Order this document by
MCCIRM/AD
Rev. 1.0

Modular Microcontroller Family

MCCI

Multichannel Communication Interface

Reference Manual

MOTOROLA

Modular Microcontroller Family

MCCI

Multichannel Communication Interface Reference Manual

Ninth Edition

List of Sections

List of Sections

Table of Contents

Table of Contents

Section 4. SPI Submodule

Section 5. SCI Submodule

List of Figures

List of Figures

List of Tables

List of Tables

Section 1. Functional Overview

1.1 Contents

1.2 General Information

The multichannel communication interface (MCCI), a module in Motorola's family of modular microcontrollers, contains three serial interfaces:

- One serial peripheral interface (SPI)
- Two serial communication interfaces (SCI)

The SPI provides easy peripheral expansion or interprocessor communication via a full-duplex, synchronous, 3-line bus: data in, data out, and a serial clock. Serial transfer of eight or 16 bits can begin with the most significant bit (MSB) or least significant bit (LSB).

The MCCI module can be configured as a master or slave device. Clock control logic allows a selection of clock polarity and a choice of two clocking protocols to accommodate most available synchronous serial peripheral devices. When the SPI is configured as a master, software selects one of 254 different bit rates for the serial clock.

The SCI is a universal asynchronous receiver transmitter (UART) serial interface with a standard non-return-to-zero (NRZ) mark/space format. It operates in either full- or half-duplex mode and contains separate transmitter- and receiver-enable bits and a double transmit buffer.

A modulus-type baud rate generator provides rates from 64 baud to 524 Kbaud with a 16.78-MHz system clock.

Word length of either eight or nine bits is software selectable.

Optional parity generation and detection provide either even or odd parity check capability.

Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is received.

1.3 Block Diagram

Figure 1-1 is a block diagram of the MCCI.

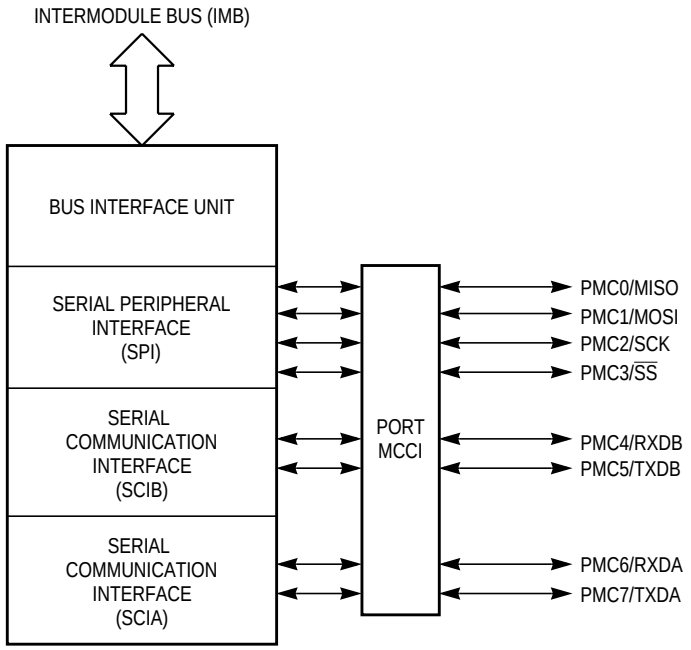


Figure 1-1. MCCI Block Diagram

1.4 Memory Map

The MCCI memory map consists of the global registers and the SPI and SCI control, status, and data registers, as shown in **Table 1-1**. The memory locations shown are offsets from the base address. For the exact location of MCCI registers in the MCU memory map, refer to the appropriate MCU user's manual.

Table 1-1. MCCI Memory Map

Access	Address Offset	15 8	7 0
S	$00	MCCI module configuration register (MMCR)	
S	$02	MCCI test register (MTEST)	
S	$04	SCI interrupt register (ILSCI)	MCCI interrupt vector register (MIVR)
S	$06	SPI interrupt register (ILSPI)	Reserved
S/U	$08	Reserved	MCCI pin assignment register (MPAR)
S/U	$0A	Reserved	MCCI data direction register (MDDR)
S/U	$0C	Reserved	MCCI port data register (PORTMC)
S/U	$0E	Reserved	MCCI port pin state register (PORTMCP)
S/U	$10–$16	Reserved	
S/U	$18	SCIA control register 0 (SCCR0A)	
S/U	$1A	SCIA control register 1 (SCCR1A)	
S/U	$1C	SCIA status register (SCSRA)	
S/U	$1E	SCIA data register (SCDRA)	
S/U	$20–$26	Reserved	
S/U	$28	SCIB control register 0 (SCCR0B)	
S/U	$2A	SCIB control register 1 (SCCR1B)	
S/U	$2C	SCIB status register (SCSRB)	
S/U	$2E	SCIB data register (SCDRB)	
S/U	$30–$36	Reserved	
S/U	$38	SPI control register (SPCR)	
S/U	$3A	Reserved	
S/U	$3C	SPI status register (SPSR)	
S/U	$3E	SPI data register (SPDR)	

S = Supervisor access only
S/U = Supervisor access only or user access, depending on SUPV bit in MMCR

The MCCI memory map includes registers that can be accessed only when the CPU is operating at the supervisor access level and registers that can be programmed to allow supervisor access only or user access.

The MCCI registers that allow supervisor access only (those with an S in the Access column of **Table 1-1**) are the MCCI global registers. These registers define parameters needed by the MCCI to integrate with the MCU.

The remaining MCCI registers (those with S/U in the Access column of **Table 1-1**) can be programmed to either access level.

Refer to **3.5.2 Privilege Levels** and to the appropriate CPU manual for more information on access levels.

Section 2. Signal Descriptions

2.1 Contents

2.2 Introduction

The MCCI has eight external pins. When not being used by the SPI or SCI submodules, these pins can be used as general-purpose I/O (input/output) port pins.

The MCCI global registers shown in this section affect the function of both SPI (serial peripheral interrupt) and SCI (serial communications interface) pins:

- MCCI data direction register, MDDR
- MCCI pin assignment register, MPAR
- MCCI port data register, MPDR

Refer to **3.7 Pin Control and General-Purpose I/O** for more information on these registers.

2.3 SPI Pins

These four pins are associated with the SPI: MISO, MOSI, SCK, and $\overline{SS}$.

The MPAR configures each pin for either SPI function or general-purpose I/O. The MDDR assigns each pin as either input or output. The WOMP bit in the SPI control register (SPCR) determines whether each SPI pin that is configured for output functions as an open-drain output or a normal CMOS output. The MMDR and WOMP assignments are valid regardless of whether the pins are configured for SPI use or general-purpose I/O.

The operation of pins configured for SCI use depends on whether the SCI is operating as a master or a slave, determined by the MSTR bit in the SPCR.

Table 2-1 summarizes SPI pins and their functions.

Table 2-1. SPI Pins

Pin	Mode	SPI Function	Port I/O Signal
MISO	Master	Serial data input to SPI	PMC0
	Slave	Serial data output from SPI	
MOSI	Master	Serial data output from SPI	PMC1
	Slave	Serial data input to SPI	
SCK	Master	Clock output from SPI	PMC2
	Slave	Clock input to SPI	
$\overline{SS}$	Master	Causes mode fault when asserted	PMC3
	Slave	Initiates serial transfer when asserted	

2.3.1 Slave Select ($\overline{SS}$)

Assertion of $\overline{SS}$, a bidirectional signal, selects the SPI submodule for a serial transfer when the SPI is operating in slave mode. Assertion of this signal when the SPI is operating in master mode causes a mode fault.

2.3.2 SPI Serial Clock (SCK)

SCK, a bidirectional signal, furnishes the clock from the SPI in master mode and provides the clock to the SPI in slave mode.

2.3.3 Master In/Slave Out (MISO)

MISO is a bidirectional signal that furnishes serial data input to the SPI in master mode and provides serial data output from the SPI in slave mode.

2.3.4 Master Out/Slave In (MOSI)

MOSI is a bidirectional signal that furnishes serial data output from the SPI in master mode and serial data input to the SPI in slave mode.

2.4 SCI Pins

Four pins are associated with the SCI: TXDA, TXDB, RXDA, and RXDB.

The state of the transmitter enable (TE) or receiver enable (RE) bit in SCI control register 1 of each SCI submodule (SCCR1A, SCCR1B) determines whether the associated pin is configured for SCI operation or general-purpose I/O.

The MDDR assigns each pin as either input or output.

The WOMC bit in SCCR1A or SCCR1B determines whether the associated RXD and TXD pins, when configured as outputs, function as open-drain output pins or normal CMOS outputs.

The MDDR and WOMC assignments are valid regardless of whether the pins are configured for SPI use or general-purpose I/O.

The SCI pins and their functions are listed in **Table 2-2**.

Table 2-2. SCI Pins

Pin Name	Mnemonic	SCI Function	Port I/O Signal
Transmit data	TXDA	Serial data output from SCIA (TE = 1)	PMC7
	TXDB	Serial data output from SCIB (TE = 1)	PMC5
Receive data	RXDA	Serial data input to SCIA (RE = 1)	PMC6
	RXDB	Serial data input to SCIB (RE = 1)	PMC4

2.4.1 Receive Data Pins (RXDA and RXDB)

RXDA and RXDB are the serial data inputs to the SCIA and SCIB interfaces, respectively.

Each pin is also available as a general-purpose I/O pin when the receiver enable (RE) bit in control register 1 (SCCR1) of the associated SCI submodule is cleared. When used for general-purpose I/O, RXDA and RXDB may be configured either as input or output as determined by the RXDA and RXDB bits in the MDDR.

2.4.2 Transmit Data Pins (TXDA and TXDB)

When used for general-purpose I/O, TXDA and TXDB can be configured either as input or output as determined by the TXDA and TXDB bits in the MDDR. The TXDA and TXDB pins are enabled for SCI use by setting the transmitter enable bit (TE) in SCCR1 of each SCI interface.

Section 3. Configuration and Control

3.1 Contents

3.2 Introduction

MCCI global registers are used by the SPI (serial peripheral interface) and both SCI (serial communications interface) interfaces.

These registers are used to:

- Configure MCCI pins
- Provide parameters for interrupt service
- Provide additional information for configuring the MCCI module

This section discusses the functions of these MCCI global registers and provides a checklist for initializing the MCCI.

3.3 MCCI Global Registers

The MCCI global registers define parameters used by the MCCI to interface with the CPU. Global registers are listed in

.

Table 3-1. MCCI Global Registers

Address Offset	Mnemonic	Name
$00	MMCR	MCCI module configuration register
$02	MTEST	MCCI test register
$04	ILSCI	SCI interrupt level register
$05	MIVR	MCCI interrupt vector register
$06	ILSPI	SPI interrupt level register
$09	MPAR	MCCI pin assignment register
$0B	MDDR	MCCI date direction register
$0D	PORTMC	MCCI port data register
$0F	PORTMCP	MCCI port data pin register

3.4 System Initialization

After reset, the MCCI remains in an idle state. Several registers must be initialized before serial operations begin. A general sequence guide for initialization follows. Registers, fields, and bits referred to in the summary are fully described later in this section or in subsequent sections of this manual.

1. Global
    a. MCCI module configuration register (MMCR)
        i. Write an interrupt arbitration number greater than 0 into the IARB field.
        ii. Clear the STOP bit if it is not already cleared.
    b. Interrupt vector and interrupt level registers (MIVR, ILSPI, and ILSCI)
        i. Write the SPI/SCI interrupt vector into MIVR.
        ii. Write the SPI interrupt request level into the ILSPI and the interrupt request levels for the two SCI interfaces into the ILSCI.
    c. Port data register
        i. Write a data word to PORTMC.
        ii. Read a port pin state from PORTMCP.
    d. Pin control registers
        i. Establish the direction of MCCI pins by writing to the MCCI data direction register (MDDR).
        ii. Assign pin functions by writing to the MCCI pin assignment register (MPAR).
2. Serial peripheral interface
    a. SPI control register (SPCR)
        i. Write a transfer rate value into the BAUD field.
        ii. Determine clock phase (CPHA) and clock polarity (CPOL).

iii. Specify an 8- or 16-bit transfer (SIZE) and MSB- or LSB-first transfer mode (LSBF).

iv. Select master or slave operating mode (MSTR).

v. Enable or disable wired-OR operation (WOMP).

vi. Enable or disable SPI interrupts (SPIE).

vii. Enable the SPI by setting the SPE bit.

3. Serial communication interface (SCIA/SCIB)

   a. To transmit, read the SCI status register (SCSR) and then write transmit data to the serial communication data register (SCDR). This clears the transmit data register empty (TDRE) and transmit complete (TC) indicators in the SCSR.

   b. SCI control register 0 (SCCR0)

      i. Write a baud rate value into the BR field.

   c. SCI control register 1 (SCCR1)

      i. Select 8- or 9-bit frame format (M).

      ii. Determine use (PE) and type (PT) of parity generation or detection.

      iii. To receive, set the receiver enable (RE) and receiver interrupt enable (RIE) bits in SCCR1. Select use (RWU) and type (WAKE) of receiver wakeup. Select idle-line detection type (ILT) and enable or disable idle-line interrupt (ILIE).

      iv. To transmit, set transmit enable (TE), transmit interrupt enable (TIE) bits in SCCR1, enable or disable wired-OR operation (WOMC), and transmit complete interrupts (TCIE). Disable break transmission (SBK) for normal operation.

3.5 Module Configuration

The MMCR contains bits and fields to:

- Place the MCCI in low-power operation
- Establish the privilege level required to access MCCI registers
- Establish the priority of the MCCI during interrupt arbitration

3.5.1 Low-Power Stop Mode

When the STOP bit in the MMCR is set, the IMB clock signal to most of the MCCI module is disabled. This places the module in an idle state and minimizes power consumption.

To ensure that the MCCI stops in a known state, assert the STOP bit before executing the CPU LPSTOP instruction. Before asserting the STOP bit, disable the SPI (clear the SPE bit), and disable the SCI receivers and transmitters (clear the RE and TE bits). Complete transfers in progress before disabling the SPI and SCI interfaces.

Once the STOP bit is asserted, it can be cleared by system software or by reset.

3.5.2 Privilege Levels

In systems that support privilege levels, the CPU can operate at either of two levels of access: user or supervisor. MCCI global registers (MMCR, MTEST, ILSCI, MIVR, and ILSPI) can be accessed only when the CPU is operating at the supervisor level. The supervisor bit (SUPV) in the MMCR determines whether the CPU can access the remaining MCCI registers from the supervisor level only or from either privilege level. In systems that do not support privilege levels, the CPU always operates at the supervisor level.

When SUPV is set, the CPU can access MCCI registers only when operating at the supervisor privilege level. When SUPV is clear, the CPU can access all MCCI registers from the user privilege level except the global registers mentioned earlier.

Attempting to access a supervisor-only register from the user privilege level causes the MCCI to respond as if an unimplemented register is being accessed. Writes have no effect and reads return 0s.

The S bit in the CPU status register determines the privilege level at which the CPU is operating (0 = user level and 1 = supervisor level). The SUPV bit in the MMCR can be set or cleared only when the CPU is operating at the supervisor privilege level (S = 1).

Refer to the appropriate CPU manual for more information on privilege levels.

3.5.3 MCCI Module Configuration Register

The MCCI module configuration regster (MMCR) contains bits and fields for placing the MCCI in stop mode, establishing the privilege level required to access certain MCCI registers, and providing the module an interrupt arbitration number. This register can be modified only when the CPU is operating at the supervisor privilege level.

Register address: $XXXX00

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	STOP	0	0	0	0	0	0	0	SUPV	0	0	0	IARB			
Write:																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-1. MCCI Module Configuration Register (MMCR)

STOP — Stop Enable Bit

0 = Normal MCCI clock operation
1 = MCCI clock operation stopped

MMCR[14:8] — Not Implemented

SUPV — Supervisor/User Bit

SUPV defines the assignable MCCI registers as either supervisor-only access or user access.

0 = User access permitted to registers controlled by the SUPV bit
1 = Supervisor access only permitted to MCCI registers

MMCR[6:4] — Not Implemented

IARB — Interrupt Arbitration Number Bit

The value in this field is used to arbitrate for the IMB when two or more modules generate simultaneous interrupts at the same interrupt-request level. The IARB field should be initialized by system software to a value from $F (highest priority) through $1 (lowest priority). Refer to **3.6 Interrupts** for more information on interrupts and interrupt arbitration.

3.5.4 MCCI Test Register

The MCCI test register (MTEST) is used only during factory testing of the MCU.

