CpE 390: Microprocessor Systems

Lecture 10

68HC12 Serial Interface – SPI

The SPI Function

- A serial synchronous communication protocol (while SCI is a serial asynchronous communication protocol)

- Mainly used in interfacing with peripherals such as shift registers, LED/LCD display drivers, memory components with serial interface, or A/D and D/A converters that does not need high speed.

- There are two types of device in a network that uses SPI protocol: a master and one or multiple slaves.

- Data transfer in SPI protocol can only be initiated by a master device.

- A microcontroller can be a master or slave device. When configured as a slave device, a microcontroller would respond to the transfer request only when its slave select (SS) input is asserted.

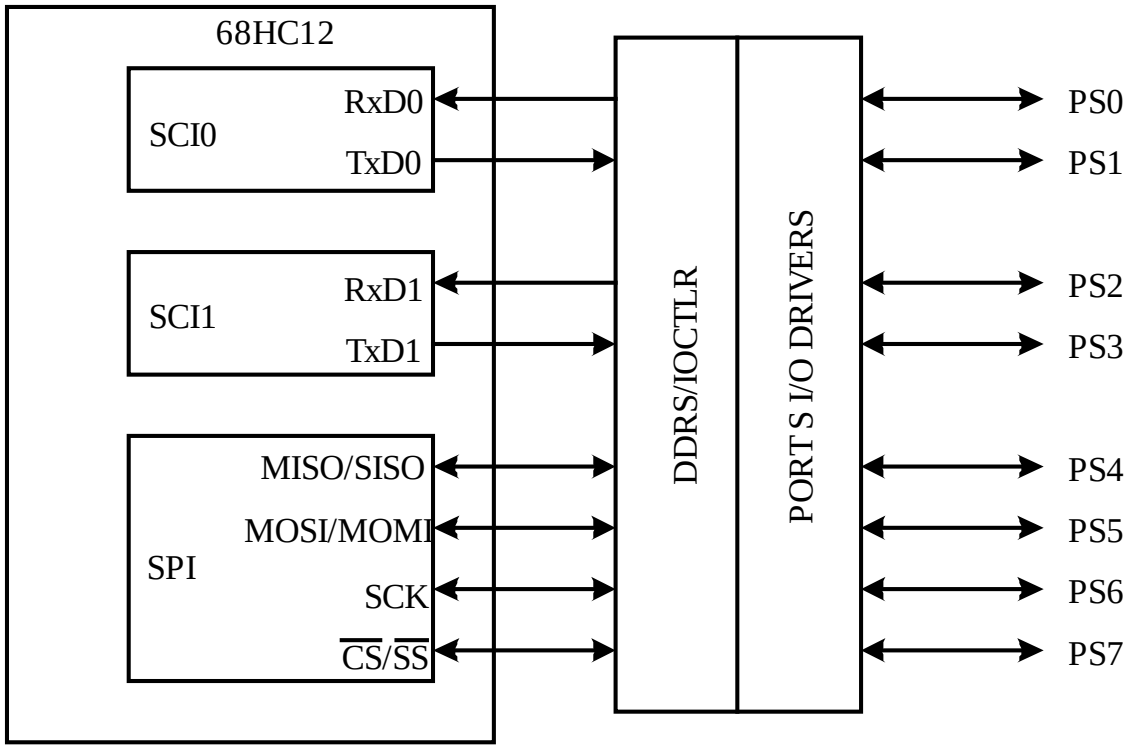
4 PORTS pins required for SPI:

SDI/MISO: master-in-slave-out (serial data input) – msb sent first

SDO/MOSI: master-out-slave-in (serial data output) – msb sent first

SCK: serial clock: synchronize data movement

~CS/~SS: chip select (or slave select)



Note. B family members (912B32, 912BC32, and 912BE32) do not have SCI1

Figure 9.8 Multiple serial interface (SCIs and SPI) block diagram

SPI Operation

- The 8-bit data register in the master and the 8-bit data register in the slave are linked together to form a distributed 16-bit shift register.
- When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the SCK clock from the master so the data is effectively exchanged between the master and the slave.
- Related Registers
    - Two control registers: **SP0CR1** & **SP0CR2**.
    - Baud rate register **SP0BR**.
    - Status register (**SP0SR**)
- Data written into the SP0DR register of the master becomes the output data for the slave.
- Data read from the SP0DR register of the master after a transfer operation is the input data to the slave.

