SECTION 14
QUEUED SERIAL MULTI-CHANNEL MODULE

14.1 Overview

The queued serial multi-channel module (QSMCM) provides three serial communication interfaces: the queued serial peripheral interface (QSPI) and two serial communications interfaces (SCI1 and SCI2). These submodules communicate with the CPU via a common slave bus interface unit (SBIU).

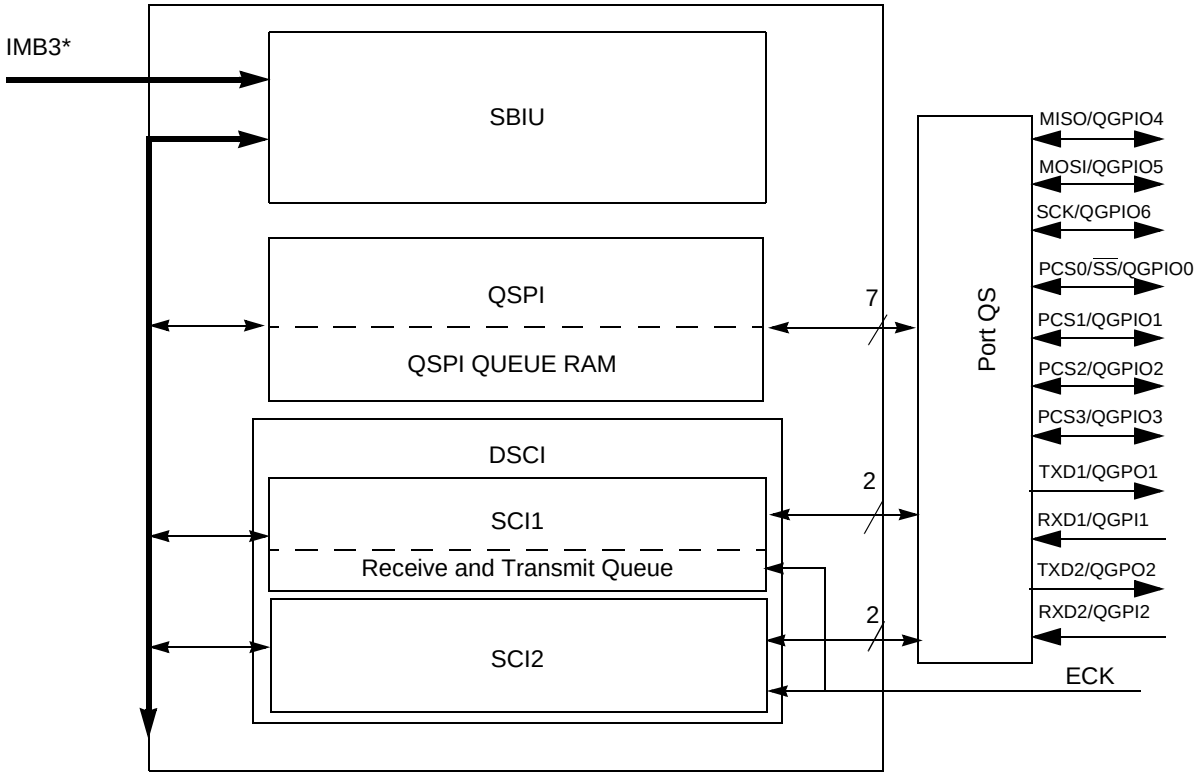
The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced from the original SPI in the QSMCM (queued serial module) to include a total of 160 bytes of queue RAM to accommodate more receive, transmit, and control information. The QSPI is fully compatible with the SPI systems found on other Motorola devices.

The dual, independent SCIs are used to communicate with external devices and other MCUs via an asynchronous serial bus. Each SCI is a full-duplex universal asynchronous receiver transmitter (UART) serial interface. The original QSMCM SCI is enhanced by the addition of an SCI and a common external baud clock source.

The SCI1 has the ability to use the resultant baud clock from SCI2 as the input clock source for the SCI1 baud rate generator. Also, the SCI1 has an additional mode of operation that allows queuing of transmit and receive data frames. If the queue feature is enabled, a set of 16 entry queues is allocated for the receive and/or transmit operation.

14.2 Block Diagram

Figure 14-1 depicts the major components of the QSMCM.



*Note: SBIU Bus and interface to IMB3 are each 16 bits wide.

Figure 14-1 QSMCM Block Diagram

14.3 Signal Descriptions

The QSMCM has 12 external pins, as shown in **Figure 14-1**. Seven of the pins, if not in use for their submodule function, can be used as general-purpose I/O port pins. The RXDx and TXDx pins can alternately serve as general-purpose input-only and output-only signals, respectively. ECK is a dedicated clock pin.

For detailed descriptions of QSMCM signals, refer to **14.6 QSMCM Pin Control Registers**, **14.7.3 QSPI Pins**, and **14.8.6 SCI Pins**.

14.4 Memory Map

The QSMCM memory map, shown in **Table 14-1**, includes the global registers, the QSPI and dual SCI control and status registers, and the QSPI RAM. The QSMCM memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **1.3 MPC555 Address Map** for a diagram of the MPC555 internal memory map.

Table 14-1 QSMCM Register Map

Access[1]	Address	MSB[2] 0	LSB 15
S	0x30 5000	QSMCM Module Configuration Register (QSMCMMCR) See **Table 14-4** for bit descriptions.	
T	0x30 5002	QSMCM Test Register (QTEST)	
S	0x30 5004	Dual SCI Interrupt Level (QDSCI_IL) See **Table 14-5** for bit descriptions.	Reserved
S	0x30 5006	Reserved	Queued SPI Interrupt Level (QSPI_IL) See **Table 14-6** for bit descriptions.
S/U	0x30 5008	SCI1Control Register 0 (SCC1R0) See **Table 14-23** for bit descriptions.	
S/U	0x30 500A	SCI1Control Register 1 (SCC1R1) See **Table 14-24** for bit descriptions.	
S/U	0x30 500C	SCI1 Status Register (SC1SR) See **Table 14-25** for bit descriptions.	
S/U	0x30 500E	SCI1 Data Register (SC1DR) See **Table 14-26** for bit descriptions.	
S/U	0x30 5010	Reserved	
S/U	0x30 5012	Reserved	
S/U	0x30 5014	Reserved	QSMCM Port Q Data Register (PORTQS) See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
S/U	0x30 5016	QSMCM Pin Assignment Register (PQSPAR) See **Table 14-10** for bit descriptions.	QSMCM Data Direction Register (DDRQS) See **Table 14-11** for bit descriptions.
S/U	0x30 5018	QSPI Control Register 0 (SPCR0) See **Table 14-13** for bit descriptions.	
S/U	0x30 501A	QSPI Control Register 1 (SPCR1) See **Table 14-15** for bit descriptions.	
S/U	0x30 501C	QSPI Control Register 2 (SPCR2) See **Table 14-16** for bit descriptions.	
S/U	0x30 501E	QSPI Control Register 3 (SPCR3) See **Table 14-17** for bit descriptions.	QSPI Status Register (SPSR) See **Table 14-18** for bit descriptions.
S/U	0x30 5020	SCI2 Control Register 0 (SCC2R0)	
S/U	0x30 5022	SCI2 Control Register 1 (SCC2R1)	
S/U	0x30 5024	SCI2 Status Register (SC2SR)	
S/U	0x30 5026	SCI2 Data Register (SC2DR)	
S/U	0x30 5028	QSCI1 Control Register (QSCI1CR) See **Table 14-33** for bit descriptions.	
S/U	0x30 502A	QSCI1 Status Register (QSCI1SR) See **Table 14-34** for bit descriptions.	

Table 14-1 QSMCM Register Map (Continued)

Access[1]	Address	MSB[2] 0	LSB 15
S/U	0x30 502C – 0x30 504A	Transmit Queue Locations (SCTQ)	
S/U	0x30 504C – 0x30 506A	Receive Queue Locations (SCRQ)	
S/U	0x30 506C – 0x30 513F[3]	Reserved	
S/U	0x30 5140 – 0x30 517F	Receive Data RAM (REC.RAM)	
S/U	0x30 5180 – 0x30 51BF	Transmit Data RAM (TRAN.RAM)	
S/U	0x30 51C0 – 0x30 51DF	Command RAM (COMD.RAM)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.
3. Note that QRAM offsets have been changed from the original (modular family) QSMCM.

The supervisor-only data space segment contains the QSMCM global registers. These registers define parameters needed by the QSMCM to integrate with the MCU. Access to these registers is permitted only when the CPU is operating in supervisor mode.

Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user accesses. The supervisor (SUPV) bit in the QSMCM module configuration register (QSMCMMCR) designates the assignable data space as either supervisor or unrestricted. If SUPV is set, then the space is designated as supervisor-only space. Access is then permitted only when the CPU is operating in supervisor mode. If SUPV is clear, both user and supervisor accesses are permitted. To clear SUPV, the CPU must be in supervisor mode.

The QSMCM assignable data space segment contains the control and status registers for the QSPI and SCI submodules, as well as the QSPI RAM. All registers and RAM can be accessed on byte (8-bits), half-word (16-bits), and word (32-bit) boundaries. Word accesses require two consecutive IMB3 bus cycles.

14.5 QSMCM Global Registers

The QSMCM global registers contain system parameters used by the QSPI and SCI submodules for interfacing to the CPU and the intermodule bus. The global registers are listed in **Table 14-2 QSMCM Global Registers**

Table 14-2 QSMCM Global Registers

Access[1]	Address	MSB[2]	LSB
S	0x30 5000	QSMCM Module Configuration Register (QSMCMMCR) See **Table 14-4** for bit descriptions.	
T	0x30 5002	QSMCM Test Register (QTEST)	
S	0x30 5004	Dual SCI Interrupt Level (QDSCI_IL) See **Table 14-5** for bit descriptions.	Reserved
S	0x30 5006	Reserved	Queued SPI Interrupt Level (QSPI_IL) See **Table 14-6** for bit descriptions.

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers reside on 8-bit boundaries. 16-bit registers reside on 16-bit boundaries.

14.5.1 Low-Power Stop Operation

When the STOP bit in QSMCMMCR is set, the system clock input to the QSMCM is disabled and the module enters a low-power operating state. QSMCMMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable in low-power stop mode. However, writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be written by the CPU and is cleared by reset.

System software must bring each submodule to an orderly stop before setting STOP to avoid data corruption. The SCI receiver and transmitter should be disabled after transfers in progress are complete. The QSPI can be halted by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set.

14.5.2 Freeze Operation

The FRZ1 bit in QSMCMMCR determines how the QSMCM responds when the IMB3 FREEZE signal is asserted. FREEZE is asserted when the CPU enters background debug mode. Setting FRZ1 causes the QSPI to halt on the first transfer boundary following FREEZE assertion. FREEZE causes the SCI1 transmit queue to halt on the first transfer boundary following FREEZE assertion.

14.5.3 Access Protection

The SUPV bit in the QMCR defines the assignable QSMCM registers as either supervisor-only data space or unrestricted data space.

When the SUPV bit is set, all registers in the QSMCM are placed in supervisor-only space. For any access from within user mode, the IMB3 address acknowledge ($\overline{\text{AACK}}$) signal is asserted and a bus error is generated.

Because the QSMCM contains a mix of supervisor and user registers, $\overline{\text{AACK}}$ is asserted for either supervisor or user mode accesses, and the bus cycle remains internal. If a supervisor-only register is accessed in user mode, the module responds as if

an access had been made to an unauthorized register location, and a bus error is generated.

14.5.4 QSMCM Interrupts

The interrupt structure of the IMB3 supports a total of 32 interrupt levels that are time multiplexed on the IRQB[0:7] lines as seen in **Figure 14-2**.

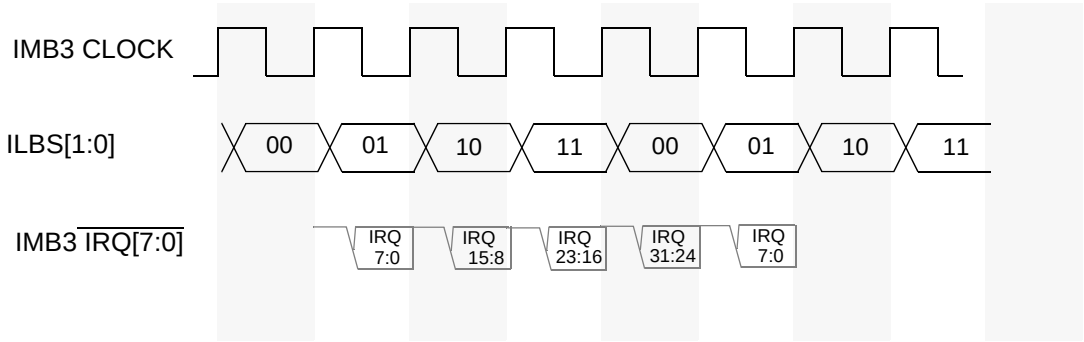


Figure 14-2 QSMCM Interrupt Levels

In this structure, all interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table 14-3 Interrupt Levels

ILBS[0:1]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The QSMCM module is capable of generating one of the 32 possible interrupt levels on the IMB3. The levels that the interrupt will drive can be programmed into the interrupt request level (ILDSCI and ILQSPI) bits located in the interrupt configuration register (QDSCI_IL and QSPI_IL). This value determines which interrupt signal (IRQB[0:7]) is driven onto the bus during the programmed time slot. **Figure 14-3** shows a block diagram of the interrupt hardware.

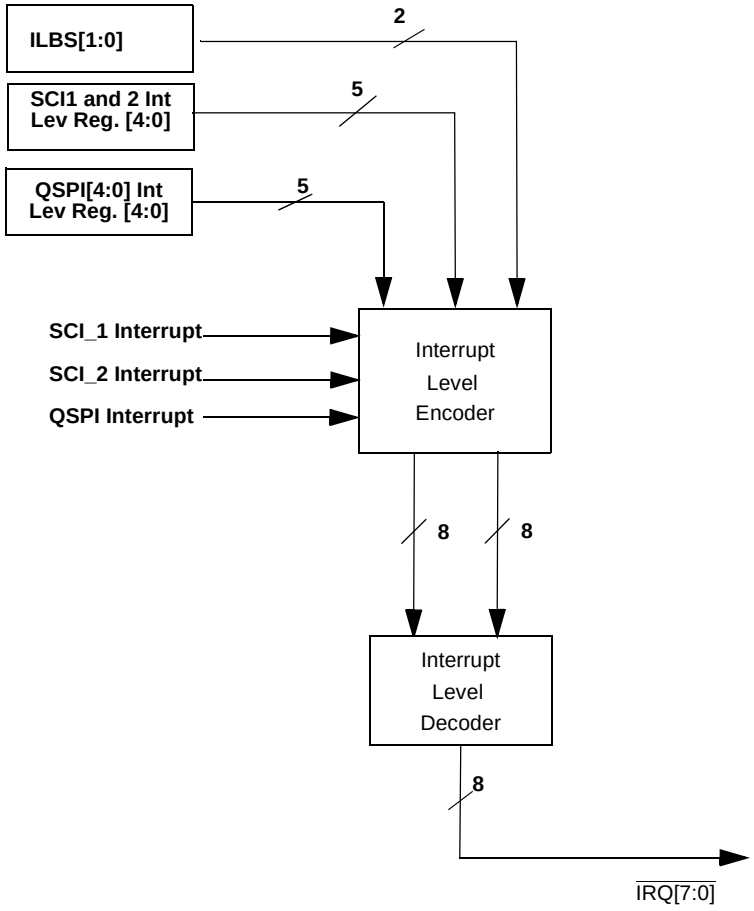


Figure 14-3 QSPI Interrupt Generation

14.5.5 QSMCM Configuration Register (QSMCMMCR)

The QSMCMMCR contains parameters for interfacing to the CPU and the intermodule bus. This register can be modified only when the CPU is in supervisor mode.

QSMCMMCR — QSMCM Configuration Register **0x30 5000**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ1	RESERVED						SUPV	RESERVED			IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 14-4 QSMCMMCR Bit Settings

Bit(s)	Name	Description
0	STOP	Stop enable. Refer to **14.5.1 Low-Power Stop Operation**. 0 = Normal clock operation 1 = Internal clocks stopped
1	FRZ1	Freeze1 bit. Refer to **14.5.2 Freeze Operation**. 0 = Ignore the FREEZE signal 1 = Halt the QSMCM (on transfer boundary)
2:7	—	Reserved
8	SUPV	Supervisor /Unrestricted. Refer to **14.5.3 Access Protection**. 0 = Assigned registers are unrestricted (user access allowed) 1 = Assigned registers are restricted (only supervisor access allowed)
9:11	—	Reserved
12:15	IARB	This field currently has no effect. It is implemented for future interrupt arbitration schemes.

14.5.6 QSMCM Test Register (QTEST)

The QTEST register is used for factory testing of the MCU.

14.5.7 QSMCM Interrupt Level Registers (QDSCI_IL, QSPI_IL)

The QDSCI_ILI and QSPI_IL registers determine the interrupt level requested by the QSMCM. The two SCI submodules (DSCI) share a 5-bit interrupt level field, ILDSCI. The QSPI uses a separate field, ILQSPI. The level value is used to determine which interrupt is serviced first when two or more modules or external peripherals simultaneously request an interrupt. The user can select among 32 levels. This register can be accessed only when the CPU is in supervisor mode.

QDSCI_IL — QSM2 Dual SCI Interrupt Level Register **0x30 5004**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Reserved			ILDSCI					RESERVED							
RESET:															
0	0	0	0	0	0	0	0								

Table 14-5 QDSCI_IL Bit Settings

Bit(s)	Name	Description
0:2	—	Reserved
3:7	ILSCI1	Interrupt level of SCIs 00000 = lowest interrupt level request (level 0) 11111 = highest interrupt level request (level 31)
8:15	—	Reserved

QSPI_IL — QSPI Interrupt Level Register **0x30 5006**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED											ILQSPI				
RESET:															
0	0	0	0	0	0	0	0								

Table 14-6 QSPI_IL Bit Settings

Bit(s)	Name	Description
0:10	—	Reserved
11:15	ILQSPI	Interrupt level of SPI 00000 = lowest interrupt level request (level 0) 11111 = highest interrupt level request (level 31)

14.6 QSMCM Pin Control Registers

Table 14-7 lists the three QSMCM pin control registers.

Table 14-7 QSMCM Pin Control Registers

Address	Register
0x30 5014	QSMCM Port Data Register (PORTQS) See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
0x30 5016	PORTQS Pin Assignment Register (PQSPAR) See **Table 14-11** for bit descriptions.
0x30 5017	PORTQS Data Direction Register (DDRQS) See **Table 14-11** for bit descriptions.

The QSMCM uses 12 pins. Eleven of the pins, when not being used by the serial subsystems, form a parallel port on the MCU. (The ECK pin is a dedicated external clock source.)

The port QS pin assignment register (PQSPAR) governs the usage of QSPI pins. Clearing a bit assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI.

PQSPAR does not affect operation of the SCI. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. When the SCIx transmitter or receiver is enabled, the associated TXDx or RXDx pin is assigned its SCI function.

The port QS data direction register (DDRQS) determines whether QSPI pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function. **Table 14-10** summarizes the effect of DDRQS bits on QSPI pin function.

DDRQS does not affect SCI pin function. TXDx pins are always outputs, and RXDx pins are always inputs, regardless of whether they are functioning as SCI pins or as PORTQS pins.

The port QS data register (PORTQS) latches I/O data. PORTQS writes drive pins defined as outputs. PORTQS reads return data present on the pins. To avoid driving undefined data, write the first data to PORTQS before configuring DDRQS.

Table 14-8 Effect of DDRQS on QSPI Pin Function

QSMCM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQS0	0	Serial data input to QSPI
			1	Disables data input
	Slave		0	Disables data output
			1	Serial data output from QSPI
MOSI	Master	DDQS1	0	Disables data output
			1	Serial data output from QSPI
	Slave		0	Serial data input to QSPI
			1	Disables data input
SCK[1]	Master	DDQS2	—	Clock output from QSPI
	Slave		—	Clock input to QSPI
PCS0/$\overline{SS}$	Master	DDQS3	0	Assertion causes mode fault
			1	Chip-select output
	Slave		0	QSPI slave select input
			1	Disables slave select input
PCS[1:3]	Master	DDQS[4:6]	0	Disables chip-select output
			1	Chip-select output
	Slave		0	Inactive
			1	Inactive

NOTES:
1. SCK/QGPIO6 is a digital I/O pin unless the SPI is enabled (SPE set in SPCR1), in which case it becomes the QSPI serial clock SCK.

14.6.1 Port QS Data Register (PORTQS)

PORTQS determines the actual input or output value of a QSMCM port pin if the pin is defined as general-purpose input or output. All QSMCM pins except the ECK pin can be used as general-purpose input and/or output. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. Writes to this register affect the pins defined as outputs; reads of this register return the actual value of the pins.

PORTQS — Port QS Data Register **0x30 5014**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED				QDRXD2	QDTXD2	QDRXD1	QDTXD1	0	QDPCS3	QDPCS2	QDPCS1	QDPCS0	QDSCK	QDMOSI	QDMISO
RESET:															
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

14.6.2 PORTQS Pin Assignment Register (PQSPAR)

PQSPAR determines which of the QSPI pins, with the exception of the SCK pin, are used by the QSPI submodule, and which pins are available for general-purpose I/O. Pins may be assigned on a pin-by-pin basis. If the QSPI is disabled, the SCK pin is automatically assigned its general-purpose I/O function (QGPIO6).

QSPI pins designated by PQSPAR as general-purpose I/O pins are controlled only by PQSDDR and PQSPDR; the QSPI has no effect on these pins. PQSPAR does not affect the operation of the SCI submodule.