3.6 Interrupts

The interrupt request level of each of the three MCCI interfaces can be programmed to a value of 0 (interrupts disabled) through 7 (highest priority). These levels are selected by the ILSCIA and ILSCIB fields in the SCI interrupt level register (ILSCI) and the ILSPI field in the SPI interrupt level register (ILSPI). In case two or more MCCI submodules request an interrupt simultaneously and are assigned the same interrupt request level, the SPI submodule is given the highest priority and SCIB is given the lowest.

When an interrupt is requested which is at a higher level than the interrupt mask in the CPU status register, the CPU initiates an interrupt acknowledge cycle. During this cycle, the MCCI compares its interrupt request level to the level recognized by the CPU. If a match occurs, arbitration with other modules begins.

Interrupting modules present their arbitration number on the intermodule bus (IMB), and the module with the highest number wins. The arbitration number for the MCCI is programmed into the interrupt arbitration (IARB) field of the MMCR. Each module should be assigned a unique arbitration number. The reset value of the IARB field is $0, which prevents the MCCI from arbitrating during an interrupt acknowledge cycle. The IARB field should be initialized by system software to a value from $F (highest priority) through $1 (lowest priority). Otherwise, the CPU identifies any interrupts generated as spurious and takes a spurious-interrupt exception.

If the MCCI wins the arbitration, it generates an interrupt vector that uniquely identifies the interrupting serial interface. The six most significant bits (MSBs) are read from the interrupt vector (INTV) field in the MCCI interrupt vector register (MIVR). The two least significant bits (LSBs) are assigned by the MCCI according to the interrupting serial interface, as indicated in **Table 3-2**.

Select a value for INTV so that each MCCI interrupt vector corresponds to one of the user-defined vectors ($40–$FF).

Refer to the appropriate CPU manual for additional information on interrupt vectors.

Table 3-2. MCCI Interrupt Vectors

Interface	INTV[1:0]
SCIA	00
SCIB	01
SPI	10

3.6.1 SCI Interrupt Level Register

The SCI interrupt level register (ILSCI) determines the level of interrupts requested by each SCI. Separate fields hold the interrupt-request levels for SCIA and SCIB. This register may be accessed only when the CPU is operating at the supervisor privilege level.

Register address: $XXXX04

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	ILSCIB			ILSCIA			MIVR (MCCI interrupt vector register)							
Reset:	0	0	0	0	0	0	0	0								

Figure 3-2. SCI Interrupt Level Register (ILSCI)

ILSCIA and ILSCIB — Interrupt Level Bits for SCIA and SCIB

ILSCIA and ILSCIB determine the interrupt-request levels of SCIA and SCIB interrupts, respectively. Program this field to a value from $0 (interrupts disabled) through $7 (highest priority).

3.6.2 MCCI Interrupt Vector Register

The MCCI interrupt vector register (MIVR) determines which three vectors in the exception vector table are to be used for MCCI interrupts. The SPI and both SCI interfaces have separate interrupt vectors adjacent to one another. When initializing the MCCI, program INTV[7:2] so that INTV[7:0] correspond to one of the user-defined vectors ($40–$FF). INTV[1:0] are determined by the serial interface causing the interrupt.

At reset, MIVR is initialized to $0F, which corresponds to the interrupt vector in the exception table.

Register address: $XXXX05

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	ILSCI (SCI interrupt level register)								INTV[7:2]						INTV[1:0}	
Write:																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

= Unimplemented

Figure 3-3. MCCI Interrupt Vector Register (MIVR)

INTV[7:2] — Interrupt Vector [7:2] Bits

High-order six bits of MCCI interrupt vector programmed by user.

INTV[1:0] — Interrupt Vector [1:0] Bits

Writes to INTV0 and INTV1 have no meaning or effect. Reads of INTV0 and INTV1 return a value of 1.

00 = SCIA is source of interrupt.

01 = SCIB is source of interrupt.

10 = SPI is source of interrupt.

3.6.3 SPI Interrupt Level Register

The SPI interrupt level register (ILSPI) determines the priority level of interrupts requested by the SPI. When the interrupt-request level programmed in this field matches the interrupt-request level of one of the SCI interfaces and both request an interrupt simultaneously, the SPI is given priority. This register may be accessed only when the CPU is operating at the supervisor privilege level.

Register address: $XXXX06

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	0	0	ILSPI			0	0	0	Reserved							
Write:																
Reset:	0	0	0	0	0	0	0	0								

Figure 3-4. SPI Interrupt Level Register (ILSPI)

ILSPI — Interrupt Level for SPI Bit

ILSPI determines the priority level of SPI interrupts. Program this field to a value from $0 (interrupts disabled) through $7 (highest priority).

3.7 Pin Control and General-Purpose I/O

The eight pins used by the SPI and SCI subsystems have alternate functions as general-purpose I/O pins. Configuring the MCCI submodule includes programming each pin for either general-purpose I/O or its serial interface function. In either function, each pin also must be programmed as input or output.

The MCCI data direction register (MDDR) assigns each MCCI pin as either input or output.

The MCCI pin assignment register (MPAR) assigns the MOSI, MISO, and $\overline{SS}$ pins as either SPI pins or general-purpose I/O. (The fourth pin, SCK, is automatically assigned to the SPI whenever the SPI is enabled, for instance, when the SPE bit in the SPI control register is set.)

The receiver enable (RE) and transmit enable (TE) bits in the SCI control registers (SCCR0A and SCCR0B) automatically assign the associated pin as an SCI pin when set or general-purpose I/O when cleared. **Table 3-3** summarizes how pin function and direction are assigned.

Table 3-3. Pin Assignments

Pin	Function Assigned By	Direction Assigned By
TXDA/PMC7	TE bit in SCCR0A	MDDR7
RXDA/PMC6	RE bit in SCCR0A	MDDR6
TXDB/PMC5	TE bit in SCCR0B	MDDR5
RXDB/PMC4	RE bit in SCCR0B	MDDR4
$\overline{SS}$/PMC3	SS bit in MPAR	MDDR3
SCK/PMC2	SPE bit in SPCR	MDDR2
MOSI/PMC1	MOSI bit in MPAR	MDDR1
MISO/PMC0	MISO bit in MPAR	MDDR0

3.7.1 MCCI Port Data Registers

Two registers are associated with port MCCI, the MCCI general-purpose I/O port. Pins used for general-purpose I/O must be configured for that function. (**Table 3-3** explains how to do this.) When using port MCCI as an output port, after configuring the pins for I/O, write the first byte to be output before writing to the MDDR. Then write to the MDDR to assign each I/O pin as either input or output. This outputs the value contained in register PORTMC for all pins defined as outputs. To output different data, write another byte to PORTMC.

Register name and address: MCCI Port Data Register (PORTMC), $XXXX0C
MCCI Port Pin State Register (PORTMCP), $XXXX0E

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: / Write:	Reserved								PMC (TXDA)	PMC6 (RXDA)	PMC5 (TXDB)	PMC4 (RXDB)	PMC3 ($\overline{SS}$)	PMC2 (SCK)	PMC1 (MOSI)	PMC0 (MISO)
Reset:									0	0	0	0	0	0	0	0

Figure 3-5. MCCI Port Data Registers (PORTMC and PORTMCP)

Writes to PORTMC, the MCCI port data register, are stored in the internal data latch. If any bit of PORTMC is configured as discrete output, the value latched for that bit is driven onto the pin. Reads of PORTMC return the value of the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value of the latch.

Reads of PORTMCP, the MCCI pin state register, always return the state of the pins regardless if the pins are configured for input or output. Writes to PORTMCP have no effect.

3.7.2 MCCI Pin Assignment Register

The MCCI pin assignment register (MPAR) determines which of the SPI pins (with the exception of the SCK pin) actually are used by the SPI submodule and which pins are available for general-purpose I/O. (The state of SCK is determined by the SPI enable bit in SPCR1.) Pins may be assigned to the SPI or to function as general-purpose I/O on a pin-by-pin basis. SPI pins designated by the MPAR as general-purpose I/O are controlled only by MDDR and PORTMC; the SPI has no effect on these pins. The MPAR does not affect the operation of the SCI submodule.

Register address: $XXXX09

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	Reserved								0	0	0	0	SS	0	MOSI	MISO
Reset:									0	0	0	0	0	0	0	0

Figure 3-6. MCCI Pin Assignment Register (MPAR)

MPAR[7:4] and MPAR2 — Not implemented

SS — Slave Select Bit

MOSI — Master Out/Slave In Bit

MISO — Master In/Slave Out Bit

These bits determine whether the associated MCCI port pin functions as a general-purpose I/O pin or is assigned to the SPI submodule.

0 = General-purpose I/O

1 = SPI function

3.7.3 MCCI Data Direction Register

The MCCI data direction register (MDDR) configures each pin as an input or output. The MDDR affects both serial interface function and general-purpose I/O function. During reset, all MCCI pins are configured as inputs.

Register address: $XXXX0B

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	Reserved								TXDA	RXDA	TXDB	RXDB	SS	SCK	MOSI	MISO
Write:																
Reset:									0	0	0	0	0	0	0	0

Figure 3-7. MCCI Data Direction Register (MDDR)

These bits assign the corresponding pin to be input or output.

0 = Input
1 = Output

Configuration and Control

Section 4. SPI Submodule

4.1 Contents

4.2 Introduction

The SPI submodule communicates with external peripherals and other MCUs via a synchronous serial bus. The SPI (serial peripheral interface) is fully compatible with the SPI systems found on other Motorola devices, such as the M68HC11 and M68HC05 Families. The SPI can perform full duplex 3-wire or half duplex 2-wire transfers. Serial transfer of eight or 16 bits can begin with the most significant bit (MSB) or least significant bit (LSB). The system can be configured as a master or slave device.

Clock control logic allows a selection of clock polarity and a choice of two clocking protocols to accommodate most available synchronous serial peripheral devices. When the SPI is configured as a master, software selects one of 254 different bit rates for the serial clock.

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. A slave-select line allows individual selection of a slave SPI device. Slave devices which are not selected do not interfere with SPI bus activities. On a master SPI device, the slave-select line optionally can be used to indicate a multiple-master bus contention.

Error-detection logic is included to support interprocessor interfacing. A write-collision detector indicates when an attempt is made to write data to the serial shift register while a transfer is in progress. A multiple-master mode-fault detector automatically disables SPI output drivers if more than one MCU simultaneously attempts to become bus master.

Table 4-1 is a quick reference guide to the bits and fields involved in SPI operation. All registers, bits, and fields referred to are discussed fully later in this section or in **Section 3. Configuration and Control**.

Table 4-1. SPI Bit/Field Quick Reference Guide

Mnemonic	Function	Register
BAUD	Serial clock baud rate	SPCR
CPHA	Clock phase	SPCR
CPOL	Clock polarity	SPCR
LSBF	Least significant bit first	SPCR
MISO	Master in/slave out	MPAR, MDDR, PORTMC, PORTMCP
MODF	Mode fault flag	SPSR
MOSI	Master out/slave in	MPAR, MDDR, PORTMC, PORTMCP
MSTR	Master/slave mode select	SPCR
SS	Slave select	MPAR, MDDR, PORTMC, PORTMCP
SCK	Serial clock	MDDR, PORTMC, PORTMCP
SIZE	Data size (8- or 16-bit)	SPCR
SPE	SPI enable	SPCR
SPIE	SPI finished interrupt enable	SPCR
SPIF	SPI transfer complete flag	SPSR
WCOL	Write collision flag	SPSR
WOMP	Wired-OR mode for SPI pins	SPCR

4.3 Block Diagram

Figure 4-1 shows a block diagram of the SPI submodule components.

SPI Submodule

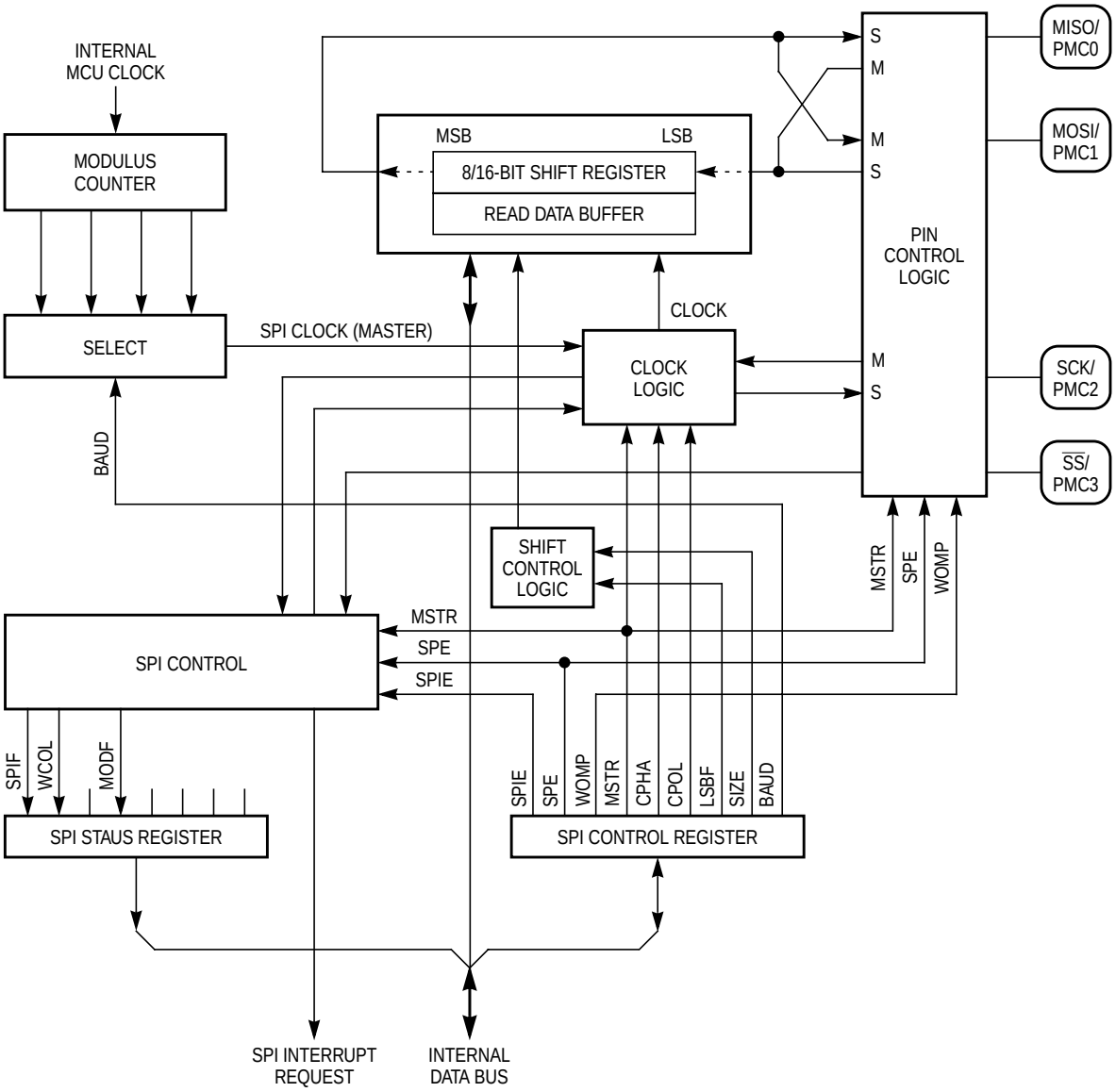


Figure 4-1. SPI Submodule Block Diagram

4.4 SPI Pins

The SPI uses four bidirectional pins. These pins may be configured for general-purpose I/O when not needed for the SPI. **Table 4-2** shows SPI pins and their functions. Refer to **Section 2. Signal Descriptions** for additional information.

Table 4-2. SPI Pin Function

Pin Name	Mode	Function
Master in/slave out (MISO)	Master Slave	Provides serial data input to the SPI Provides serial data output from the SPI
Master out/slave in (MOSI)	Master Slave	Provides serial output from the SPI Provides serial input to the SPI
Serial clock (SCK)	Master Slave	Provides clock output from the SPI Provides clock input to the SPI
Slave select ($\overline{SS}$)	Master Slave	Detects bus-master mode fault Selects the SPI for an externally initiated serial transfer

4.5 Operating Modes

The SPI operates in either master or slave mode. Master mode is used when the MCU originates data transfers. Slave mode is used when an external device initiates serial transfers to the MCU. The MSTR bit in SPCR selects master or slave operation.

4.5.1 Master Mode

Setting the MSTR bit in SPCR selects master mode operation. In master mode, the SPI can initiate serial transfers but cannot respond to externally initiated transfers. When the slave-select input of a device configured for master mode is asserted, a mode fault occurs.

