur). Data (D1) is written to TX, which clears TDE, and the transmitter is enabled by software. When the frame sync occurs, data (D1) is transferred to the transmit shift register, setting TDE. Data (D1) is shifted out during the first word time, and the output flags are updated. These flags will remain stable until the next frame sync. The TDE interrupt is then serviced by writing data (D2) to TX, clearing TDE. After the TSR completes transmission, the transmit pin is three-stated until the next frame sync

Figure 6-88 shows a speaker phone example that uses a DSP56002 and two codecs. No additional logic is required to connect the codecs to the DSP. The two serial output flags in this example (OF1 and OF0) are used as chip selects to enable the appropriate codec for I/O. This procedure allows the transmit lines to be ORed together. The appropriate output flag pin changes at the same time as the first bit of the transmit word and remains stable until the next transmit word (see Figure 6-89). Applications include serial-device chip selects, implementing multidrop protocols, generating Bell PCM signaling frame syncs, and outputting status information.

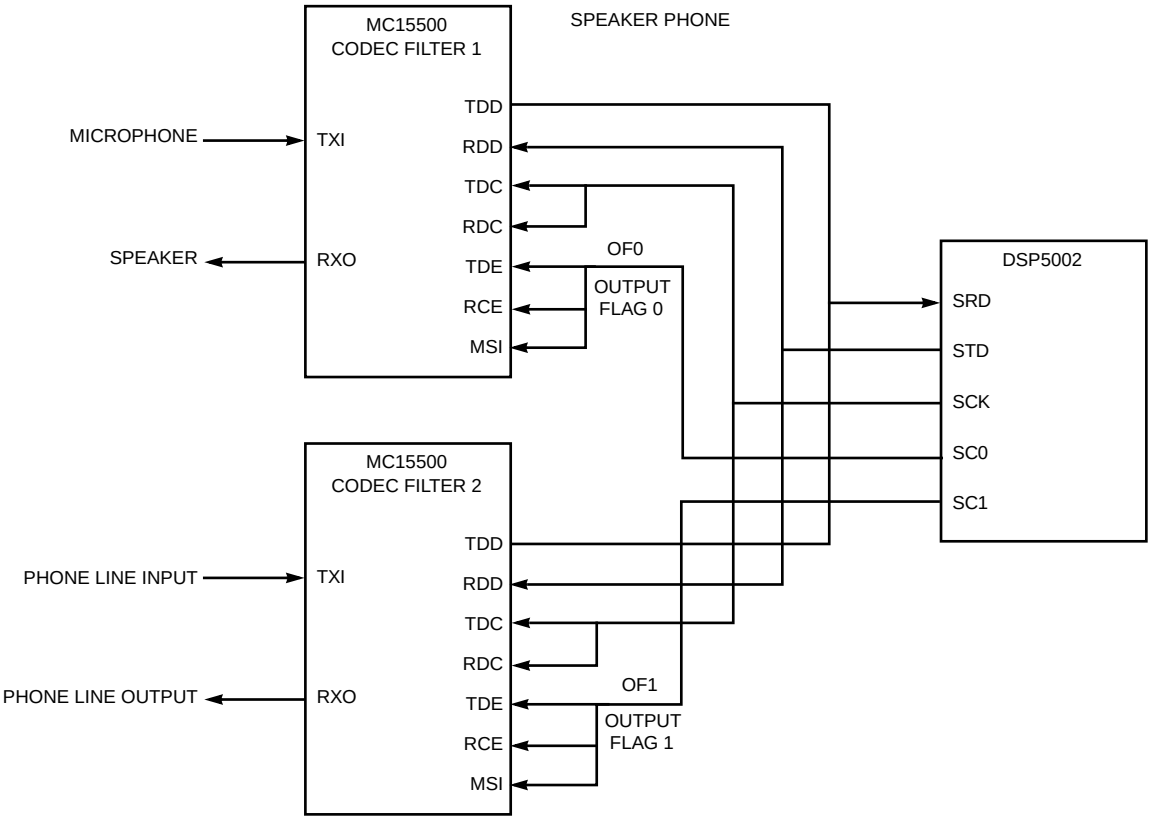
Initializing the flags (see Figure 6-89) is accomplished by setting SYN, SCD1, and SCD0. No other control bits affect the flags. The synchronous control bit must be set (SYN=1) to select the SC1 and SC0 pins as flags. SCD1 and SCD0 select whether SC1 and SC0 are inputs or outputs (input=0, output=1). The other bits selected in Figure 6-89 are chosen for the speaker phone example in Figure 6-88. In this example, the codecs require that the SSI be set for normal mode (MOD=0) with a gated clock (GCK=1) out (SCKD=1).