Table 14-9 summarizes the QSMCM pin functions.

Table 14-9 QSMCM Pin Functions

PORTQS Function	QSMCM Function
QGPI2	RXD2
QGPO2	TXD2
QGPI1	RXD1
QGPO1	TXD1
QGPIO6	SCK
QGPIO5	MOSI
QGPIO4	MISO
QGPIO3	PCS3
QGPIO2	PCS2
QGPIO1	PCS1
QGPIO0	PCS0

PQSPAR — PORTQS Pin Assignment Register **0x30 5016**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	QPAPCS3	QPAPCS2	QPAPCS1	QPAPCS0	0	QPAMOSI	QPAMISO	DDRQS*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 14-11**.

Table 14-10 PQSPAR Bit Settings

Bit(s)	Name	Description
0	—	Reserved
1	QPAPCS3	0 = Pin is assigned QGPIO3 1 = Pin is assigned PCS3 function
2	QPAPCS2	0 = Pin is assigned QGPIO2 1 = Pin is assigned PCS2 function
3	QPAPCS1	0 = Pin is assigned QGPIO1 1 = Pin is assigned PCS1 function
4	QPAPCS0	0 = Pin is assigned QGPIO0 1 = Pin is assigned PCS0 function
5	—	Reserved
6	QPAMOSI	0 = Pin is assigned QGPIO5 1 = Pin is assigned MOSI function
7	QPAMISO	0 = Pin is assigned QGPIO4 1 = Pin is assigned MISO function
8:15	DDRQS	PORSTQS data direction register. See **14.6.3 PORTQS Data Direction Register (DDRQS)**.

14.6.3 PORTQS Data Direction Register (DDRQS)

DDRQS assigns QSPI pin as an input or an output regardless of whether the QSPI submodule is enabled or disabled. All QSPI pins are configured during reset as general-purpose inputs.

This register does not affect SCI operation. The TXD1 and TXD2 remain output pins dedicated to the SCI submodules, and the RXD1, RXD2 and ECK pins remain input pins dedicated to the SCI submodules.

DDRQS — PORTQS Data Direction Register **0x30 5016**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PQSPAR*								0	QDDPCS3	QDDPCS2	QDDPCS1	QDDPCS0	QDDSCK	QDDMOSI	QDDMISO
RESET:								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 14-10**.

Table 14-11 DDRQS Bit Settings

Bit(s)	Name	Description
0:7	PQSPAR	PORTSQS pin assignment register. See **14.6.2 PORTQS Pin Assignment Register (PQSPAR)**.
8	—	Reserved
9	QDDPCS3	QSPI pin data direction for the pin PCS3 0 = Pin direction is input 1 = Pin direction is output
10	QDDPCS2	QSPI pin data direction for the pin PCS2 0 = Pin direction is input 1 = Pin direction is output
11	QDDPCS1	QSPI pin data direction for the pin PCS1 0 = Pin direction is input 1 = Pin direction is output
12	QDDPCS0	QSPI pin data direction for the pin PCS0 0 = Pin direction is input 1 = Pin direction is output
13	QDDSCK	QSPI pin data direction for the pin SCK 0 = Pin direction is input 1 = Pin direction is output
14	QPDMOSI	QSPI pin data direction for the pin MOSI 0 = Pin direction is input 1 = Pin direction is output
15	QPDMISO	QSPI pin data direction for the pin MISO 0 = Pin direction is input 1 = Pin direction is output

14.7 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) is used to communicate with external devices through a synchronous serial bus. The QSPI is fully compatible with SPI systems found on other Motorola products, but has enhanced capabilities. The QSPI can perform full duplex three-wire or half duplex two-wire transfers. Several transfer rates, clocking, and interrupt-driven communication options are available. **Figure 14-4** is a block diagram of the QSPI.

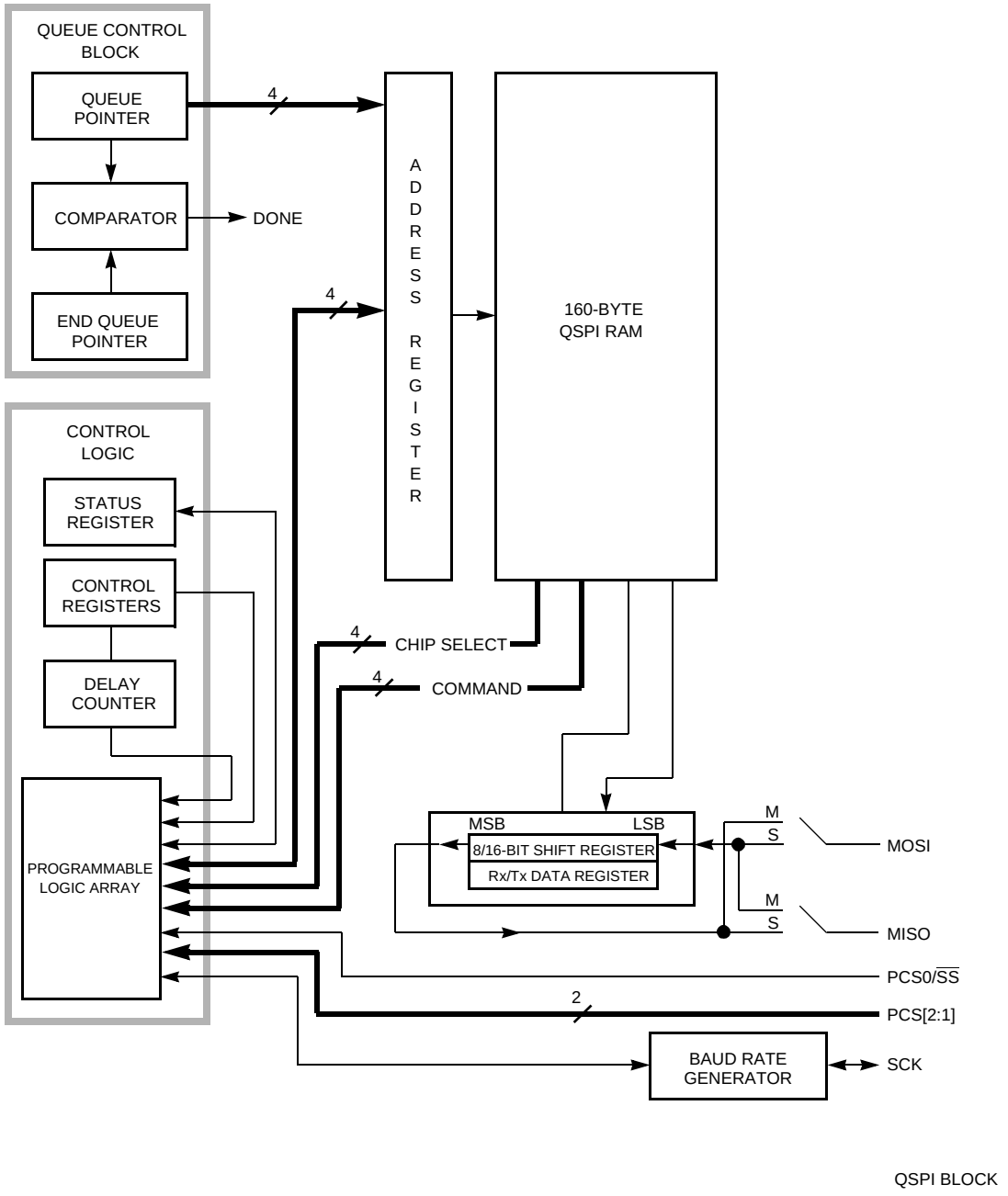


Figure 14-4 QSPI Block Diagram

Serial transfers of eight to 16 bits can be specified. Programmable transfer length simplifies interfacing to devices that require different data lengths.

An inter-transfer delay of approximately 0.8 to 204 µs (using a 40-MHz system clock) can be programmed. The default delay is 17 clocks (0.425 µs at 40 MHz). Programmable delay simplifies the interface to devices that require different delays between transfers.

A dedicated 160-byte RAM is used to store received data, data to be transmitted, and a queue of commands. The CPU can access these locations directly. This allows serial peripherals to be treated like memory-mapped parallel devices.

The command queue allows the QSPI to perform up to 32 serial transfers without CPU intervention. Each queue entry contains all the information needed by the QSPI to independently complete one serial transfer.

A pointer identifies the queue location containing the data and command for the next serial transfer. Normally, the pointer address is incremented after each serial transfer, but the CPU can change the pointer value at any time. Support for multiple-tasks can be provided by segmenting the queue.

The QSPI has four peripheral chip-select pins. The chip-select signals simplify interfacing by reducing CPU intervention. If the chip-select signals are externally decoded, 16 independent select signals can be generated.

Wrap-around mode allows continuous execution of queued commands. In wrap-around mode, newly received data replaces previously received data in the receive RAM. Wrap-around mode can simplify the interface with A/D converters by continuously updating conversion values stored in the RAM.

Continuous transfer mode allows transfer of an uninterrupted bit stream. From 8 to 512 bits can be transferred without CPU intervention. Longer transfers are possible, but minimal intervention is required to prevent loss of data. A standard delay of 17 system clocks (0.8 µs with a 40-MHz system clock) is inserted between the transfer of each queue entry.

14.7.1 QSPI Registers

The QSPI memory map, shown in **Table 14-12**, includes the QSMCM global and pin control registers, four QSPI control registers (SPCR[0:3]), the status register (SPSR), and the QSPI RAM. Registers and RAM can be read and written by the CPU. The memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **1.3 MPC555 Address Map** for a diagram of the MPC555 internal memory map.

Table 14-12 QSPI Register Map

Access[1]	Address	MSB[2]	LSB
S/U	0x30 5018	QSPI Control Register 0 (SPCR0) See **Table 14-13** for bit descriptions.	
S/U	0x30 501A	QSPI Control Register 1 (SPCR1) See **Table 14-15** for bit descriptions.	
S/U	0x30 501C	QSPI Control Register 2 (SPCR2) See **Table 14-16** for bit descriptions.	
S/U	0x30 501E/ 0x30 501F	QSPI Control Register 3 (SPCR3) See **Table 14-17** for bit descriptions.	QSPI Status Register (SPSR) See **Table 14-18** for bit descriptions.
S/U	0x30 5140 – 0x30 517F	Receive Data RAM (32 half-words)	
S/U	0x30 5180 – 0x30 51BF	Transmit Data RAM (32 half-words)	
S/U	0x30 51C0 – 0x30 51DF	Command RAM (32 bytes)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.

To ensure proper operation, set the QSPI enable bit (SPE) in SPCR1 only after initializing the other control registers. Setting this bit starts the QSPI.

Rewriting the same value to a control register does not affect QSPI operation with the exception of writing NEWQP in SPCR2. Rewriting the same value to these bits causes the RAM queue pointer to restart execution at the designated location.

Before changing control bits, the user should halt the QSPI. Writing a different value into a control register other than SPCR2 while the QSPI is enabled may disrupt operation. SPCR2 is buffered, preventing any disruption of the current serial transfer. After the current serial transfer is completed, the new SPCR2 value becomes effective.

14.7.1.1 QSPI Control Register 0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU has read/write access to SPCR0, but the QSPI has read access only. SPCR0 must be initialized before QSPI operation begins. Writing a new value to SPCR0 while the QSPI is enabled disrupts operation.

SPCR0 — QSPI Control Register 0 **0x30 5018**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Table 14-13 SPCR0 Bit Settings

Bit(s)	Name	Description
0	MSTR	Master/slave mode select 0 = QSPI is a slave device and only responds to externally generated serial transfers. 1 = QSPI is the system master and can initiate transmission to external SPI devices.
1	WOMQ	Wired-OR mode for QSPI pins. This bit controls the QSPI pins regardless of whether they are used as general-purpose outputs or as QSPI outputs, and regardless of whether the QSPI is enabled or disabled. 0 = Pins designated for output by DDRQS operate in normal mode. 1 = Pins designated for output by DDRQS operate in open drain mode.
2:5	BITS	Bits per transfer. In master mode, when BITSE is set in a command RAM byte, BITS determines the number of data bits transferred. When BITSE is cleared, eight bits are transferred regardless of the value in BITS. In slave mode, the BITS field always determines the number of bits the QSPI will receive during each transfer before storing the received data. Data transfers from 8 to 16 bits are supported. Illegal (reserved) values default to eight bits. **Table 14-14** shows the number of bits per transfer.
6	CPOL	Clock polarity. CPOL is used to determine the inactive state of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices. 0 = The inactive state of SCK is logic zero. 1 = The inactive state of SCK is logic one.
7	CPHA	Clock phase. CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices. 0 = Data is captured on the leading edge of SCK and changed on the trailing edge of SCK. 1 = Data is changed on the leading edge of SCK and captured on the trailing edge of SCK
8:15	SPBR	Serial clock baud rate. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into SPBR. The following equation determines the SCK baud rate: $\text{SCK Baud Rate} = \frac{f_{SYS}}{2 \times SPBR}$ Refer to **14.7.5.2 Baud Rate Selection** for more information.

Table 14-14 Bits Per Transfer

BITS[3:0]	Bits per Transfer
0000	16
0001 to 0111	Reserved (defaults to 8)
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

14.7.1.2 QSPI Control Register 1

SPCR1 enables the QSPI and specifies transfer delays. The CPU has read/write access to SPCR1, but the QSPI has read access only to all bits except SPE. SPCR1 must be written last during initialization because it contains SPE. The QSPI automati-

cally clears this bit after it completes all serial transfers or when a mode fault occurs. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

SPCR1 — QSPI Control Register 1 **0x30 501A**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPE	DSCKL							DTL							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 14-15 SPCR1 Bit Settings

Bit(s)	Name	Description
0	SPE	QSPI enable. Refer to **14.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O. 1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.
1:7	DSCKL	Delay before SCK. When the DSCK bit is set in a command RAM byte, this field determines the length of the delay from PCS valid to SCK transition. The following equation determines the actual delay before SCK: $\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$ where DSCKL equals is in the range of 1 to 127. Refer to **14.7.5.3 Delay Before Transfer** for more information.
8:15	DTL	Length of delay after transfer. When the DT bit is set in a command RAM byte, this field determines the length of the delay after a serial transfer. The following equation is used to calculate the delay: $\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$ where DTL is in the range of 1 to 255. A zero value for DTL causes a delay-after-transfer value of 8192 ÷ fSYS (204.8 µs with a 40-MHz system clock). Refer to **14.7.5.4 Delay After Transfer** for more information.

14.7.1.3 QSPI Control Register 2

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU has read/write access to SPCR2, but the QSPI has read access only. Writes to this register are buffered. New SPCR2 values become effective only after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. Reads of SPCR2 return the current value of the register, not the buffer.

SPCR2 — QSPI Control Register 2 **0x30 501C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPIFIE	WREN	WRTO	ENDQP					Reserved			NEWQP				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-16 SPCR2 Bit Settings

Bit(s)	Name	Description
0	SPIFIE	SPI finished interrupt enable. Refer to **14.7.4.2 QSPI Interrupts**. 0 = QSPI interrupts disabled 1 = QSPI interrupts enabled
1	WREN	Wrap enable. Refer to **14.7.5.7 Master Wraparound Mode**. 0 = Wraparound mode disabled. 1 = Wraparound mode enabled.
2	WRTO	Wrap to. When wraparound mode is enabled and after the end of queue has been reached, WRTO determines which address the QSPI executes next. The end of queue is determined by an address match with ENDQP. 0 = Wrap to pointer address 0x0 1 = Wrap to address in NEWQP
3:7	ENDQP	Ending queue pointer. This field determines the last absolute address in the queue to be completed by the QSPI. After completing each command, the QSPI compares the queue pointer value of the just-completed command with the value of ENDQP. If the two values match, the QSPI sets SPIF to indicate it has reached the end of the programmed queue. Refer to **14.7.4 QSPI Operation** for more information.
8:10	—	Reserved
11:15	NEWQP	New queue pointer value. This field contains the first QSPI queue address. Refer to **14.7.4 QSPI Operation** for more information.

14.7.1.4 QSPI Control Register 3

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enable, and the halt control bit. The CPU has read/write access to SPCR3, but the QSPI has read access only. SPCR3 must be initialized before QSPI operation begins. Writing a new value to SPCR3 while the QSPI is enabled disrupts operation.

SPCR3 — QSPI Control Register **0x30 501E**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Reserved					LOOPQ	HMIE	HALT	SPSR*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 14-18**.

Table 14-17 SPCR3 Bit Settings

Bit(s)	Name	Description
0:4	—	Reserved
5	LOOPQ	QSPI loop mode. LOOPQ controls feedback on the data serializer for testing. 0 = Feedback path disabled. 1 = Feedback path enabled.
6	HMIE	HALTA and MODF interrupt enable. HMIE enables interrupt requests generated by the HALTA status flag or the MODF status flag in SPSR. 0 = HALTA and MODF interrupts disabled. 1 = HALTA and MODF interrupts enabled.
7	HALT	Halt QSPI. When HALT is set, the QSPI stops on a queue boundary. It remains in a defined state from which it can later be restarted. Refer to **14.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI operates normally. 1 = QSPI is halted for subsequent restart.
8:15	—	SPSR. See**Table 14-18** for bit descriptions.

14.7.1.5 QSPI Status Register

The SPSR contains information concerning the current serial transmission. Only the QSPI can set bits in this register. To clear status flags, the CPU reads SPSR with the flags set and then writes the SPSR with zeros in the appropriate bits. Writes to CPTQP have no effect.

SPSR — QSPI Status Register **0x30 501E**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPCR3*								SPIF	MODF	HALTA	CPTQP				
								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 14-17**.

Table 14-18 SPSR Bit Settings

Bit(s)	Name	Description
0:7	SPCR3	See bit descriptions in **Table 14-17**.
8	SPIF	QSPI finished flag. SPIF is set after execution of the command at the address in ENDQP in SPCR2. If wraparound mode is enabled (WREN = 1), the SPIF is set, after completion of the command defined by ENDQP, each time the QSPI cycles through the queue. 0 = QSPI is not finished 1 = QSPI is finished
9	MODF	Mode fault flag. The QSPI asserts MODF when the QSPI is in master mode (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver. Refer to **14.7.8 Mode Fault** for more information. 0 = Normal operation 1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ($\overline{SS}$ input taken low).
10	HALTA	Halt acknowledge flag. HALTA is set when the QSPI halts in response to setting the HALT bit in SPCR3. HALTA is also set when the IMB3 FREEZE signal is asserted, provided the FRZ1 bit in the QSMCMMCR is set. To prevent undefined operation, the user must not modify any QSPI control registers or RAM while the QSPI is halted. If HMIE in SPCR3 is set the QSPI sends interrupt requests to the CPU when HALTA is asserted. 0 = QSPI is not halted. 1 = QSPI is halted
11:15	CPTQP	Completed queue pointer. CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value 0x0 or a pointer to the last command completed in the previous queue. If the QSPI is halted, CPTQP may be used to determine which commands have not been executed. The CPTQP may also be used to determine which locations in the receive data segment of the QSPI RAM contain valid received data.

14.7.2 QSPI RAM

The QSPI contains a 160-byte block of dual-ported static RAM that can be accessed by both the QSPI and the CPU. Because of this dual access capability, up to two wait states may be inserted into CPU access time if the QSPI is in operation.

The size and type of access of the QSPI RAM by the CPU affects the QSPI access time. The QSPI allows byte, half-word, and word accesses. Only word accesses of the RAM by the CPU are coherent because these accesses are an indivisible operation. If the CPU makes a coherent access of the QSPI RAM, the QSPI cannot access the QSPI RAM until the CPU is finished. However, a word or misaligned word access is not coherent because the CPU must break its access of the QSPI RAM into two parts, which allows the QSPI to access the QSPI RAM between the two accesses by the CPU.

The RAM is divided into three segments: receive data RAM, transmit data RAM, and command data RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored for transmission to an external device. Command data defines transfer parameters. **Figure 14-5** shows RAM organization.

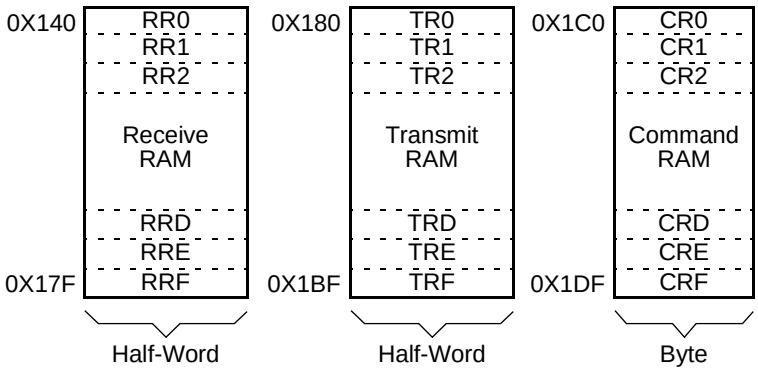


Figure 14-5 QSPI RAM

14.7.2.1 Receive RAM

Data received by the QSPI is stored in this segment, to be read by the CPU. Data stored in the receive RAM is right-justified, i.e., the least significant bit is always in the right-most bit position within the word regardless of the serial transfer length. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, half-word, or word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

14.7.2.2 Transmit RAM

Data that is to be transmitted by the QSPI is stored in this segment. The CPU normally writes one word of data into this segment for each queue command to be executed. If the corresponding peripheral, such as a serial input port, is used solely to input data, then this segment does not need to be initialized.

Data must be written to transmit RAM in a right-justified format. The QSPI cannot modify information in the transmit RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

14.7.2.3 Command RAM

Command RAM is used by the QSPI in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 32 bytes. Each byte is divided into two fields. The peripheral chip-select field, enables peripherals for transfer. The command control field provides transfer options.