3.4 System Initialization provides general initialization procedures. When using the SPI in master mode, include these specific steps:

1. Write to the MMCR, MIVR, and ILSPI as outlined in **3.4 System Initialization**.

2. Write to the MPAR to assign these three pins to the SPI: MISO, MOSI, and (optionally) $\overline{SS}$. MISO is used for serial data input in master mode, and MOSI is used for serial data output. Either or both may be necessary, depending on the particular application. $\overline{SS}$ is used to generate a mode fault in master mode. If this SPI is the only possible master in the system, the $\overline{SS}$ pin may be used for general-purpose I/O.

3. Write to the MDDR to direct the data flow on SPI pins. Configure the SCK (serial clock) and MOSI pins as outputs. Configure MISO and (optionally) $\overline{SS}$ as inputs.

4. Write to the SPCR to assign values for BAUD, CPHA, CPOL, SIZE, LSBF, WOMP, and SPIE. Set the MSTR bit to select master operation. Set the SPE bit to enable the SPI.

5. Enable the slave device.

6. Write appropriate data to the SPI data register to initiate the transfer.

When the SPI reaches the end of the transmission, it sets the SPIF flag in the SPSR. If the SPIE bit in the SPCR is set, an interrupt request is generated when SPIF is asserted. After the SPSR is read with SPIF set and then the SPDR is read or written to, the SPIF flag is cleared automatically.

Data transfer is synchronized with the internally generated serial clock (SCK). Control bits CPHA and CPOL in SPCR control clock phase and polarity. Combinations of CPHA and CPOL determine the SCK edge on which the master MCU drives outgoing data from the MOSI pin and latches incoming data from the MISO pin. (Refer to **4.6 SPI Clock Phase and Polarity Controls**.)

4.5.2 Slave Mode

Clearing the MSTR bit in SPCR selects slave mode operation. In slave mode, the SPI is unable to initiate serial transfers. Transfers are initiated by an external bus master. Typically, slave mode is used on a multimaster SPI bus. Only one device can be bus master (operate in master mode) at any given time.

3.4 System Initialization provides general initialization procedures. When using the SPI in slave mode, include these specific steps:

1. Write to the MMCR and interrupt registers as outlined in **3.4 System Initialization**.
2. Write to the MPAR to assign these three pins to the SPI: MISO, MOSI, and $\overline{SS}$. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the input serial clock. $\overline{SS}$ selects the SPI when asserted.
3. Write to the MDDR to direct the data flow on SPI pins. Configure the SCK, MOSI, and $\overline{SS}$ pins as inputs. Configure MISO as an output.
4. Write to the SPCR to assign values for CPHA, CPOL, SIZE, LSBF, WOMP, and SPIE. Set the MSTR bit to select master operation. Set the SPE bit to enable the SPI. (The BAUD field in the SPCR of the slave device has no effect on SPI operation.)

When SPE is set and MSTR is clear, a low state on the $\overline{SS}$ pin initiates slave mode operation. (The $\overline{SS}$ pin is used only as an input.)

After a byte or word of data is transmitted, the SPI sets the SPIF flag. If the SPIE bit in SPCR is set, an interrupt request is generated when SPIF is asserted.

Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine the SCK edge on which the slave MCU latches incoming data from the MOSI pin and drives outgoing data from the MISO pin.

4.6 SPI Clock Phase and Polarity Controls

Two bits in the SPI control register (SPCR) determine SCK phase and polarity. The clock polarity (CPOL) bit selects clock polarity (high true or low true clock). The clock phase control bit (CPHA) selects one of two transfer formats and affects the timing of the transfer. The clock phase and polarity should be the same for the master and slave devices. In some cases, the phase and polarity may be changed between transfers to allow a master device to communicate with slave devices with different requirements. The flexibility of the SPI system allows it to be directly interfaced to almost any existing synchronous serial peripheral.

4.6.1 CPHA = 0 Transfer Format

Figure 4-2 is a timing diagram of an 8-bit, MSB-first SPI transfer in which CPHA equals 0. Two waveforms are shown for SCK: one for CPOL equal to 0 and another for CPOL equal to 1. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are directly connected between the master and the slave. The MISO signal shown is the output from the slave and the MOSI signal shown is the output from the master. The $\overline{SS}$ line is the chip-select input to the slave.

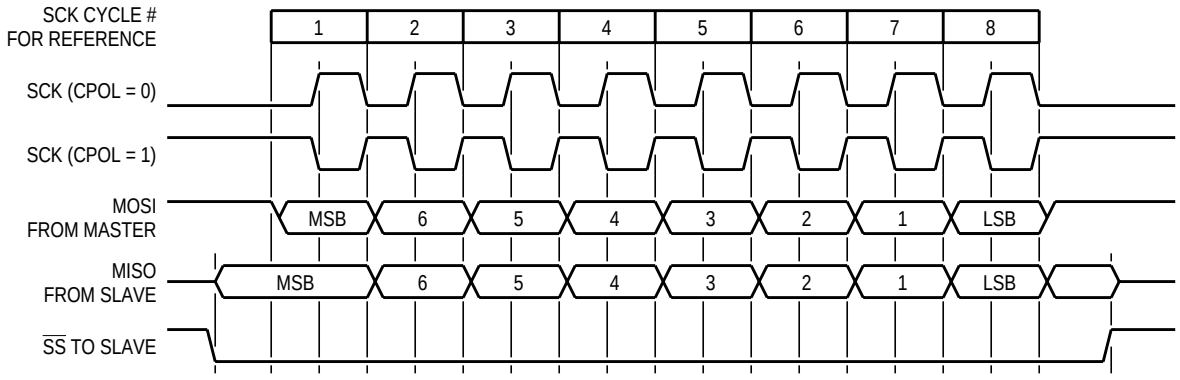


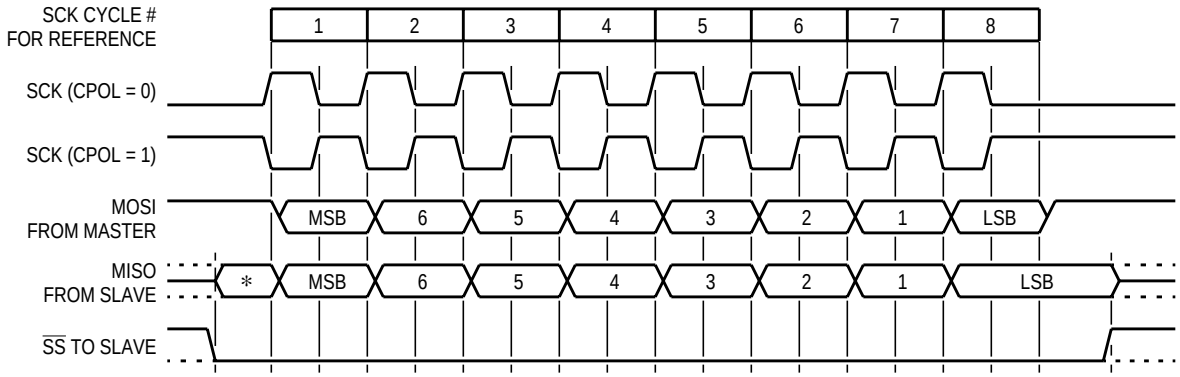
Figure 4-2. CPHA = 0 SPI Transfer Format

For a master, writing to the SPDR initiates the transfer. For a slave, the falling edge of $\overline{SS}$ indicates the start of a transfer. The SCK signal remains inactive for the first half of the first SCK cycle. Data is latched on the first and each succeeding odd clock edge, and the SPI shift register is left-shifted on the second and succeeding even clock edges. SPIF is set at the end of the eighth SCK cycle.

When CPHA equals 0, the $\overline{SS}$ line must be negated and reasserted between each successive serial byte. If the slave writes data to the SPI data register while $\overline{SS}$ is asserted (low), a write collision error results. To avoid this problem, the slave should read bit 3 of PORTMCP, which indicates the state of the $\overline{SS}$ pin, before writing to the SPDR again.

4.6.2 CPHA = 1 Transfer Format

Figure 4-3 is a timing diagram of an 8-bit, MSB-first SPI transfer in which CPHA equals 1. Two waveforms are shown for SCK, one for CPOL equal to 0 and another for CPOL equal to 1. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are directly connected between the master and the slave. The MISO signal shown is the output from the slave and the MOSI signal shown is the output from the master. The $\overline{SS}$ line is the slave select input to the slave.



* Not defined, but normally LSB of previously transmitted character.

Figure 4-3. CPHA = 1 SPI Transfer Format

For a master, writing to the SPDR initiates the transfer. For a slave, the first edge of SCK indicates the start of a transfer. The SPI is left-shifted on the first and each succeeding odd clock edge, and data is latched on the second and succeeding even clock edges.

SCK is inactive for the last half of the eighth SCK cycle. For a master, SPIF is set at the end of the eighth SCK cycle (after the 17th SCK edge). Since the last SCK edge occurs in the middle of the eighth SCK cycle, however, the slave has no way of knowing when the end of the last SCK cycle occurs. The slave, therefore, considers the transfer complete after the last bit of serial data has been sampled, which corresponds to the middle of the eighth SCK cycle.

When CPHA is 1, the $\overline{SS}$ line may remain at its active low level between transfers. This format is sometimes preferred in systems having a single fixed master and only one slave that needs to drive the MISO data line.

4.7 SPI Serial Clock Baud Rate

Baud rate is selected by writing a value from 2 to 255 into the BAUD field in the SPCR of the master MCU. (Writing a BAUD value into the SPCR of the slave device has no effect.) The SPI uses a modulus counter to derive SPI serial clock (SCK) baud rate from the MCU system clock.

This equation determines the SCK baud rate:

SCK baud rate = system clock/(2 x BAUD)

Therefore, given a desired SCK baud rate, compute BAUD like this:

BAUD = system clock/(2 x SCK baud rate)

Giving BAUD a value of 0 or 1 disables the baud rate generator. SCK is disabled and assumes its inactive state value.

BAUD has 254 active values. **Table 4-3** lists several possible baud values and the corresponding SCK frequency based on a 16.78-MHz system clock.

Table 4-3. Examples of SCK Frequencies

System Clock Frequency	Required Division Ratio	Value of SPBR	Actual SCK Frequency
16.78 MHz	4	2	4.19 MHz
	8	4	2.10 MHz
	16	8	1.05 MHz
	34	17	493 kHz
	168	84	100 kHz
	510	255	33 kHz

4.8 Wired-OR Open-Drain Outputs

Typically, SPI bus outputs are not open-drain unless multiple SPI masters are in the system. If needed, the WOMP bit in SPCR can be set to provide wired-OR, open-drain outputs. An external pullup resistor should be used on each output line. WOMP affects all SPI pins regardless of whether they are assigned to the SPI or used as general-purpose I/O.

4.9 Transfer Size and Direction

The SIZE bit in the SPCR selects a transfer size of 8 (SIZE = 0) or 16 (SIZE = 1) bits. The LSBF bit in the SPCR determines whether serial shifting to and from the data register begins with the LSB (LSBF = 1) or MSB (LSBF = 0).

4.10 Write Collision

A write collision occurs if an attempt is made to write the SPDR while a transfer is in progress. Since the SPDR is not double buffered in the transmit direction, a successful write to SPDR would cause data to be written directly into the SPI shift register. Because this would corrupt any transfer in progress, a write collision error is generated instead. The transfer continues undisturbed, the data that caused the error is not written to the shifter, and the WCOL bit in SPSR is set. No SPI interrupt is generated.

A write collision is normally a slave error because a slave has no control over when a master initiates a transfer. Since a master is in control of the transfer, software can avoid a write collision error generated by the master. The SPI logic can, however, detect a write collision in a master as well as in a slave.

What constitutes a transfer in progress depends on the SPI configuration. For a master, a transfer starts when data is written to the SPDR and ends when SPIF is set. For a slave, the beginning and ending points of a transfer depend on the value of CPHA. When CPHA = 0, the transfer begins when $\overline{SS}$ is asserted and ends when it is negated. When CPHA = 1, a transfer begins at the edge of the first SCK cycle and ends when SPIF is set. Refer to **4.6 SPI Clock Phase and Polarity Controls** for more information on transfer periods and on avoiding write collision errors.

When a write collision occurs, the WCOL bit in the SPSR is set. To clear WCOL, read the SPSR while WCOL is set, and then either read the SPDR (either before or after SPIF is set) or write the SPDR after SPIF is set. (Writing the SPDR before SPIF is set results in a second write collision error.) This process clears SPIF as well as WCOL.

4.11 Mode Fault

When the SPI system is configured as a master and $\overline{SS}$ input line is asserted, a mode fault error occurs, and the MODF bit in the SPSR is set. Only an SPI master can experience a mode fault error, caused when a second SPI device becomes a master and selects this device as if it were a slave. To avoid latchup caused by contention between two pin drivers, the MCU does the following when it detects a mode fault error:

1. Forces the MSTR control bit to 0 to reconfigure the SPI as a slave
2. Forces the SPE control bit to 0 to disable the SPI system
3. Sets the MODF status flag and generates an SPI interrupt if SPIE = 1
4. Clears the appropriate bits in the MDDR to configure all SPI pins except the $\overline{SS}$ pin as inputs

After correcting the problems that led to the mode fault, clear MODF by reading the SPSR while MODF is set and then writing to the SPCR. Control bits SPE and MSTR may be restored to their original set state during this clearing sequence or after the MODF bit has been cleared. Hardware does not allow the user to set the SPE and MSTR bits while MODF is a logic 1 except during the proper clearing sequence.

4.12 SPI Registers

SPI registers are shown in **Table 4-4**. The addresses indicated are offsets from the MCCI base address.

Table 4-4. SPI Registers

Address	Name	Usage
$38	SPCR	SPI control register
$3A	—	Reserved
$3C	SPSR	SPI status register
$3E	SPDR	SPI data register

Software can read and write the SPCR and SPDR. Software can read the SPSR, but only the SPI hardware can write to this register. The SPCR must be initialized before the SPI is enabled to ensure defined operation. Reset values are shown at the bottom of each register diagram.

4.12.1 SPI Control Register

The SPI control register (SPCR) contains parameters for configuring the SPI. The register can be read or written at any time.

Register address: $XXXX38

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	WOMP	MSTR	CPOL	CPHA	LSBF	SIZE	BAUD							
Reset:	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Figure 4-4. SPI Control Register (SPCR)

SPIE — SPI Interrupt Enable Bit

0 = SPI interrupts disabled
1 = SPI interrupts enabled

SPE — SPI Enable Bit

To prevent unpredicted operation, the other SPI control fields should be configured properly before or at the same time SPE is set.

0 = SPI disabled
1 = SPI enabled

WOMP — Wired-OR Mode for SPI Pins

0 = Outputs have normal MOS drivers.
1 = Pins designated for output by MDDR have open-drain drivers, regardless if the pins are used as SPI outputs or for general-purpose I/O, and regardless if the SPI is enabled.

MSTR — Master/Slave Mode Select Bit

0 = SPI is a slave device.
1 = SPI is system master.

CPOL — Clock Polarity Bit

0 = The inactive state value of SCK is logic level 0.
1 = The inactive state value of SCK is logic level 1.

CPHA — Clock Phase Bit

0 = Data is captured on the leading edge of SCK and changed on the following edge of SCK.

1 = Data is changed on the leading edge of SCK and captured on the following edge of SCK.

LSBF — Least Significant Bit First

0 = Serial data transfer starts with MSB.

1 = Serial data transfer starts with LSB.

SIZE — Transfer Data Size Bit

0 = 8-bit data transfer

1 = 16-bit data transfer

BAUD — SPI Serial Clock Baud Rate Bit

The SPI baud rate is selected by writing a value from 2 to 255 into the BAUD field of the SPCR of the master MCU. Giving BAUD a value of 0 or 1 disables SCK. (The disabled state is determined by CPOL). At reset, BAUD is initialized so that SCK has a 2.1-MHz SCK frequency, given a 16.8-MHz system clock.

4.12.2 SPI Status Register

The SPI status register (SPSR) contains SPI status information. Only the SPI can set the bits in this register. The CPU reads the register to obtain status information.

Register address: $XXXX3C

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIF	WCOL	0	MODF	0	0	0	0	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-5. SPI Status Register (SPSR)

SPIF — SPI Finished Flag

Clearing SPIF is accomplished by reading the SPSR while SPIF is set and then writing to or reading the SPDR.