	7	6	5	4	3	2	1	0	
	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF	$00D0
value after reset	0	0	0	0	0	1	0	0	

SPIE: SPI interrupt enable bit
- 0 = SPI interrupts are inhibited
- 1 = SPI interrupt is requested every time the SPIF or MODF status flag is set.

SPE: SPI system enable bit
- 0 = SPI hardware is initialized but is in the disabled state
- 1 = SPI enabled and pins PS4-PS7 are dedicated to SPI function

SWOM: Port S wired-OR mode bit

Controls only pins PS4-PS7.
- 0 = PS4-PS7 output buffers operate normally
- 1 = PS4-PS7 output buffers behave as open-drain outputs

MSTR: SPI master/slave mode select bit
- 0 = slave mode
- 1 = master mode

CPOL and CPHA: SPI clock polarity, clock phase bits

Functions are shown in Figure 9.18a and 9.18b

SSOE: slave select output enable bit

The $\overline{SS}$ output feature is enabled only in master mode by asserting the SSOE and DDRS7.

LSBF: SPI LSB first enable bit
- 0 = data is transferred most-significant bit first
- 1 = data is transferred least-significant bit first

Figure 9.20 SPI control register 1 (SP0CR1)

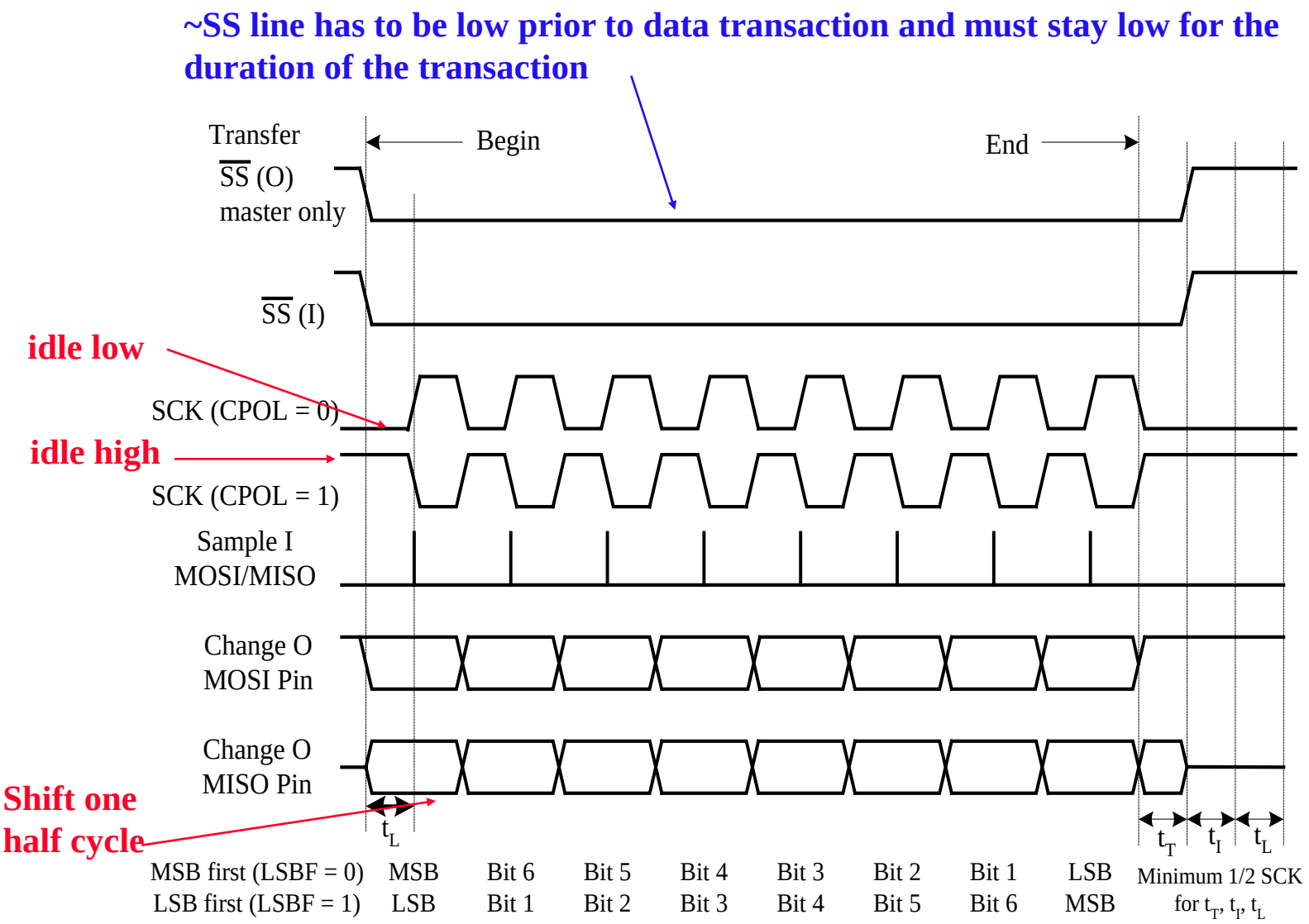


Figure 9.18a SPI Clock format 0 (CPHA = 0)