Serial input flags, IF1 and IF0, are latched at the same time as the first bit is sampled in the receive data word (see Figure 6-90). Since the input was latched, the signal on the input flag pin can change without affecting the input flag until the first bit of the next receive data word. To initialize SC1 or SC0 as input flags, the synchronous control bit in CRB must be set to one (SYN=1) and SCD1 set to zero for pin SC1, and SCD0 must be set to zero for pin SC0. The input flags are bits 1 and 0 in the SSISR (at X:$FFEE).



NOTES:
1. Fn = flags associated with Dn data.
2. Output flags are double buffered with transmit data.
3. Output flags change when data is transferred from TX to the transmit data shift register.
4. Initial flag outputs (*) = last flag output value.
5. Data and flags transition after external frame sync but not before rising edge of clock.

Figure 6-87 Output Flag Timing



NOTE: SC0 and SC1 are output flag 0 and 1 used to software select either filter 1 or 2.

Figure 6-88 Output Flag Example

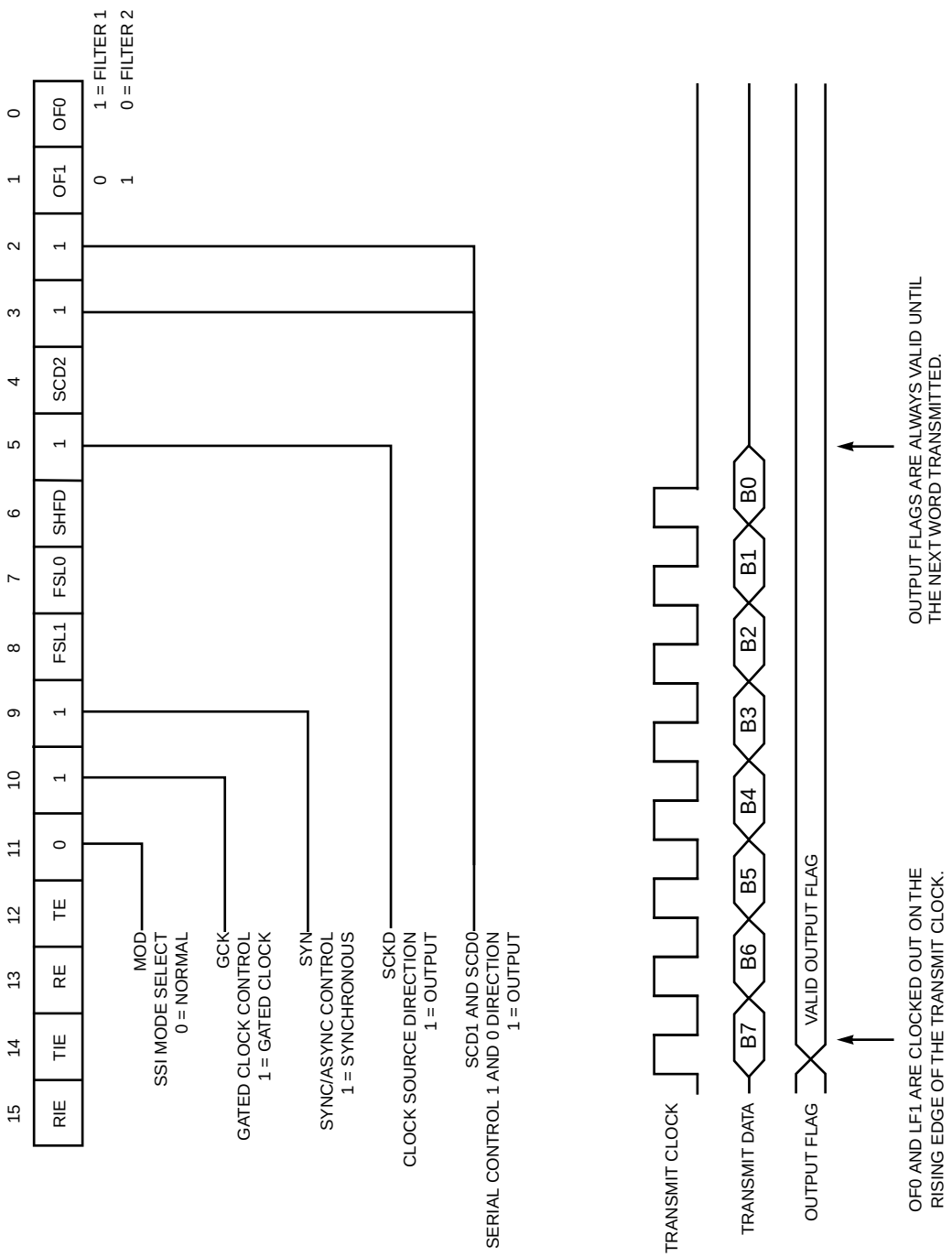


Figure 6-89 Output Flag Initialization

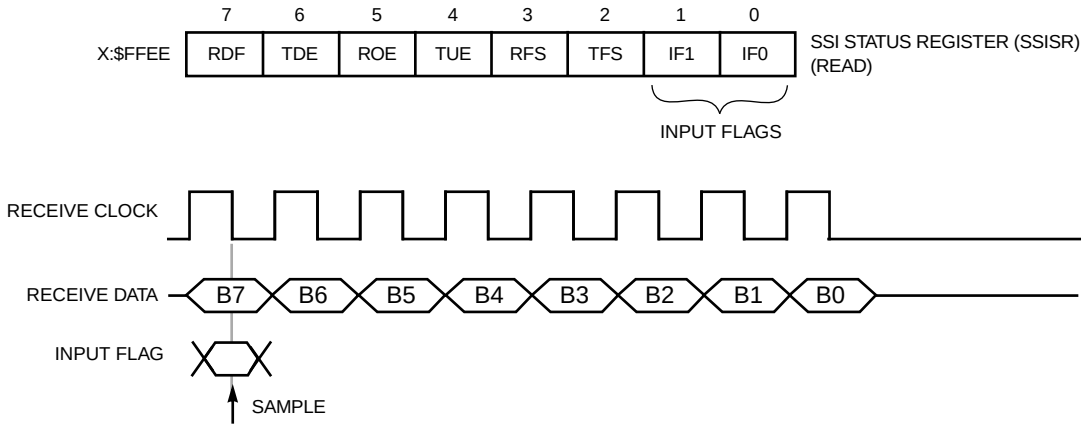


Figure 6-90 Input Flags

6.4.9 Example Circuits