0 = Transfer not completed
1 = Transfer completed

WCOL — Write Collision Bit

Clearing WCOL is accomplished by reading the SPSR while WCOL is set and then either reading the SPDR prior to the SPIF bit being set or reading or writing the SPDR after the SPIF bit is set.

0 = No write collision occurred
1 = Write collision occurred

MODF — Mode Fault Flag

Clearing MODF is accomplished by reading the SPSR while MODF is set and then writing to the SPDR.

0 = Normal operation
1 = Another SPI node requested to become the network SPI master while the SPI was enabled in master mode ($\overline{SS}$ input taken low)

4.12.3 SPI Data Register

The SPI data register (SPDR) is used to transmit and receive data on the serial bus. A write to this register in the master device initiates transmission or reception of another byte or word. After a byte or word of data is transmitted, the SPIF status bit is set in both the master and slave devices.

A read of the SPDR actually reads a buffer. If the first SPIF is not cleared by the time a second transfer of data from the shift register to the read buffer is initiated, an overrun condition occurs. In cases of overrun, the byte or word causing the overrun is lost.

A write to the SPDR is not buffered and places data directly into the shift register for transmission.

Register address: $XXXX3E

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	UPPB								LOWB							
Reset:	Reset state is unknown															

Figure 4-6. SPI Data Register (SPDR)

UPPB — Upper Byte Bit

In a 16-bit transfer (SIZE = 1), the address of the upper byte is used to access the most significant eight bits of the data. Bit 15 of the SPDR is the MSB of the 16-bit data.

LOWB — Lower Byte Bit

In an 8-bit transfer (SIZE = 0), the data is accessed at the address of the lower byte. The MSB in an 8-bit transfer is bit 7 of the SPDR. In a 16-bit transfer (SIZE = 1), the lower byte holds the least significant eight bits of the data.

SPI Submodule

Section 5. SCI Submodule

5.1 Contents

5.2 Introduction

The SCI submodule contains two independent SCI systems. Each is a full-duplex universal asynchronous receiver transmitter (UART). This SCI system is fully compatible with SCI systems found on other Motorola devices, such as the M68HC11 and M68HC05 Families.

The SCI uses a standard non-return-to-zero (NRZ) transmission format. An on-chip baud-rate generator derives standard baud-rate frequencies from the MCU oscillator. Both the transmitter and the receiver are double buffered, so that back-to-back characters can be handled easily even if the CPU is delayed in responding to the completion of an individual character. The SCI transmitter and receiver are functionally independent but use the same data format and baud rate.

SCI operation can be polled by means of status flags in the SCI status register (SCSR), or interrupt-driven operation can be employed by means of the interrupt-enable bits in SCI control register 1 (SCCR1).

The two independent SCI systems are called SCIA and SCIB. These SCIs are identical in register set and hardware configuration, providing an application with full flexibility in using the dual SCI system. References to SCI registers in this section do not always distinguish between the two SCI systems. A reference to SCCR1, for example, applies to both SCCR1A (SCIA control register 1) and SCCR1B (SCIB control register 1).

Table 5-1 is a quick reference guide to all the bits and fields referred to in this section. All registers, bits, and fields referred to are discussed fully later in this section or in **Section 3. Configuration and Control.** Refer to **Table 1-1. MCCI Memory Map** (the MCCI memory map) for a list of the names, mnemonics, and address offsets of registers involved in SCI operation.

Table 5-1. SCI Bit/Field Quick Reference Guide

Mnemonic	Function	Register
BR	Baud rate	SCCR0
FE	Framing error flag	SCSR
IDLE	Idle line detected flag	SCSR
ILIE	Idle line interrupt enable	SCCR1
ILT	Idle line detect type	SCCR1
LOOPS	SCI loop mode	SCCR1
M	Mode select (8- or 9-bit)	SCCR1
NF	Noise error flag	SCSR
OR	Overrun error flag	SCSR
PE	Parity enable	SCCR1
PF	Parity error flag	SCSR
PT	Parity type	SCCR1
R[8:0]	Receive 8–0	SCDR
RAF	Receiver active flag	SCSR
RDRF	Receive data register full flag	SCSR
RE	Receiver enable	SCCR1
RIE	Receiver interrupt enable	SCCR1
RWU	Receiver wakeup	SCCR1
RXDA, RXDB	Receive data	MDDR, MPDR
SBK	Send break	SCCR1
T[8:0]	Transmit 8–0	SCDR
TC	Transmit complete flag	SCSR
TCIE	Transmit complete interrupt enable	SCCR1
TDRE	Transmit data register empty flag	SCSR
TE	Transmit enable	SCCR1
TIE	Transmit interrupt enable	SCCR1
TXDA, TXDB	Transmit data	MDDR, MPDR
WAKE	Wakeup type	SCCR1
WOMC	Wired-OR mode for SCI pins	SCCR1

5.3 SCI Pins

Four unidirectional pins are associated with the SCI. When the receiver or transmitter associated with a given pin is not enabled (for instance, when the RE or TE bit in the associated SCCR1 register is not set), the pin automatically reverts to its general-purpose I/O function. **Table 5-2** summarizes SCI pin function.

Table 5-2. SCI Pin Function

Pin Name	Mnemonic	SCI Function
Receive data	RXDA RXDB	Serial data input to SCIA receiver Serial data input to SCIB receiver
Transmit data	TXDA TXDB	Serial data output from SCIA transmitter Serial data output from SCIB transmitter

5.4 Serial Formats

Data can be transmitted and received in a number of formats. The following terms concerning data format are used in this section:

- Bit time — The time required to transmit or receive one bit of data; one cycle of the baud frequency
- Start bit — One bit time of logic 0 that indicates the beginning of a data frame. A start bit must begin with a 1-to-0 transition.
- Stop bit — One bit time of logic 1 that indicates the end of a data frame
- Frame — A complete unit of serial information. The SCI can use 10- or 11-bit frames.
- Data frame — A start bit, a specified number of data bits (one of which may be used as an address mark or parity bit or an extra stop bit) and a final stop bit
- Idle frame — A frame that consists of all 1s. An idle frame has no start bit.
- Break frame — A frame that consists of all 0s. A break frame has no stop bits.

- MSB — Most significant bit; in a data frame, the bit preceding the hardware-generated stop bit. Depending on the hardware and software configuration, the MSB (most significant bit) may represent a data bit, parity bit, address mark bit, or extra stop bit.

The SCI transmitter automatically provides a start bit as the first bit of each frame and a stop bit as the final bit. In addition, it generates a parity bit preceding the stop bit, if parity is enabled. The SCI receiver automatically strips the start and stop bits. If parity is enabled, the receiver strips the parity bit as well. Receiving and transmitting devices must use the same data frame format.

The SCI provides hardware support for both 10- and 11-bit frames. The most common 10-bit data frame format for NRZ serial interface is one start bit, eight data bits (LSB first), and one stop bit. The most common 11-bit data frame contains one start bit, eight data bits, a parity or address mark bit, and one stop bit.

The serial mode bit in SCCR1 specifies the number of bits per frame. The PE bit determines whether the MSB is a parity bit. (For transmitted data, a parity bit is generated; for received data, the parity bit is checked.) When parity is disabled, software can use the MSB as an address mark bit or a second stop bit. When parity is enabled, the address mark or second stop bit options are not available.

Table 5-3 lists the possible frame formats. Notice that hardware provides four frame formats, according to the values of M and PE. Within these four formats, software can provide additional variations.

Table 5-3. Data Frame Formats

M	PE	Frame Format
0	0	1 start bit, 8 data bits, 1 stop bit 1 start bit, 7 data bits, 1 address mark bit, 1 stop bit 1 start bit, 7 data bits, 2 stop bits
0	1	1 start bit, 7 data bits, 1 parity bit, 1 stop bit
1	0	1 start bit, 8 data bits, 1 address mark bit, 1 stop bit 1 start bit, 8 data bits, 2 stop bits
1	1	1 start bit, 8 data bits, 1 parity bit, 1 stop bit

5.5 Parity Checking

The parity type (PT) bit in SCCR1 selects even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The parity enable (PE) bit in SCCR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB in a frame is used for the parity function. For transmitted data, a parity bit is generated; for received data, the parity bit is checked. When parity checking is enabled, the parity flag (PF) in the SCI status register (SCSR) is set if a parity error is detected.

5.6 Baud Clock

The SCI baud clock is programmed by writing a 13-bit value to the baud rate (BR) field in SCI control register 0 (SCCR0). The baud clock is derived from the MCU system clock by a modulus counter. Writing a value of 0 to BR disables the baud rate generator.

Baud clock rate is:

SCI baud clock rate = system clock/(32 x BR)

where BR is in the range {1, 2, 3, . . . 8191}. As the preceding equation indicates, the baud rate depends on the values for both BR and the system clock. For a 16.78-MHz system clock, **Table 5-4** shows sample baud rates for different values of BR. The maximum baud rate with this system clock speed is 524 Kbaud.

The SCI receiver operates asynchronously. To synchronize with an incoming data stream, the SCI baud clock generator produces a receive time (RT) sampling clock with a frequency 16 times that of the SCI baud clock. The SCI determines the position of bit boundaries from transitions within the received waveform and adjusts sampling points to the proper positions within the bit period. Refer to **5.7.11 Frame Detection and Synchronization** for more information.

Table 5-4. Examples of SCI Baud Rates

Nominal Baud Rate	Actual Baud Rate	Percent Error	BR
500,000	524,288.00	4.86	1
38,400	37,499.14	–2.48	14
32,768	32,768.00	0.00	16
19,200	19,418.07	1.14	27
9,600	9,532.51	–0.70	55
4,800	4,809.98	0.21	109
2,400	2,404.99	0.21	218
1,200	1,199.74	–0.02	437
600	599.87	–0.02	874
300	299.94	–0.02	1,748
110	110.01	0.01	4,766
64	64.00	0.01	8,191

5.7 SCI Transmitter

The SCI transmitter consists of a transmit serial shifter and a parallel transmit data register (TDR) located in the SCDR. The transmitter is double buffered: One byte can be loaded into the TDR while another character is being shifted out from the serial shifter to the TXD pin.

Transmitter logic adds a start bit (logic 0) and a stop bit (logic 1) to the data characters presented by the CPU for transmission. The transmitter can be configured to send characters with eight (M = 0) or nine (M = 1) data bits. When the TDR is able to accept a new data character, the TDRE status flag is set, and an interrupt can optionally be generated. Another status flag (TC) and optional interrupt are produced when the transmitter has finished sending everything in its queue. In addition to data characters, the transmitter is capable of sending idle-line characters and break characters, which are useful in multidrop SCI networks. Two characters can normally be in the transmit queue, but three characters can be queued when at least one of them is an idle-line or break character.

Figure 5-1 is a block diagram of the SCI transmitter.

SCI Submodule

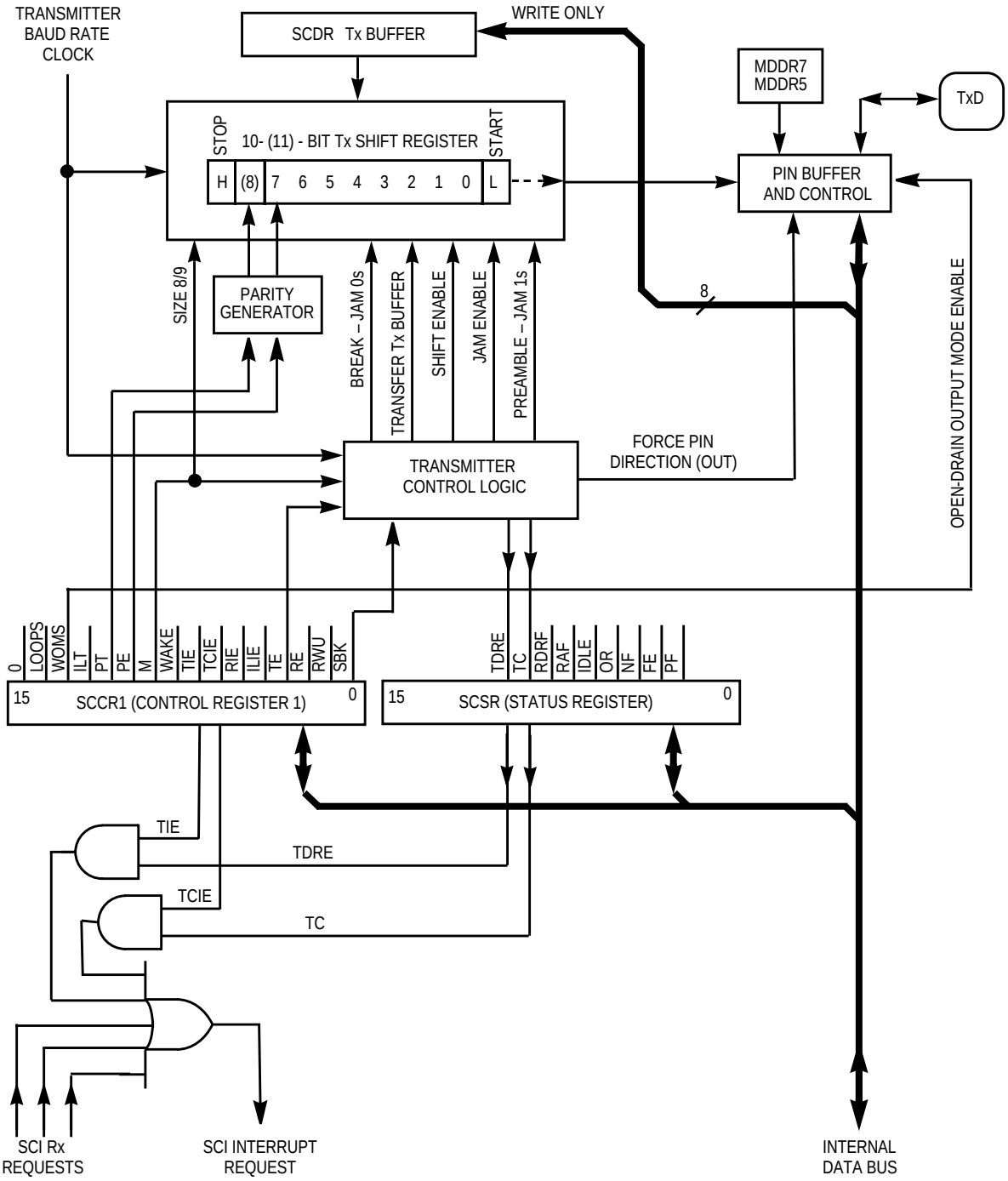


Figure 5-1. SCI Transmitter Block Diagram

5.7.1 Initializing the Transmitter

A general outline for initializing the MCCI is provided in **3.4 System Initialization**. When using the SCI transmitter, include these specific steps:

1. Write to the MCCI global registers as outlined in **3.4 System Initialization**. Configure the TXD pin as an output in the MDDR.
2. Read the SCI status register (SCSR), then write to the SCI data register (SCDR). This clears the transmit data register empty (TDRE) and transmit complete (TC) indicators in the SCSR, preventing these flags from generating interrupts as soon as the transmitter is enabled.
3. Write a baud rate value into the BR field of SCCR0.
4. Write to SCCR1 to set or clear all appropriate bits. Select an 8- or 9-bit frame format (M) and determine use (PE) and type (PT) of parity check. Set the transmit enable (TE) and transmit interrupt enable (TIE) bits. Enable or disable transmit-complete interrupts (TCIE) and wired-OR operation (WOMC). Disable the send-break (SBK) function for normal operation.
5. Write the byte to be transmitted to the SCDR.

When TE changes from 0 to 1, the MCU checks the state of the serial shifter. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

To transmit subsequent bytes, wait for TDRE to be set, then write to the SCDR. SCI supports both polling and interrupt-driven transmission, as described in the following subsection.

5.7.2 Status Flags and Interrupts

Two status flags are associated with the SCI transmitter. These flags are read (polled) by software to tell when the corresponding condition exists. Alternately, an interrupt-enable bit can be set to enable each of these status conditions to generate interrupt requests when the corresponding condition is present. Status flags are automatically set by hardware logic conditions, but software clears these flags, providing an interlock mechanism for logic to know when software has noticed the status indication. The software clearing sequence for these flags consists of steps that normally are performed in response to the flags.

When the transmitter is first enabled, the TDRE and TC flags are normally already set. To prevent an immediate interrupt from occurring from these sources, read the SCSR and then write to the SCDR before enabling the transmitter. This procedure clears the TDRE and TC flags.

The TDRE flag indicates that there is room in the transmit queue to store another data character in the TDR. The TIE bit is the interrupt-enable bit for TDRE. When TIE equals 0, TDRE must be polled; when TIE equals 1, an interrupt is requested whenever TDRE is set.