Four clock formats are used by the SPI system determined by the CPOL and CPHA bits of the SP0CR1

-The 00 and 11 combinations of CPOL and CPHA are for data transfers using the rising edge of the SCK clock whereas the 01 and 10 combinations are for data transfers using the falling edge of the SCK clock.

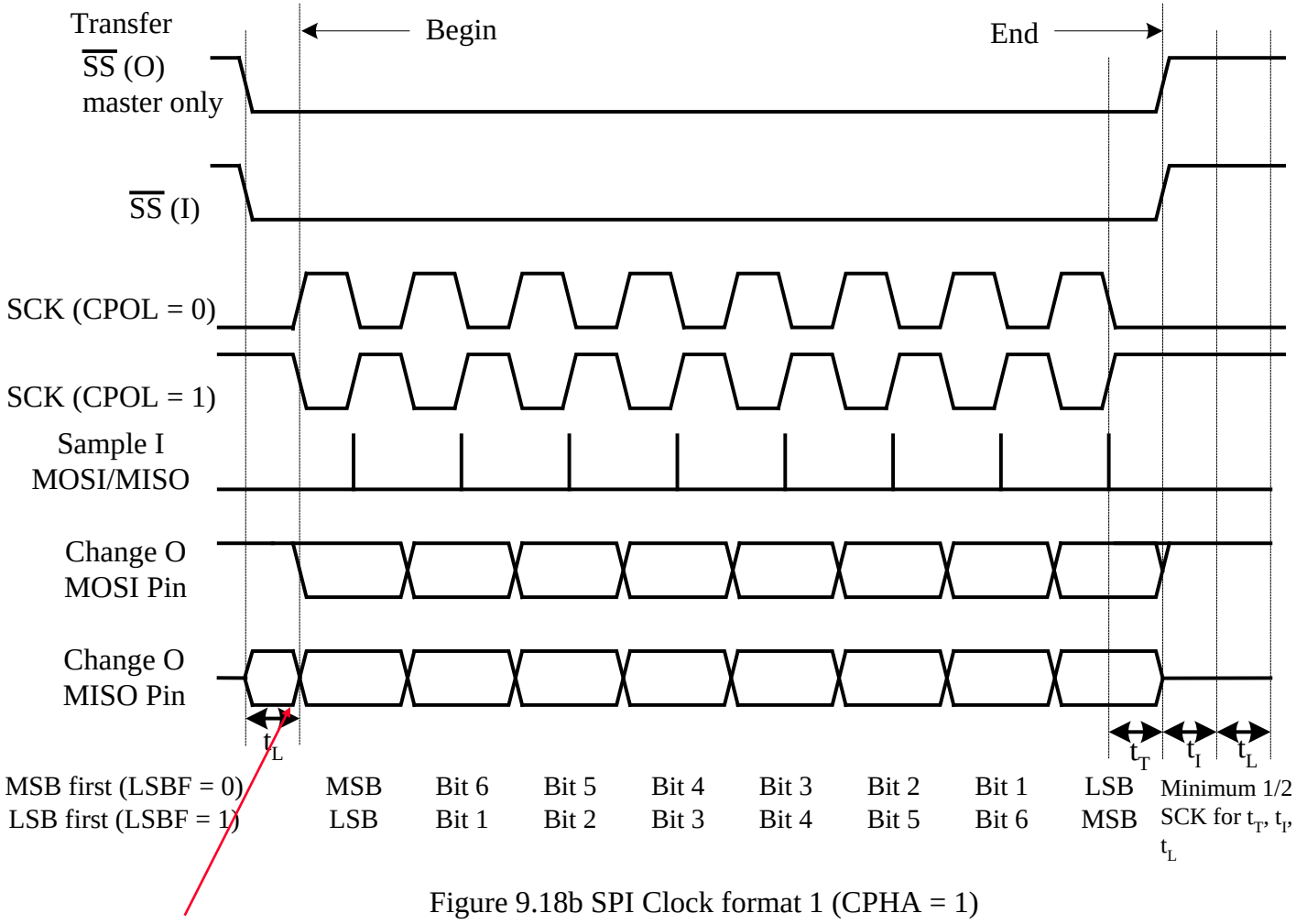
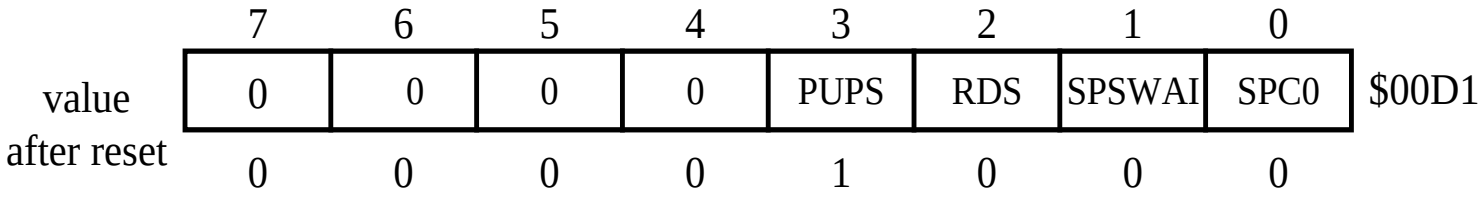