A maximum of 32 commands can be in the queue. These bytes are assigned an address from 0x00 to 0x1F. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP. (Both of these fields are in SPCR2.)

CR[0:F] — Command RAM **0x30 51C0 – 0x30 51DF**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
Command Control				Peripheral Chip Select			

NOTES:
1. The PCS0 bit represents the dual-function PCS0/$\overline{SS}$.

Table 14-19 Command RAM Bit Settings

Bit(s)	Name	Description
0	CONT	Continue 0 = Control of chip selects returned to PORTQS after transfer is complete. 1 = Peripheral chip selects remain asserted after transfer is complete.
1	BITSE	Bits per transfer enable 0 = Eight bits 1 = Number of bits set in BITS field of SPCR0.
2	DT	Delay after transfer 0 = Delay after transfer is 17 ÷ fSYS. 1 = SPCR1 DTL[7:0] specifies delay after transfer PCS valid to SCK.
3	DSCK	PCS to SCK Delay 0 = PCS valid to SCK delay is one-half SCK. 1 = SPCR1 DSCKL[6:0] specifies delay from PCS valid to SCK.
4:7	PCS[3:0]	Peripheral chip selects. Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select may be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided proper fanout is observed. PCS0 shares a pin with the slave select ($\overline{SS}$) signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

Refer to **14.7.5 Master Mode Operation** for more information on the command RAM.

14.7.3 QSPI Pins

Seven pins are associated with the QSPI. When not needed by the QSPI, they can be configured for general-purpose I/O. **Table 14-20** identifies the QSPI pins and their functions. Register DDRQS determines whether the pins are designated as input or output. The user must initialize DDRQS for the QSPI to function correctly.

Table 14-20 QSPI Pin Functions

Pin Names	Mnemonic	Mode	Function
Master in slave out	MISO	Master Slave	Serial data input to QSPI Serial data output from QSPI
Master out slave in	MOSI	Master Slave	Serial data output from QSPI Serial data input to QSPI
Serial clock	SCK[1]	Master Slave	Clock output from QSPI clock Input to QSPI
Peripheral chip selects	PCS[1:3]	Master	Outputs select peripheral(s)
Peripheral chip select[2] Slave select[3]	PCS0/ $\overline{SS}$	Master Slave	Output selects peripheral(s) Input selects the QSPI
Slave select[4]	$\overline{SS}$	Master	May cause mode fault

NOTES:
1. All QSPI pins (except SCK) can be used as general-purpose I/O if they are not used by the QSPI while the QSPI is operating. SCK can only be used for general-purpose I/O if the QSPI is disabled.
2. An output (PCS0) when the QSPI is in master mode.
3. An input ($\overline{SS}$) when the QSPI is in slave mode.
4. An input ($\overline{SS}$) when the QSPI is in master mode; useful in multimaster systems.

14.7.4 QSPI Operation

The QSPI uses a dedicated 160-byte block of static RAM accessible by both the QSPI and the CPU to perform queued operations. The RAM is divided into three segments: 32 command control bytes, 64 transmit data bytes, and 64 receive data bytes.

Once the CPU has set up a queue of QSPI commands, written the transmit data segment with information to be sent, and enabled the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating completion, and then either interrupts the CPU or waits for CPU intervention.

QSPI RAM is organized so that one byte of command data, one word of transmit data, and one word of receive data correspond to each queue entry, 0x0 to 0x2F.

The CPU initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU or waits for intervention.

There are four queue pointers. The CPU can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), contained in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

The internal pointer is initialized to the same value as NEWQP. During normal operation, the command pointed to by the internal pointer is executed, the value in the internal pointer is copied into CPTQP, the internal pointer is incremented, and then the

sequence repeats. Execution continues at the internal pointer address unless the NEWQP value is changed. After each command is executed, ENDQP and CPTQP are compared. When a match occurs, the SPIF flag is set and the QSPI stops and clears SPE, unless wraparound mode is enabled.

At reset, NEWQP is initialized to 0x0. When the QSPI is enabled, execution begins at queue address 0x0 unless another value has been written into NEWQP. ENDQP is initialized to 0x0 at reset but should be changed to the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When NEWQP changes, the internal pointer value also changes. However, if NEWQP is written while a transfer is in progress, the transfer is completed normally. Leaving NEWQP and ENDQP set to 0x0 transfers only the data in transmit RAM location 0x0.

14.7.4.1 Enabling, Disabling, and Halting the SPI

The SPE bit in the SPCR1 enables or disables the QSPI submodule. Setting SPE causes the QSPI to begin operation. If the QSPI is a master, setting SPE causes the QSPI to begin initiating serial transfers. If the QSPI is a slave, the QSPI begins monitoring the PCS0/$\overline{SS}$ pin to respond to the external initialization of a serial transfer.

When the QSPI is disabled, the CPU may use the QSPI RAM. When the QSPI is enabled, both the QSPI and the CPU have access to the QSPI RAM. The CPU has both read and write access to all 160 bytes of the QSPI RAM. The QSPI can read-only the transmit data segment and the command control segment and can write-only the receive data segment of the QSPI RAM.

The QSPI turns itself off automatically when it is finished by clearing SPE. An error condition called mode fault (MODF) also clears SPE. This error occurs when PCS0/$\overline{SS}$ is configured for input, the QSPI is a system master (MSTR = 1), and PCS0/$\overline{SS}$ is driven low externally.

Setting the HALT bit in SPCR3 stops the QSPI on a queue boundary. The QSPI halts in a known state from which it can later be restarted. When HALT is set, the QSPI finishes executing the current serial transfer (up to 16 bits) and then halts. While halted, if the command control bit (CONT of the QSPI RAM) for the last command was asserted, the QSPI continues driving the peripheral chip select pins with the value designated by the last command before the halt. If CONT was cleared, the QSPI drives the peripheral chip-select pins to the value in register PORTQS.

If HALT is set during the last command in the queue, the QSPI completes the last command, sets both HALTA and SPIF, and clears SPE. If the last queue command has not been executed, asserting HALT does not set SPIF or clear SPE. QSPI execution continues when the CPU clears HALT.

To stop the QSPI, assert the HALT bit in SPCR3, then wait until the HALTA bit in SPSR is set. SPE can then be safely cleared, providing an orderly method of shutting down the QSPI quickly after the current serial transfer is completed. The CPU can disable the QSPI immediately by clearing SPE. However, loss of data from a current serial transfer may result and confuse an external SPI device.

14.7.4.2 QSPI Interrupts

The QSPI has three possible interrupt sources but only one interrupt vector. These sources are SPIF, MODF, and HALTA. When the CPU responds to a QSPI interrupt, the user must ascertain the interrupt cause by reading the SPSR. Any interrupt that was set may then be cleared by writing to SPSR with a zero in the bit position corresponding to the interrupt source.

The SPIFIE bit in SPCR2 enables the QSPI to generate an interrupt request upon assertion of the SPIF status flag. Because it is buffered, the value written to SPIFIE applies only upon completion of the queue (the transfer of the entry indicated by ENDPQ). Thus, if a single sequence of queue entries is to be transferred (i.e., no WRAP), then SPIFIE should be set to the desired state before the first transfer.

If a sub-queue is to be used, the same CPU write that causes a branch to the sub-queue may enable or disable the SPIF interrupt for the sub-queue. The primary queue retains its own selected interrupt mode, either enabled or disabled.

The SPIF interrupt must be cleared by clearing SPIF. Subsequent interrupts may then be prevented by clearing SPIFIE. Clearing SPIFIE does not immediately clear an interrupt already caused by SPIF.

14.7.4.3 QSPI Flow

The QSPI operates in either master or slave mode. Master mode is used when the MCU initiates data transfers. Slave mode is used when an external device initiates transfers. Switching between these modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSMCM and QSPI registers must be initialized properly.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from the transmit RAM and received by the receive RAM.

In slave mode, operation proceeds in response to $\overline{SS}$ pin assertion by an external SPI bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multi-master operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

Figure 14-6 shows QSPI initialization. **Figure 14-7** through **Figure 14-11** show QSPI master and slave operation. The CPU must initialize the QSMCM global and pin registers and the QSPI control registers before enabling the QSPI for either mode of operation. The command queue must be written before the QSPI is enabled for master

mode operation. Any data to be transmitted should be written into transmit RAM before the QSPI is enabled. During wraparound operation, data for subsequent transmissions can be written at any time.

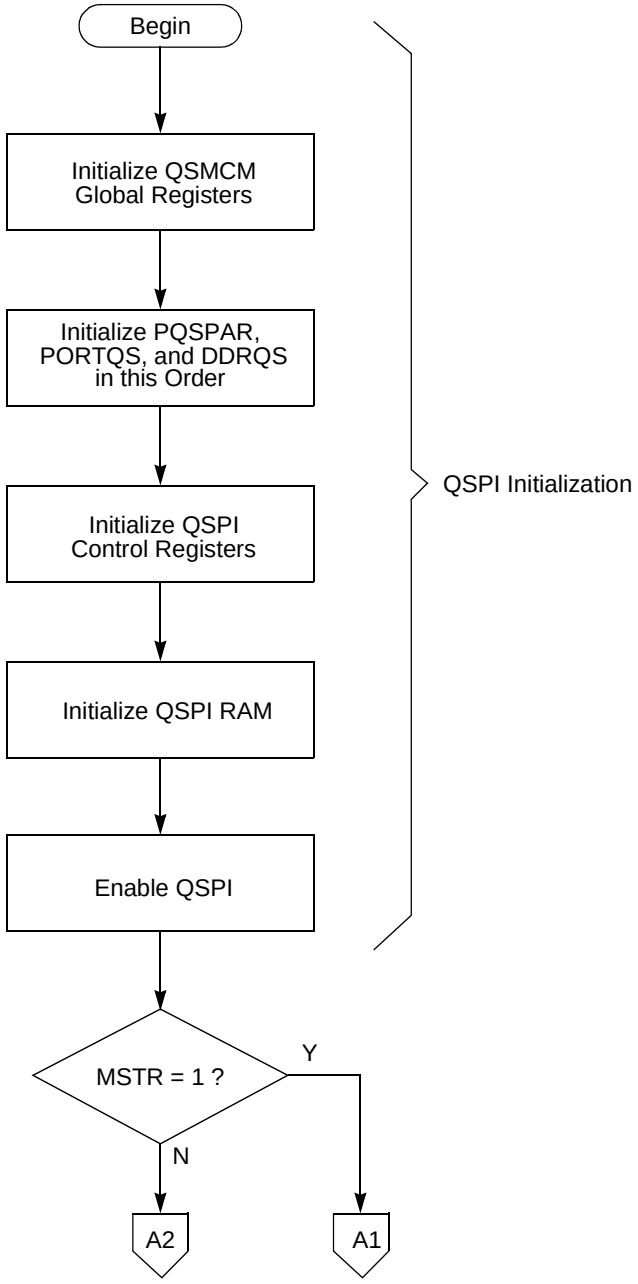


Figure 14-6 Flowchart of QSPI Initialization Operation

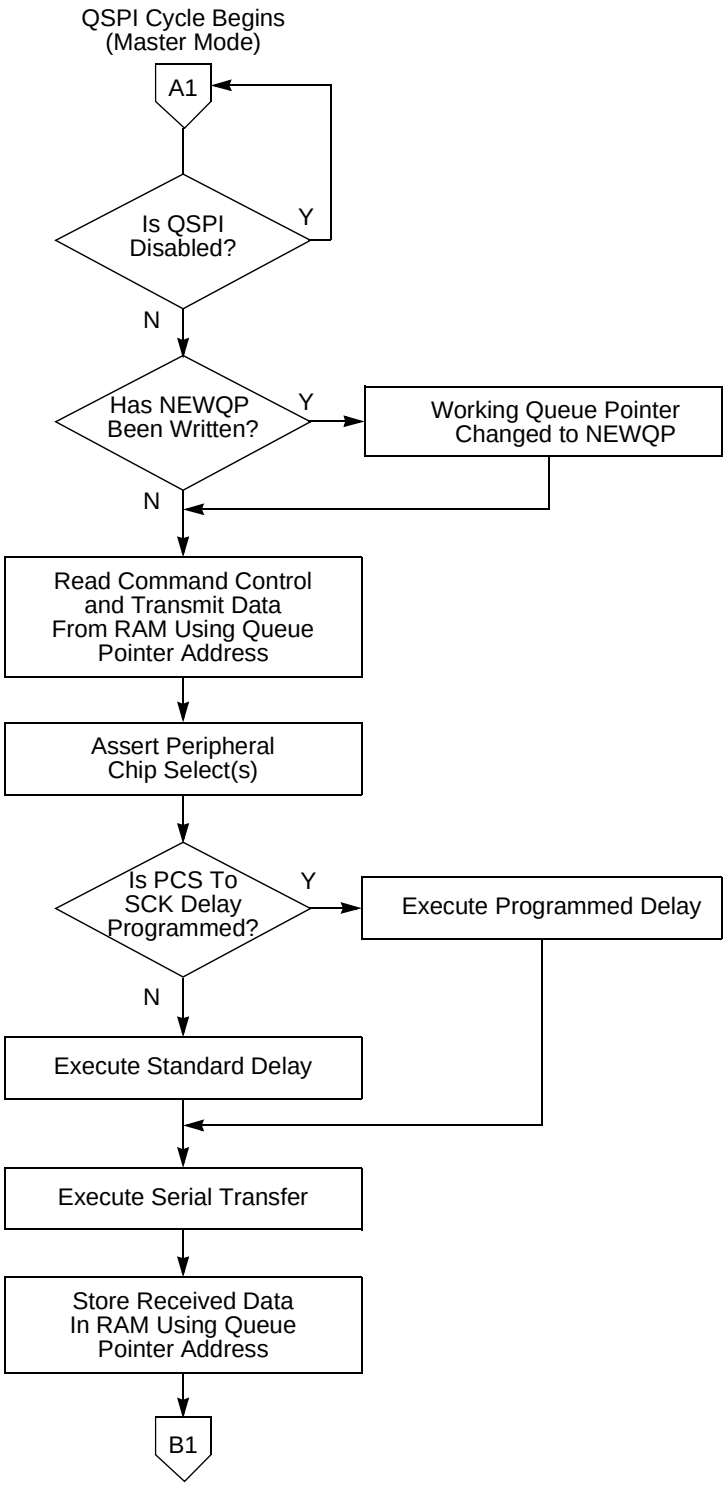


Figure 14-7 Flowchart of QSPI Master Operation (Part 1)

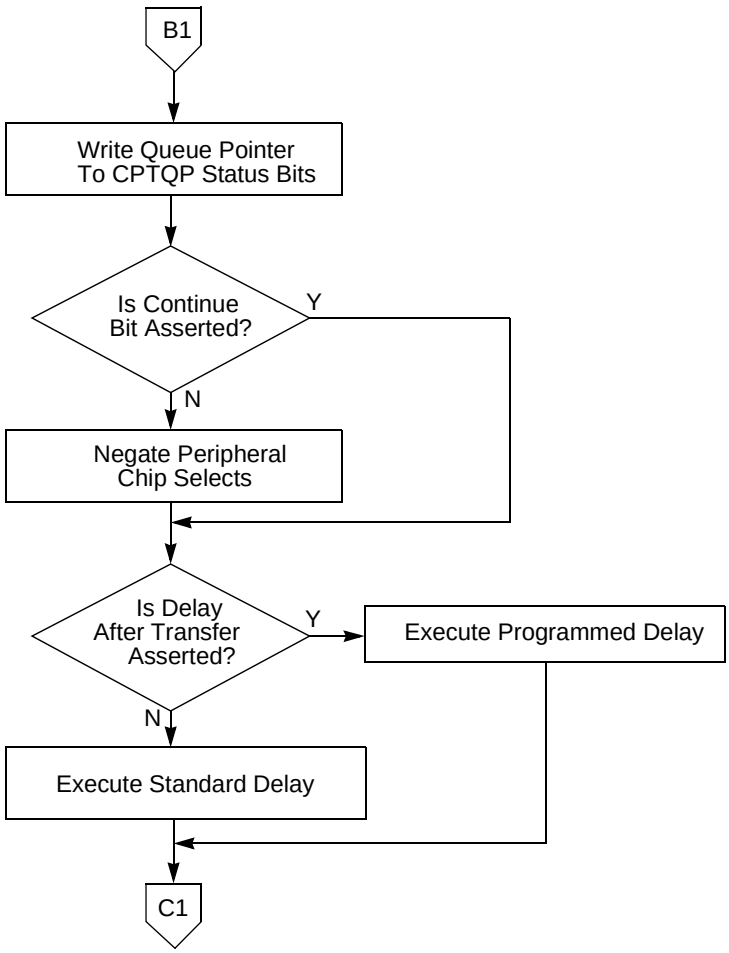


Figure 14-8 Flowchart of QSPI Master Operation (Part 2)

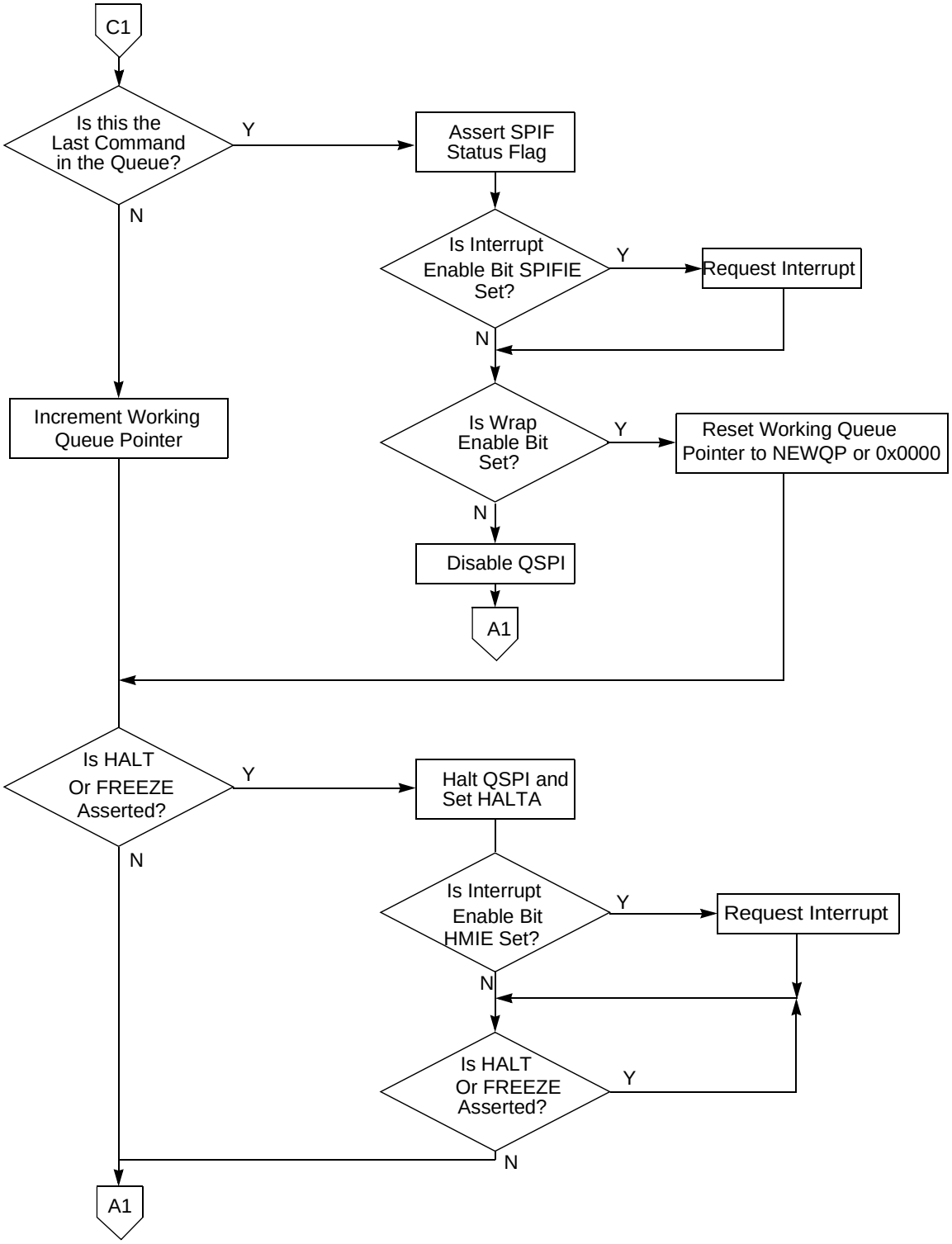


Figure 14-9 Flowchart of QSPI Master Operation (Part 3)