The TC flag indicates that the transmitter has finished transmitting everything in its queue, including any idle preamble or break character that has been queued. The TCIE bit is the interrupt-enable bit for the TC. When TCIE equals 0, TC must be polled; when TCIE equals 1, an interrupt is requested whenever TC is set.

The TC bit is useful in systems in which the SCI is driving a modem. When TC is set at the end of a transmission, the modem can be disabled. In older SCI systems, the TDRE status bit was the only indication that a transmission was near completion. Since TDRE only indicated that the last character had transferred to the transmit shift register, software had to delay an amount of time greater than or equal to the time it took for this last character to finish transmitting serially. Since the delay time depended on the baud rate, it was difficult to know when it was safe to disable the modem. The TC bit offers a more convenient way to tell when the transmitter has completed the transmission.

One interrupt vector is associated with each SCI subsystem; therefore, the interrupt service routine must begin by reading the SCSR to determine which interrupt or interrupts caused the service routine to be called. Possible interrupt sources include the two transmitter sources previously discussed and two receiver-related sources.

5.7.3 Disabling the SCI Transmitter

After loading the last byte into the TDR and receiving the interrupt from TDRE in the SCSR (indicating that the data has transferred into the transmit serial shifter), clear TE to disable the transmitter. When TE is cleared, the transmitter is disabled after all pending idle, data, and break frames are transmitted. The TC flag is set, and the TXD pin reverts to use for general-purpose I/O. Buffered data is not transmitted after TE is cleared. To avoid terminating transmission with data in the buffer, do not clear TE until TDRE is set.

If TE remains set after all pending idle, data, and break frames are shifted out, TDRE and TC are set and TXD is held at logic level 1.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. In this case, configure the TXD pin as an output by writing to the MDDR, and then write a 1 to PMC7 (for SCIA) or PMC5 (for SCIB) in PORTMC, the data register for the MCCI general-purpose I/O port. When the transmitter releases control of the TXD pin, it reverts to driving a logic 1 output.

5.7.4 Break Characters

Break characters are streams of 10 or 11 logic 0s (depending on the M bit in SCCR1). Break characters have no start or stop bits. As long as the SBK bit in SCCR1 is set, break characters are queued and sent: The TXD line remains continuously at 0. When SBK is cleared, at least one bit time of logic 1 appears on the TXD line as soon as the last break character is finished. This high bit time ensures that a receiver can detect the falling edge at the beginning of the start bit for the next data character.

If the transmitter is transmitting a character when software toggles SBK on and off, exactly one break character is produced after the current character is transmitted. If the transmitter is idle when SBK is toggled, it is uncertain whether one or two break characters will be sent. When SBK is set to 1, a break character is queued. When the transmit shift register becomes available and synchronization requirements are met with respect to the internal baud clock, the queued break character is jammed into the shift register to be serially sent and, if the SBK bit is still 1, another break is queued. The transfer mechanism from the queue to the shifter is internally synchronized to the baud clock; however, the relationship of this clock to operating software is not normally known. The instructions to write 1 and then write 0 to the SBK bit execute very quickly relative to the normal baud-rate frequency, but there is still a small chance that the baud-rate clock edge could occur between writing the 1 and writing the 0 to SBK.

5.7.5 Queued Idle Characters

Idle characters are streams of 10 or 11 bits of logic 1 (depending on the M bit in SCCR1). They are produced when the transmitter is enabled from a disabled state (for instance, when TE is changed from 0 to 1). When the transmitter is enabled, this idle character acts as a preamble. The character-length period of logic 1 ensures that any receiver connected to this transmitter will be resynchronized so that it can properly recognize the leading edge of the start bit for the next character.

Software can queue an idle character into a serial data stream by clearing and then setting TE. This queueing function is useful when using the idle-line variation of receiver wakeup (WAKE = 0 in SCCR1). In a multidrop SCI network, all receivers evaluate the first character or characters of a message to decide whether or not this message is important to this receiver. If not, receiver wakeup is invoked in the receiver by writing a 1 to the RWU bit in SCCR1. A 1 in RWU causes the receiver to ignore any other characters in the message, thus allowing the MCU to perform more useful functions than responding to interrupts from the SCI. The SCI receiver still monitors characters normally but does not set status flags or generate interrupts. When idle-line wakeup is used, the SCI receiver logic automatically clears RWU (waking up the receiver)

when it sees a full character time of logic 1. During a message, there must never be any gap between characters within a message because even a single bit time of idle can trigger wakeup if the previous character was $FF. The queued idle function allows exactly one character time of idle to be inserted into the data stream to maintain maximum efficiency and data throughput. Before queued idle was available, software had to avoid writing to the TDR for two or more character times after seeing TDRE go high, which caused the TXD line to go idle for enough time to trigger RWU.

To queue an idle character, write the last character to the TDR and wait for TDRE to become set. (This indicates that the last character has transferred to the transmit shifter to be transmitted serially.) Write 0 and then 1 to TE. Since the last character is still being transmitted, the transmitter will not give up control of the TXD pin, and the character being transmitted is undisturbed. The 0-to-1 transition of TE queues the idle character to be sent as soon as the transmit shifter becomes available. As soon as TE is written back to 1, the first character of the next message can be written to the TDR. In this unusual case, the transmit queue can be three characters deep: the last data character of the previous message still transmitting, the queued idle character, and the first character of the next message in the parallel TDR.

Since an idle character is queued at the rising edge of the setting of the TE bit, exactly one idle character results from the queuing procedure. There is never any possibility of a second idle character being produced due to uncertainty about the relationship between the software and the internal baud-rate clock (as there is with queued break characters).

5.7.6 Wired-OR Open-Drain Outputs

When more than one transmitter is used on the same SCI bus, set the WOMC bit in SCCR1 to select wired-OR open-drain operation for the TXD pin. The WOMC bit also controls the RXD pin when it is configured as an output. Use external pullup resistors on these pins for wired-OR operation.

WOMC controls TXD function regardless of whether the pin is used for SCI transmissions (TE = 1) or as a general-purpose I/O pin (TE = 0). Clearing this bit causes TXD and RXD to function as normal CMOS outputs. WOMC has no effect on TXD or RXD pins that are configured as general-purpose inputs.

5.7.7 SCI Receiver

Each SCI receiver contains a receive serial shifter and a parallel receive data register (RDR) located in the SCI data register (SCDR). The receiver is double buffered: While one character is shifted into the receive serial shift register, another character can be held in the RDR. This double-buffered arrangement gives software some time to notice a received character and read it before the next serial character is finished. Without double buffering, the transmitting device would be required to insert delays between transmitted characters to avoid a receiver overrun.

The receiver enable (RE) bit in SCCR1 enables the receiver. The M bit in SCCR1 determines frame size (10 or 11 bits). After a stop bit is detected, the received data is transferred from the shifter to the SCDR, and the receive data register full (RDRF) status flag is set. When a character is ready to be transferred to the receive buffer but the previous character has not yet been read, an overrun condition results. When this occurs, data is not transferred and the overrun (OR) status flag is set to indicate the error.

The wakeup block uses the WAKE control bit in SCCR1 to decide whether to use address mark or the all 1s signal (idle line) to wake up the receiver. When the selected condition is detected, the wakeup logic clears the receiver wakeup (RWU) bit in SCCR1, waking up the receiver.

The SCSR contains two receiver-related flags that can be polled by software or optionally cause an SCI interrupt request. The receiver interrupt enable (RIE) control bit enables the RDRF status flag to generate hardware interrupt requests. The idle line interrupt enable (ILIE) control bit allows the IDLE status flag to generate SCI interrupt requests.

The input of the receive serial shifter is connected to the sampling logic of the receiver bit processor. The receiver bit processor logic drives a state machine that determines the logic level for each bit time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter. Once data is shifted into the receive serial shifter, it is moved synchronously with the MCU system clock. **5.7.11 Frame Detection and Synchronization** explains the process in greater detail.

Figure 5-2 is a block diagram of the SCI receiver.

5.7.8 Initializing the Receiver

A general outline for initializing the MCCI is provided in **3.4 System Initialization.** To initialize the SCI receiver, include these specific steps:

1. Write to the MCCI global registers as outlined in **3.4 System Initialization**. Configure the RXD pin as an input in the MDDR.
2. Write a baud rate value into the BR field of SCCR0.
3. Write to SCCR1 to set or clear all appropriate bits. Select 8- or 9-bit frame format (M) and use (PE) and type (PT) of parity check. Select use (RWU) and type (WAKE) of receiver wakeup. Select idle-line detection type (ILT) and enable or disable idle-line interrupt (ILIE). For most applications, set receiver interrupt enable (RIE). Enable the receiver by setting the receiver enable (RE) bit.

The RDRF flag in the SCSR is set as each bit is received and transferred into the RDR. RDRF must be cleared before the next character is transferred into the RDR. To clear RDRF, read the SCSR and then read the SCDR.

SCI Submodule

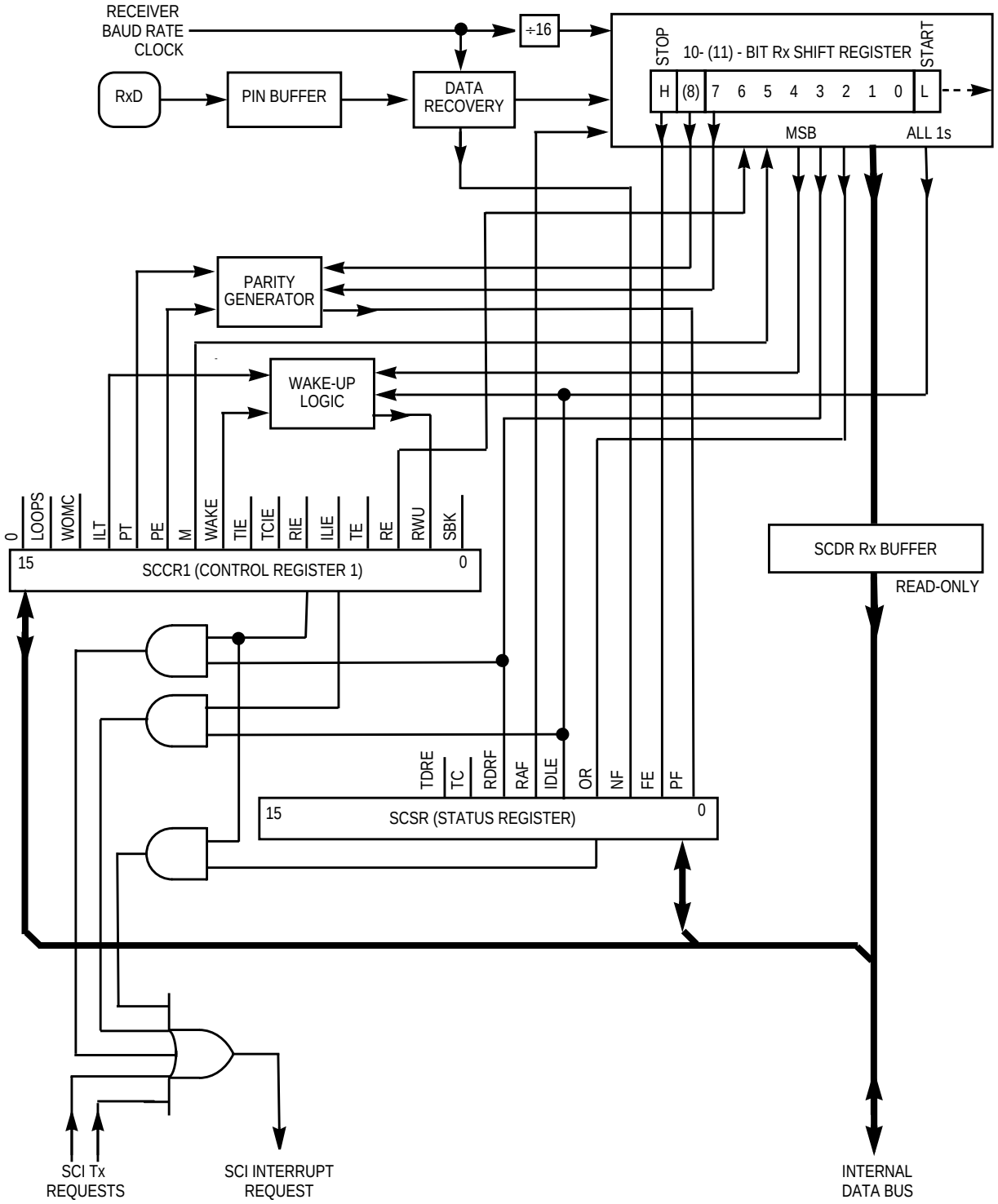


Figure 5-2. SCI Receiver Block Diagram

5.7.9 Receiver Status Flags and Interrupts

Six status flags are associated with the SCI receiver. RDRF is set when a character has been received and transferred into the parallel RDR. The OR flag is set if RDRF is already set when a new character is ready to be transferred into the parallel RDR. The PF, NF, and FE flags alert the user to parity, noise, and framing error conditions, respectively. The IDLE flag is set when the MCU detects an idle line condition (a frame of all 1s).

All status flags associated with a serially received frame are set simultaneously. When a completed frame is received, either the RDRF or OR flag is always set. The receiver status flags NF, FE, and PF are set simultaneously with RDRF as appropriate. (Although error conditions are detected as bits are received, these flags are not set until data is transferred from the serial shifter to RDR.)

Status flags can be polled at any time by software. RDRF and IDLE can optionally generate an automatic interrupt request. Because NF, FE, and PF are set at the same time as RDRF, they do not have separate interrupt enables.

To clear all receiver status flags, read the SCSR and then read the SCDR.

5.7.9.1 RDR Full Flag (RDRF)

RDRF is set when a character has been received and transferred into the parallel RDR. RDRF optionally can generate an automatic interrupt request. If RIE equals 1, an interrupt is generated whenever RDRF changes from 0 to 1. If RIE equals 0, RDRF does not generate interrupts, and the receiver must be polled.

Although an interrupt is not explicitly associated with NF, FE, or PF, an interrupt can be generated with RDRF and the error flags checked in this manner.

5.7.9.2 Noise Flag (NF)

NF is set when the SCI receiver detects noise on a valid start bit, on any of the data bits, or on the stop bit. It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times for noise. If the three samples are not at the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until the entire frame is received and RDRF is set. Refer to **5.7.11 Frame Detection and Synchronization** for more information on noise detection.

5.7.9.3 Framing Error Flag (FE)

FE is set when the SCI receiver detects a 0 where a stop bit (one) is expected. A framing error results when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. FE is not set until the entire frame is received and RDRF is set. A break can also cause FE to be set.

Framing errors are not always detected: If a data bit in the expected stop bit time happens to be a logic 1, the FE flag is not set.

5.7.9.4 Overrun Flag (OR)

OR is set when a new byte is ready to be transferred from the receive serial shifter to register RDR, and RDR is already full (RDRF is still set). In an overrun condition, the character that caused the overrun is lost, but the previously received character in the SCDR is not disturbed. Since framing, noise, and parity errors are detected while data is in the serial shifter, FE, NF, and PF cannot occur simultaneously with OR.

5.7.9.5 Parity Error Flag (PF)

PF is set when the SCI receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set. Although an interrupt is not explicitly associated with PF, an interrupt may be generated with RDRF and PF checked in this manner.

5.7.9.6 Idle Line Flag (IDLE)

During normal serial transmission, no idle time occurs between frames. Even when all the data bits in a frame are logic 1s, the start bit provides one bit time of logic 0 during the frame. An idle line, in contrast, is a sequence of 1s equal to the current frame size. (Frame size is determined by the state of the M bit in SCCR1.)

The receiver hardware can detect an idle line. This function can be used to indicate when a group of serial transmissions is finished. Idle line detection is always enabled. When an idle line condition is detected, the IDLE flag in SCSR is set.

The SCI receiver has both short and long idle-line detection capability. The idle line type (ILT) bit in SCCR1 determines which type of detection is used. For short idle-line detection, the receiver bit processor counts contiguous logic 1 bit times whenever they occur. Short detection provides the earliest possible recognition of an idle line condition, because the stop bit and contiguous logic 1s before and after it are counted. Long idle-line detection starts counting idle time only after a valid stop bit is received. Only a complete idle frame results in the detection of an idle line.

When idle-line receiver wakeup is used (see **5.7.10 Receiver Wakeup**), long idle-line detection prevents the receiver from being awakened prematurely if the message preceding the start of the idle line contained 1s in advance of its stop bit. (Since receiver status flags, however, cannot be set when RWU = 1, the IDLE flag cannot be used with receiver wakeup.)