Figure 9.18b SPI Clock format 1 (CPHA = 1)



PUPS: Pullup port S enable bit (not available in 912D60)
0 = no internal pullups on port S
1 = All port S input pins have an active pullup device. If a pin is programmed as output, the pullup device becomes inactive.

RDS: Reduce drive of port S bit (not available in 912D60)
0 = Port S output drivers operate normally
1 = All port S output pins have reduced drive capability for lower power and less noise.

SPSWAI: serial interface stop in wait mode (available only in 912D60, 912DG128, 912DT128, M9S12DP256)
0 = Serial port S clock operate normally
1 = Halt serial interface clock generation in Wait mode

SPC0: Serial pin control 0 bit
This bit decides serial pin configurations with MSTR control bit. All possible combinations are shown in Table 9.5.

Figure 9.21 SPI control register 2 (SP0CR2)

In bidirectional mode, the SPI uses only one serial data pin for an external device interface (see Fig. 9.19)

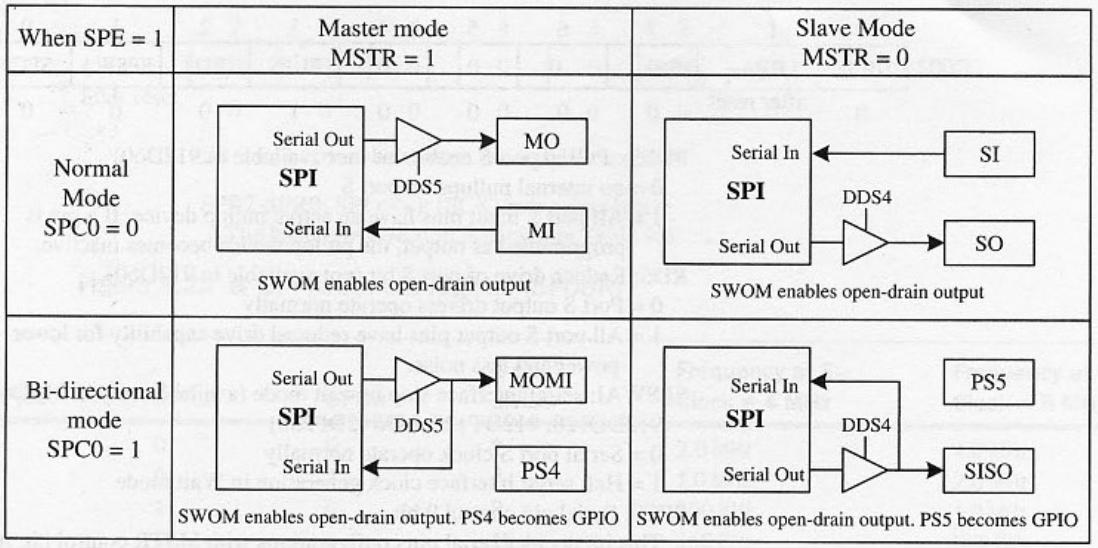


Figure 9.19 ■ Normal mode and bidirectional mode

Table 9.5 MISO and MOSI pin configurations

Pin mode		SPC0(1)	MSTR	MISO(2)	MOSI(3)	SCK (4)	SS (5)
#1	normal	0	0	slave out	slave in	SCK in	SS in
#2			1	master in	master out	SCK out	SS I/O
#3	bidirectional	1	0	slave I/O	general-purpose I/O	SCK in	SS in
#4			1	general-purpose I/O	master I/O	SCK out	SS I/O