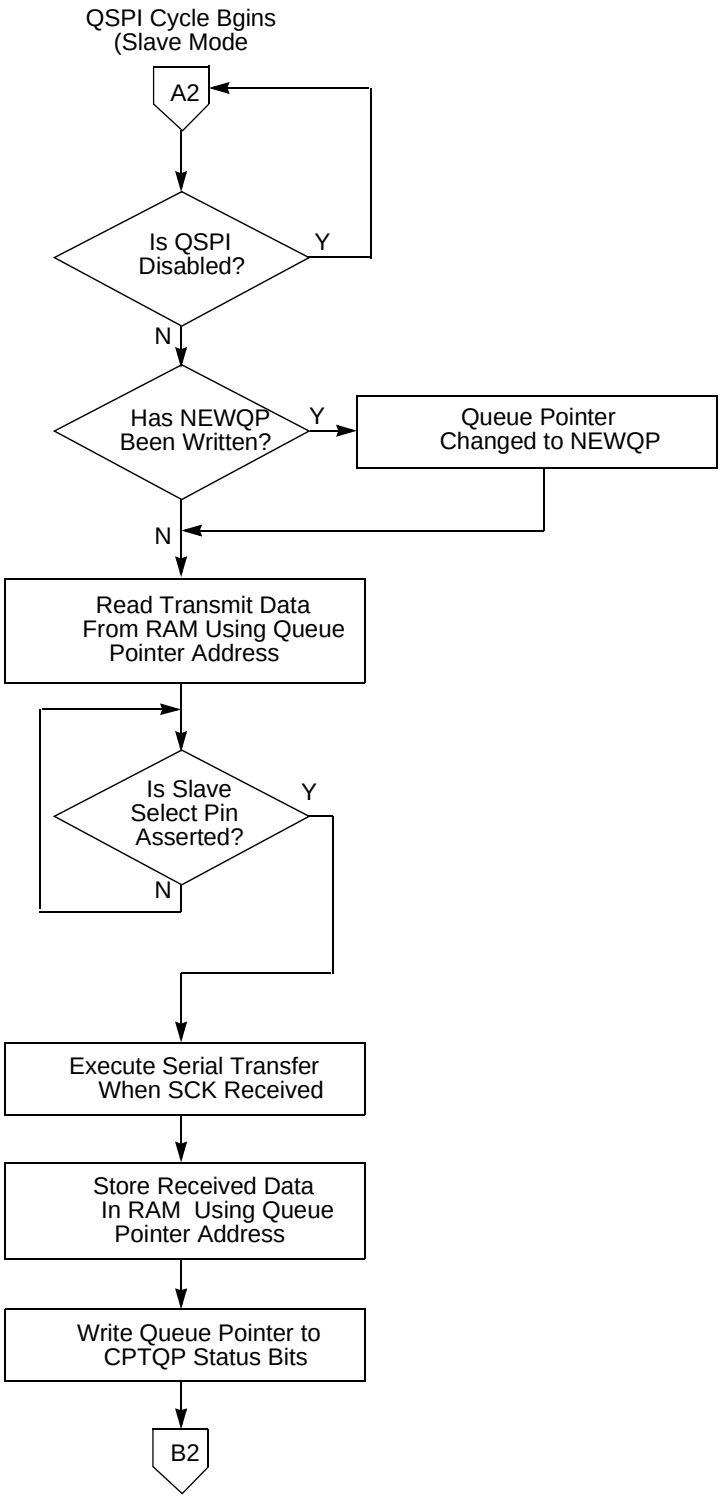


Figure 14-10 Flowchart of QSPI Slave Operation (Part 1)

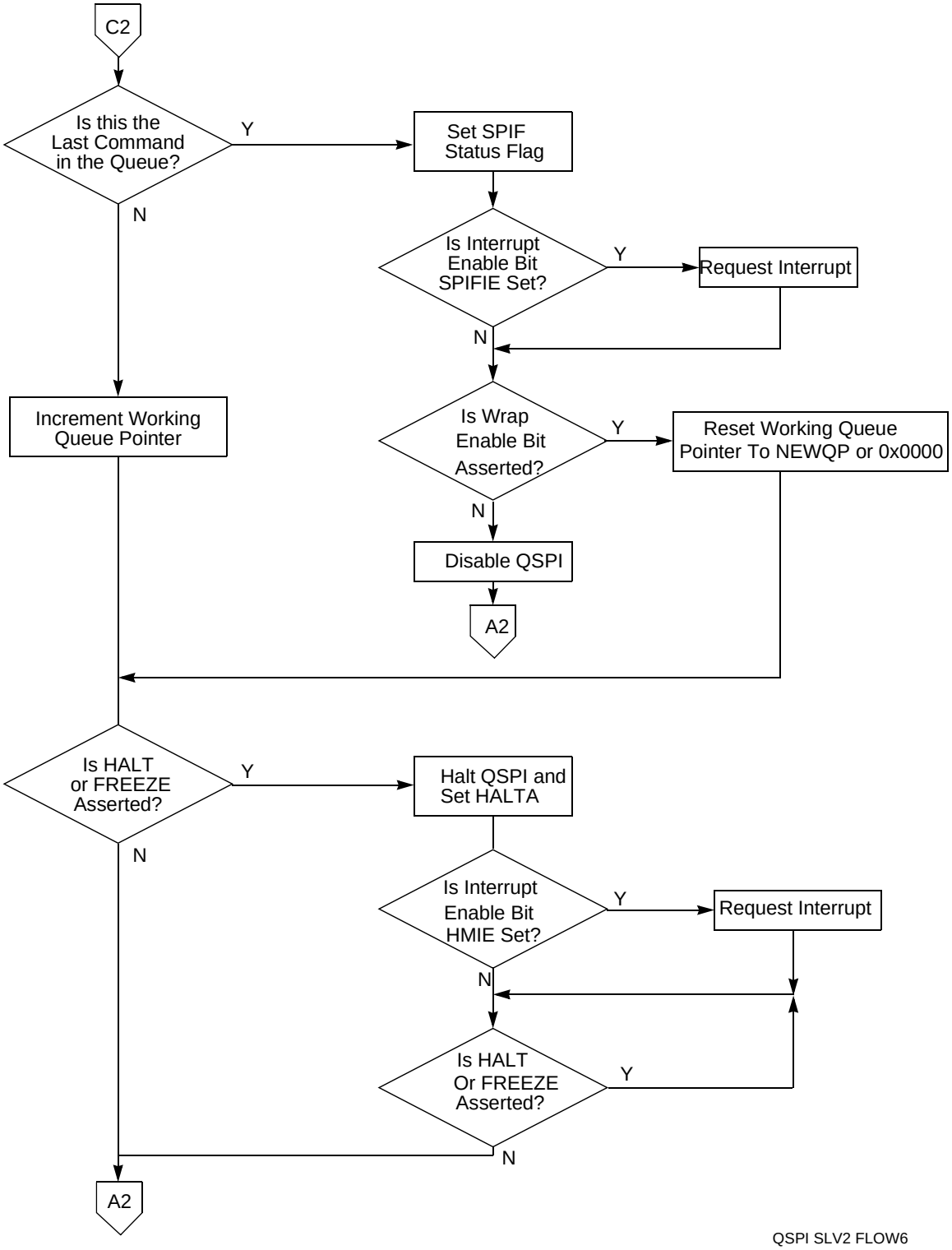


Figure 14-11 Flowchart of QSPI Slave Operation (Part 2)

Normally, the SPI bus performs synchronous bi-directional transfers. The serial clock on the SPI bus master supplies the clock signal SCK to time the transfer of data. Four

possible combinations of clock phase and polarity can be specified by the CPHA and CPOL bits in SPCR0.

Data is transferred with the most significant bit first. The number of bits transferred per command defaults to eight, but can be set to any value from eight to sixteen bits by writing a value into the BITS field in SPCR0 and setting BITSE in command RAM.

Typically, SPI bus outputs are not open drain unless multiple SPI masters are in the system. If needed, the WOMQ bit in SPCR0 can be set to provide wired-OR, open drain outputs. An external pull-up resistor should be used on each output line. WOMQ affects all QSPI pins regardless of whether they are assigned to the QSPI or used as general-purpose I/O.

14.7.5 Master Mode Operation

Setting the MSTR bit in SPCR0 selects master mode operation. In master mode, the QSPI 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.

Before QSPI operation begins, PQSPAR must be written to assign the necessary pins to the QSPI. The pins necessary for master mode operation are MISO, MOSI, SCK, and one or more of the chip-select pins. 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. SCK is the serial clock output in master mode and must be assigned to the QSPI for proper operation.

The PORTQS data register must next be written with values that make the QGPIO6/SCK (bit 13 QDSCK of PORTQS) and QGPIO[3:0]/PCS[3:0] (bits 12:9 QDPCS[3:0] of PORTQS) outputs inactive when the QSPI completes a series of transfers. Pins allocated to the QSPI by PQSPAR are controlled by PORTQS when the QSPI is inactive. PORTQS I/O pins driven to states opposite those of the inactive QSPI signals can generate glitches that momentarily enable or partially clock a slave device.

For example, if a slave device operates with an inactive SCK state of logic one (CPOL = 1) and uses active low peripheral chip-select PCS0, the QDSCK and QDPCS0 bits in PORTQS must be set to 0b11. If QDSCK and QDPCS0 = 0b00, falling edges will appear on QGPIO6/SCK and GPIO0/PCS0 as the QSPI relinquishes control of these pins and PORTQS drives them to logic zero from the inactive SCK and PCS0 states of logic one.

Before master mode operation is initiated, QSMCM register DDRQS is written last to direct the data flow on the QSPI pins used. Configure the SCK, MOSI and appropriate chip-select pins PCS[3:0] as outputs. The MISO pin must be configured as an input.

After pins are assigned and configured, write appropriate data to the command queue. If data is to be transmitted, write the data to transmit RAM. Initialize the queue pointers as appropriate.

QSPI operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

When the CONT bit in a command RAM byte is set, PCS pins are continuously driven to specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers. The data in PORTQS must match the inactive states of SCK and any peripheral chip-selects used.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

14.7.5.1 Clock Phase and Polarity

In master mode, data transfer is synchronized with the internally-generated serial clock SCK. Control bits, CPHA and CPOL, in SPCR0, control clock phase and polarity. Combinations of CPHA and CPOL determine upon which SCK edge to drive outgoing data from the MOSI pin and to latch incoming data from the MISO pin.

14.7.5.2 Baud Rate Selection

Baud rate is selected by writing a value from two to 255 into the SPBR field in SPCR0. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU system clock.

The following expressions apply to the SCK baud rate:

$$\text{SCK Baud Rate} = \frac{f_{SYS}}{2 \times \text{SPBR}}$$

or

$$\text{SPBR} = \frac{f_{SYS}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state. At reset, the SCK baud rate is initialized to one eighth of the system clock frequency.

Table 14-21 provides some example SCK baud rates with a 40-MHz system clock.

Table 14-21 Example SCK Frequencies with a 40-MHz System Clock

Division Ratio	SPBR Value	SCK Frequency
4	2	10.00 MHz
6	3	6.67 MHz
8	4	5.00 MHz
14	7	2.86 MHz
28	14	1.43 MHz
58	29	689 kHz
280	140	143 kHz
510	255	78.43 kHz

14.7.5.3 Delay Before Transfer

The DSCK bit in each command RAM byte inserts either a standard (DSCK = 0) or user-specified (DSCK = 1) delay from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the length of the user-defined delay before the assertion of SCK. The following expression determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$$

where DSCKL is in the range from 1 to 127.

NOTE

A zero value for DSCKL causes a delay of 128 system clocks, which equals 3.2 µs for a 40-MHz system clock. Because of design limits, a DSCKL value of one defaults to the same timing as a value of two.

When DSCK equals zero, DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half the SCK period.

14.7.5.4 Delay After Transfer

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion. Writing a value to the DTL field in SPCR1 specifies a delay period. The DT bit in each command RAM byte determines whether the standard delay period (DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$$

where DTL is in the range from one to 255.

A zero value for DTL causes a delay-after-transfer value of 8192 ÷ system clock frequency (204.8 µs with a 40-MHz system clock).

If DT is zero in a command RAM byte, a standard delay is inserted.

$$\text{Standard Delay after Transfer} = \frac{17}{f_{SYS}}$$

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

Adequate delay between transfers must be specified for long data streams because the QSPI requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

14.7.5.5 Transfer Length

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value from eight (0b1000) to 16 (0b0000) bits, inclusive. Reserved values (from 0b0001 to 0b0111) default to eight bits. The programmed value must be written into the BITS field in SPCR0. The BITSE bit in each command RAM byte determines whether the default value (BITSE = 0) or the BITS value (BITSE = 1) is used.

14.7.5.6 Peripheral Chip Selects

Peripheral chip-select signals are used to select an external device for serial data transfer. Chip-select signals are asserted when a command in the queue is executed. Signals are asserted at a logic level corresponding to the value of the PCS[3:0] bits in each command byte. More than one chip-select signal can be asserted at a time, and more than one external device can be connected to each PCS pin, provided proper fanout is observed. PCS0 shares a pin with the slave select $\overline{SS}$ signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

To configure a peripheral chip select, set the appropriate bit in PQSPAR, then configure the chip-select pin as an output by setting the appropriate bit in DDRQS. The value of the bit in PORTQS that corresponds to the chip-select pin determines the base state of the chip-select signal. If the base state is zero, chip-select assertion must be active high (PCS bit in command RAM must be set); if base state is one, assertion must be active low (PCS bit in command RAM must be cleared). PORTQS bits are cleared dur-

ing reset. If no new data is written to PORTQS before pin assignment and configuration as an output, the base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

14.7.5.7 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wraparound mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven QSPI service is used, the service routine must clear the SPIF bit to end the current interrupt request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not end the current request.

Wraparound mode is exited by clearing the WREN bit or by setting the HALT bit in SPCR3. Exiting wraparound mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.

14.7.6 Slave Mode

Clearing the MSTR bit in SPCR0 selects slave mode operation. In slave mode, the QSPI is unable to initiate serial transfers. Transfers are initiated by an external SPI bus master. Slave mode is typically used on a multi-master SPI bus. Only one device can be bus master (operate in master mode) at any given time.

Before QSPI operation is initiated, QSMCM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO, MOSI, SCK, and PCS0/$\overline{SS}$. 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 serial clock input in slave mode and must be assigned to the QSPI for proper operation. Assertion of the active-low slave select signal $\overline{SS}$ initiates slave mode operation.

Before slave mode operation is initiated, DDRQS must be written to direct data flow on the QSPI pins used. Configure the MOSI, SCK and PCS0/$\overline{SS}$ pins as inputs. The MISO pin must be configured as an output.

After pins are assigned and configured, write data to be transmitted into transmit RAM. Command RAM is not used in slave mode, and does not need to be initialized. Set the queue pointers, as appropriate.

When SPE is set and MSTR is clear, a low state on the slave select PCS0/$\overline{SS}$ pin begins slave mode operation at the address indicated by NEWQP. Data that is received is stored at the pointer address in receive RAM. Data is simultaneously loaded into the data serializer from the pointer address in transmit RAM and transmitted. Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine upon which SCK edge to latch incoming data from the MISO pin and to drive outgoing data from the MOSI pin.

Because the command RAM is not used in slave mode, the CONT, BITSE, DT, DSCK, and peripheral chip-select bits have no effect. The PCS0/$\overline{SS}$ pin is used only as an input.

The SPBR, DT and DSCKL fields in SPCR0 and SPCR1 bits are not used in slave mode. The QSPI drives neither the clock nor the chip-select pins and thus cannot control clock rate or transfer delay.

Because the BITSE option is not available in slave mode, the BITS field in SPCR0 specifies the number of bits to be transferred for all transfers in the queue. When the number of bits designated by BITS[3:0] has been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads new transmit data from transmit RAM into the data serializer. The working queue pointer address is used the next time PCS0/$\overline{SS}$ is asserted, unless the RCPU writes to NEWQP first.

The QSPI shifts one bit for each pulse of SCK until the slave select input goes high. If $\overline{SS}$ goes high before the number of bits specified by the BITS field is transferred, the QSPI resumes operation at the same pointer address the next time $\overline{SS}$ is asserted. The maximum value that the BITS field can have is 16. If more than 16 bits are transmitted before $\overline{SS}$ is negated, pointers are incremented and operation continues.

The QSPI transmits as many bits as it receives at each queue address, until the BITS value is reached or $\overline{SS}$ is negated. $\overline{SS}$ does not need to go high between transfers as the QSPI transfers data until reaching the end of the queue, whether $\overline{SS}$ remains low or is toggled between transfers.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

Slave wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2. Slave wraparound operation is identical to master wraparound operation.

14.7.6.1 Description of Slave Operation

After reset, the QSMCM registers and the QSPI control registers must be initialized as described above. Although the command control segment is not used, the transmit and receive data segments may, depending upon the application, need to be initial-

ized. If meaningful data is to be sent out from the QSPI, the user should write the data to the transmit data segment before enabling the QSPI.

If SPE is set and MSTR is not set, a low state on the slave select (PCS0/$\overline{SS}$) pin commences slave mode operation at the address indicated by NEWQP. The QSPI transmits the data found in the transmit data segment at the address indicated by NEWQP, and the QSPI stores received data in the receive data segment at the address indicated by NEWQP. Data is transferred in response to an external slave clock input at the SCK pin.

Because the command control segment is not used, the command control bits and peripheral chip-select codes have no effect in slave mode operation. The QSPI does not drive any of the four peripheral chip-selects as outputs. PCS0/$\overline{SS}$ is used as an input.

Although CONT cannot be used in slave mode, a provision is made to enable receipt of more than 16 data bits. While keeping the QSPI selected (PCS0/$\overline{SS}$ is held low), the QSPI stores the number of bits, designated by BITS, in the current receive data segment address, increments NEWQP, and continues storing the remaining bits (up to the BITS value) in the next receive data segment address.

As long as PCS0/$\overline{SS}$ remains low, the QSPI continues to store the incoming bit stream in sequential receive data segment addresses, until either the value in BITS is reached or the end-of-queue address is used with wraparound mode disabled.

When the end of the queue is reached, the SPIF flag is asserted, optionally causing an interrupt. If wraparound mode is disabled, any additional incoming bits are ignored.

If wraparound mode is enabled, storing continues at either address 0x0 or the address of NEWQP, depending on the WRTO value. When using this capability to receive a long incoming data stream, the proper delay between transfers must be used. The QSPI requires time, approximately 0.425 µs with a 40-MHz system clock, to prefetch the next transmit RAM entry for the next transfer. Therefore, the user may select a baud rate that provides at least a 0.6-µs delay between successive transfers to ensure no loss of incoming data. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Because the BITSE option in the command control segment is no longer available, BITS sets the number of bits to be transferred for all transfers in the queue until the CPU changes the BITS value. As mentioned above, until PCS0/$\overline{SS}$ is negated (brought high), the QSPI continues to shift one bit for each pulse of SCK. If PCS0/$\overline{SS}$ is negated before the proper number of bits (according to BITS) is received, the next time the QSPI is selected it resumes storing bits in the same receive-data segment address where it left off. If more than 16 bits are transferred before negating the PCS0/$\overline{SS}$, the QSPI stores the number of bits indicated by BITS in the current receive data segment address, then increments the address and continues storing as described above. Note that PCS0/$\overline{SS}$ does not necessarily have to be negated between transfers.

Once the proper number of bits (designated by BITS) are transferred, the QSPI stores the received data in the receive data segment, stores the internal working queue pointer value in CPTQP, increments the internal working queue pointer, and loads the new transmit data from the transmit data segment into the data serializer. The internal working queue pointer address is used the next time PCS0/$\overline{SS}$ is asserted, unless the CPU writes to the NEWQP first.

The DT and DSCK command control bits are not used in slave mode. As a slave, the QSPI does not drive the clock line nor the chip-select lines and, therefore, does not generate a delay.

In slave mode, the QSPI shifts out the data in the transmit data segment. The transmit data is loaded into the data serializer (refer to **Figure 14-1**) for transmission. When the PCS0/$\overline{SS}$ pin is pulled low the MISO pin becomes active and the serializer then shifts the 16 bits of data out in sequence, most significant bit first, as clocked by the incoming SCK signal. The QSPI uses CPHA and CPOL to determine which incoming SCK edge the MOSI pin uses to latch incoming data, and which edge the MISO pin uses to drive the data out.

The QSPI transmits and receives data until reaching the end of the queue (defined as a match with the address in ENDQP), regardless of whether PCS0/SS remains selected or is toggled between serial transfers. Receiving the proper number of bits causes the received data to be stored. The QSPI always transmits as many bits as it receives at each queue address, until the BITS value is reached or PCS0/SS is negated.

14.7.7 Slave Wraparound Mode

When the QSPI reaches the end of the queue, it always sets the SPIF flag, whether wraparound mode is enabled or disabled. An optional interrupt to the CPU is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled. A description of SPIFIE bit can be found in 4.3.3 QSPI Control Register 2 (SPCR2).

In wraparound mode, the QSPI cycles through the queue continuously. Each time the end of the queue is reached, the SPIF flag is set. If the CPU fails to clear SPIF, it remains set, and the QSPI continues to send interrupt requests to the CPU (assuming SPIFIE is set). The user may avoid causing CPU interrupts by clearing SPIFIE.

As SPIFIE is buffered, clearing it after the SPIF flag is asserted does not immediately stop the CPU interrupts, but only prevents future interrupts from this source. To clear the current interrupt, the CPU must read QSPI register SPSR with SPIF asserted, followed by a write to SPSR with zero in SPIF (clear SPIF). Execution continues in wraparound mode even while the QSPI is requesting interrupt service from the CPU. The internal working queue pointer is incremented to the next address and the commands are executed again. SPE is not cleared by the QSPI. New receive data overwrites previously received data located in the receive data segment.

Wraparound mode is properly exited in two ways: a) The CPU may disable wraparound mode by clearing WREN. The next time end of the queue is reached, the QSPI

sets SPIF, clears SPE, and stops; and, b) The CPU sets HALT. This second method halts the QSPI after the current transfer is completed, allowing the CPU to negate SPE. The CPU can immediately stop the QSPI by clearing SPE; however, this method is not recommended, as it causes the QSPI to abort a serial transfer in process.

14.7.8 Mode Fault

MODF is asserted by the QSPI when the QSPI is the serial master (MSTR = 1) and the slave select (PCS0/$\overline{SS}$) input pin is pulled low by an external driver. This is possible only if the PCS0/$\overline{SS}$ pin is configured as input by QDDR. This low input to $\overline{SS}$ is not a normal operating condition. It indicates that a multimaster system conflict may exist, that another MCU is requesting to become the SPI network master, or simply that the hardware is incorrectly affecting PCS0/$\overline{SS}$. SPE in SPCR1 is cleared, disabling the QSPI. The QSPI pins revert to control by QPDR. If MODF is set and HMIE in SPCR3 is asserted, the QSPI generates an interrupt to the CPU.