The DSP-to-DSP serial network shown in Figure 6-91 uses no additional logic chips for the network connection. All serial data is synchronized to the data source (all serial clocks and serial syncs are common). This basic configuration is useful for decimation and data reduction when more processing power is needed than one DSP can provide. Cascading DSPs in this manner is useful in several network topologies including star and ring networks.

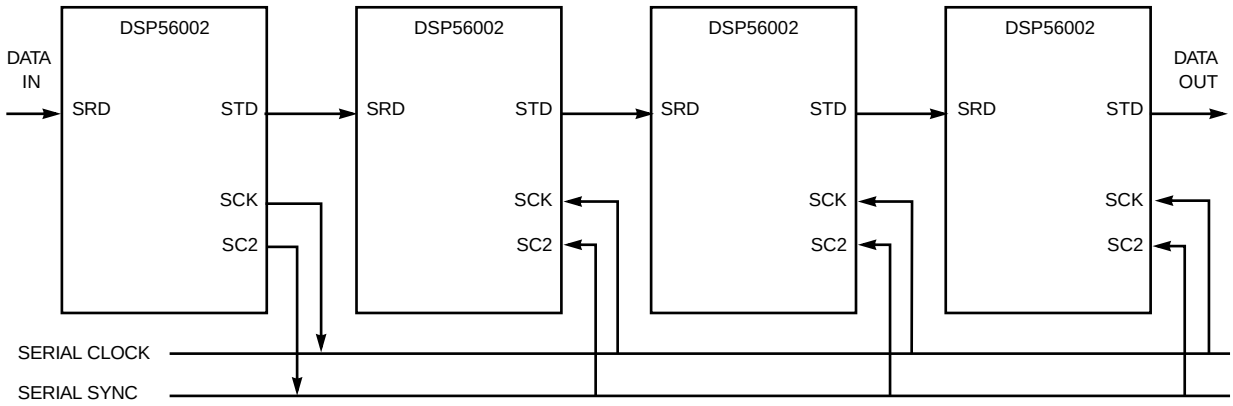


Figure 6-91 SSI Cascaded Multi-DSP System

TDM networks are useful to reduce the wiring needed for connecting multiple processors. A TDM parallel topology, such as the one shown in Figure 6-92, is useful for interpolating filters. Serial data can be received simultaneously by all DSPs, processing can occur in parallel, and the results are then multiplexed to a single serial data out line. This configuration can be cascaded and/or looped back on itself as needed to fit a particular application (see Figure 6-93). The serial and parallel configurations can be combined to form the array processor shown in Figure 6-94. A nearest neighbor array, which is applicable to matrix relaxation processing, is shown in Figure 6-95. To simplify the drawing, only the center DSP is connected in this illustration. In use, all DSPs would have four three-state buffers connected to their STD pin. The flags (SC0 and SC1) on the control master operate the three-state buffers, which control the direction that data is transferred in the matrix (north, south, east, or west).