1. The serial pin control 0 bit enables bidirectional configurations.
2. Slave output is enabled if DDRS4 = 1, SS = 0, and MSTR = 0, (#1, #3)
3. Master output is enabled if DDRS5 = 1 and MSTR = 1, (#2, #4)
4. SCK output is enabled if DDRS6 = 1 and MSTR = 1, (#2, #4)
5. SS output is enabled if DDRS7 = 1, SSOE = 1, and MSTR = 1, (#2, #4)

	7	6	5	4	3	2	1	0	
	0	0	0	0	0	SPR2	SPR1	SPR0	$00D2
value after reset	0	0	0	0	0	0	0	0	

Bit 7 to 3 cannot be written but will be read as 0s

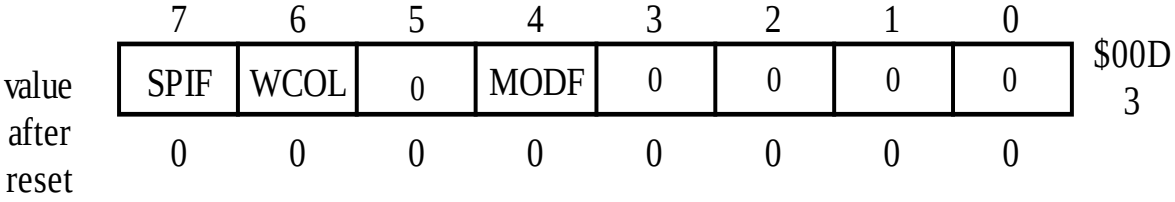
SPR2-SPR0: SPI clock (SCK) rate select bits

The baud rate selection is shown in Table 9.6.

Figure 9.22 SPI baud rate register (SP0BR)

Table 9.6 SPI clock rate selection

SPR2	SPR1	SPR0	E clock divisor	Frequency at E Clock = 4 MHz	Frequency at E Clock = 8 MHz
0	0	0	2	2.0 MHz	4.0 MHz
0	0	1	4	1.0 MHz	2.0 MHz
0	1	0	8	500 KHz	1.0 MHz
0	1	1	16	250 KHz	500 KHz
1	0	0	32	125 KHz	250 KHz
1	0	1	64	62.5 KHz	125 KHz
1	1	0	128	31.3 KHz	62.5 KHz
1	1	1	256	15.6 KHz	31.3 KHz



SPIF: SPI interrupt request bit

SPIF is set after the eight SCK cycles in a data transfer, and it is cleared by reading the SP0SR register (with SPIF set) followed by an access to the SPI data register

WCOL: write collision

0 = no write collision

1 = Indicates that a serial transfer is in progress when the microcontroller tried to write new data into the SP0DR register.

MODF: mode error interrupt status flag

This bit is set if the MSTR control bit is set and the slave select input pin becomes low. This condition is not permitted in normal operation. This flag is cleared by a read of the SP0SR register followed by a write to the SP0CR1 register.

Figure 9.23 SCI status register (SP0SR)

- The application program can check bit 7 of the SP0SR or wait for the SPI interrupt to find out if the SPI transfer has completed.
- When transferring data in high frequency using the SPI format, using the polling method is more efficient due to the overhead involved in interrupt handling

- The 8-bit data register (SP0DR) is both the input and output register for SPI data.

- When the CPU writes a byte into SP0DR, eight clock signals are generated to shift out the 8-bit data written into this register.

- All port S pins can be configured to have reduced drive capability. All port S input pins can be programmed to have pullup devices. Both are done via the PURDS register.

- Port S pins that are not used in the SPI or SCI can be used for general I/O. They can be accessed by reading/writing from/to the PORTS register. Their data direction can be set by programming the DDRS register.

