

# Motorola Semiconductor Engineering Bulletin

---

## EB275

### Example Using the Queued Serial Peripheral Interface on Modular MCUs

By Sharon Darley  
Austin, Texas

#### Introduction

---

The QSPI (queued serial peripheral interface) uses a synchronous serial bus to communicate with external peripherals and other MCUs. It is compatible with the serial peripheral interface (SPI) found on the M68HC11 and M68HC05 Families of MCUs. However, it is unique in that it has a queue with programmable queue pointers that allow up to 16 transfers without CPU intervention. Furthermore, it has a wrap-around mode that allows continuous transfers to and from the queue with no CPU intervention. This queue is useful in applications such as control of an A/D convertor.

This engineering bulletin explains how to initialize the QSPI in wrap-around mode. Modifying the code to disable the wrap-around mode is very simple, and it is explained in the comments. Also explained is how to enable interrupts.



## Example Program

---

This example program initializes the QSPI to transmit a data stream with no interrupts continuously. The CPU16 code was assembled with P&E Microcomputer Systems' IASM16 assembler, and the CPU32 code was assembled with P&E Microcomputer Systems' IASM32 assembler.

### CPU16 Code

```
SPCR1      EQU      $FC1A
PORTQS     EQU      $FC15
PQSPAR     EQU      $FC16
DDRQS      EQU      $FC17
SPSR       EQU      $FC1F
SPCR0      EQU      $FC18
SPCR2      EQU      $FC1C
SPCR3      EQU      $FC1E
SYNCR      EQU      $FA04
SYPCR      EQU      $FA21
TXDRAM     EQU      $FD20
CMDRAM     EQU      $FD40

          ORG      $200                ;begin program at $200, immediately after
                                          ;the exception table

INIT_SIM
          LDAB     #$0F
          TBK
          TBK
          TBK
          LDAA     #$7F
          STAA    SYNCR                ;increase clock speed
          CLR     SYPCR                ;disable software watchdog

INIT_QSPI
          LDD     SPCR1
          ANDD   #$7F
          STD     SPCR1                ;Clear the SPE bit to disable
                                          ;the QSPI. Enabling the QSPI is the last
                                          ;step in the initialization sequence.
```

\* The next commands read and clear the flags in SPSR. These flags are the  
\* QSPI finished flag (SPIF), the mode fault flag (MODF), and the halt  
\* acknowledge flag (HALTA). The SPIF bit is usually the flag of interest. It is  
\* set by the QSPI upon completion of a serial transfer when the address of the  
\* command being executed matches the ENDQP. If wrap-around mode is enabled, the  
\* SPIF bit is set each time the QSPI cycles through the queue. If interrupts  
\* are enabled, assertion of the SPIF bit causes an interrupt.

```
LDAB SPSR
ANDB #$00
STAB SPSR                ;read and clear flags in SPSR
```

\* The next commands define the initial states of the chip select signals in  
\* PORTQS (formerly called QPDR). The chip selects may be active high or active  
\* low. The initial state set in the PORTQS is the inactive state. The active  
\* state is selected in the command RAM. In this example, the initial state of  
\* the chip select lines is high, and the initial state of SCK is low. This  
\* defines the chip selects to be active low and SCK to be active high. The SCI  
\* TXD signal bit is not affected.

```
LDAB #$7B
STAB PORTQS              ;define initial states of chip
                          ;selects/SCK
STAB PQSPAR              ;Assign all pins to the QSPI. Pins can be
                          ;assigned to the QSPI or for general
                          ;purpose I/O on a pin by pin basis.
LDAB #$7E
STAB DDRQS               ;Select the direction of the signal lines
                          ;as outputs, except for MISO.
LDD #$8002
STD SPCR0                ;Configure the QSPI as master, select
                          ;8 data bits per transfer, set the
                          ;inactive
                          ;state of SCK as low, capture data on the
                          ;leading edge of SCK, baud rate is 4.19
                          ;MHz
```

\* The next commands set the control parameters. Interrupts are not enabled. To  
\* enable interrupts upon assertion of the SPIF bit, set SPCR2[15]. To clear  
\* an interrupt, read and then clear the SPIF bit. Wrap-around mode is enabled.  
\* NEWQP is set to zero, and ENDQP is set to \$F. Thus, the QSPI will  
\* continuously transmit the data between \$0 and \$F in the queue. To disable  
\* wrap-around mode so that the QSPI only goes through the queue once, clear the  
\* WREN bit (SPCR2[14]) to a zero.

```
        LDD #$4F00
        STD SPCR2           ;NEWQP = 0, ENDQP = $F, WREN is enabled
        CLRB
        TBXK
        STAB SPCR3        ;Disable loop mode, HALTA and MODF
                          ;interrupts, and HALT.
        LDX #DATA         ;Point X to the data to be transmitted.
        LDY #TXDRAM       ;Point Y to the transmit data RAM.
        LDZ #CMDRAM       ;Point Z to the command RAM
        LDE #$10          ;Set a counter to count down from 16 ($10),
                          ;since there are 16 queue entries to fill.
LOOP    LDD 0,X
        STD 0,Y           ;Begin a loop to fill the transmit RAM.
        AIX #$02         ;Store the data right-justified.
        AIY #$02
```