The bus architecture shown in Figure 6-96 allows data to be transferred between any two DSPs. However, the bus must be arbitrated by hardware or a software protocol to prevent collisions. The master/slave configuration shown in Figure 6-97 also allows data to be transferred between any two DSPs but simplifies network control.

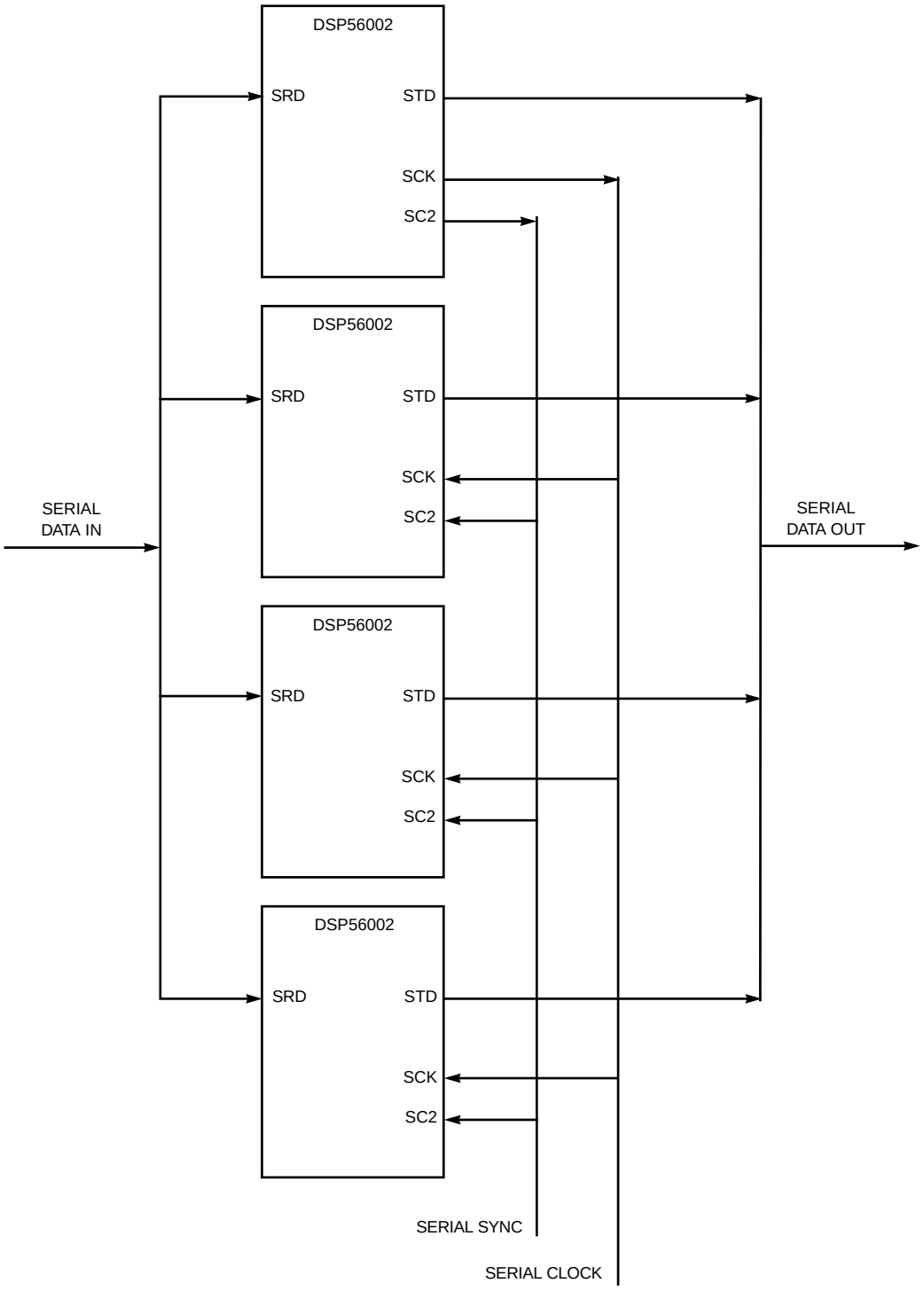