The CPU may clear MODF by reading SPSR with MODF asserted, followed by writing SPSR with a zero in MODF. After correcting the mode fault problem, the QSPI can be re-enabled by asserting SPE.

The PCS0/$\overline{SS}$ pin may be configured as a general-purpose output instead of input to the QSPI. This inhibits the mode fault checking function. In this case, MODF is not used by the QSPI.

14.8 Serial Communication Interface

The dual, independent, serial communication interface (DSCI) communicates with external devices through an asynchronous serial bus. The two SCI modules are functionally equivalent, except that the SCI1 also provides 16-deep queue capabilities for the transmit and receive operations. The SCIs are fully compatible with other Motorola SCI systems. The DSCI has all of the capabilities of previous SCI systems as well as several significant new features.

Figure 14-12 is a block diagram of the SCI transmitter. **Figure 14-13** is a block diagram of the SCI receiver.

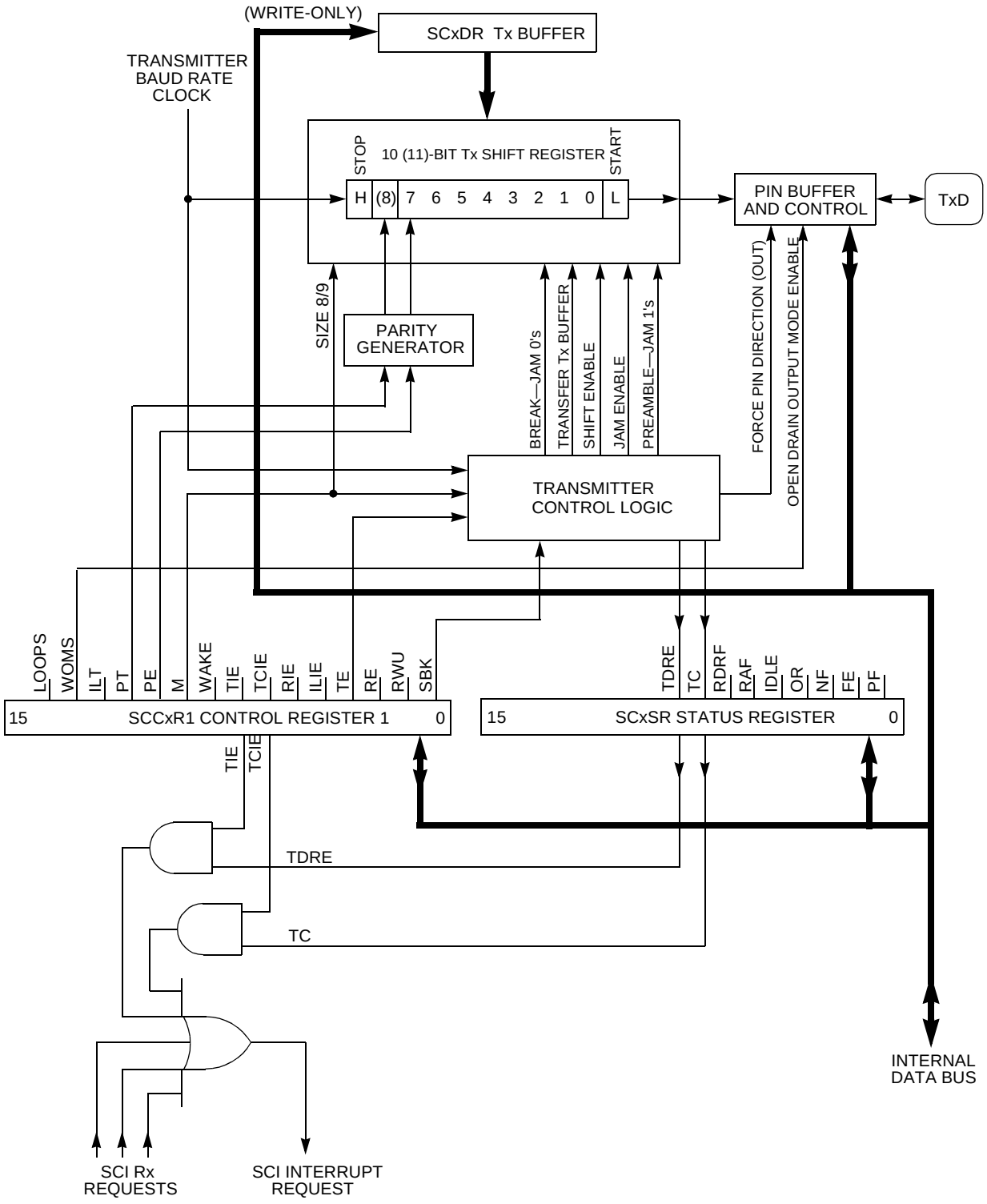


Figure 14-12 SCI Transmitter Block Diagram

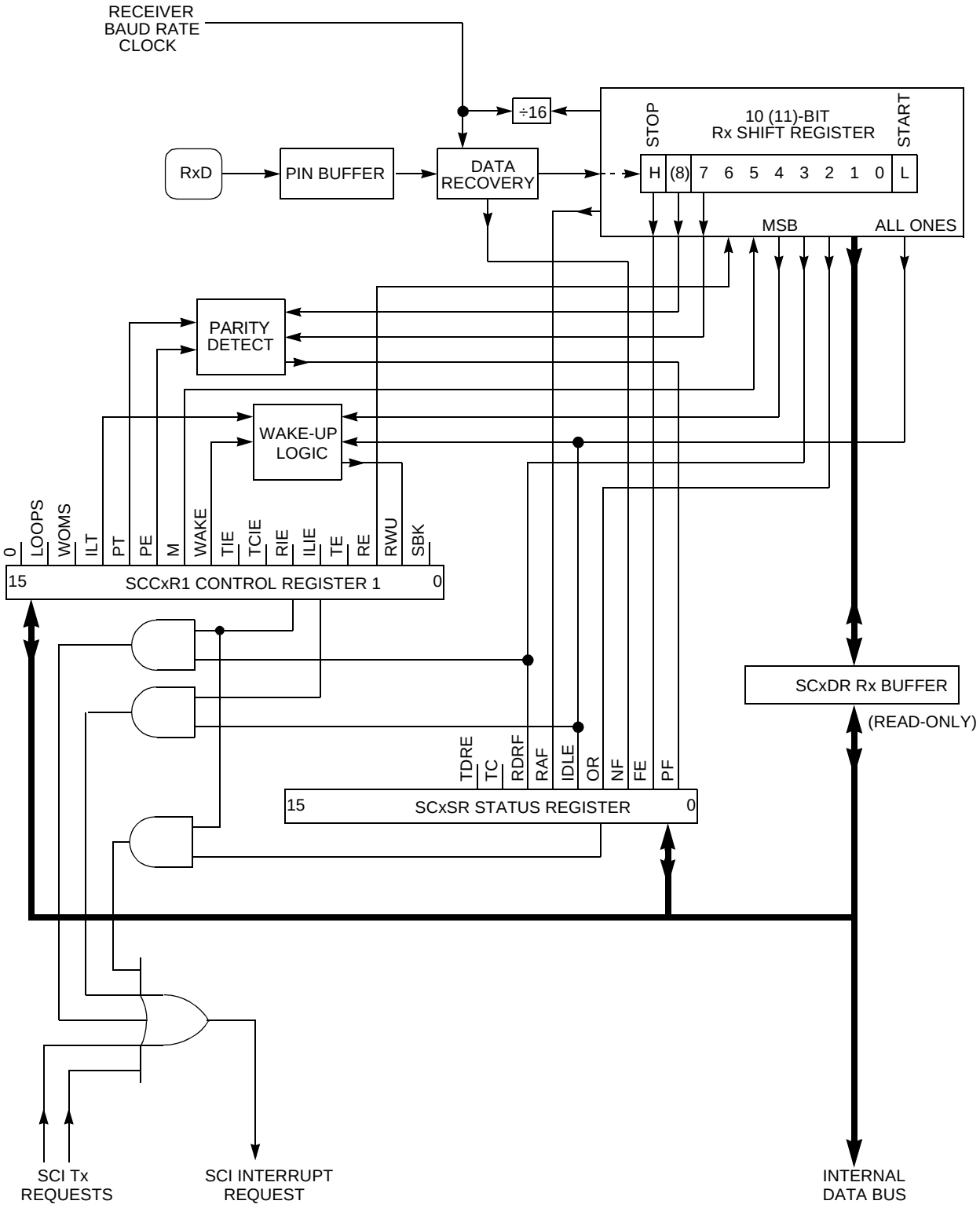


Figure 14-13 SCI Receiver Block Diagram

14.8.1 SCI Registers

The SCI programming model includes the QSMCM global and pin control registers and the DSCI registers.

The DSCI registers, listed in **Table 14-22**, consist of five control registers, three status registers, and 34 data registers. All registers may be read or written at any time by the CPU. Rewriting the same value to any DSCI register does not disrupt operation; however, writing a different value into a DSCI register when the DSCI 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 register SCxSR can be cleared at any time.

Table 14-22 SCI Registers

Address	Name	Usage
0x30 5008	SCC1R0	SCI1 Control Register 0 See **Table 14-23** for bit descriptions.
0x30 500A	SCC1R1	SCI1 Control Register 1 See **Table 14-24** for bit descriptions.
0x30 500C	SC1SR	SCI1 Status Register See **Table 14-25** for bit descriptions.
0x30 500E (non-queue mode only)	SC1DR	SCI1 Data Register Transmit Data Register (TDR1)* Receive Data Register (RDR1)* See **Table 14-26** for bit descriptions.
0x30 5020	SCC2R0	SCI2 Control Register 0
0x30 5022	SCC2R1	SCI2 Control Register 1
0x30 5024	SC2SR	SCI2 Status Register
0x30 5026	SC2DR	SCI2 Data Register Transmit Data Register (TDR2)* Receive Data Register (RDR2)*
0x30 5028	QSCI1CR	QSCI1 Control Register Interrupts, wrap, queue size and enables for receive and transmit, QTPNT. See **Table 14-33** for bit descriptions.
0x30 502A	QSCI1SR	QSCI1 Status Register OverRun error flag, queue status flags, QRPNT, and QPEND. See **Table 14-34** for bit descriptions.
0x30 502C — 0x30 504A	QSCI1 Transmit Queue Memory Area	QSCI1 Transmit Queue Data locations (on half-word boundary)
0x30 504C-6A	QSCI1 Receive Queue Memory Area	QSCI1 Receive Queue Data locations (on half-word boundary)

*Reads access the RDRx; writes access the TDRx.

During SCIx initialization, two bits in the SCCxR1 should be written last: the transmitter enable (TE) and receiver enable (RE) bits, which enable SCIx. Registers SCCxR0 and SCCxR1 should both be initialized at the same time or before TE and RE are asserted. A single half-word write to SCCxR1 can be used to initialize SCIx and enable the transmitter and receiver.

14.8.2 SCI Control Register 0

SCCxR0 contains the SCIx baud rate selection field and two bits controlling the clock source. The baud rate must be set before the SCI is enabled. The CPU can read and write SCCxR0 at any time.

Changing the value of SCCxR0 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR0 — SCI Control Register 0 **0x30 5008**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
OTHR	LNKBD	0	SCxBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 14-23 SCCxR0 Bit Settings

Bit(s)	Name	Description
0	OTHR	Select baud rate source clock other than system clock. OTHR determines which clock source is used by the baud rate generator—the internal system clock or another source determined by the LNKBD bit. Refer to **14.8.7.3 Baud Clock** for more information. 0 = Internal system clock source selected 1 = Other clock source selected, determined by LNKBD bit
1	LNKBD	Link baud. The SCI1 can use the resultant baud rate clock from SCI2 as the input clock source for the SCI baud rate generator or use the clock provided externally on the ECK pin. The link baud option is not available for SCI2. Refer to **14.8.7.3 Baud Clock** for more information. 0 = External clock selected 1 = Link baud selected
2	—	Reserved
3:15	SCxBR	SCI baud rate. The SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCxBR disables the baud rate generator. Baud clock rate is calculated as follows: $\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$ where SCxBR is in the range of 1 to 8191. Refer to **14.8.7.3 Baud Clock** for more information.

14.8.3 SCI Control Register 1

SCCxR1 contains SCIx configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read or write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCCxR1 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	LOOPS	WOMS	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

Table 14-24 SCCxR1 Bit Settings

Bit(s)	Name	Description
0	—	Reserved
1	LOOPS	Loop mode 0 = Normal SCI operation, no looping, feedback path disabled. 1 = SCI test operation, looping, feedback path enabled.
2	WOMS	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.
3	ILT	Idle-line detect type. Refer to **14.8.7.8 Idle-Line Detection**. 0 = Short idle-line detect (start count on first one). 1 = Long idle-line detect (start count on first one after stop bit(s)).
4	PT	Parity type. Refer to **14.8.7.4 Parity Checking**. 0 = Even parity. 1 = Odd parity.
5	PE	Parity enable. Refer to **14.8.7.4 Parity Checking**. 0 = SCI parity disabled. 1 = SCI parity enabled.
6	M	Mode select. Refer to **14.8.7.2 Serial Formats**. 0 = 10-bit SCI frame. 1 = 11-bit SCI frame.
7	WAKE	Wakeup by address mark. Refer to **14.8.7.9 Receiver Wake-Up**. 0 = SCI receiver awakened by idle-line detection. 1 = SCI receiver awakened by address mark (last bit set).
8	TIE	Transmit interrupt enable 0 = SCI TDRE interrupts disabled. 1 = SCI TDRE interrupts enabled.
9	TCIE	Transmit complete interrupt enable 0 = SCI TC interrupts disabled. 1 = SCI TC interrupts enabled.
10	RIE	Receiver interrupt enable 0 = SCI RDRF and OR interrupts disabled. 1 = SCI RDRF and OR interrupts enabled.
11	ILIE	Idle-line interrupt enable 0 = SCI IDLE interrupts disabled. 1 = SCI IDLE interrupts enabled.
12	TE	Transmitter enable 0 = SCI transmitter disabled (TXD pin can be used as general-purpose output) 1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter).
13	RE	Receiver Enable 0 = SCI receiver disabled (RXD pin can be used as general-purpose input). 1 = SCI receiver enabled (RXD pin is dedicated to SCI receiver).

Table 14-24 SCCxR1 Bit Settings (Continued)

Bit(s)	Name	Description
14	RWU	Receiver wakeup. Refer to **14.8.7.9 Receiver Wake-Up**. 0 = Normal receiver operation (received data recognized). 1 = Wakeup mode enabled (received data ignored until receiver is awakened).
15	SBK	Send break 0 = Normal operation. 1 = Break frame(s) transmitted after completion of current frame.

14.8.4 SCI Status Register (SCxSR)

SCxSR contains flags that show SCI operating conditions. These flags are cleared either by SCIx hardware or by a read/write sequence. The sequence consists of reading the SCxSR (either the upper byte, lower byte, or the entire half-word) with a flag bit set, then reading (or writing, in the case of flags TDRE and TC) the SCxDR (either the lower byte or the half-word).

The contents of the two 16-bit registers SCxSR and SCxDR appear as upper and lower half-words, respectively, when the SCxSR is read into a 32-bit register. An upper byte access of SCxSR is meaningful only for reads. Note that a word read can simultaneously access both registers SCxSR and SCxDR. This action clears the receive status flag bits that were set at the time of the read, but does not clear the TDRE or TC flags. To clear TC, the SCxSR read must be followed by a write to register SCxDR (either the lower byte or the half-word). The TDRE flag in the status register is read-only.

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 read or written the SCxDR, the newly set status bit is not cleared. Instead, SCxSR must be read again with the bit set and SCxDR must be read or written before the status bit is cleared.

NOTE

None of the status bits are cleared by reading a status bit while it is set and then writing zero to that same bit. Instead, the procedure outlined above must be followed. Note further that reading either byte of SCxSR causes all 16 bits to be accessed, and any status bits already set in either byte are armed to clear on a subsequent read or write of SCxDR.

SCxSR — SCIx Status Register **0x30 500C, 0x30 5024**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED							TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:															
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Table 14-25 SCxSR Bit Settings

Bit(s)	Name	Description
0:6	—	Reserved
7	TDRE	Transmit data register empty. TDRE is set when the byte in TDRx is transferred to the transmit serial shifter. If this bit is zero, the transfer is yet to occur and a write to TDRx will overwrite the previous value. New data is not transmitted if TDRx is written without first clearing TDRE. 0 = Transmit data register still contains data to be sent to the transmit serial shifter. 1 = A new character can now be written to the transmit data register. For transmit queue operation, this bit should be ignored by software.
8	TC	Transmit complete. TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle-line), or queued breaks (logic zero). 0 = SCI transmitter is busy. 1 = SCI transmitter is idle. For transmit queue operation, TC is cleared when SCxSR is read with TC set, followed by a write to SCTQ[0:15].
9	RDRF	Receive data register full. RDRF is set when the contents of the receive serial shifter are transferred to register RDRx. If one or more errors are detected in the received word, the appropriate flag(s) (NF, FE, or PF) are set within the same clock cycle. 0 = Receive data register is empty or contains previously read data. 1 = Receive data register contains new data. For receiver queue operation, this bit should be ignored by software.
10	RAF	Receiver active flag. RAF indicates whether the receiver is busy. This flag is set when the receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters. 0 = SCI receiver is idle. 1 = SCI receiver is busy.
11	IDLE	Idle line detected. IDLE is set when the receiver detects an idle-line condition (reception of a minimum of 10 or 11 consecutive ones as specified by ILT in SCCxR1). This bit is not set by the idle-line condition when RWU in SCCxR1 is set. Once cleared, IDLE is not set again until after RDRF is set (after the line is active and becomes idle again). If a break is received, RDRF is set, allowing a subsequent idle line to be detected again. Under certain conditions, the IDLE flag may be set immediately following the negation of RE in SCCxR1. 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. For receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15].
12	OR	Overrun error. OR is set when a new byte is ready to be transferred from the receive serial shifter to register RDRx, and RDRx is already full (RDRF is still set). Data transfer is inhibited until OR is cleared. Previous data in RDRx remains valid, but additional data received during an overrun condition (including the byte that set OR) is lost. Note that whereas the other receiver status flags (NF, FE, and PF) reflect the status of data already transferred to RDRx, the OR flag reflects an operational condition that resulted in a loss of data to RDRx. 0 = RDRF is cleared before new data arrives. 1 = RDRF is not cleared before new data arrives.

Table 14-25 SCxSR Bit Settings (Continued)

Bit(s)	Name	Description
13	NF	Noise error flag. NF is set when the receiver detects noise on a valid start bit, on any of the data bits, or on the stop bit(s). 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. Although no interrupt is explicitly associated with NF, an interrupt can be generated with RDRF, and the interrupt handler can check NF. 0 = No noise detected in the received data. 1 = Noise detected in the received data. For receiver queue operation NF is cleared when SCxSR is read with NF set, followed by a read of SCRQ[0:15].
14	FE	Framing error. FE is set when the receiver detects a zero where a stop bit (one) was 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. Although no interrupt is explicitly associated with FE, an interrupt can be generated with RDRF, and the interrupt handler can check FE. 0 = No framing error detected in the received data. 1 = Framing error or break detected in the received data.
15	PF	Parity error. PF is set when the receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with PF, an interrupt can be generated with RDRF, and the interrupt handler can check PF. 0 = No parity error detected in the received data. 1 = Parity error detected in the received data.

14.8.5 SCI Data Register (SCxDR)

The SCxDR consists of two data registers located at the same address. The receive data register (RDRx) is a read-only register that contains data received by the SCI serial interface. Data is shifted into the receive serial shifter and is transferred to RDRx. The transmit data register (TDRx) is a write-only register that contains data to be transmitted. Data is first written to TDRx, then transferred to the transmit serial shifter, where additional format bits are added before transmission.

SCxDR — SCI Data Register **0x30 500E**

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

Table 14-26 SCxSR Bit Settings

Bit(s)	Name	Description
0:6	—	Reserved
7:15	R[8:0]/ T[8:0]	R[7:0]/T[7:0] contain either the eight data bits received when SCxDR is read, or the eight data bits to be transmitted when SCxDR is written. R8/T8 are used when the SCI is configured for nine-bit operation (M = 1). When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect. Accesses to the lower byte of SCxDR triggers the mechanism for clearing the status bits or for initiating transmissions whether byte, half-word, or word accesses are used.

14.8.6 SCI Pins

The RXD1 and RXD2 pins are the receive data pins for the SCI1 and SCI2, respectively. TXD1 and TXD2 are the transmit data pins for the two SCI modules. An external clock pin, ECK, is common to both SCIs. The pins and their functions are listed in **Table 14-27**.

Table 14-27 SCI Pin Functions

Pin Names	Mnemonic	Mode	Function
Receive Data	RXD1, RXD2	Receiver disabled Receiver enabled	General purpose input Serial data input to SCI
Transmit Data	TXD1, TXD2	Transmitter disabled Transmitter enabled	General purpose output Serial data output from SCI
External Clock	ECK	Receiver disabled Receiver enabled Transmitter disabled Transmitter enabled	Not used Alternate input source to baud Not used Alternate input source to baud

14.8.7 SCI Operation

The SCI can operate in polled or interrupt-driven mode. Status flags in SCxSR reflect SCI conditions regardless of the operating mode chosen. The TIE, TCIE, RIE, and ILIE bits in SCCxR1 enable interrupts for the conditions indicated by the TDRE, TC, RDRF, and IDLE bits in SCxSR, respectively.