	7	6	5	4	3	2	1	0	
	0	RDPS2	RDPS1	RDPS0	0	PUPS2	PUPS1	PUPS0	$00DB
value after reset	0	0	0	0	0	0	0	0	

Bit 7 and 3 cannot be written but will be read as 0s

RDPS2: reduce drive PS7-PS4
- 0 = Port S output drives for bits 7 to 4 operate normally
- 1 = Port S output pins for bits 7 to 4 have reduced drive capability for lower power and less noise

RDPS1: reduce drive of PS3 and PS2
- 0 = Port S output drives for bits 3 and 2 operate normally
- 1 = Port S output pins for bits 3 and 2 have reduced drive capability for lower power and less noise

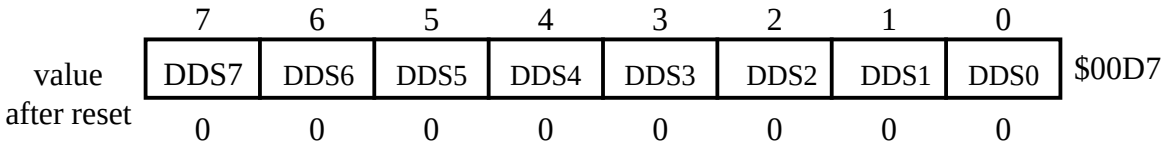
RPDS0: reduce drive of PS1 and PS0
- 0 = Port S output drives for bits 1 and 0 operate normally
- 1 = Port S output pins for bits 1 and 0 have reduced drive capability for lower power and less noise

PUPS2: Pullup port S enable PS7-PS4
- 0 = No internal pullups on port S bits 7 to 4
- 1 = Port S input pins for bits 7 to 4 have an active pullup device. If a pin is programmed as output, the pullup device becomes inactive.

PUPS1: Pullup port S enable PS3 and PS2
- 0 = No internal pullups on port S bits 3 and 2
- 1 = Port S input pins for bits 3 and 2 have an active pullup device.

PUPS0: Pullup port S enable PS1 and PS0
- 0 = No internal pullups on port S bits 1 and 0
- 1 = Port S input pins for bits 1 and 0 have an active pullup device.

Figure 9.24 Pullup and reduced drve register for port S (PURDS)



After reset, all general-purpose I/O are configured for input only.
0 = configure the corresponding I/O pin for input
1 = configure the corresponding I/O pin for output

DDS2 & DDS0 -- Data direction for port S bit 2 and 0
If the SCI receiver are configured for 2-wire SCI operation, corresponding port S pins are input regardless of the state of these bits.

DDS3 & DDS1 -- Data direction for port S bit 3 and 1
If the SCI receiver are configured for 2-wire SCI operation, corresponding port S pins are output regardless of the state of these bits.

DDS6-DDS4: Data direction for port S bits 6 to 4
If the SPI is enabled and expects the corresponding port S pin to be an input, it will be an input regardless of the state of the DDRS bit. If the SPI is enabled and expects the bit to be an output, it wil be an output only if the DDRS bit is set.

DDS7 -- Data direction for port S bit 7
In SPI slave mode, DDS7 has no effect; the PS7 pin is dedicated as the SS input. In SPI master mode, DDS7 determines whether PS7 is an error detect input to the SPI or a general-purpose or slave select output line.

Figure 9.25 Port S data direction register (DDRS)

Example 9.9 Suppose that there is an SPI-compatible peripheral output device that has the following characteristics:

- Has a *clk* input pin that is used as the data shifting clock signal
- Has a *data_in* pin to shift in data on the rising edge of the clk input
- Has a CS pin, which must be low to enable the data to be shifted in
- Highest data shift rate is 2 MHz
- Most significant bit shifted in first

Describe how to connect the SPI pins of the HC912BC32 and this device. Write a sequence of instruction to configure the SPI module properly for data transfer.

Solution:

- The pin connection is shown in Figure 9.26.
- Configure the SS, SCK, and MOSI pins for output
- Also configure PS1 for output and PS0 for input

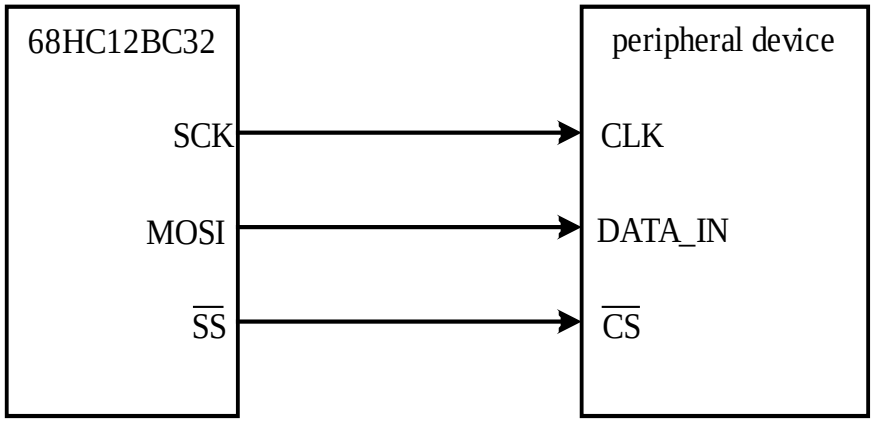


Figure 9.26 Circuit connection between the 68HC12 SPI and the peripheral device

The SP0BR Register Setting

- The data transfer rate is 2 MHz for the 8 MHz E clock, set the SPI clock divide factor to 4. Write the value $01 into the SP0BR register.