Figure 6-92 SSI TDM Parallel DSP Network

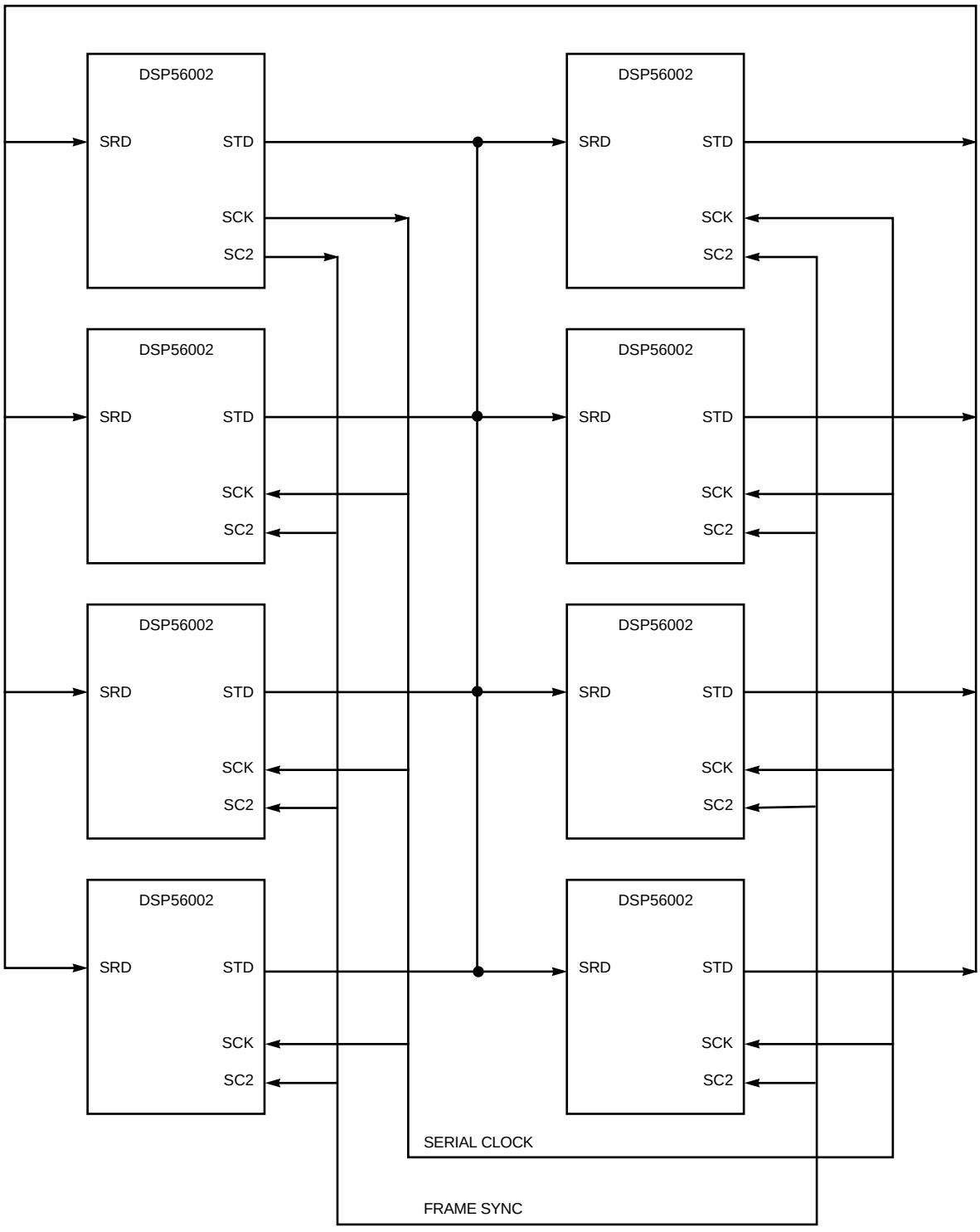


Figure 6-93 SSI TDM Connected Parallel Processing Array

