Using the HC12 SPI in Master Mode

0.2in

- When using the HC12 SPI in Master Mode you can talk to several devices on over the same serial lines
- When sending data to a device:
  1. Select the device (usually be bringing a general purpose I/O line active)
  2. Write data to the SPI data register (starts the serial clock, and starts sending data out the MOSI pin)
  3. Wait until the SPI flag is set (which indicates data transfer is finished) by reading the SPI status register until the SPIF bit is 1 (this is also the first step in clearing the SPIF)
  4. Repeat steps 3 and 4 if more data to send
  5. Deselect the device (usually be bringing a general purpose I/O line inactive)

```
PORTS = PORTS & ~0x80;          /* Bring SS low to select device */

SP0DR = data;                   /* Write data to device */
while ((SP0DR & 0x80)==0) ;     /* Wait for transfer to finish */

PORTS = PORTS | 0x80;           /* Bring SS high to deselect device */
```

Using the HC12 SPI in Master Mode

0.2in

- When reading data from a device:
  1. Select the device (usually be bringing a general purpose I/O line active)
  2. Write a junk byte to the SPI data register (starts the serial clock)
  3. Wait until the SPI flag is set (which indicates data transfer is finished) by reading the SPI status register until the SPIF bit is 1 (this is also the first step in clearing the SPIF)
  4. Read data from the SPI data register (gets data and completes clearing the SPIF flag)
  5. Repeat steps 2, 3 and 4 if more data to receive
  6. Deselect the device (usually be bringing a general purpose I/O line inactive)

```
PORTS = PORTS & ~0x80;          /* Bring SS low to select device */

SP0DR = 0;                      /* Write junk byte to device */
while ((SP0DR & 0x80)==0) ;     /* Wait for transfer to finish */
data = SP0DR;                   /* Get data byte */

PORTS = PORTS | 0x80;           /* Bring SS high to deselect device */
```

Using the HC12 SPI in Slave Mode

0.2in

- When using the HC12 SPI is Slave Mode you can send and/or receive data only when the master device brings your Slave Select line low
- You know when a transfer is complete when the SPIF flag is set
- When using the HC12 SPI in Slave Mode you often use interrupts so you respond to a transfer in an interrupt service routine
- If you need to send data to the master SPI, you need to put that data in SP0DR *before* the master starts the data transfer
- You may need to use a general purpose I/O line to let the master know you have data to send it
- Example: Receiving data from an SPI master:
    1. Wait for SPIF flag to become set — this happens when master device brings your slave select low, and sends 8 ticks on the serial clock
    2. Read the SP0DR to get data from master

```
while ((SP0DR & 0x80)==0) ;   /* Wait for transfer to finish */
data = SP0DR;                 /* Get data byte */
```

The Maxim MAX522 D/A Converter Chip

- In next week's lab you will connect a Maxim MAX522 D/A converter chip to your HC12 board. A copy of the data sheet for the Maxim MAX522 D/A converter is available from the same location you got these notes, or you can download it from the Maxim website at www.maxim-ic.com.
- Be sure to have a copy of the data sheet when reading these notes.
- As outlined in the General Description part of the MAX522 data sheet. the MAX522 is:
  - A two-channel 8-bit D/A converter
  - Compatible with the SPI serial interface
  - Can run at a serial clock speed of 5 MHz
- To use the MAX522 with the HC12 you need to know how to make the hardware connections, and how to set up the HC12 software protocol to be compatible with the MAX522
- As shown in the Functional Diagram of the data sheet, the MAX522 has a $\overline{\mathrm{CS}}$ input (which can be connected to a general purpose I/O pin of the HC12, or to the $\overline{\mathrm{SS}}$ line of the HC12 SPI interface), a SCLK line (which should be connected to the HC12's SCK line) and a $\mathrm{D_{in}}$ line (which should be connected to the HC12's MOSI line)
- The REF input of the MAX522 will be connected to the +5V power of the HC12. This will allow the MAX522 output voltage to range from 0V to +5V

Using the Maxim MAX522 D/A Converter Chip

- The serial transfer protocol shown in Figure 2 of the data sheet shows the following:
  - $\overline{\text{CS}}$ is active low
  - The serial clock is idle low, so you should use CPOL = 0
  - The data is valid on the first clock edge, so you should use CPHA = 0
  - The data is sent out with Most Significant Bit (MSB) first, so you should use LSBF = 0
  - The HC12 needs to transfer two bytes with $\overline{\text{CS}}$ low, so you should use SSOE = 0
- The General Description says the the MAX522 will work with a serial clock speed of up to 5 MHz. Further details of this are given in Figure 3 of the data sheet, and in the Timing Characteristics table on Page 3.