In addition, in some applications, CPU overhead can cause a bit time of logic level 1 to occur between frames. This bit time does not affect content, but if it occurs after a frame of 1s when short detection is enabled, the receiver flags an idle line.

When the idle-line interrupt enable (ILIE) bit in SCCR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading the SCSR and SCDR in sequence. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

5.7.10 Receiver Wakeup

The receiver wakeup function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Receivers not addressed become dormant for the remainder of the message, eliminating any further software overhead to service the remaining characters of the message.

Software places the receiver in wakeup mode by setting the receiver wakeup (RWU) bit in SCCR1. While RWU is set, receiver status flags cannot be set. For this reason idle-line detection cannot be used with receiver wakeup. Hardware clears RWU using one of two methods: idle-line wakeup or address-mark wakeup. (It is possible to clear RWU with software, but this is not the normal procedure.)

The WAKE bit in SCCR1 determines which type of wakeup is used. When WAKE = 0, idle-line wakeup is selected. When WAKE = 1, address-mark wakeup is selected. Both types require a software-based device addressing and recognition scheme.

5.7.10.1 Idle-Line Wakeup

Idle-line wakeup allows a receiver to sleep until an idle line is detected. When an idle-line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The byte is received normally and transferred to the RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep until the next idle line. This method of receiver wakeup requires a minimum of one idle-line frame time between messages and allows no idle time between frames in a message. The ILT bit determines whether short or long idle-line detection is invoked.

5.7.10.2 Address-Mark Wakeup

Address-mark wakeup uses a special frame format. The first frame of each transmission must be an address frame, indicated by a logic level 1 in the MSB. For all other characters, the MSB must equal 0.

When the MSB of a frame is set, the receiver clears RWU and wakes up. The byte is received normally and is transferred to the RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address-mark wakeup allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency due to an additional bit time per frame.

5.7.11 Frame Detection and Synchronization

After RE is set, bit processor logic begins to sample the signal on the RXD pin. The sampling process identifies a valid start bit and synchronizes the receiver with the incoming frame. A receive time (RT) clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the RT clock with the incoming data stream. From this point on, the data movement is synchronized with the MCU system clock. Understanding this process can be useful in determining the amount of baud-rate frequency mismatch that can be tolerated. It also indicates how well this SCI receiver can handle noise.

RT clock rate is 16 times the baud clock rate. Each bit time of a received frame is divided into 16 sample periods designated RT1–RT16. Designations are assigned relative to the time a start bit is detected. The receiver active flag (RAF) in SCSR is set when a valid start bit is identified.

Synchronicity is maintained throughout the frame. Sampling continues and the receiver resynchronizes on each 1-to-0 transition in the frame. Bit time normally begins at RT1 and ends at RT16. After the frame ends, the receiver begins the process of identifying the next start bit.

Synchronization at the beginning of each incoming frame eliminates cumulative timing errors due to small differences between the baud rates of the receiver and the transmitter. Synchronization on each 1-to-0 transition in the frame increases tolerance to small frequency variations in the received data stream.

5.7.11.1 Start Bit Detection

When the receiver is first enabled or when a stop bit is received at the end of a frame, an asynchronous search is initiated to find the leading edge of the next start bit. The goal of this asynchronous search is to gain bit-time synchronization between the serial data stream and the internal RT clock. Once synchronization has been established, the RT clock controls where the MCU perceives the bit-time boundaries to be.

The first step in locating a start bit is to find a sample where RXD is 0 preceded by three consecutive samples of logic 1. These four samples are called start-bit qualifiers. Until the start-bit qualifiers are detected, the RT clock is reset to state RT1 after each sample. Once the qualifiers are found, the beginning of a start bit is tentatively assumed, and subsequent samples are assigned successive RT state numbers. Next, start-bit verification samples are taken at RT3, RT5, and RT7. If any two of the three verification samples are logic 1s, the low at the start-bit qualifiers and the start-bit verification requirements are met, synchronization has been achieved, and the RT count state is used to determine the position of bit-time boundaries.

5.7.11.2 Logic Level Detection

Data samples are taken at RT8, RT9, and RT10 of each bit time, including start and stop bit times, to determine the logic level of the bit time and to set a working noise flag, if necessary. The logic level of the bit time is considered to be the majority of all samples taken during bit time. Even if the samples at RT8, RT9, and RT10 suggest it should be 1, however, the logic level of the start bit is always assumed to be 0.

If there is any disagreement among the samples taken during any bit time in a frame (including the start and stop bit times), the internal noise flag is set. At the end of a character reception, data is transferred from the receive shifter to the parallel RDR, and the RDRF flag is set. If noise was detected during reception of the character, the NF bit in the SCSR is set at the same time as RDRF.

5.7.11.3 Start Bit Recognition Examples

The operation of the receiver bit processor is shown in the following figures. These examples demonstrate the search for a valid start bit and the synchronization procedure as outlined. The possibility of noise durations greater than one bit time are not considered in these examples. **Figure 5-3** illustrates the ideal case with no noise present.

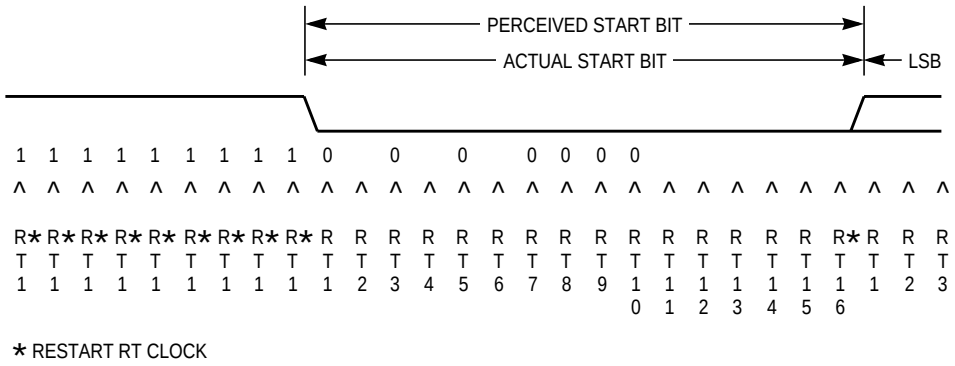


Figure 5-3. Start Search Example 1

Figure 5-4 shows the start bit search and resynchronization process being restarted because the first low detected was determined to be noise rather than the beginning of a start bit time. Since the noise occurred before the start bit was found, it will not cause the internal noise flag to be set.

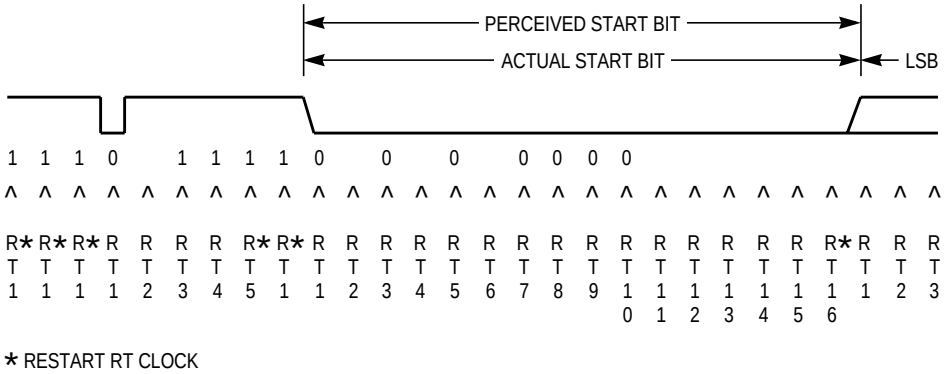


Figure 5-4. Start Search Example 2

Figure 5-5 shows that noise is perceived as the beginning of a start bit. Note that the high level sensed at RT3 causes the internal noise flag to be set. Even though this figure shows improper alignment of the perceived bit-time boundaries to the actual bit-time boundaries, the logic sense samples taken at RT8, RT9, and RT10 fall well within the correct actual bit time. The start bit and all other bits in the frame should be received correctly.

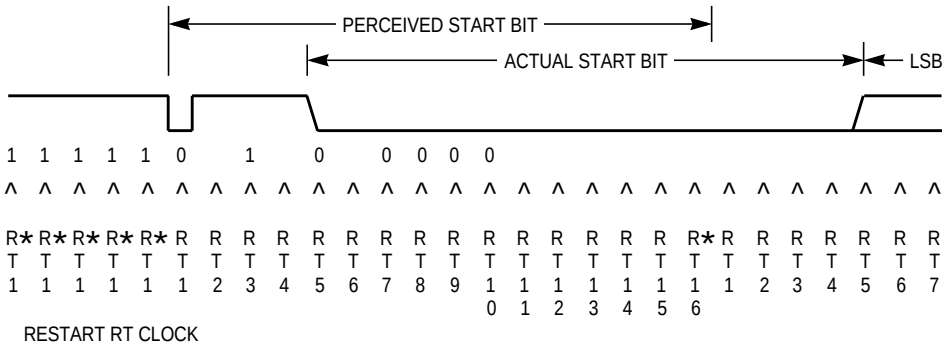


Figure 5-5. Start Search Example 3

Figure 5-6 shows how a large burst of noise is perceived as the beginning of a start bit. Note that RT5 is sensed as logic 1, setting the internal noise flag. This figure also illustrates a worst-case alignment of the perceived bit-time boundaries to the actual bit-time boundaries; however, RT8, RT9, and RT10 all fall within the correct actual bit time. The start bit is detected and the incoming data stream is correctly sensed.

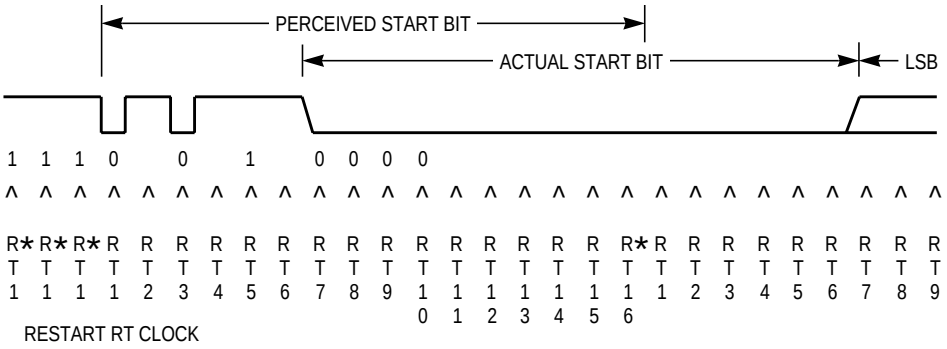


Figure 5-6. Start Search Example 4

Figure 5-7 illustrates the effect of noise early within the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the internal noise flag.

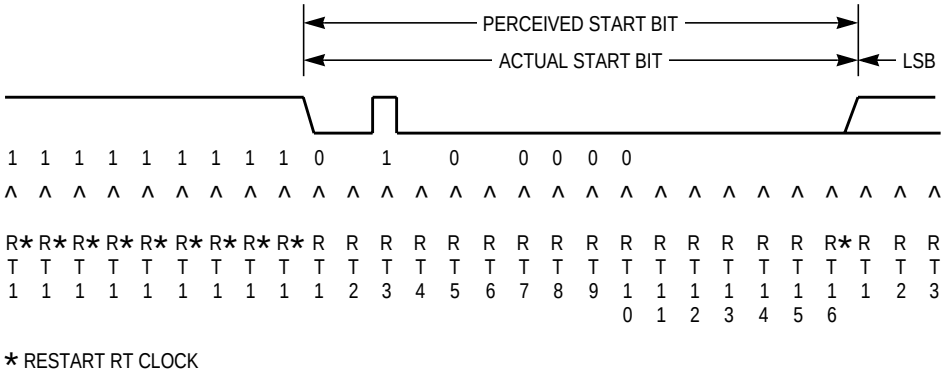


Figure 5-7. Start Search Example 5

Figure 5-8 shows a large burst of noise near the beginning of the start bit that causes the start bit search to be restarted. During RT1 following RT7, a search for a new start bit could not be started as the previous three RT samples are not all high. The receiver bit processor misses this start bit. The frame might be partially received or missed entirely, depending on the data in the frame and when the start bit search logic synchronized upon what appeared to be a start bit. If a valid stop bit is not detected, the FE flag is set in SCSR.

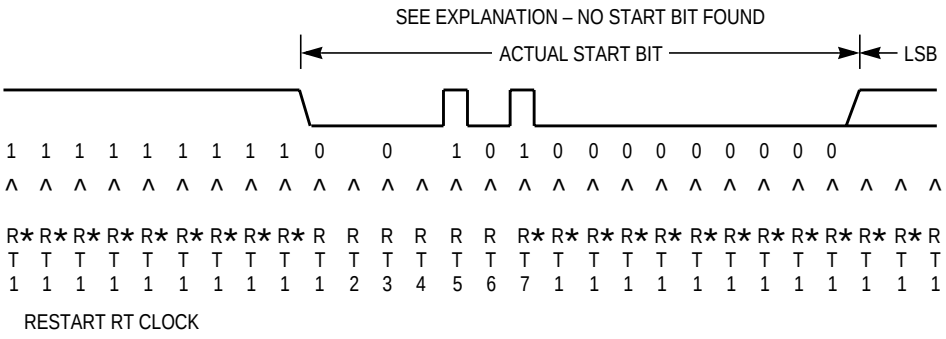


Figure 5-8. Start Search Example 6

Figure 5-9 depicts the case in which the majority of RT8, RT9, and RT10 return a logic level 1. However, the start bit is a special case in which the majority voting scheme does not apply. In review, at least three of the samples taken at RT1, RT3, RT5, and RT7 must be low. The start bit is detected and the RT clock is synchronized. Because RT8–RT10 were not unanimous, NF is set.

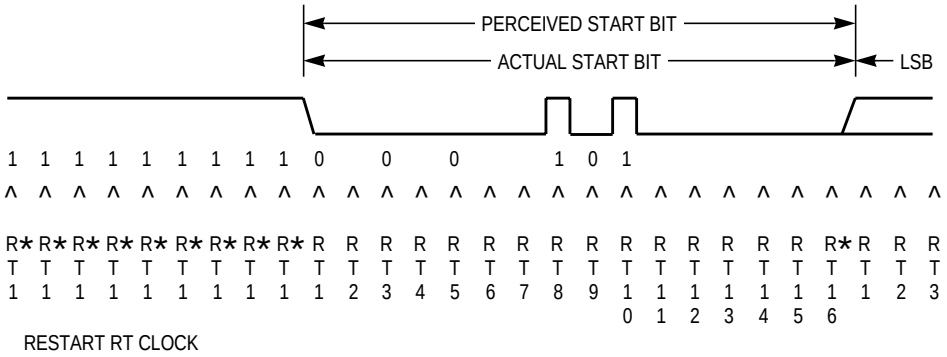


Figure 5-9. Start Search Example 7

5.8 SCI Registers

The SCI programming model includes the MCCI global and pin control registers and eight SCI registers. Each of the two SCI units contains two SCI control registers, one status register, and one data register. The SCI registers are listed in Table 5-5. The addresses are offsets from the base address.

All registers may be read or written at any time by the CPU. Rewriting the same value to any SCI register does not disrupt operation; however, writing a different value into an SCI register when the SCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in the SCSR may be cleared at any time.

Table 5-5. SCI Registers

Address	Name	Usage
$18	SCCR0A	SCIA control register 0
$1A	SCCR1A	SCIA control register 1
$1C	SCSRA	SCIA status register
$1E	SCDRA	SCIA data register Transmit data register (TDR)* Receive data register (RDR)*
$20–$27	—	Reserved
$28	SCCR0B	SCIB control register 0
$2A	SCCR1B	SCIB control register 1
$2C	SCSRB	SCIB status register
$2E	SCDRB	SCIB data register Transmit data register (TDR)* Receive data register (RDR)*
$30		Reserved

*Reads access the RDR; writes access the TDR.

When initializing the SCI, set the transmitter enable (TE) and receiver enable (RE) bits in SCCR1 last. A single word write to SCCR1 can be used to initialize the SCI and enable the transmitter and receiver.

5.8.1 SCI Control Register 0

SCCRO contains the baud rate selection field. The baud rate must be set before the SCI is enabled. The CPU can read and write this register at any time.