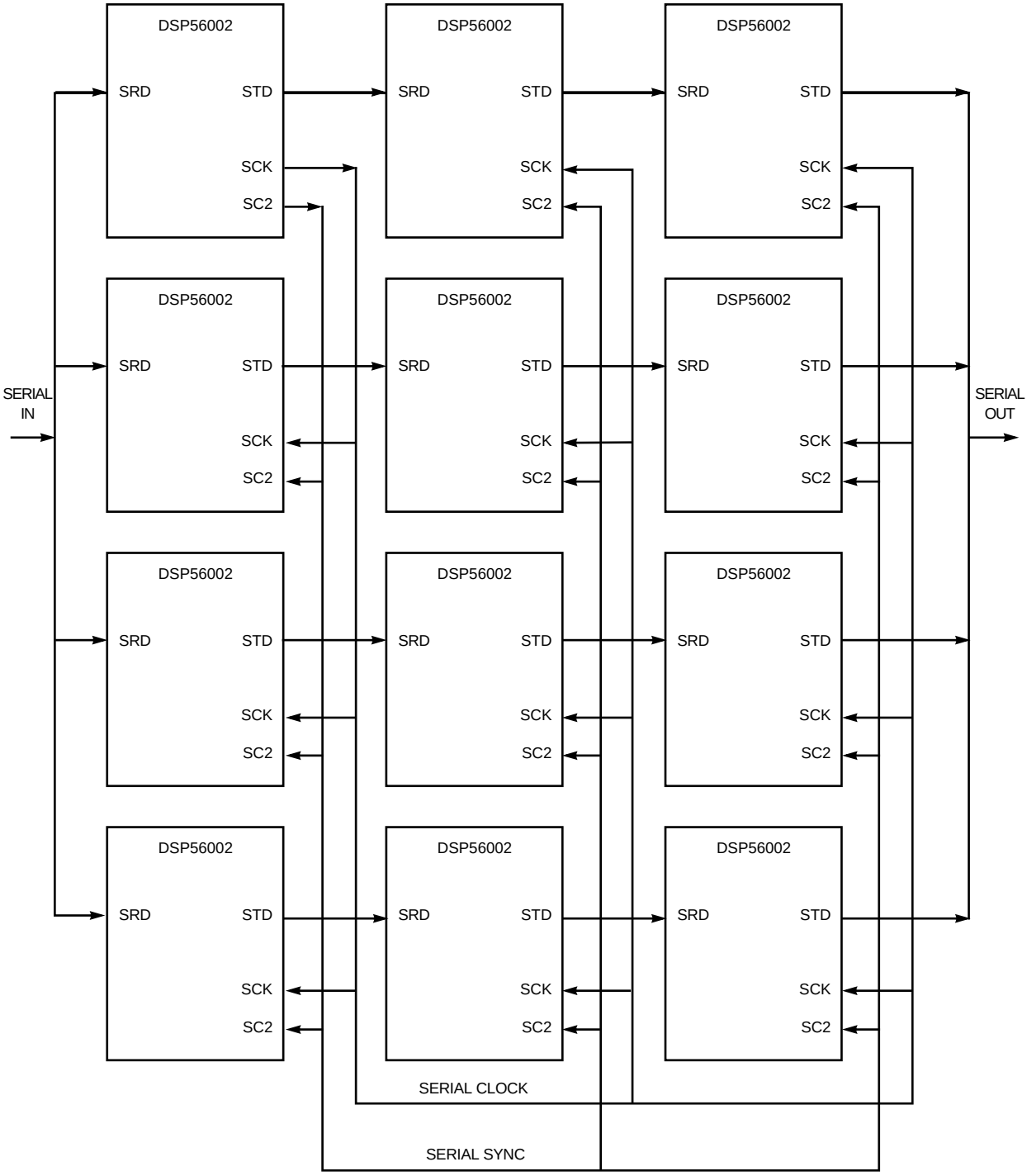


Figure 6-94 SSI TDM Serial/Parallel Processing Array

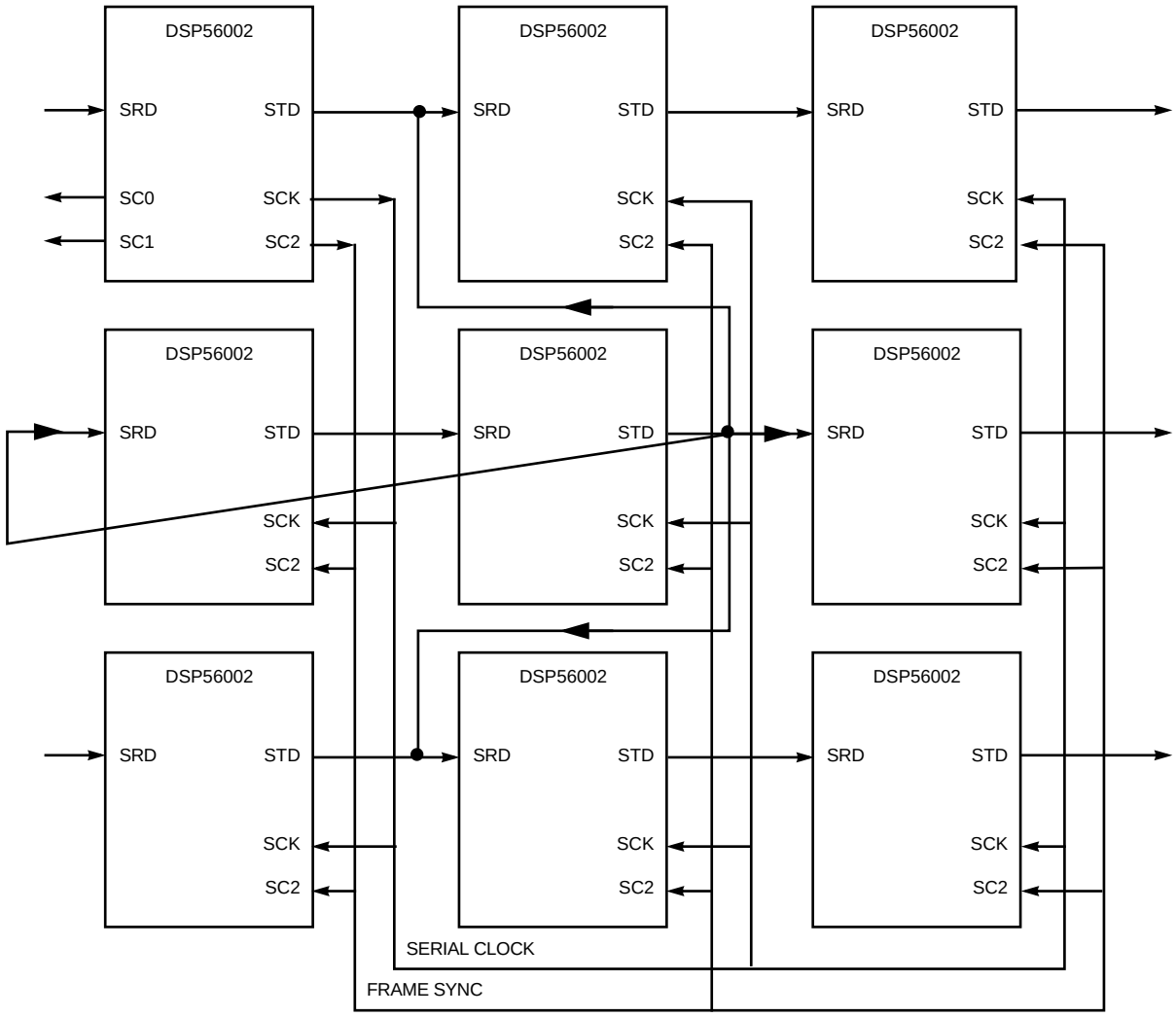


Figure 6-95 SSI Parallel Processing — Nearest Neighbor Array