SP0BR register

0	0	0	0	0	SPR2	SPR1	SPR0
					0	**0**	**1**

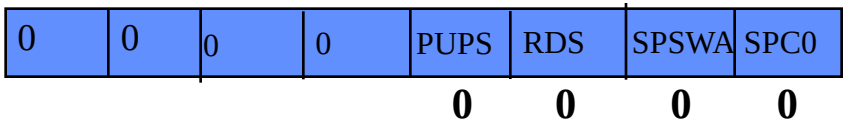
The SP0CR1 Register setting

- Enable SPI subsystem, master mode, and SS pin
- Disable SPI interrupt
- Normal SPI pins
- Rising edge to shift data and most significant bit first
- Write the vale $52 into the SP0CR1

SP0CR1 register

SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF
0	**1**	**0**	**1**	**0**	**0**	**1**	**0**

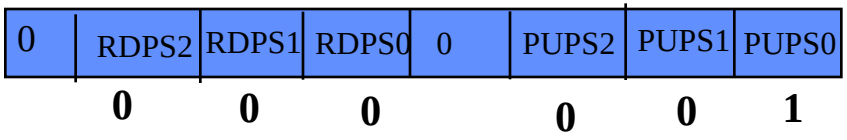
SP0CR2 register



The SP0CR2 Register Setting

- Enable Port S pins internal pullup
- Select normal port S output drive
- Normal MOSI pin
- Write the value $00 into the SP0CR2 register

PURDS register



The PURDS Register Setting

- Select normal drive
- Enable pullup
- Write the value $01 into this register
- The following instruction sequence will achieve the specified parameters:

```
movb    #$E2,DDRS
movb    #$01,SP0BR
movb    #$52,SP0CR1
movb    #$00,SP0CR2
movb    #$01,PURDS
```

SPI Circuit Connection

- In a system that use SPI to exchange data, one device must be the master and other devices must be slave devices. The master device controls the data transfer and can control more than one slave devices simultaneously

- In a single-slave configuration, the circuit would be connected as Fig.9.27. The user can set the SS output enable feature which will automatically goes low to enable the slave device before the data transfer is started.-

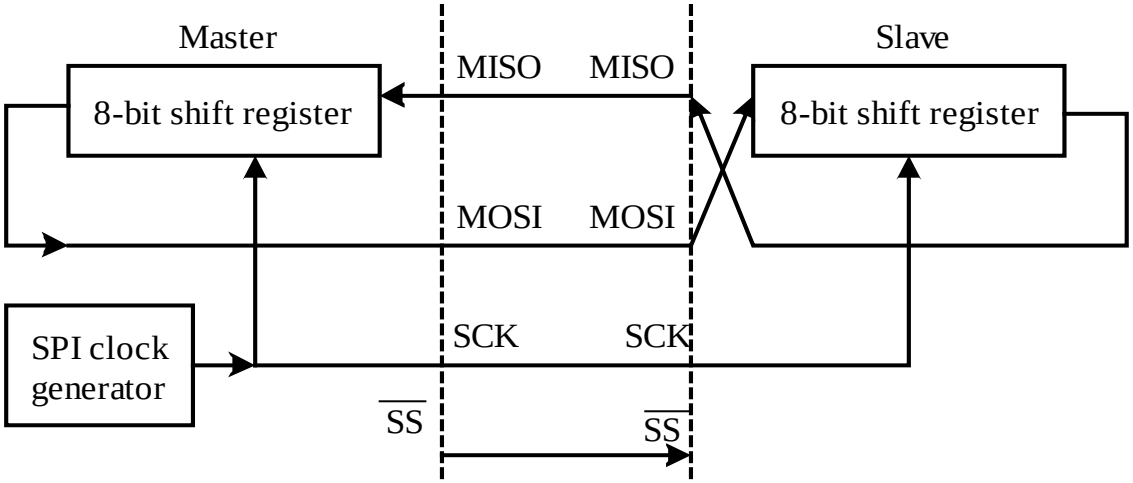


Figure 9.27 SPI master-slave interconnection

- In this multi-slave environment, the 68HC12 can select any slave device for data transfer. We use port P pins to drive the ~SS inputs for peripheral devices