Register name and address: SCI Control Register 0 (SCCR0A), $XXXX18
SCI Control Register 0 (SCCR0B), $XXXX28

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: / Write:				BR												
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

= Unimplemented

Figure 5-10. SCI Control Register 0 (SCCR0A and SCCR0B)

SCCR0[15:13] — Not Implemented

BR — Baud Rate Field

The SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of 0 to BR disables the baud rate generator. Refer to **5.6 Baud Clock** for more information on setting the baud rate.

5.8.2 SCI Control Register 1

SCCR1 contains a number of SCI configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read and write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Register name and address: SCI Control Register 1 (SCCR1A), $XXXX1A
SCI Control Register 1 (SCCR1B), $XXXX2A

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: / Write:		LOOPS	WOMC	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented

Figure 5-11. SCI Control Register 1 (SCCR1A and SCCR1B)

Bit 15 — Not Implemented

LOOPS — Loop Mode Bit

The LOOPS bit in SCCR1 controls a feedback path on the data serial shifter. When LOOPS is set, SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled prior to entering loop mode.

0 = Normal SCI operation, no looping, feedback path disabled
1 = Test SCI operation, looping, feedback path enabled

WOMC — Wired-OR Mode for SCI Pins

0 = If configured as an output, TXD is a normal CMOS output.
1 = If configured as an output, TXD is an open-drain output.

ILT — Idle-Line Detect Type Bit

0 = Short idle-line detect (start count on first 1)
1 = Long idle-line detect (start count on first 1 after stop bit(s))

PT — Parity Type Bit

0 = Even parity
1 = Odd parity

PE — Parity Enable Bit
- 0 = SCI parity disabled
- 1 = SCI parity enabled

M — Mode Select Bit
- 0 = 10-bit frame
- 1 = 11-bit frame

WAKE — Wakeup by Address Mark Bit
- 0 = Receiver awakened by idle-line detection
- 1 = Receiver awakened by address mark

TIE — Transmit Interrupt Enable Bit
- 0 = TDRE interrupts inhibited
- 1 = TDRE interrupts enabled

TCIE — Transmit Complete Interrupt Enable Bit
- 0 = TC interrupts inhibited
- 1 = TC interrupts enabled

RIE — Receiver Interrupt Enable Bit
- 0 = RDRF interrupts inhibited
- 1 = RDRF interrupts enabled

ILIE — Idle-Line Interrupt Enable Bit
- 0 = IDLE interrupts inhibited
- 1 = IDLE interrupts enabled

TE — Transmitter Enable Bit
- 0 = Transmitter disabled (TXD pin may be used as I/O.)
- 0 = Transmitter enabled (TXD pin dedicated to SCI transmitter)

RE — Receiver Enable Bit
- 0 = Receiver disabled (status bits inhibited)
- 1 = Receiver enabled

RWU — Receiver Wakeup Bit
- 0 = Normal receiver operation (received data recognized)
- 1 = Wakeup mode enabled (received data ignored until awakened)

SBK — Send Break Bit
- 0 = Normal operation
- 1 = Break frame(s) transmitted after completion of current frame

5.8.3 SCI Status Register

The SCSR contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a read/write sequence. To clear SCI transmitter flags, read the SCSR and then write to the SCDR. To clear SCI receiver flags, read the SCSR and then read the SCDR. A long-word read can consecutively access both the SCSR and the SCDR. This action clears receiver status flag bits that were set at the time of the read, but does not clear TDRE or TC flags.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits, but before the CPU has written or read the SCDR, the newly set status bit is not cleared. The SCSR must be read again with the bit set, and the SCDR must be written to or read before the status bit is cleared.

Reading either byte of the SCSR causes all 16 bits to be accessed, and any status bit already set in either byte will be cleared on a subsequent read or write of the SCDR.

Register name and address: SCI Status Register (SCSRA), $XXXX1C
SCI Status Register (SCSRB), $XXXX2C

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:								TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
Write:																
Reset:	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

= Unimplemented

Figure 5-12. SCI Status Registers (SCSRA and SCSRB)

Bits 15–9 — Not implemented

TDRE — Transmit Data Register Empty Flag
- 0 = The TDR still contains data to be sent to the transmit serial shifter.
- 1 = A new character may now be written to register TDR.

TC — Transmit Complete Flag
- 0 = SCI transmitter is busy.
- 1 = SCI transmitter is idle.

0RDRF — Receive Data Register Full Flag

0 = The RDR is empty or contains previously read data.

1 = The RDR contains new data.

RAF — Receiver Active Flag

0 = SCI receiver is idle.

1 = SCI receiver is busy.

IDLE — Idle-Line Detected Flag

Under certain conditions, the IDLE flag may be set immediately following the negation of RE (SCCR1). System designs should ensure this causes no detrimental effects.

0 = SCI receiver did not detect an idle-line condition.

1 = SCI receiver detected an idle-line condition.

OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.

1 = RDRF is not cleared before new data arrives.

NF — Noise Error Flag

0 = No noise detected on the received data

1 = Noise occurred on the received data.

FE — Framing Error Flag

0 = No framing error on the received data

1 = Framing error or break occurred on the received data.

PF — Parity Error Flag

0 = No parity error on the received data

1 = Parity error occurred on the received data.

5.8.4 SCI Data Register

The SCDR contains two data registers at the same address. The RDR is a read-only register that contains data received by the SCI serial interface. The data comes into the receive serial shifter and is transferred to the RDR. The TDR is a write-only register that contains data to be transmitted. The data is first written to the TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission.

Register name and address: SCI Data Register (SCDRA), $XXXX1E
SCI Data Register (SCDRB), $XXXX2F

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
Write:																
Reset:	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

= Unimplemented U = Unaffected

Figure 5-13. SCI Data Registers (SCDRA and SCDRB)

R[8:0] — Receive[8:0] Bits

R8 is the MSB when the SCI system is configured for 11-bit frames (M = 1). If used, R8 represents the address mark, an extra stop bit, or the parity mark. When the SCI system is configured for 10-bit frames (M = 0), this bit has no meaning or effect. R7 is then the MSB before the stop bit and may represent either data, a parity bit, a second stop bit, or an address mark. R[6:0] represent data bits.

T[8:0] — Transmit[8:0] Bits

T8 is the MSB when the SCI system is configured for 11-bit frames (M = 1). When used, T8 represents either the address mark, an extra stop bit, or the parity mark. When the SCI system is configured for 10-bit frames (M = 0), this bit has no meaning or effect. T7 is then the MSB and may represent either data, a parity bit, an extra stop bit, or an address mark. T[6:0] represent data bits.

SCI Submodule

Appendix A. Electrical Characteristics

Table A-1. Serial Peripheral Interface Timing Characteristics

No,	Characteristic[1]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{op(m)}$ $f_{op(s)}$	 dc dc	 1/4 1/4	 System clock frequency
1	Cycle time Master Slave	 $t_{cyc(m)}$ $t_{cyc(s)}$	 4 4	 510 —	 t_{cyc}
2	Slave enable lead time	t_{lead}	2	—	t_{cyc}
3	Slave enable lag time	t_{lag}	2	—	t_{cyc}
4	Clock (SCK) high or low time Master Slave	 $t_{sw(m)}$ $t_{sw(s)}$	 $2\ t_{cyc} - 60$ $2\ t_{cyc} - N$[2]	 $255\ t_{cyc}$ —	 ns
6	Data setup time (inputs) Master Slave	 $t_{su(m)}$ $t_{su(s)}$	 30 20	 — —	 ns
7	Data hold time (inputs) Master Slave	 $t_{h(m)}$ $t_{h(s)}$	 0 20	 — —	 ns
8	Slave access time	t_a	—	1	t_{cyc}
9	Slave MISO disable time	t_{dis}	—	2	t_{cyc}
10	Data valid (after SCK edge) Master Slave	 $t_{v(m)}$ $t_{v(s)}$	 — —	 20 20	 ns
11	Data hold time (outputs) Master Slave	 $t_{ho(m)}$ $t_{ho(s)}$	 — —	 0 0	 ns
12	Output rise time SCK, MOSI, and MISO	t_r	—	30	ns
13	Output fall time SCK, MOSI, and MISO	t_f	—	30	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 200 pF load on all SPI pins, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. N = External SCK rise plus external SCK fall time

Figure A-1, Figure A-2, Figure A-3, and Figure A-4 show the timing relationships of SPI pins in master and slave modes for both values of the clock phase (CPHA) bit. The clock polarity (CPOL) bit determines the inactive state of the serial clock but has no effect on the timing values.

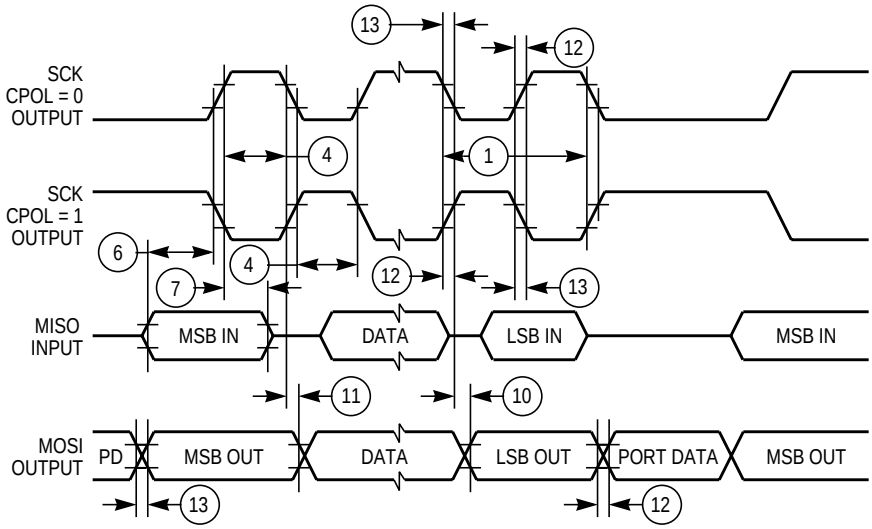


Figure A-1. SPI Timing — Master, CPHA 0

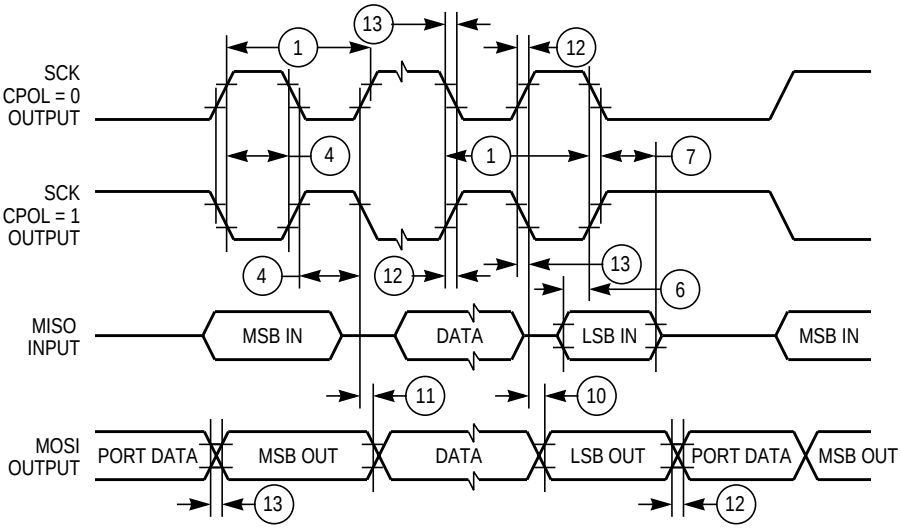


Figure A-2. SPI Timing — Master, CHPA 1

Table A-2. Key to Figure A-1 and Figure A-2 (Abstracted from Table A-1)

No.	Characteristic[1]	Symbol	Min	Max	Unit
1	Master cycle time	$t_{cyc(m)}$	4	510	t_{cyc}
4	Master clock (SCK) high or low time	$t_{sw(m)}$	2 t_{cyc} – 60	255 t_{cyc}	ns
6	Master data setup time (inputs)	$t_{su(m)}$	30	—	ns
7	Master data hold time (inputs)	$t_{h(m)}$	0	—	ns
10	Master data valid (after SCK edge)	$t_{v(m)}$	—	50	ns
11	Master data hold time (outputs)	$t_{ho(m)}$	—	50	ns
12	Output rise time SCK, MOSI, and MISO	t_r	—	30	ns
13	Output fall time SCK, MOSI, and MISO	t_f	—	30	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 200 pF load on all SPI pins, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted

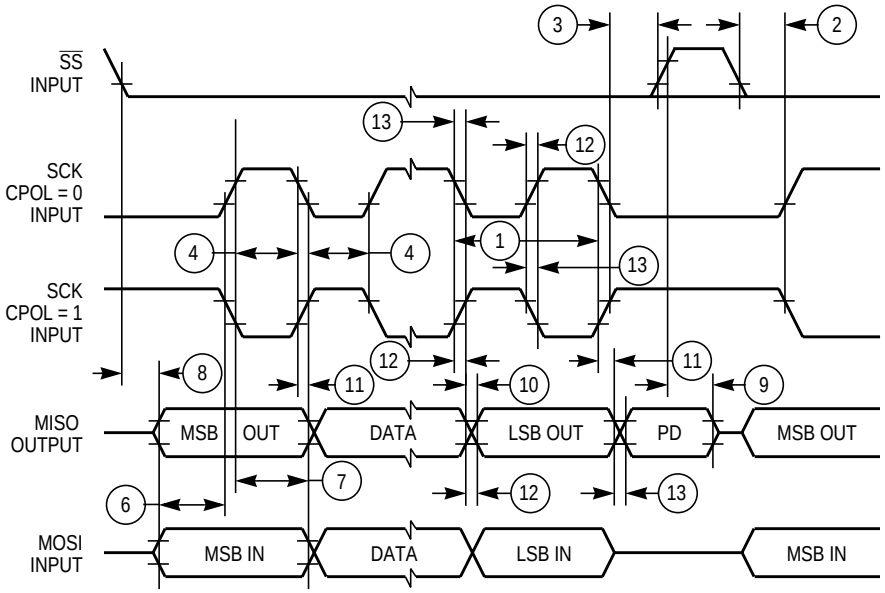


Figure A-3. SPI Timing — Slave, CPHA 0

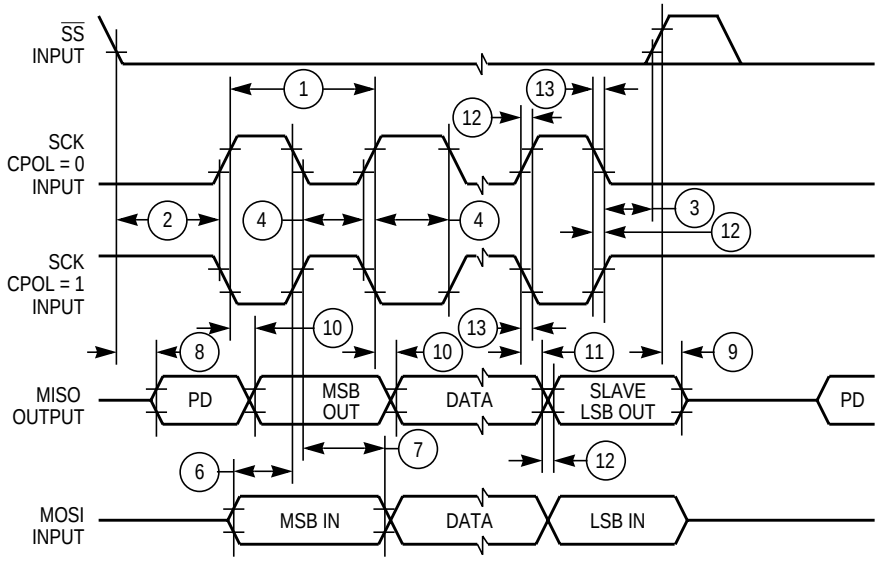


Figure A-4. SPI Timing — Slave, CPHA 1

Table A-3. Key to Figure A-3 and Figure A-4 (Abstracted from Table A-1)

No.	Characteristic[1]	Symbol	Min	Max	Unit
1	Slave cycle time	$t_{cyc(s)}$	4	—	t_{cyc}
2	Slave enable lead time	t_{lead}	2	—	t_{cyc}
3	Slave enable lag time	t_{lag}	2	3	t_{cyc}
4	Slave clock (SCK) high or low time	$t_{sw(s)}$	$2\ t_{cyc} - N$[2]	—	ns
6	Slave data setup time (inputs)	$t_{su(s)}$	20	—	ns
7	Slave data hold time (inputs)	$t_{h(s)}$	20	—	ns
8	Slave access time	t_a	—	1	t_{cyc}
9	Slave MISO disable time	t_{dis}	—	2	t_{cyc}
10	Slave data valid (after SCK edge)	$t_{v(s)}$	—	50	ns
11	Slave data hold time (outputs)	$t_{ho(s)}$	0	50	ns
12	Output rise time SCK, MOSI, and MISO	t_r	—	30	ns
13	Output fall time SCK, MOSI, and MISO	t_f	—	30	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 200 pF load on all SPI pins, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. N = External SCK rise plus external SCK fall time

Electrical Characteristics

Appendix B. Memory Map and Registers

B.1 Contents

B.2 MCCI Memory Map

Access	Address	15 8	7 0
S	$00	MCCI module configuration register (MMCR)	
S	$02	MCCI test register (MTEST)	
S	$04	SCI interrupt register (ILSCI)	MCCI interrupt vector register (MIVR)
S	$06	SPI interrupt register (ILSPI)	Reserved
S/U	$08	Reserved	MCCI pin assignment register (MPAR)
S/U	$0A	Reserved	MCCI data direction register (MDDR)
S/U	$0C	Reserved	MCCI port data register (PORTMC)
S/U	$0E	Reserved	MCCI port pin state register (PORTMCP)
S/U	$10–$16	Reserved	
S/U	$18	SCIA control register 0 (SCCR0A)	
S/U	$1A	SCIA control register 1 (SCCR1A)	
S/U	$1C	SCIA status register (SCSRA)	
S/U	$1E	SCIA data register (SCDRA)	
S/U	$20–$26	Reserved	
S/U	$28	SCIB control register 0 (SCCR0B)	
S/U	$2A	SCIB control register 1 (SCCR1B)	
S/U	$2C	SCIB status register (SCSRB)	
S/U	$2E	SCIB data register (SCDRB)	
S/U	$30–$36	Reserved	
S/U	$38	SPI control register (SPCR)	
S/U	$3A	Reserved	
S/U	$3C	SPI status register (SPSR)	
S/U	$3E	SPI data register (SPDR)	

S = Supervisor access only
S/U = Supervisor access only or user access, depending on SUPV bit in MMCR
Addresses are offsets from MCCI base address

B.3 MCCI Registers

This section summarizes the MCCI registers.