14.8.7.1 Definition of Terms

- Bit-time — The time required to transmit or receive one bit of data, which is equal to one cycle of the baud frequency.
- Start bit — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time samples of logic one.
- Stop bit— One bit-time of logic one that indicates the end of a data frame.
- Frame — A complete unit of serial information. The SCI can use 10-bit or 11-bit frames.
- Data frame — A start bit, a specified number of data or information bits, and at least one stop bit.
- Idle frame — A frame that consists of consecutive ones. An idle frame has no start bit.

- Break frame — A frame that consists of consecutive zeros. A break frame has no stop bits.

14.8.7.2 Serial Formats

All data frames must have a start bit and at least one stop bit. Receiving and transmitting devices must use the same data frame format. The SCI provides hardware support for both 10-bit and 11-bit frames. The M bit in SCCxR1 specifies the number of bits per frame.

The most common data frame format for NRZ (non-return to zero) serial interfaces is one start bit, eight data bits (LSB first), and one stop bit (ten bits total). The most common 11-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and 11-bit frames are shown in **Table 14-28**.

Table 14-28 Serial Frame Formats

10-bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	—	2
1	7	1	1
1	8	—	1
11-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	1	2
1	8	1	1

14.8.7.3 Baud Clock

The SCI baud rate is programmed by writing a 13-bit value to the SCxBR field in SCI control register zero (SCCxR0). The baud rate is derived from the MCU system clock by a modulus counter. Writing a value of zero to SCxBR[12:0] disables the baud rate generator. The baud rate is calculated as follows:

$$\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$$

or

$$\text{SCxBR} = \frac{f_{SYS}}{32 \times \text{SCI Baud Rate Desired}}$$

where SCxBR is in the range {1, 2, 3, ..., 8191}.

Note that if the OTHR bit in SCCxR0 is set, the system clock frequency in the previous equations is replaced with the frequency of the selected baud clock source—either an external clock source, or the link baud clock source, depending on the value of the LNKBD bit in SCCxR0.

LNKBD enables SCI1 to use the resultant baud rate clock from SCI2 as the input clock source for the SCI1 baud rate generator.

NOTE

This option is not available for SCI2. If LNKBD is set for SCI2 the input clock source for SCI2 will be disabled.

If selected, the external clock frequency must be less than four times the system clock frequency so that correct synchronization for this signal can be provided. The same external clock is common to both independent SCIs.

Table 14-29 summarizes the possible sources for the SCI baud clock.

Table 14-29 SCI Baud Clock Sources

OTHR	LNKBD	Clock Source
0	X	Internal system clock
1	0	External clock source
1	1	Link baud clock source

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receive time sampling clock with a frequency 16 times that of the SCI baud rate. 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.

Table 14-30 shows possible baud rates for a 40-MHz system clock. The maximum baud rate with this system clock speed is 1250 Kbaud.

Table 14-30 Examples of SCIx Baud Rates[1]

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCxBR
1,250,000.00	1,250,000.00	0.00	1
57,600.00	56,818.18	-1.36	22
38,400.00	37,878.79	-1.36	33
32,768.00	32,894.74	0.39	38
28,800.00	29,069.77	0.94	43
19,200.00	19,230.77	0.16	65
14,400.00	14,367.81	-0.22	87
9,600.00	9,615.38	0.16	130
4,800.00	4,807.69	0.16	260
2,400.00	2,399.23	-0.03	521
1,200.00	1,199.62	-0.03	1042
600.00	600.09	0.02	2083
300.00	299.98	-0.01	4167

NOTES:
1. These rates are based on a 40-MHz system clock.

More accurate baud rates can be obtained by varying the system clock frequency.

Using the external clock, standard baud rates such as 1200, 2400, and 9600 can be achieved with reasonable error (less than 0.1%). **Table 14-31** shows an example of standard baud rates produced from a common crystal frequency, 3.6864 MHz.

$$\text{SCIx Baud} = \text{External Clock}/(32 * \text{SCxBR})(5\text{-}1)$$

where SCxBR equals {1, 2, 3, ... 8191}. Note that zero is a disallowed value for SCxBR.

Table 14-31 Examples of SCIx Baud Rates from an 3.6864 MHz external clock

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCxBR
115,200.00	115,200.00	0.0	1
57,600.00	57,600.00	0.0	2
28,800.00	28,800.00	0.0	4
14,400.00	14,400.00	0.0	8
9,600.00	9,600.00	0.0	12
2,400.00	2,400.00	0.0	48
1,200.00	1,200.00	0.0	96
64.00	64.00	0.0	1800

14.8.7.4 Parity Checking

The PT bit in SCCxR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The PE bit in SCCxR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of data in a frame (i.e., the bit preceding the stop bit) 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 PF bit in the SCI status register (SCxSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 14-32** shows possible data and parity formats.

Table 14-32 Effect of Parity Checking on Data Size

M	PE	Result
0	0	8 data bits
0	1	7 data bits, 1 parity bit
1	0	9 data bits
1	1	8 data bits, 1 parity bit

14.8.7.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The transmitter is double-buffered, which means that data can be loaded into the TDRx while other data is shifted out. The TE bit in SCCxR1 enables (TE = 1) and disables (TE = 0) the transmitter.

The shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in SCCxR1 determines whether TXD is an open drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function, regardless of whether the pin is used by the SCI or as a general-purpose output pin.

Data to be transmitted is written to SCxDR, then transferred to the serial shifter. Before writing to TDRx, the user should check the transmit data register empty (TDRE) flag in SCxSR. When TDRE = 0, the TDRx contains data that has not been transferred to the shifter. Writing to SCxDR again overwrites the data. If TDRE = 1, then TDRx is empty, and new data may be written to TDRx, clearing TDRE.

As soon as the data in the transmit serial shifter has shifted out and if a new data frame is in TDRx (TDRE = 0), then the new data is transferred from TDRx to the transmit serial shifter and TDRE is set automatically. An interrupt may optionally be generated at this point.

The transmission complete (TC) flag in SCxSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCxSR while TC is set, then writing new data to SCxDR, or writing to SCTQ[0:15] for transmit queue operation.

The state of the serial shifter is checked when the TE bit is set. 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.

The SBK bit in SCCxR1 is used to insert break frames in a transmission. A non-zero integer number of break frames are transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE is cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To ensure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

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 one (mark).

When TE is cleared, the transmitter is disabled after all pending idle, data, and break frames are transmitted. The TC flag is set, and control of the TXD pin reverts to PQSPAR and DDRQS. Buffered data is not transmitted after TE is cleared. To avoid losing data in the buffer, do not clear TE until TDRE is set.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. Configure the TXD pin as an output, then write a one to either

QDTX1 or QDTX2 of the PORTQS register. See **14.6.1**. When the transmitter releases control of the TXD pin, it reverts to driving a logic one output.

To insert a delimiter between two messages, to place non-listening receivers in wake-up mode between transmissions, or to signal a re-transmission by forcing an idle-line, clear and then set TE before data in the serial shifter has shifted out. The transmitter finishes the transmission, then sends a preamble. After the preamble is transmitted, if TDRE is set, the transmitter marks idle. Otherwise, normal transmission of the next sequence begins.

Both TDRE and TC have associated interrupts. The interrupts are enabled by the transmit interrupt enable (TIE) and transmission complete interrupt enable (TCIE) bits in SCCxR1. Service routines can load the last data frame in a sequence into SCxDR, then terminate the transmission when a TDRE interrupt occurs.

Two SCI messages can be separated with minimum idle time by using a preamble of 10 bit-times (11 if a 9-bit data format is specified) of marks (logic ones). Follow these steps:

1. Write the last data frame of the first message to the TDRx
2. Wait for TDRE to go high, indicating that the last data frame is transferred to the transmit serial shifter
3. Clear TE and then set TE back to one. This queues the preamble to follow the stop bit of the current transmission immediately.
4. Write the first data frame of the second message to register TDRx

In this sequence, if the first data frame of the second message is not transferred to TDRx prior to the finish of the preamble transmission, then the transmit data line (TXDx pin) marks idle (logic one) until TDRx is written. In addition, if the last data frame of the first message finishes shifting out (including the stop bit) and TE is clear, TC goes high and transmission is considered complete. The TXDx pin reverts to being a general-purpose output pin.

14.8.7.6 Receiver Operation

The receiver can be divided into two segments. The first is the receiver bit processor logic that synchronizes to the asynchronous receive data and evaluates the logic sense of each bit in the serial stream. The second receiver segment controls the functional operation and the interface to the CPU including the conversion of the serial data stream to parallel access by the CPU.

Receiver Bit Processor — The receiver bit processor contains logic to synchronize the bit-time of the incom-ing data and to evaluate the logic sense of each bit. To accomplish this an RT clock, which is 16 times the baud rate, is used to sample each bit. Each bit-time can thus be divided into 16 time periods called RT1–RT16. The receiver looks for a possible start bit by watching for a high-to-low transition on the RXDx pin and by assigning the RT time labels appropriately.

When the receiver is enabled by writing RE in SCCxR1 to one, the receiver bit processor logic begins an asynchronous search for a start bit. The goal of this search is

to gain synchronization with a frame. The bit-time synchronization is done at the beginning of each frame so that small differences in the baud rate of the receiver and transmitter are not cumulative. SCIx also synchronizes on all one-to-zero transitions in the serial data stream, which makes SCIx tolerant to small frequency variations in the received data stream.

The sequence of events used by the receiver to find a start bit is listed below.

1. Sample RXDx input during each RT period and maintain these samples in a serial pipeline that is three RT periods deep.
2. If RXDx is low during this RT period, go to step 1.
3. If RXDx is high during this RT period, store sample and proceed to step 4.
4. If RXDx is low during this RT period, but not high for the previous three RT periods (which is noise only), set an internal working noise flag and go to step 1, since this transition was not a valid start bit transition.
5. If RXDx is low during this RT period and has been high for the previous three RT periods, call this period RT1, set RAF, and proceed to step 6.
6. Skip RT2 but place RT3 in the pipeline and proceed to step 7.
7. Skip RT4 and sample RT5. If both RT3 and RT5 are high (RT1 was noise only), set an internal working noise flag. Go to step 3 and clear RAF. Otherwise, place RT5 in the pipeline and proceed to step 8.
8. Skip RT6 and sample RT7. If any two of RT3, RT5, or RT7 is high (RT1 was noise only), set an internal working noise flag. Go to step 3 and clear RAF. Otherwise, place RT7 in the pipeline and proceed to step 9.
9. A valid start bit is found and synchronization is achieved. From this point on until the end of the frame, the RT clock will increment starting over again with RT1 on each one-to-zero transition or each RT16. The beginning of a bit-time is thus defined as RT1 and the end of a bit-time as RT16.

Upon detection of a valid start bit, synchronization is established and is maintained through the reception of the last stop bit, after which the procedure starts all over again to search for a new valid start bit. During a frame's reception, SCIx resynchronizes the RT clock on any one-to-zero transitions.

Additional logic in the receiver bit processor determines the logic level of the re-ceived bit and implements an advanced noise-detection function. During each bit-time of a frame (including the start and stop bits), three logic-sense samples are taken at RT8, RT9, and RT10. The logic sense of the bit-time is decided by a majority vote of these three samples. This logic level is shifted into register RDRx for every bit except the start and stop bits.

If RT8, RT9, and RT10 do not all agree, an internal working noise flag is set. Additionally for the start bit, if RT3, RT5, and RT7 do not all agree, the internal working noise flag is set. If this flag is set for any of the bit-times in a frame, the NF flag in SCxSR is set concurrently with the RDRF flag in SCxSR when the data is transferred to register RDRx. The user must determine if the data received with NF set is valid. Noise on the RXDx pin does not necessarily corrupt all data.

The operation of the receiver bit processor is shown in **Figure 14-14**. This example demonstrates the search for a valid start bit and the synchronization procedure as outlined above. The possibilities of noise durations greater than one bit-time are not considered in this examples.

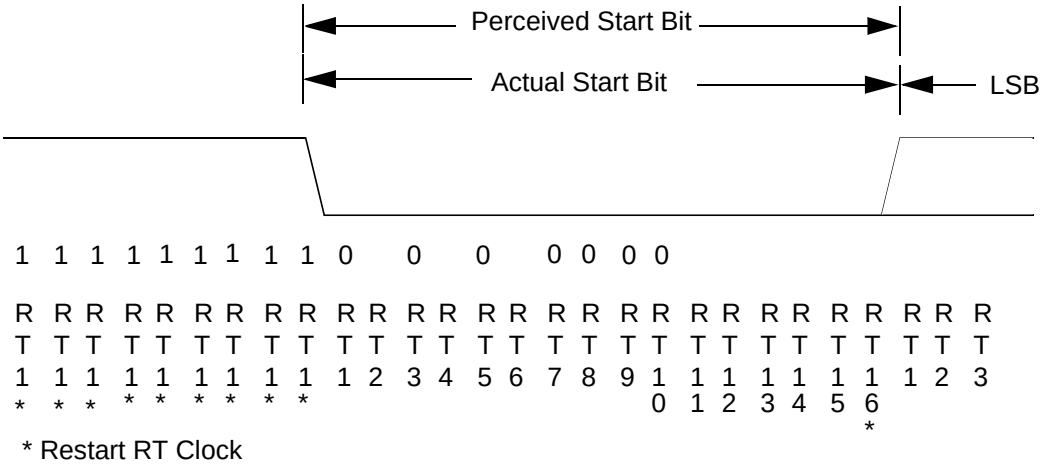


Figure 14-14 Start Search Example

14.8.7.7 Receiver Functional Operation

The RE bit in SCCxR1 enables (RE = 1) and disables (RE = 0) the receiver. The receiver contains a receive serial shifter and a parallel receive data register (RDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The receiver is double-buffered, allowing data to be held in the RDRx while other data is shifted in.

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. A receive time clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the receive time clock with the incoming data stream. From this point on, data movement is synchronized with the MCU system clock. Operation of the receiver state machine is detailed in the *Queued Serial Module Reference Manual* (QSMRM/AD).

The number of bits shifted in by the receiver depends on the serial format. However, all frames must end with at least one stop bit. When the stop bit is received, the frame is considered to be complete, and the received data in the serial shifter is transferred to the RDRx. The receiver data register flag (RDRF) is set when the data is transferred.

The stop bit is always a logic one. If a logic zero is sensed during this bit-time, the FE flag in SCxSR is set. A framing error is usually caused by mismatched baud rates between the receiver and transmitter or by a significant burst of noise. Note that a

framing error is not always detected; the data in the expected stop bit-time may happen to be a logic one.

Noise errors, parity errors, and framing errors can be detected while a data stream is being received. Although error conditions are detected as bits are received, the noise flag (NF), the parity flag (PF), and the framing error (FE) flag in SCxSR are not set until data is transferred from the serial shifter to the RDRx.

RDRF must be cleared before the next transfer from the shifter can take place. If RDRF is set when the shifter is full, transfers are inhibited and the overrun error (OR) flag in SCxSR is set. OR indicates that the RDRx needs to be serviced faster. When OR is set, the data in the RDRx is preserved, but the data in the serial shifter is lost.

When a completed frame is received into the RDRx, either the RDRF or OR flag is always set. If RIE in SCCxR1 is set, an interrupt results whenever RDRF is set. The receiver status flags NF, FE, and PF are set simultaneously with RDRF, as appropriate. These receiver flags are never set with OR because the flags apply only to the data in the receive serial shifter. The receiver status flags do not have separate interrupt enables, since they are set simultaneously with RDRF and must be read by the user at the same time as RDRF.

When the CPU reads SCxSR and SCxDR in sequence, it acquires status and data, and also clears the status flags. Reading SCxSR acquires status and arms the clearing mechanism. Reading SCxDR acquires data and clears SCxSR.

14.8.7.8 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronically and no idle time occurs between frames. Even when all the data bits in a frame are logic ones, the start bit provides one logic zero bit-time during the frame. An idle line is a sequence of contiguous ones equal to the current frame size. Frame size is determined by the state of the M bit in SCCxR1.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle-line type (ILT) bit in SCCxR1 determines which type of detection is used. When an idle-line condition is detected, the IDLE flag in SCxSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle-line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

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

When the ILIE bit in SCCxR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCxSR and SCxDR in sequence. For

receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15]. 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.

14.8.7.9 Receiver Wake-Up

The receiver wake-up 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. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wake-up mode by setting the RWU bit in SCCxR1. While RWU is set, receiver status flags and interrupts are disabled. Although the software can clear RWU, it is normally cleared by hardware during wake-up.

The WAKE bit in SCCxR1 determines which type of wake-up is used. When WAKE = 0, idle-line wake-up is selected. When WAKE = 1, address-mark wake-up is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wake-up 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 data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wake-up to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address mark wake-up uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The data frame is received normally, transferred to the RDRx, 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 wake-up allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

14.8.7.10 Internal Loop Mode

The LOOPS bit in SCCxR1 controls a feedback path in the data serial shifter. When LOOPS is set, the SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

14.9 SCI Queue Operation

14.9.1 Queue Operation of SCI1 for Transmit and Receive

The SCI1 serial module allows for queueing on transmit and receive data frames. In the standard mode, in which the queue is disabled, the SCI1 operates as previously defined (i.e. transmit and receive operations done via SC1DR). However, if the SCI1 queue feature is enabled (by setting the QTE and/or QRE bits within QSCI1CR) a set of 16 entry queues is allocated for the receive and/or transmit operation. Through software control the queue is capable of continuous receive and transfer operations within the SCI1 serial unit.

14.9.2 Queued SCI1 Status and Control Registers

The SCI1 queue uses the following registers:

- QSCI1 control register (QSCI1CR, address offset 0x28)
- QSCI1 status register (QSCI1SR, address offset 0x2A)

14.9.2.1 QSCI1 Control Register

QSCI1CR — QSCI1 Control Register **0x30 5028**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QTPNT				QTHFI	QBHFI	QTHEI	QBHEI	0	QTE	QRE	QTWE	QTSZ			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-33 QSCI1CR Bit Settings

Bit(s)	Name	Description
0:3	QTPNT	Queue transmit pointer. QTPNT is a 4-bit counter used to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This feature allows for ease of testability. This field is writable in test mode only; otherwise it is read-only.
4	QTHFI	Receiver queue top-half full interrupt. When set, QTHFI enables an SCI1 interrupt whenever the QTHF flag in QSCI1SR is set. The interrupt is blocked by negating QTHFI. This bit refers to the queue locations SCRQ[0:7]. 0 = QTHF interrupt inhibited 1 = Queue top-half full (QTHF) interrupt enabled
5	QBHFI	Receiver queue bottom-half full interrupt. When set, QBHFI enables an SCI1 interrupt whenever the QBHF flag in QSCI1SR is set. The interrupt is blocked by negating QBHFI. This bit refers to the queue locations SCRQ[8:15]. 0 = QBHF interrupt inhibited 1 = Queue bottom-half full (QBHF) interrupt enabled
6	QTHEI	Transmitter queue top-half empty interrupt. When set, QTHEI enables an SCI1 interrupt whenever the QTHE flag in QSCI1SR is set. The interrupt is blocked by negating QTHEI. This bit refers to the queue locations SCTQ[0:7]. 0 = QTHE interrupt inhibited 1 = Queue top-half empty (QTHE) interrupt enabled
7	QBHEI	Transmitter queue bottom-half empty interrupt. When set, QBHEI enables an SCI1 interrupt whenever the QBHE flag in QSCI1SR is set. The interrupt is blocked by negating QBHEI. This bit refers to the queue locations SCTQ[8:15]. 0 = QBHE interrupt inhibited 1 = Queue bottom-half empty (QBHE) interrupt enabled
8	—	Reserved
9	QTE	Queue transmit enable. When set, the transmit queue is enabled and the TDRE bit should be ignored by software. The TC bit is redefined to indicate when the entire queue is finished transmitting. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QTE. 0 = Transmit queue is disabled 1 = Transmit queue is enabled
10	QRE	Queue receive enable. When set, the receive queue is enabled and the RDRF bit should be ignored by software. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QRE. 0 = Receive queue is disabled 1 = Receive queue is enabled
11	QTWE	Queue transmit wrap enable. When set, the transmit queue is allowed to restart transmitting from the top of the queue after reaching the bottom of the queue. After each wrap of the queue, QTWE is cleared by hardware. 0 = Transmit queue wrap feature is disabled 1 = Transmit queue wrap feature is enabled
12:15	QTSZ	Queue transfer size. The QTSZ bits allow programming the number of data frames to be transmitted. From 1 (QTSZ = 0b0000) to 16 (QTSZ = 0b1111) data frames can be specified. QTSZ is loaded into QPEND initially or when a wrap occurs.

