Freescale Semiconductor
Application Note

AN2878
Rev. 0, 01/2005

Using the Serial Peripheral Interface (SPI) Module on 68HC(9)08 Microcontrollers

by: Rogelio Reyna García
RTAC Americas Mexico

Overview

This document is intended to serve as a quick reference for an embedded engineer to get the serial peripheral interface module (SPI) up and running for any 68HC(9)08 MCU. Basic knowledge about the functional description and configuration options will give the user a better understanding on how the SPI module works. This application note provides an example which illustrates one use of the SPI module within the 68HC(9)08 Family of microcontrollers. The example mentioned is intended to be modified to suit the specific needs for any application.

Serial Peripheral Interface Module (SPI)

The serial peripheral interface module allows a full-duplex, synchronous, serial communications with peripheral devices.

freescale™ semiconductor

SPI communication can be implemented using 4 lines: MOSI (master-output/slave-input), MISO (master-input/slave-output), SS (slave select), and SPSCK (SPI serial clock).

In SPI communications, only a master can initiate transmissions, which begins by driving the SS line low. Then, data is transferred between master and slave using their shift registers at each SPSCK configurable clock edge.

Some features of the SPI module are:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four baud rates supported (max = bus freq/2)
- Serial clock with programmable polarity and phase
- Interrupt driven operations with 4 flags: receiver full, transmitter empty, mode fault error or overflow error

Please refer to Appendix A for information on the differences between each HC(9)08 Family and their capabilities in regards to the SPI module.

Registers

The main registers for configuring the SPI module are:

1. The SPI control register (SPCR) configures the SPI module:

SPI Control Register (SPCR)									
	Read: / Write:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
	Reset:	0	0	1	0	1	0	0	0

Figure 1. SPI Control Register (SPCR)

- SPE enables the SPI module
- SPTIE enables the Interrupts generated by the SPTE bit (transmitter interrupt)
- SPRIE enables the interrupts generated by the SPRF bit (receiver interrupt)
- SPMSTR selects master mode (1) or slave mode (0) operation
- SPWOM disables the pull-up devices on SPSCK, MOSI, and MISO so that those pins become open drain outputs (for multimaster operation)

2. The SPI status and control register (SPSCR) reflects the state of the SPI module:

SPI Status and Control Register (SPSCR)									
	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
	Write:								
	Reset:	0	0	0	0	1	0	0	0

Figure 2. SPI Status and Control Register (SPSCR)

- SPRF flags when the receiver's data register is full
- ERRIE enables the MODF and OVRF flags to generate CPU interrupt requests
- OVRF and MODF indicate an "overflow" on receiver data register and "mode fault" respectively
- SPTE flags when the transmitter's data register is empty
- SPR1 and SPR0 determine the baud rate select according to Table 1

Table 1. Baud Rate Selector

SPR1:SPR0	Baud Rate Divisor (BD)
00	2
01	8
10	32
11	128

The following formula is used to calculate the SPI baud rate:

$$\text{Baudrate} = \frac{\text{CGMOUT}}{2 \times \text{BD}}$$

3. The SPI data register (SPDR) permits the user access to the SPI shift register:

SPI Data Register (SPDR)									
	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Unaffected by reset							

Figure 3. SPI Data Register (SPDR)

The SPI data register represents both transmit data register and receive data register. When you read this register, you read the receive data register. When you write to this register, you write to the transmit data register.

Code and Explanation

The example code is defined in two projects which can be found in AN2878SW.zip from www.freescale.com.

Project “SPI_MASTER”

The project “SPI_Master” implements the SPI in master mode. The main functions are:

- main: Endless loop sending characters out the SPI module
- MCUInit: Configures the hardware
- SPIInit: Configures the SPI module as a master
- SPIRIsr: Responds to the “receive full” interrupt
- SPISendChar: Function used to send a byte

The simplest way to configure a master SPI module is designing it to be the only master on the SPI bus as this example has been defined.

However if your application requires multiple masters on the bus, you must configure:

- SPI outputs as open-drain (wired-or mode)
- SPI SS pin as SPI input (mode fault enabled)

The SPI module is normally used with various slaves. To communicate with a specific slave, its SS signal must be low and the SS signals from its neighbors must be high in order to avoid collisions. Therefore, the slave select (SS) signal must be generated by software.

In this example, we use just one slave, so we need a GPIO pin to manage it.

```
void SPISendChar(unsigned char data) {
   while (!SPSCR_SPTE);       /* Wait until transmit buffer is empty */
            PTD_PTD0 = 0;                           /* Select the Slave Peripheral*/
            SPDR = data;                            /* Start to send data */
}
```

Above is the SPISendChar function. This is used to send a byte through the SPI module. It waits for the transmit buffer to be empty and then pulls the SS line of the device down and then moves the data into the transmit buffer to start transmission.

```
void interrupt 10 SPIRIsr (void)
{
        while (PTD_PTD3);      /* Once a byte is received, wait until the clock goes to its
                                  default state */
        PTD_PTD0 = 1;          /* Put the SS line on high after a byte is sent (compatible with
                                  CPHA = 0)*/
        SPSCR;                 /* Read the SPI Status and Control Register and, */
        SPDR;                  /* also read the SPI Data Register to acknowledge interrupt */
}
```

Above is the receive full interrupt function SPIRIsr. It waits for the clock to go low, and then puts the SS line high. Then, it acknowledges the Interrupt by reading SPSCR and SPDR.

Further details of the actual coding are in the CodeWarrior project.

Project “SPI_SLAVE”

The project “SPI_Slave” implements the SPI in slave mode. The main functions are:

- main: Endless loop sending characters through the SPI module
- MCUInit: Configures the hardware
- SPIInit: Configures the SPI module as a slave without interrupts
- SPIRIsr: Function used to receive a byte

This project simply configures the SPI as a slave. When the SPI module receives a byte, it Interrupts the MCU and executes the SPIRIsr function, which outputs on port C part of the byte received.

Considerations

It is important to notice that the software of this application note was developed in Metrowerks CodeWarrior for Freescale HC08 v. 3.0.

Both projects were tested using MC68HC908GP32 running with a 4.9152 crystal.

Please note that SPI is a protocol designed for in board communication. However, if cable is needed, it must not be longer than 20 cm.

References

- MC68HC908GP32 Data Sheet

Appendix A

Table 2 provides information for serial peripheral interface modules of the different HC(9)08 MCUs.

Table 2. Presence of SPI Module in HC(9)08 Families

Family	SPI
AB	Yes
AP	Yes
AS	Yes
AZ	Yes
BD	No
EY	Yes
GP	Yes
GR	Yes
GT	Yes
GZ	Yes
JB	No
JK	No
JL	No
JT	No
KH	No
KX	No
LD	No
LJ	Yes
LK	Yes
MR*	Yes
QT	No
QY	No
RF	No
SR	No

*Family member MC68HC908MR8 doesn't have an SPI module

This page is intentionally blank

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2878
Rev. 0, 01/2005

freescale™
semiconductor