B.3.1 MCCI Module Configuration Register

Register address: $XXXX00

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	STOP	0	0	0	0	0	0	0	SUPV	0	0	0	IARB			
Write:																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B-1. MCCI Module Configuration Register (MMCR)

STOP — Stop Enable Bit

0 = Normal MCCI clock operation
1 = MCCI clock operation stopped

MMCR[14:8] — Not Implemented

SUPV — Supervisor/Unrestricted Bit

SUPV defines the assignable MCCI registers as either supervisor-only access or user access.

0 = Unrestricted access
1 = Supervisor access

MMCR[6:4] — Not Implemented

IARB — Interrupt Arbitration Number Bit

The value in this field is used to arbitrate for the IMB when two or more modules generate simultaneous interrupts at the same priority level. The IARB field should be initialized by system software to a value from $F (highest priority) through $1 (lowest priority).

B.3.2 MCCI Test Register

The MCCI test register (MTEST) is used during factory testing of the MCU. Accesses to MTEST must be made only while the MCU is in test mode.

B.3.3 SCI Interrupt Level Register

Register address: $XXXX04

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	ILSCIB			ILSCIA			MIVR (MCCI interrupt vector register)							
Reset:	0	0	0	0	0	0	0	0								

Figure B-2. SCI Interrupt Level Register (ILSCI)

ILSCIA and ILSCIB — Interrupt Level for SCIA and SCIB

ILSCIA and ILSCIB determine the priority levels of SCIA and SCIB interrupts, respectively. This field should be programmed to a value between $0 (interrupts disabled) and $7 (highest priority).

B.3.4 MCCI Interrupt Vector Register

Register address: $XXXX05

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	ILSCI (SCI interrupt level register)								INTV[7:2]						INTV[1:0}	
Write:																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

= Unimplemented

Figure B-3. MCCI Interrupt Vector Register (MIVR)

INTV[7:2] — Interrupt Vector [7:2] Bits

High-order six bits of MCCI interrupt vector programmed by user.

INTV[1:0] — Interrupt Vector [1:0] Bits

Writes to INTV0 and INTV1 have no meaning or effect. Reads of INTV0 and INTV1 return a value of 1.

00 = SCIA is source of interrupt.

01 = SCIB is source of interrupt.

10 = SPI is source of interrupt.

B.3.5 SPI Interrupt Level Register

Register address: $XXXX06

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	0	0	ILSPI			0	0	0	Reserved							
Write:																
Reset:	0	0	0	0	0	0	0	0								

Figure B-4. SPI Interrupt Level Register (ILSPI)

ILSPI — Interrupt Level for SPI Bit

ILSPI determines the priority level of SPI interrupts. Program this field to a value from $0 (interrupts disabled) through $7 (highest priority).

B.3.6 MCI Port Data Registers

Register name and address: MCCI Port Data Register (PORTMC), $XXXX0C
MCCI Port Pin State Register (PORTMCP), $XXXX0E

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	Reserved								PMC (TXDA)	PMC6 (RXDA)	PMC5 (TXDB)	PMC4 (RXDB)	PMC3 ($\overline{SS}$)	PMC2 (SCK)	PMC1 (MOSI)	PMC0 (MISO)
Write:																
Reset:									0	0	0	0	0	0	0	0

Figure B-5. MCCI Port Data Registers (PORTMC and PORTMCP)

Two registers are associated with port MCCI, the MCCI general-purpose input/output (I/O) port. Writes to PORTMC, the MCCI port data register, are stored in the internal data latch. If any bit of PORTMC is configured as discrete output, the value latched for that bit is driven onto the pin. Reads of PORTMC return the value of the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value of the latch.

Reads of PORTMCP, the MCCI pin state register, always return the state of the pins regardless of whether the pins are configured as input or output. Writes to PORTMCP have no effect.

B.3.7 MCCI Pin Assignment Register

Register address: $XXXX09

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	Reserved								0	0	0	0	SS	0	MOSI	MISO
Write:																
Reset:									0	0	0	0	0	0	0	0

Figure B-6. MCCI Pin Assignment Register (MPAR)

MPAR[7:4] and MPAR2 — Not implemented

SS — Slave Select Bit

MOSI — Master Out/Slave In Bit
MISO — Master In/Slave Out Bit

These bits determine whether the associated MCCI port pin functions as a general-purpose I/O pin or is assigned to the SPI submodule.

B.3.8 MCCI Data Direction Register

Register address: $XXXX0B

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: / Write:	Reserved								TXDA	RXDA	TXDB	RXDB	SS	SCK	MOSI	MISO
Reset:									0	0	0	0	0	0	0	0

Figure B-7. MCCI Data Direction Register (MDDR)

These bits assign the corresponding pin, when configured for general-purpose I/O, to be input or output.

0 = Input
1 = Output

B.3.9 SPI Control Register

Register address: $XXXX38

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: / Write:	SPIE	SPE	WOMP	MSTR	CPOL	CPHA	LSBF	SIZE	BAUD							
Reset:	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Figure B-8. SPI Control Register (SPCR)

SPIE — SPI Interrupt Enable Bit

0 = SPI interrupts disabled
1 = SPI interrupts enabled

SPE — SPI Enable Bit

To prevent unpredicted operation, the other SPI control fields should be configured properly before or at the same time SPE is set.

0 = SPI disabled
1 = SPI enabled

WOMP — Wired-OR Mode for SPI Pins

0 = Outputs have normal MOS drivers.

1 = Pins designated for output by MDDR have open-drain drivers, regardless of whether the pins are used as SPI outputs or for general-purpose I/O, and regardless of whether the SPI is enabled.

MSTR — Master/Slave Mode Select Bit

0 = SPI is a slave device.

1 = SPI is system master.

CPOL — Clock Polarity Bit

0 = The inactive state value of SCK is logic level 0.

1 = The inactive state value of SCK is logic level 1.

CPHA — Clock Phase Bit

0 = Data is captured on the leading edge of SCK and changed on the following edge of SCK.

1 = Data is changed on the leading edge of SCK and captured on the following edge of SCK.

LSBF — Least Significant Bit First

0 = Serial data transfer starts with MSB.

1 = Serial data transfer starts with LSB.

SIZE — Transfer Data Size Bit

0 = 8-bit data transfer

1 = 16-bit data transfer

BAUD — SPI Serial Clock Baud Rate Bit

The SPI baud rate is selected by writing a value from 2 to 255 into the BAUD field of the SPCR of the master MCU. Giving BAUD a value of 0 or 1 disables SCK. (The disabled state is determined by CPOL). At reset, BAUD is initialized so that SCK has a 2.1-MHz SCK frequency, given a 16.8-MHz system clock.

B.3.10 SPI Status Register

Register address: $XXXX3C

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIF	WCOL	0	MODF	0	0	0	0	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B-9. SPI Status Register (SPSR)

SPIF — SPI Finished Flag
- 0 = Transfer not completed
- 1 = Transfer completed

WCOL — Write Collision Bit
- 0 = No write collision occurred
- 1 = Write collision occurred

MODF — Mode Fault Flag
- 0 = Normal operation
- 1 = Another SPI node requested to become the network SPI master while the SPI was enabled in master mode ($\overline{SS}$ input taken low)

B.3.11 SPI Data Register

Register address: $XXXX3E

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	UPPB								LOWB							
Write:																
Reset:	Reset state is unknown															

Figure B-10. SPI Data Register (SPDR)

UPPB — Upper Byte Bit

In a 16-bit transfer (SIZE = 1), the address of the upper byte is used to access the most significant eight bits of the data. Bit 15 of the SPDR is the MSB of the 16-bit data.

LOWB — Lower Byte Bit

In an 8-bit transfer (SIZE = 0), the data is accessed at the address of the lower byte. The MSB in an 8-bit transfer is bit 7 of the SPDR. In a 16-bit transfer (SIZE = 1), the lower byte holds the least significant eight bits of the data.

B.3.12 SCI Control Register 0

Register name and address: SCI Control Register 0 (SCCR0A), $XXXX18
SCI Control Register 0 (SCCR0B), $XXXX28

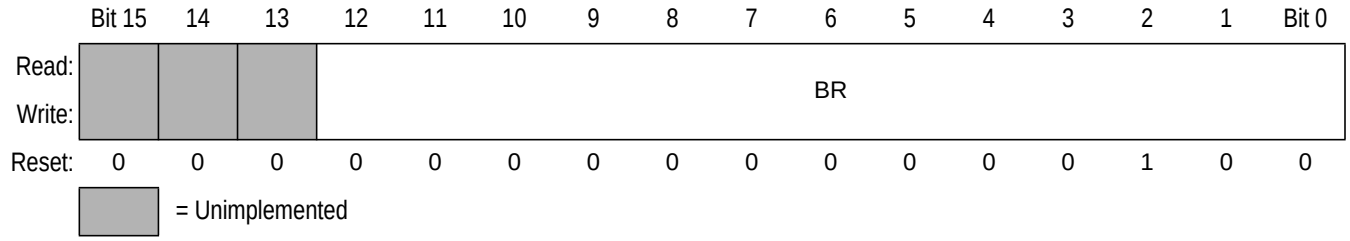


Figure B-11. SCI Control Register 0 (SCCR0A and SCCR0B)

SCCR0[15:13] — Not Implemented

BR — Baud Rate Field

The SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of 0 to BR disables the baud rate generator.

B.3.13 SCI Control Register 1

Register name and address: SCI Control Register 1 (SCCR1A), $XXXX1A
SCI Control Register 1 (SCCR1B), $XXXX2A

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: / Write:		LOOPS	WOMC	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented

Figure B-12. SCI Control Register 1 (SCCR1A and SCCR1B)

Bit 15 — Not Implemented

LOOPS — Loop Mode Bit

The LOOPS bit in SCCR1 controls a feedback path on the data serial shifter. When LOOPS is set, SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled prior to entering loop mode.

0 = Normal SCI operation, no looping, feedback path disabled
1 = Test SCI operation, looping, feedback path enabled

WOMC — Wired-OR Mode for SCI Pins

0 = If configured as an output, TXD is a normal CMOS output.
1 = If configured as an output, TXD is an open-drain output.

ILT — Idle-Line Detect Type Bit

0 = Short idle-line detect (start count on first 1)
1 = Long idle-line detect (start count on first 1 after stop bit(s))

PT — Parity Type Bit

0 = Even parity
1 = Odd parity

PE — Parity Enable Bit

0 = SCI parity disabled
1 = SCI parity enabled

M — Mode Select Bit

0 = 10-bit frame
1 = 11-bit frame

WAKE — Wakeup by Address Mark Bit
0 = Receiver awakened by idle-line detection
1 = Receiver awakened by address mark

TIE — Transmit Interrupt Enable Bit
0 = TDRE interrupts inhibited
1 = TDRE interrupts enabled

TCIE — Transmit Complete Interrupt Enable Bit
0 = TC interrupts inhibited
1 = TC interrupts enabled

RIE — Receiver Interrupt Enable Bit
0 = RDRF interrupts inhibited
1 = RDRF interrupts enabled

ILIE — Idle-Line Interrupt Enable Bit
0 = IDLE interrupts inhibited
1 = IDLE interrupts enabled

TE — Transmitter Enable Bit
0 = Transmitter disabled (TXD pin may be used as I/O.)
0 = Transmitter enabled (TXD pin dedicated to SCI transmitter)

RE — Receiver Enable Bit
0 = Receiver disabled (status bits inhibited)
1 = Receiver enabled

RWU — Receiver Wakeup Bit
0 = Normal receiver operation (received data recognized)
1 = Wakeup mode enabled (received data ignored until awakened)

SBK — Send Break Bit
0 = Normal operation
1 = Break frame(s) transmitted after completion of current frame

B.3.14 SCI Status Register

Register name and address: SCI Status Register (SCSRA), $XXXX1C
SCI Status Register (SCSRB), $XXXX2C

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:								TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
Write:																
Reset:	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

= Unimplemented

Figure B-13. SCI Status Registers (SCSRA and SCSRB)

TDRE — Transmit Data Register Empty Flag
- 0 = The TDR still contains data to be sent to the transmit serial shifter.
- 1 = A new character may now be written to register TDR.

TC — Transmit Complete Flag
- 0 = SCI transmitter is busy.
- 1 = SCI transmitter is idle.

0RDRF — Receive Data Register Full Flag
- 0 = The RDR is empty or contains previously read data.
- 1 = The RDR contains new data.

RAF — Receiver Active Flag
- 0 = SCI receiver is idle.
- 1 = SCI receiver is busy.

IDLE — Idle-Line Detected Flag
- 0 = SCI receiver did not detect an idle-line condition.
- 1 = SCI receiver detected an idle-line condition.

OR — Overrun Error Flag
- 0 = RDRF is cleared before new data arrives.
- 1 = RDRF is not cleared before new data arrives.

NF — Noise Error Flag
- 0 = No noise detected on the received data.
- 1 = Noise occurred on the received data.

FE — Framing Error Flag

0 = No framing error on the received data

1 = Framing error or break occurred on the received data.

PF — Parity Error Flag

0 = No parity error on the received data

1 = Parity error occurred on the received data.

B.3.15 SCI Data Register

Register name and address: SCI Data Register (SCDRA), $XXXX1E
SCI Data Register (SCDRB), $XXXX2F

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
Write:																
Reset:	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

= Unimplemented U = Unaffected

Figure B-14. SCI Data Registers (SCDRA and SCDRB)

R[8:0] — Receive[8:0] Bits

R8 is the MSB when the SCI system is configured for 11-bit frames (M = 1). If used, R8 represents the address mark, an extra stop bit, or the parity mark. When the SCI system is configured for 10-bit frames (M = 0), this bit has no meaning or effect. R7 is then the MSB before the stop bit and may represent either data, a parity bit, a second stop bit, or an address mark. R[6:0] represent data bits.

T[8:0] — Transmit[8:0] Bits

T8 is the MSB when the SCI system is configured for 11-bit frames (M = 1). When used, T8 represents either the address mark, an extra stop bit, or the parity mark. When the SCI system is configured for 10-bit frames (M = 0), this bit has no meaning or effect. T7 is then the MSB and may represent either data, a parity bit, an extra stop bit, or an address mark. T[6:0] represent data bits.

Memory Map and Registers

Index

Index

L

M

P

Q

R

S

T

U

W

Index

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140 or 1-800-441-2447. Customer Focus Center, 1-800-521-6274

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-8573

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-26668334

Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/; TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848

HOME PAGE: http://motorola.com/sps/

MOTOROLA

MCCIRM/AD