14.9.2.2 QSCI1 Status Register

QSCI1SR — QSCI1 Status Register **0x30 502A**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED			QOR	QTHF	QBHF	QTHE	QBHE	QRPNT				QPEND			
RESET:															
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Table 14-34 QSCI1SR Bit Settings

Bit(s)	Name	Description
0:2	—	Reserved
3	QOR	Receiver queue overrun error. The QOR is set when a new data frame is ready to be transferred from the SC1DR to the queue and the queue is already full (QTHF or QBHF are still set). Data transfer is inhibited until QOR is cleared. Previous data transferred to the queue remains valid. Additional data received during a queue overrun condition is not lost provided the receive queue is re-enabled before OR (SC1SR) is set. The OR flag is set when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost. 0 = The queue is empty before valid data is in the SC1DR 1 = The queue is not empty when valid data is in the SC1DR
4	QTHF	Receiver queue top-half full. QTHF is set when the receive queue locations SCRQ[0:7] are completely filled with new data received via the serial shifter. QTHF is cleared when register QSCI1SR is read with QTHF set, followed by a write of QTHF to zero. 0 = The queue locations SCRQ[0:7] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[0:7] are completely full of newly received data
5	QBHF	Receiver queue bottom-half full. QBHF is set when the receive queue locations SCRQ[8:15] are completely filled with new data received via the serial shifter. QBHF is cleared when register QSCI1SR is read with QBHF set, followed by a write of QBHF to zero. 0 = The queue locations SCRQ[8:15] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[8:15] are completely full of newly received data
6	QTHE	Transmitter queue top-half empty. QTHE is set when all the data frames in the transmit queue locations SCTQ[0:7] have been transferred to the transmit serial shifter. QTHE is cleared when register QSCI1SR is read with QTHE set, followed by a write of QTHE to zero. 0 = The queue locations SCTQ[0:7] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[0:7]
7	QBHE	Transmitter queue bottom-half empty. QBHE is set when all the data frames in the transmit queue locations SCTQ[8:15] has been transferred to the transmit serial shifter. QBHE is cleared when register QSCI1SR is read with QBHE set, followed by a write of QBHE to zero. 0 = The queue locations SCTQ[8:15] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[8:15]
8:11	QRPNT	Queue receive pointer. QRPNT is a 4-bit counter used to indicate the position where the next valid data frame will be stored within the receive queue. This field is writable in test mode only; otherwise it is read-only.
12:15	QPEND	Queue pending. QPEND is a 4-bit decrementer used to indicate the number of data frames in the queue that are awaiting transfer to the SC1DR. This field is writable in test mode only; otherwise it is read-only. From 1 (QPEND = 0b0000) to 16 (or done, QPEND = 1111) data frames can be specified.

14.9.3 QSCI1 Transmitter Block Diagram

The block diagram of the enhancements to the SCI transmitter is shown in **Figure 14-15**.

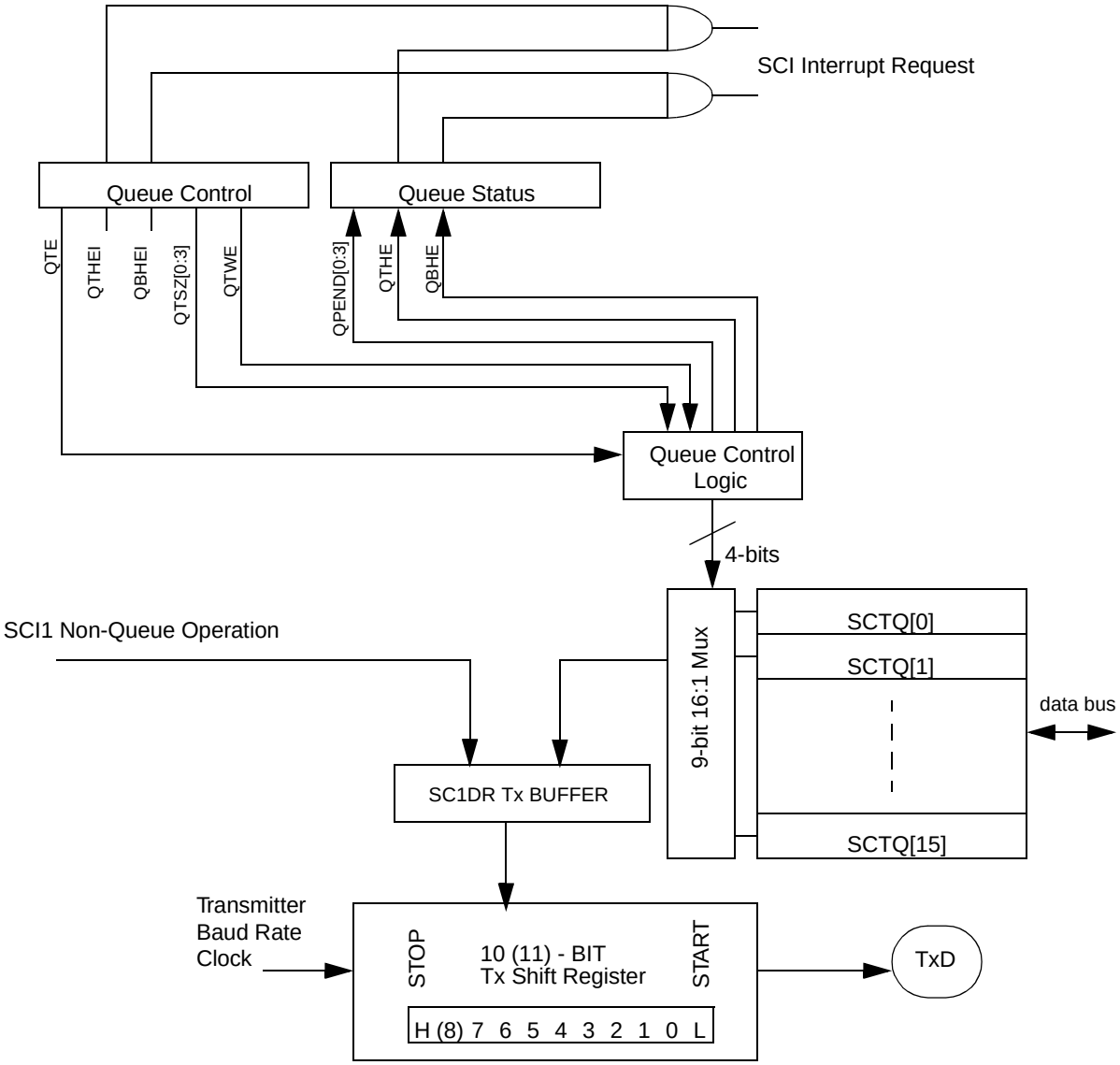


Figure 14-15 Queue Transmitter Block Enhancements

14.9.4 QSCI1 Additional Transmit Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue transmit enable (QTE) bit set by software. When enabled, (QTE = 1) the TDRE bit should be ignored by software and the TC bit is redefined (as described later).
- When the queue is disabled (QTE = 0), the SCI functions in single buffer transfer mode where the queue size is set to one (QTSZ = 0000), and TDRE and TC function as previously defined. Locations SCTQ[0:15] can be used as general purpose 9-bit registers. All other bits pertaining to the queue should be ignored by software.
- Programmable queue up to 16 transmits (SCTQ[0:15]) which may allow for infinite and continuous transmits.

- Available transmit wrap function to prevent message breaks for transmits greater than 16. This is achieved by the transmit wrap enable (QTWE) bit. When QTWE is set, the hardware is allowed to restart transmitting from the top of the queue (SCTQ[0]). After each wrap, QTWE is cleared by hardware.
  - Transmissions of more than 16 data frames must be performed in multiples of 16 (QTSZ = 0b1111) except for the last set of transmissions. For any single non-continuous transmissions of 16 or less or the last transmit set composed of 16 or fewer data frames, the user is allowed to program QTSZ to the corresponding value of 16 or less where QTWE = 0.
- Interrupt generation when the top half (SCTQ[0:7]) of the queue has been emptied (QTHE) and the bottom half (SCTQ[8:15]) of the queue has been emptied (QBHE). This may allow for uninterrupted and continuous transmits by indicating to the CPU that it can begin refilling the queue portion that is now emptied.
  - The QTHE bit is set by hardware when the top half is empty or the transmission has completed. The QTHE bit is cleared when the QSCI1SR is read with QTHE set, followed by a write of QTHE to zero.
  - The QBHE bit is set by hardware when the bottom half is empty or the transmission has completed. The QBHE bit is cleared when the QSCI1SR is read with QBHE set, followed by a write of QBHE to zero.
  - In order to implement the transmit queue, QTE must be set (QSCI1CR), TE must be set (SCC1R1), QTHE must be cleared (QSCI1SR), and TDRE must be set (SC1SR).
- Enable and disable options for the interrupts QTHE and QBHE as controlled by QTHEI and QBHEI respectfully.
- Programmable 4-bit register queue transmit size (QTSZ) for configuring the queue to any size up to 16 transfers at a time. This value may be rewritten after transmission has started to allow for the wrap feature.
- 4-bit status register to indicate the number of data transfers pending (QPEND). This register counts down to all 0’s where the next count rolls over to all 1’s. This counter is writable in test mode; otherwise it is read-only.
- 4-bit counter (QTPNT) is used as a pointer to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This counter is writable in test mode; otherwise it is read-only.
- A transmit complete (TC) bit re-defined when the queue is enabled (QTE = 1) to indicate when the entire queue (including when wrapped) is finished transmitting. This is indicated when QPEND = 1111 and the shifter has completed shifting data out. TC is cleared when the SCxSR is read with TC = 1 followed by a write to SCTQ[0:15]. If the queue is disabled (QTE = 0), the TC bit operates as originally designed.
- When the transmit queue is enabled (QTE = 1), writes to the transmit data register (SC1DR) have no effect.

14.9.5 QSCI1 Transmit Flow Chart Implementing the Queue

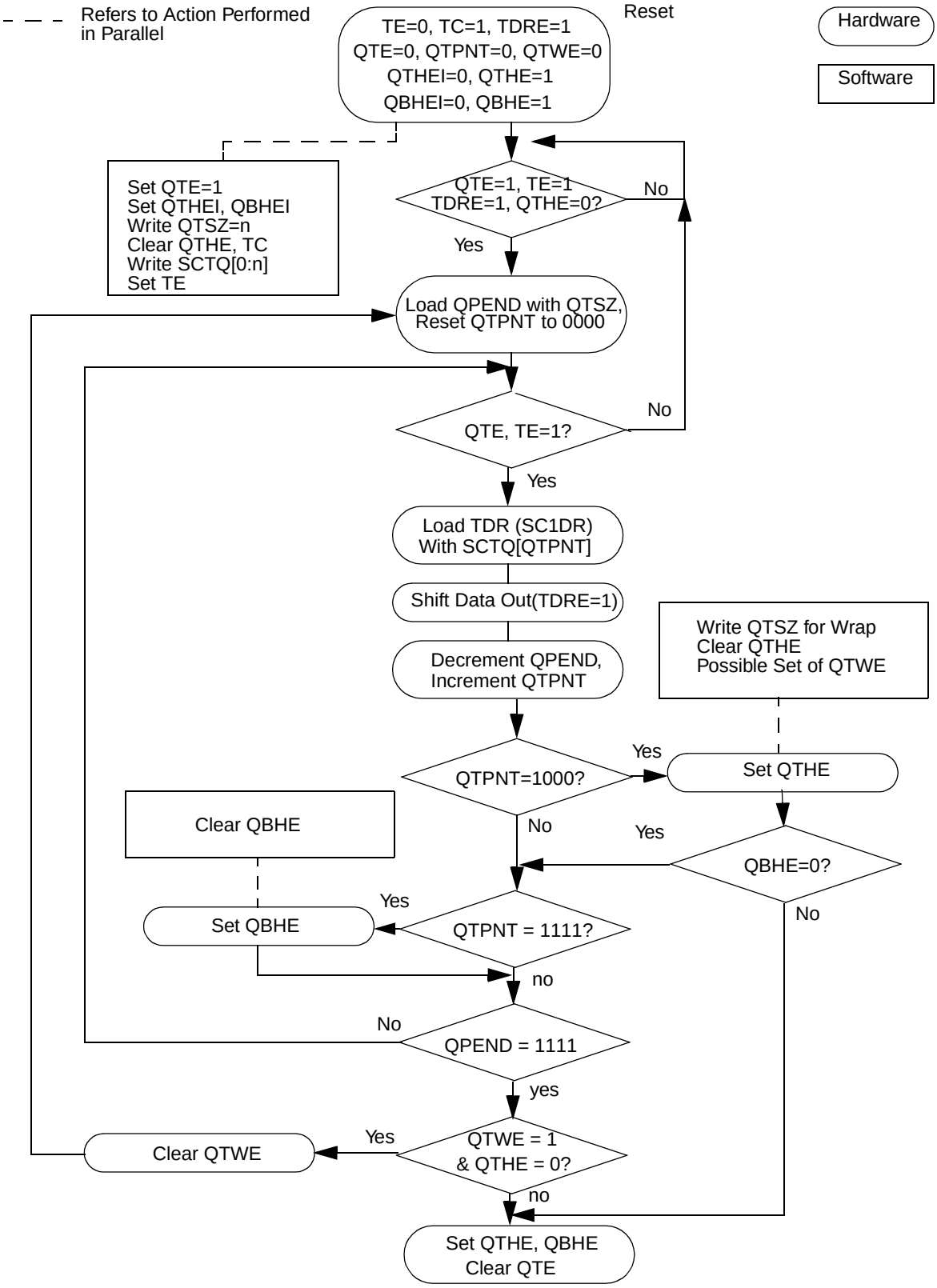


Figure 14-16 Queue Transmit Flow

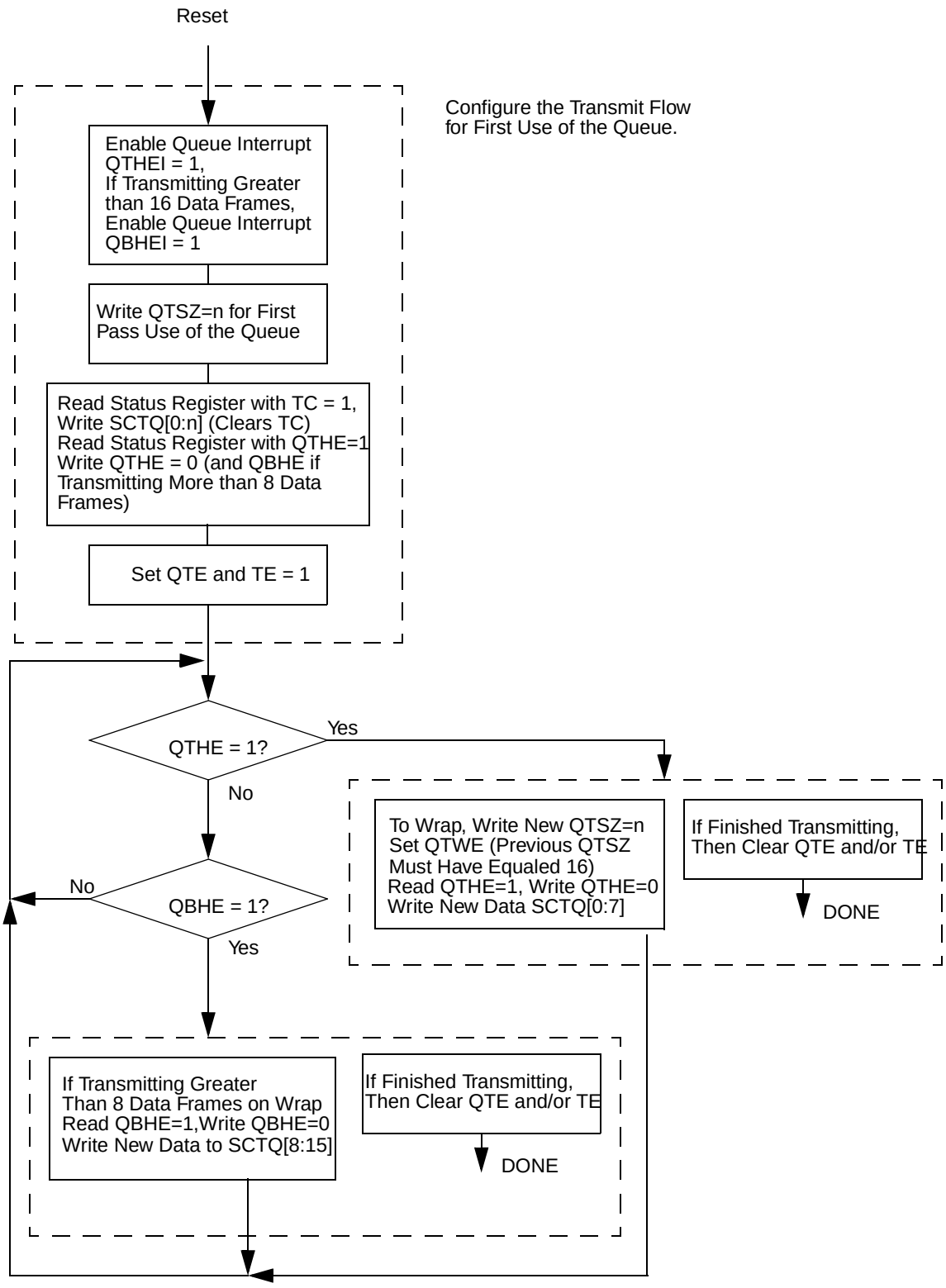


Figure 14-17 Queue Transmit Software Flow

14.9.6 Example QSCI1 Transmit for 17 Data Bytes

Figure 14-18 below shows a transmission of 17 data frames. The bold type indicates the current value for QTPNT and QPEND. The italic type indicates the action just performed by hardware. Regular type indicates the actions that should be performed by software before the next event.

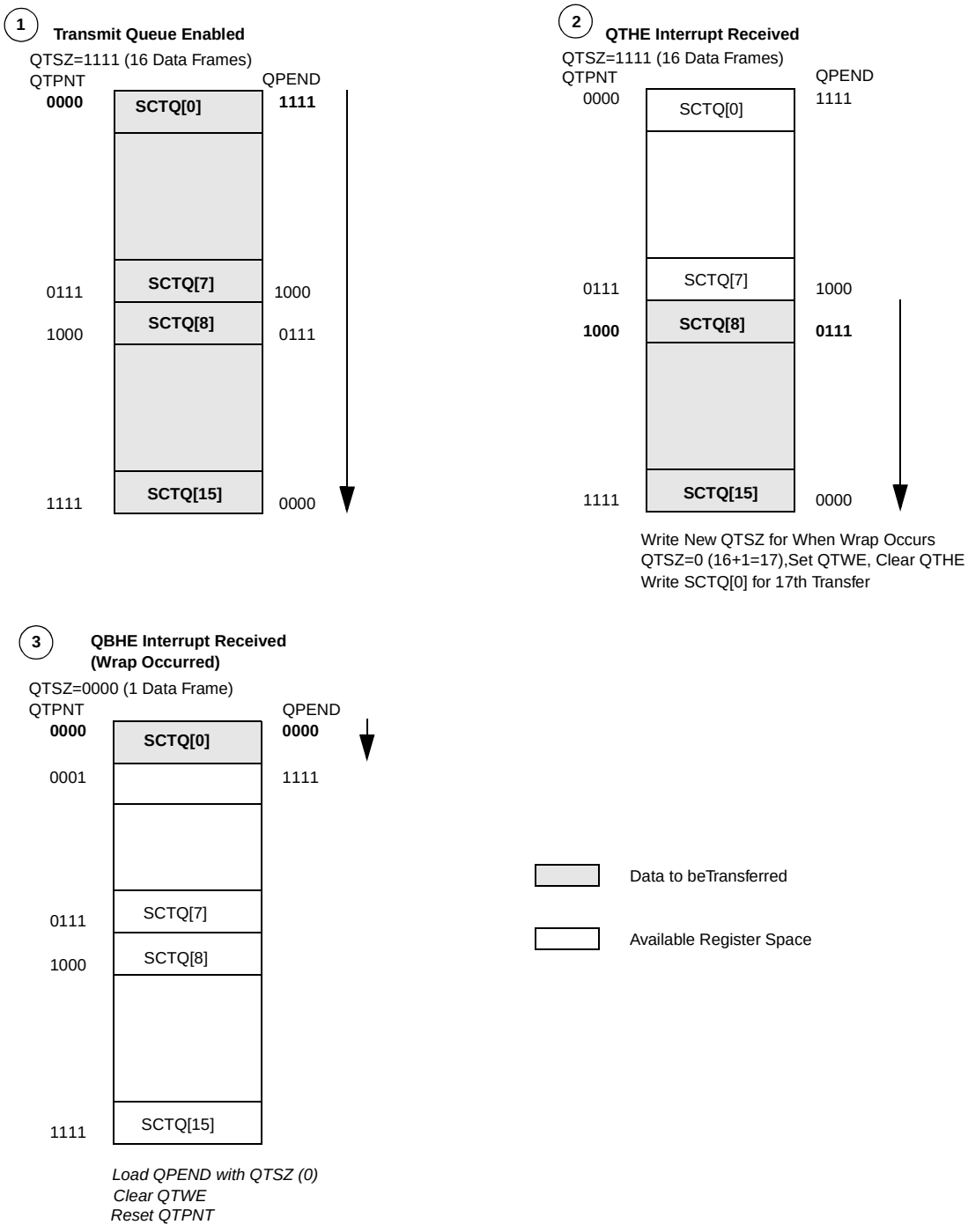


Figure 14-18 Queue Transmit Example for 17 Data Bytes

14.9.7 Example SCI Transmit for 25 Data Bytes

Figure 14-19 below is an example of a transmission of 25 data frames.