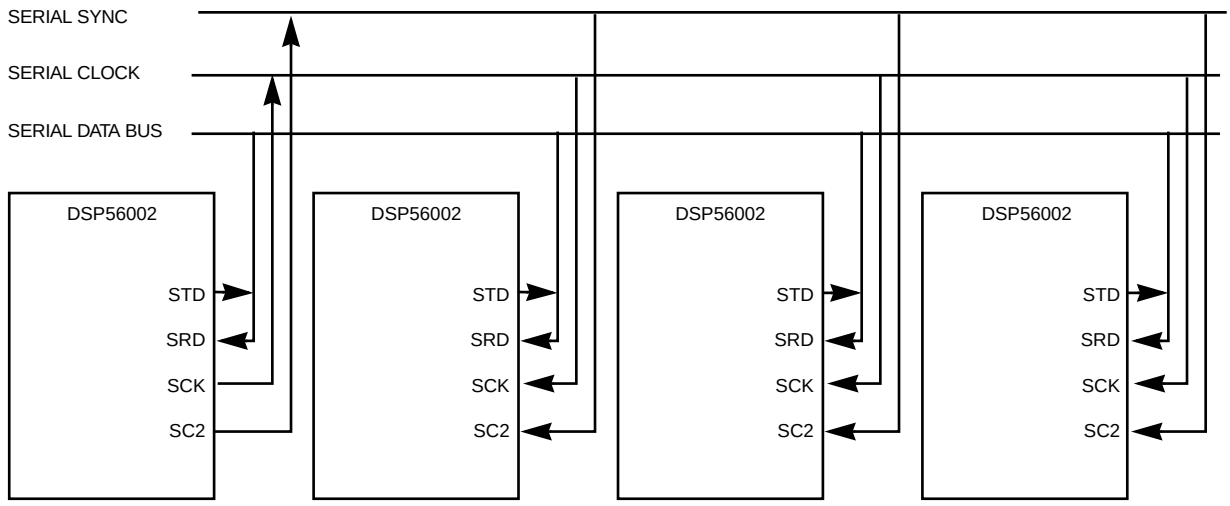


Figure 6-96 SSI TDM Bus DSP Network

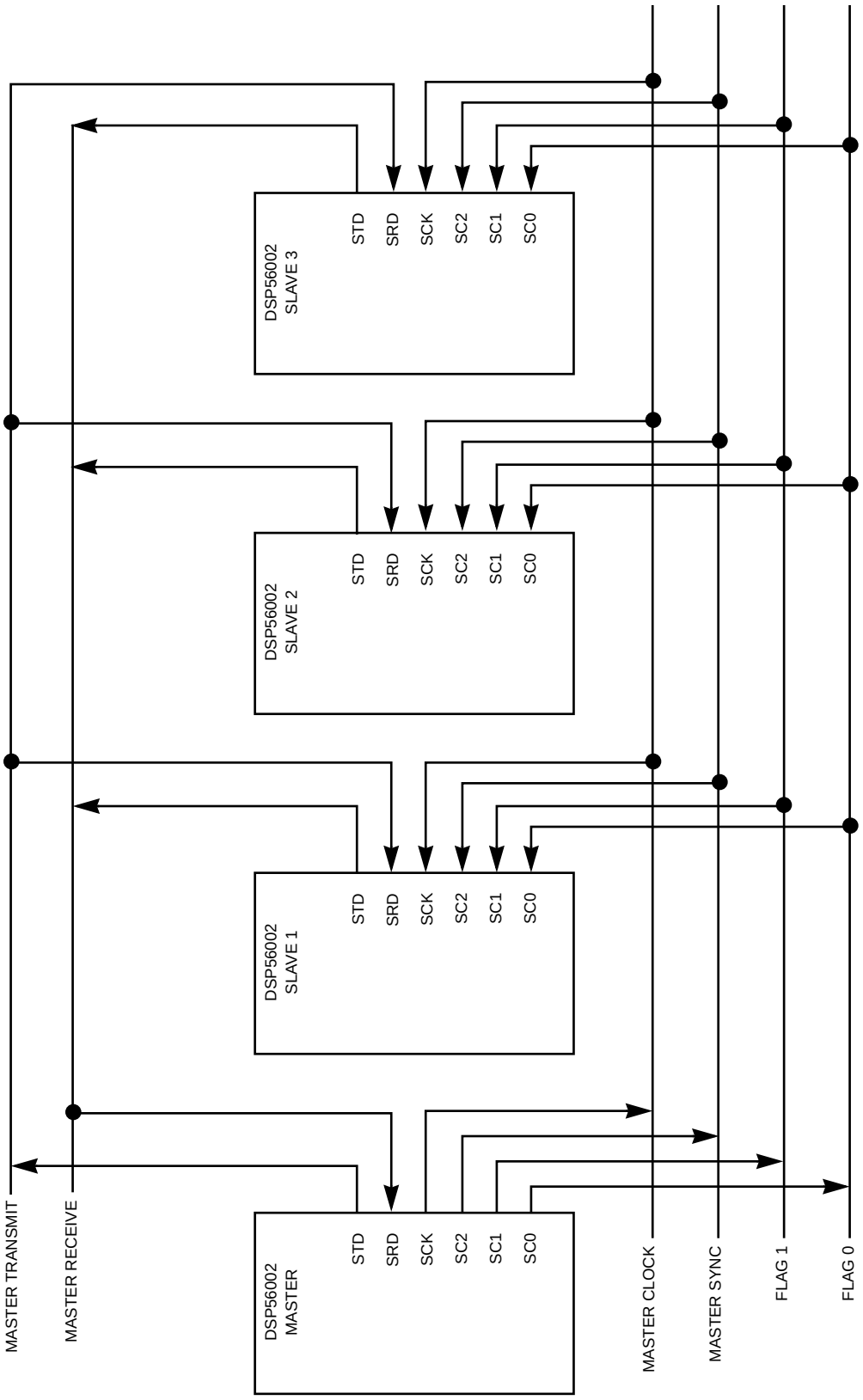


Figure 6-97 SSI TDM Master-Slave DSP Network