Using the MAX522 D/A Converter with the HC12

- To send data to the MAX522 you need to send a control byte followed by a data byte, as shown in Figure 2
- The control byte format is shown in Table 2, with an example in Table 3. The 8 control bits work as follows:
  - Bit UB3 must be 1, and Bit UB4 must be 0. Bits UB1 and UB2 are don't-cares
  - SA = 0 $\rightarrow$ make Channel A active
    SA = 1 $\rightarrow$ shut down Channel A
  - SB = 0 $\rightarrow$ make Channel B active
    SB = 1 $\rightarrow$ shut down Channel B
  - LA = 0 $\rightarrow$ do not load data into Channel A
    LA = 1 $\rightarrow$ load data into Channel A
  - LB = 0 $\rightarrow$ do not load data into Channel B
    LB = 1 $\rightarrow$ load data into Channel B
- The example shown in Table 3 has SB = 0, SA = 0, LB = 1, and LA = 1. This command will load the same data into Channels A and B, and will make both channels active. The data which goes into the DACs is 0x80, or 128_{10},
- Table 4 shows what the analog output will be for a given analog input. An input of 0x80 will give an analog output of the reference voltage/2. We will use a reference voltage of 5 V, so the digital value of 0x80 should give an output voltage of 2.5 V

A C program to use the MAX522 with the HC12 SPI

```
#include <hc12b32.h>

main()
{
    DDRS = DDRS | 0xE0;           /* SS, SCLK, MOSI outputs */
    PORTS = PORTS | 0x80;         /* Deselect slave */

    SP0CR1 = 0x50;  /* 0 1 0 1 0 0 0 0
                       | | | | | | | |
                       | | | | | | | \____ MSB first
                       | | | | | | \______ multiple bytes with SS asserted
                       | | | | | \________ 0 phase (data valid on 1st edge)
                       | | | | \__________ 0 polarity (clock active low)
                       | | | \____________ Master mode
                       | | \______________ not open drain
                       | \________________ Enable SPI
                       \__________________ No interrupts
                    */

    SP0CR2 = 0;     /* Normal (not bi-directional) mode */

    SP0BR = 0x00;   /* 4 MHz SPI clock */

    PORTS = PORTS & ~0x80;         /* Select slave */

    SP0DR = 0x31;  /* 0 0 1 1 0 0 0 1
                          | |   | |
                          | |   | \____ Load DAC A
                          | |   \______ Don't load DAC B
                          | \__________ DAC A active
                          \____________ DAC B shut down
                    */
    while ((SP0SR & 0x80) == 0) ;  /* Wait for transmit complete */

    SP0DR = 51;                    /* Set DAC A for 1.0 V */
    while ((SP0SR & 0x80) == 0) ;  /* Wait for transmit complete */

    PORTS = PORTS | 0x80;          /* Deselect slave */
```

```
    PORTS = PORTS & ~0x80;          /* Select slave */

    SP0DR = 0x22;  /* 0 0 1 0 0 0 1 0              9
                           | |   | |
                           | |   | \____ Don't load DAC A
                           | |   \_______ Load DAC B
                           | \___________ DAC A active
                           \_____________ DAC B active
                     */
    while ((SP0SR & 0x80) == 0) ;  /* Wait for transmit complete */

    SP0DR = 189;                   /* Set DAC B for 3.7 V */
    while ((SP0SR & 0x80) == 0) ;  /* Wait for transmit complete */

    PORTS = PORTS | 0x80;          /* Deselect slave */
}
```