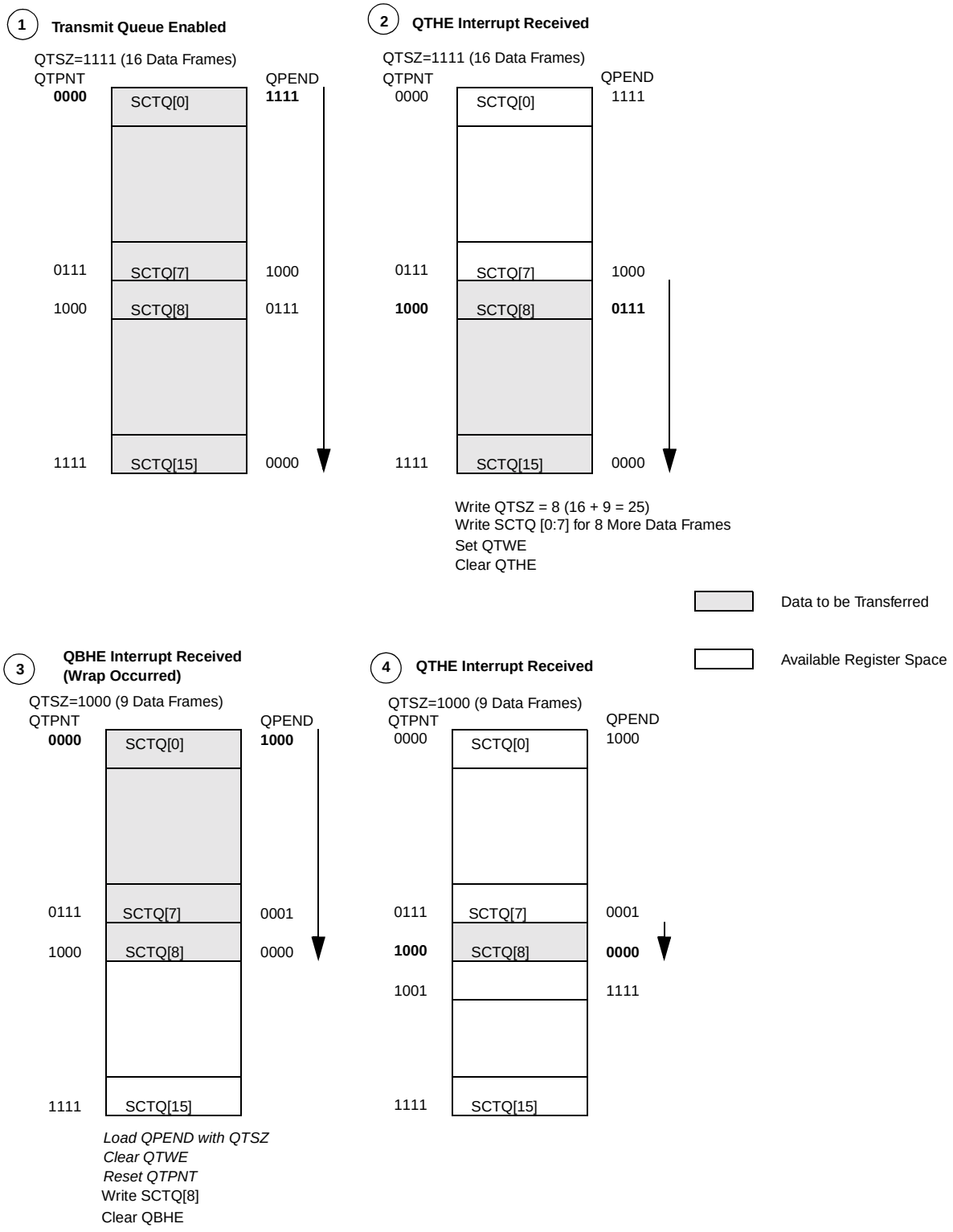


Figure 14-19 Queue Transmit Example for 25 Data Frames

14.9.8 QSCI1 Receiver Block Diagram

The block diagram of the enhancements to the SCI receiver is shown below in **Figure 14-20**.

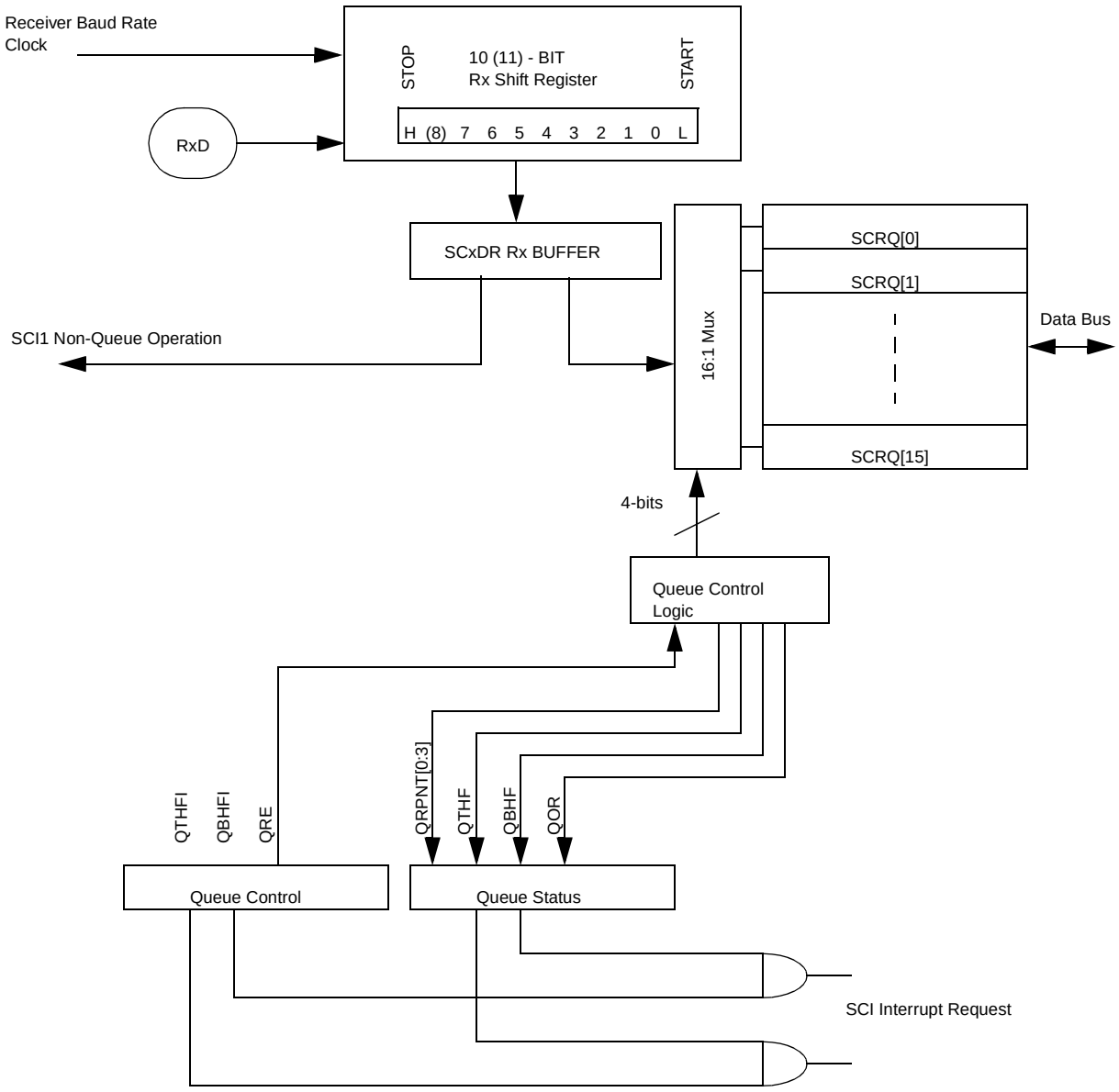


Figure 14-20 Queue Receiver Block Enhancements

14.9.9 QSCI1 Additional Receive Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue receiver enable (QRE) bit set by software. When the queue is enabled, software should ignore the RDRF bit.
- When the queue is disabled (QRE = 0), the SCI functions in single buffer receive mode (as originally designed) and RDRF and OR function as previously defined.

Locations SCRQ[0:15] can be used as general purpose 9-bit registers. Software should ignore all other bits pertaining to the queue.

- Only data that has no errors (FE and PF both false) is allowed into the queue. The status flags FE and PF, if set, reflect the status of data not allowed into the queue. The receive queue is disabled until the error flags are cleared via the original SCI mechanism and the queue is re-initialized. The pointer QRPNT indicates the queue location where the data frame would have been stored.
- Queue size capable to receive up to 16 data frames (SCRQ[0:15]) which may allow for infinite and continuous receives.
- Interrupt generation can occur when the top half (SCRQ[0:7]) of the queue has been filled (QTHF) and the bottom half (SCRQ[8:15]) of the queue has been filled (QBHF). This may allow for uninterrupted and continuous receives by indicating to the CPU to start reading the queue portion that is now full.
  - — The QTHF bit is set by hardware when the top half is full or the receive has completed. The QTHF bit is cleared when the SCxSR is read with QTHF set, followed by a write of QTHF to zero.
  - — The QBHF bit is set by hardware when the bottom half is full or the receive has completed. The QBHF bit is cleared when the SCxSR is read with QBHF set, followed by a write of QBHF to zero.
- In order to implement the receive queue, the following conditions must be met: QRE must be set (QSCI1CR); RE must be set (SCC1R1); QOR and QTHF must be cleared (QSCI1SR); and OR, PF, and FE must be cleared (SC1SR).
- Enable and disable options for the interrupts QTHF and QBHF as controlled by the QTHFI and QBHFI, respectfully.
- 4-bit counter (QRPNT) is used as a pointer to indicate where the next valid data frame will be stored.
- A queue overrun error flag (QOR) to indicate when the queue is already full when another data frame is ready to be stored into the queue (similar to the OR bit in single buffer mode). The QOR bit can be set for QTHF = 1 or QBHF = 1, depending on where the store is being attempted.
- The queue can be exited when an idle line is used to indicate when a group of serial transmissions is finished. This can be achieved by using the ILIE bit to enable the interrupt when the IDLE flag is set. The CPU can then clear QRE and/or RE allowing the receiver queue to be exited.
- For receiver queue operation, IDLE is cleared when SC1SR is read with IDLE set, followed by a read of SCRQ[0:15].
- For receiver queue operation, NF is cleared when the SC1SR is read with NF set, followed by a read of SCRQ[0:15]. When noise occurs, the data is loaded into the receive queue, and operation continues unaffected. However, it may not be possible to determine which data frame in the receive queue caused the noise flag to be asserted.
- The queue is successfully filled (16 data frames) if error flags (FE and PF) are clear, QTHF and QBHF are set, and QRPNT is reset to all zeroes.
- QOR indicates that a new data frame has been received in the data register

(SC1DR), but it cannot be placed into the receive queue due to either the QTHF or QBHF flag being set (QSCI1SR). Under this condition, the receive queue is disabled (QRE = 0). Software may service the receive queue and clear the appropriate flag (QTHF, QBHF). Data is not lost provided that the receive queue is reenabled before OR (SC1SR) is set, which occurs when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost.

14.9.10 QSCI1 Receive Flow Chart Implementing The Queue

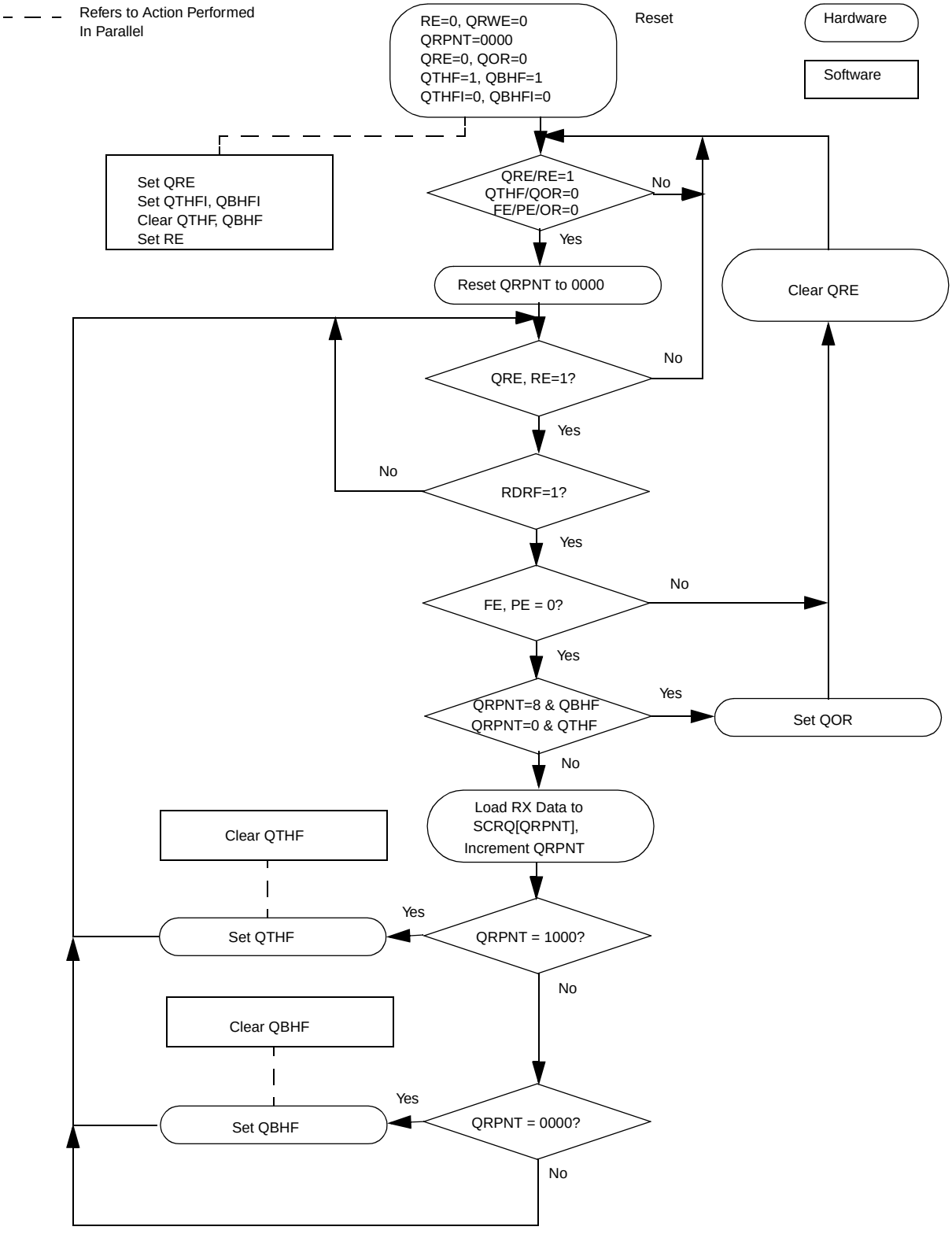


Figure 14-21 Queue Receive Flow

14.9.11 QSCI1 Receive Queue Software Flow Chart

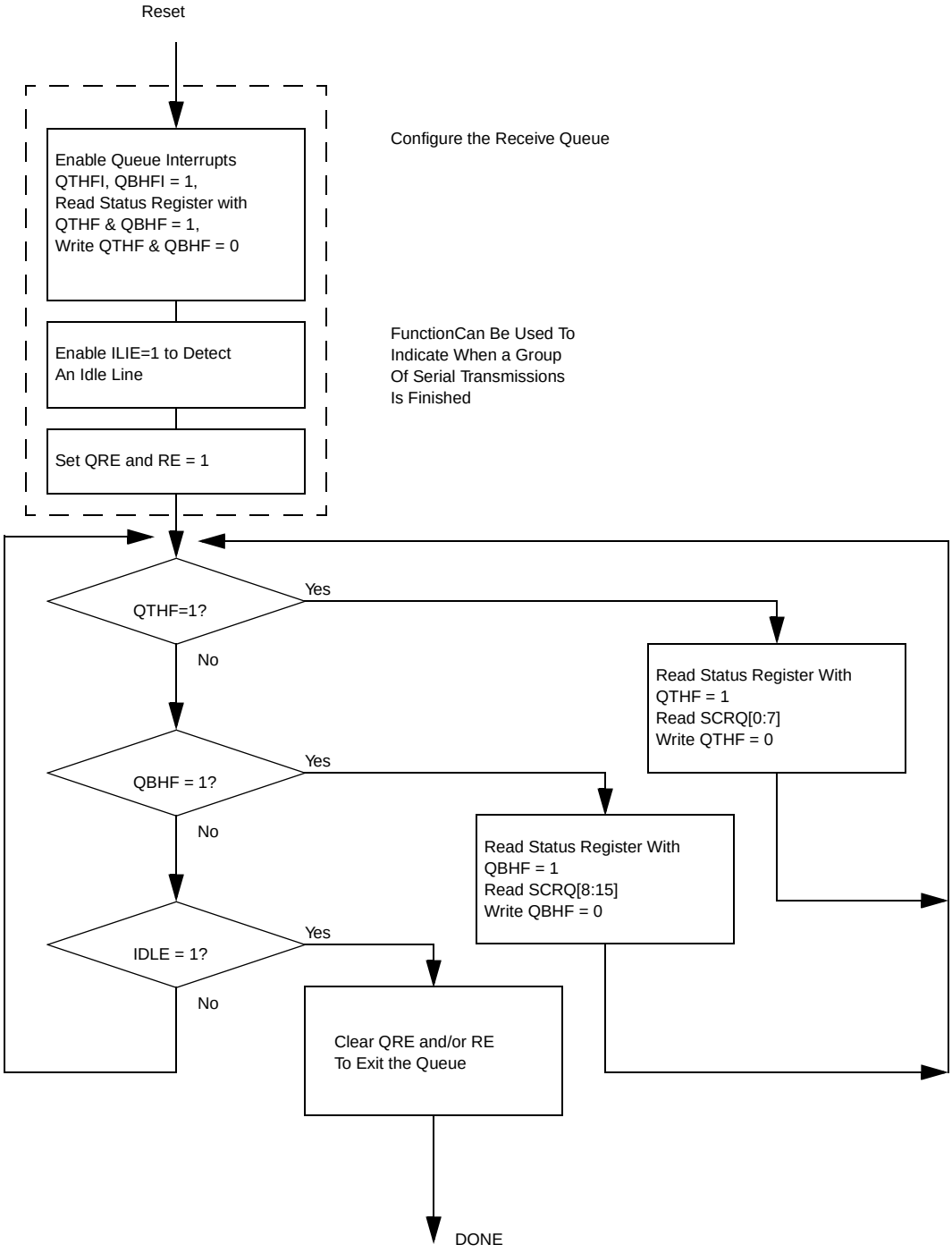


Figure 14-22 Queue Receive Software Flow

14.9.12 Example QSCI1 Receive Operation of 17 Data Frames

Figure 14-23 shows an example receive operation of 17 data frames. The bold type indicates the current value for the QRPNT. Action of the queue may be followed by starting at the top of the figure and going left to right and then down the page.

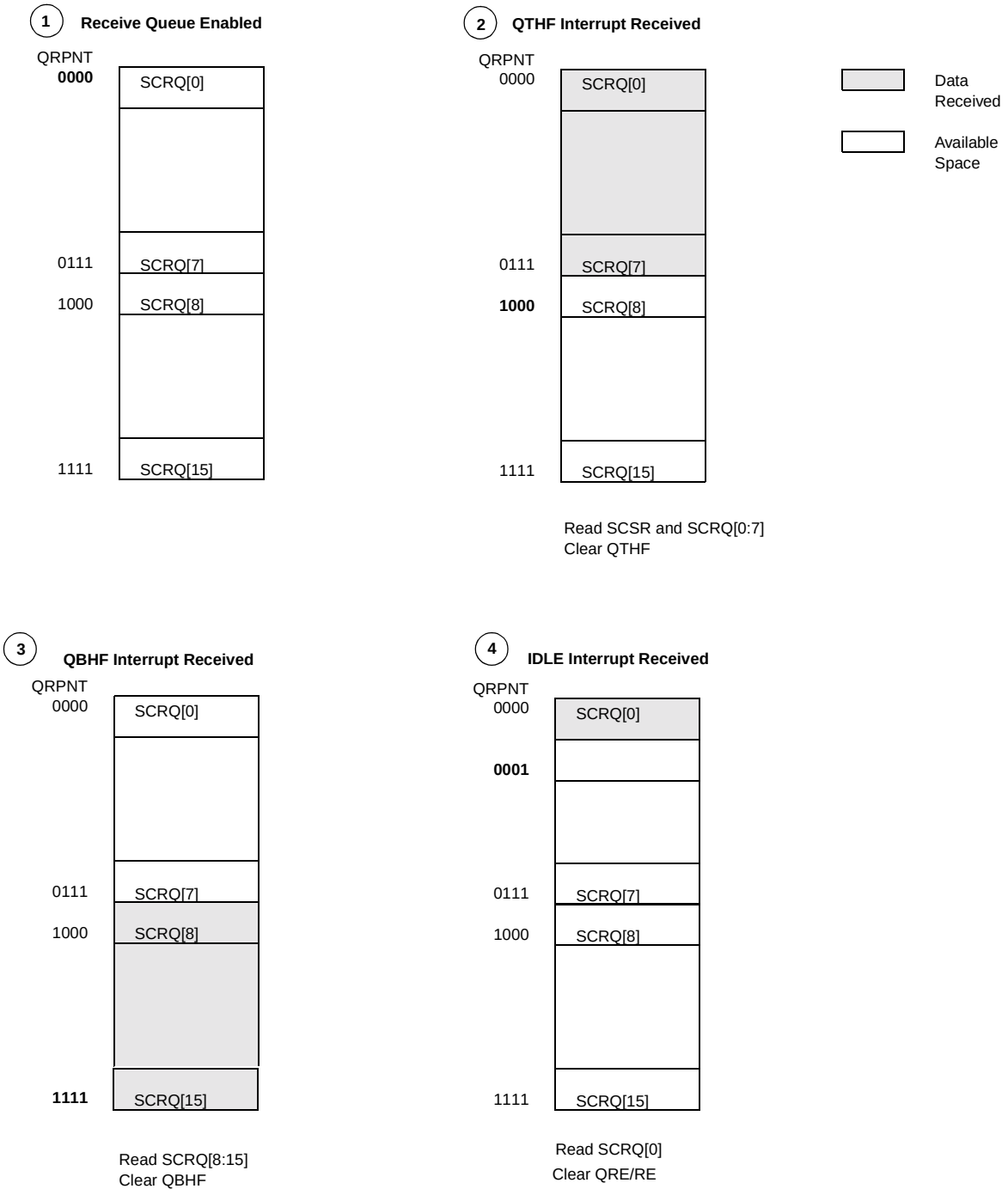


Figure 14-23 Queue Receive Example for 17 Data Bytes