\* The next commands fill the command RAM in a right-justified manner. There is  
\* one byte of control information for each QSPI command to be executed in the  
\* queue. Here, all four chip selects drive low during each serial transfer.

```
        CLRB
        STAB 0,Z
        INCZ              ;fill command RAM: chip selects active
                          ;low
        SUBE #$01         ;Subtract one from the counter
        BNE LOOP         ;Fill next queue entry if not done
        LDD #$8000
        STD SPCR1        ;Begin operation by setting the SPE bit.
FINISH  BRA FINISH       ;Normally, this would begin the next
                          ;task.
DATA    DB 16            ;Set aside memory space for the data to be
                          ;transmitted. This program does not
                          ;initialize the data.
```

## CPU32 Code

```

SPCR1 EQU $FFFC1A
PORTQS EQU $FFFC15
PQSPAR EQU $FFFC16
DDRQS EQU $FFFC17
SPSR EQU $FFFC1F
SPCR0 EQU $FFFC18
SPCR2 EQU $FFFC1C
SPCR3 EQU $FFFC1E
SYNCR EQU $FFFA04
SYPCR EQU $FFFA21
TXDRAM EQU $FFFD20
CMDRAM EQU $FFFD40

ORG $400 ;begin program at $400, immediately after
;the exception table

INIT_SIM
    MOVE.B #$7F,(SYNCR).L ;increase clock speed
    CLR.B (SYPCR).L ;disable software watchdog
INIT_QSPI
    ANDI.W #$7F,(SPCR1).L ;Clear the SPE bit in SPCR1 to disable
;the QSPI. Enabling the QSPI is the last
;step in the initialization sequence.

* The next command reads and clears the flags in SPSR. These flags are the
* QSPI finished flag (SPIF), the mode fault flag (MODF), and the halt
* acknowledge flag (HALTA). The SPIF bit is usually the flag of interest. It is
* set by the QSPI upon completion of a serial transfer when the address of the
* command being executed matches the ENDQP. If wrap-around mode is enabled, the
* SPIF bit is set each time the QSPI cycles through the queue. If interrupts
* are enabled, assertion of the SPIF bit causes an interrupt.

    ANDI.B #$00,(SPSR).L

```

\* The next command defines the initial states of the chip selects in PORTQS  
\* (formerly called QPDR). The chip selects may be active high or active low.  
\* The initial state set in the PORTQS is the inactive state. The active state  
\* is selected in the command RAM. In this example, the initial state of the  
\* chip select lines is high, and the initial state of SCK is low. This defines  
\* the chip selects to be active low and SCK to be active high. The SCI TXD  
\* signal bit is not affected.


```
MOVE.B #$7B,(PORTQS).L
MOVE.B #$7B,(PQSPAR).L
;Assign all pins to the QSPI. Pins can be
;assigned to the QSPI or for general
;purpose I/O on a pin by pin basis.
MOVE.B #$7E,(DDRQS).L
;Select the direction of the signal lines
;as outputs, except for MISO.
MOVE.W #$8002,(SPCR0).L
;Configure the QSPI as master, select
;8 data bits per transfer, inactive
;state of SCK is low, capture data on the
;leading edge of SCK, baud rate = 4.19
;MHz
```

\* The next command sets the control parameters. Interrupts are not enabled. To  
\* enable interrupts upon assertion of the SPIF bit, set SPCR2[15]. To clear  
\* an interrupt, read and then clear the SPIF bit. Wrap-around mode is enabled.  
\* NEWQP is set to zero, and ENDQP is set to \$F. Thus, the QSPI will  
\* continuously transmit the data between \$0 and \$F in the queue. To disable  
\* wrap-around mode so that the QSPI only goes through the queue once, clear the  
\* WREN bit (SPCR2[14]) to a zero.

```
MOVE.W #$4F00,(SPCR2).L
MOVE.B #$00,(SPCR3).L
;Disable loop mode, HALTA and MODF
;interrupts, and HALT.
MOVEA.L #DATA,A0
;Point A0 to the address of the data to be
;transmitted.
MOVEA.L #TXDRAM,A1
;Point A1 to the transmit data RAM.
MOVEA.L #CMDRAM,A2
;Point A2 to the command RAM
MOVE.W #$10,D0
;Set a counter to count down from 16
;($10), since there are 16 queue entries
CLR.L D1
;Clear D1. It will be used to fill the
;transmit RAM.
LOOP MOVE.B (A0)+,D1
;Begin a loop to fill the transmit RAM.
MOVE.W D1,(A1)+
;Store the data right-justified.
```

\* The next command fills the command RAM in a right-justified manner. There is  
\* one byte of control information for each QSPI command to be executed in the  
\* queue. Here, all four chip selects will drive low during each transfer.

```
        MOVE.B #$00,(A2)+  
        SUBI.W #$01,D0           ;Subtract one from the counter  
        BNE LOOP                ;Fill next queue entry if not done  
        MOVE.W #$8000,(SPCR1).L ;Begin operation by setting the SPE bit.  
                                   ;This is the last step of initialization.  
FINISH  
        BRA FINISH              ;Normally, this would begin the next  
                                   ;task.  
DATA    DB 16                  ;Set aside memory space for the data to be  
                                   ;transmitted. This program does not  
                                   ;initialize the data.
```

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

#### How to reach us:

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

**JAPAN:** Motorola Japan Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinagawa-ku, Tokyo, Japan, 03-5487-8488

**ASIA/PACIFIC:** Motorola Semiconductors H.K. Ltd., Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, New Territories, Hong Kong, 852-26629298

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

**HOME PAGE:** <http://motorola.com/sps/>

Mfax is a trademark of Motorola, Inc.



**MOTOROLA**

© Motorola, Inc., 1999

EB275/D