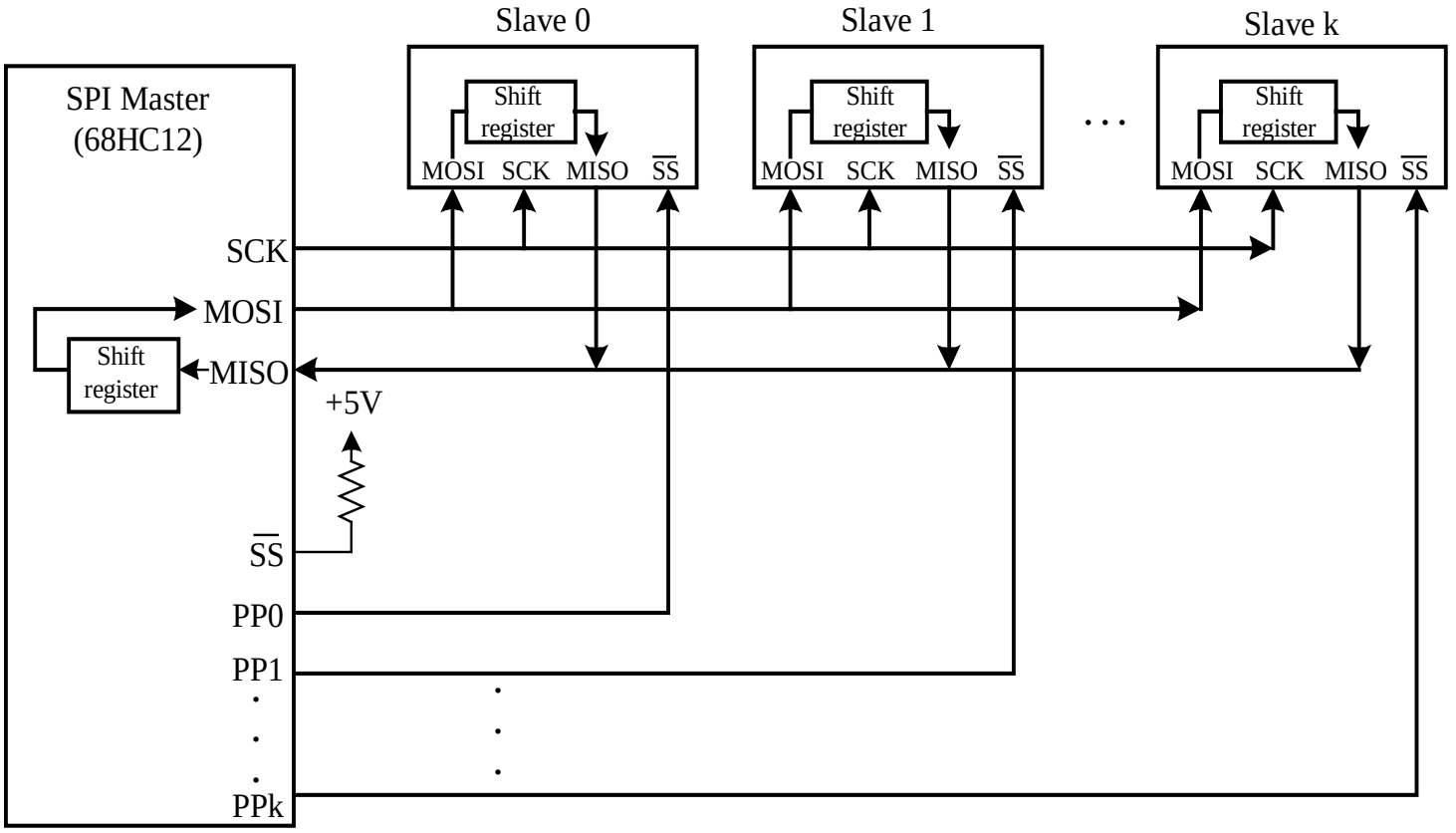


Figure 9.28 Single-master and multiple-slave device connection (method 1)

- In this multi-slave environment, the shift registers of the SPI master and slaves becomes a ring. In this configuration, a minimal number of pins control a large number of peripheral devices.
- However, the master does not have freedom to select an arbitrary slave device for data transfer without going through other slave devices.

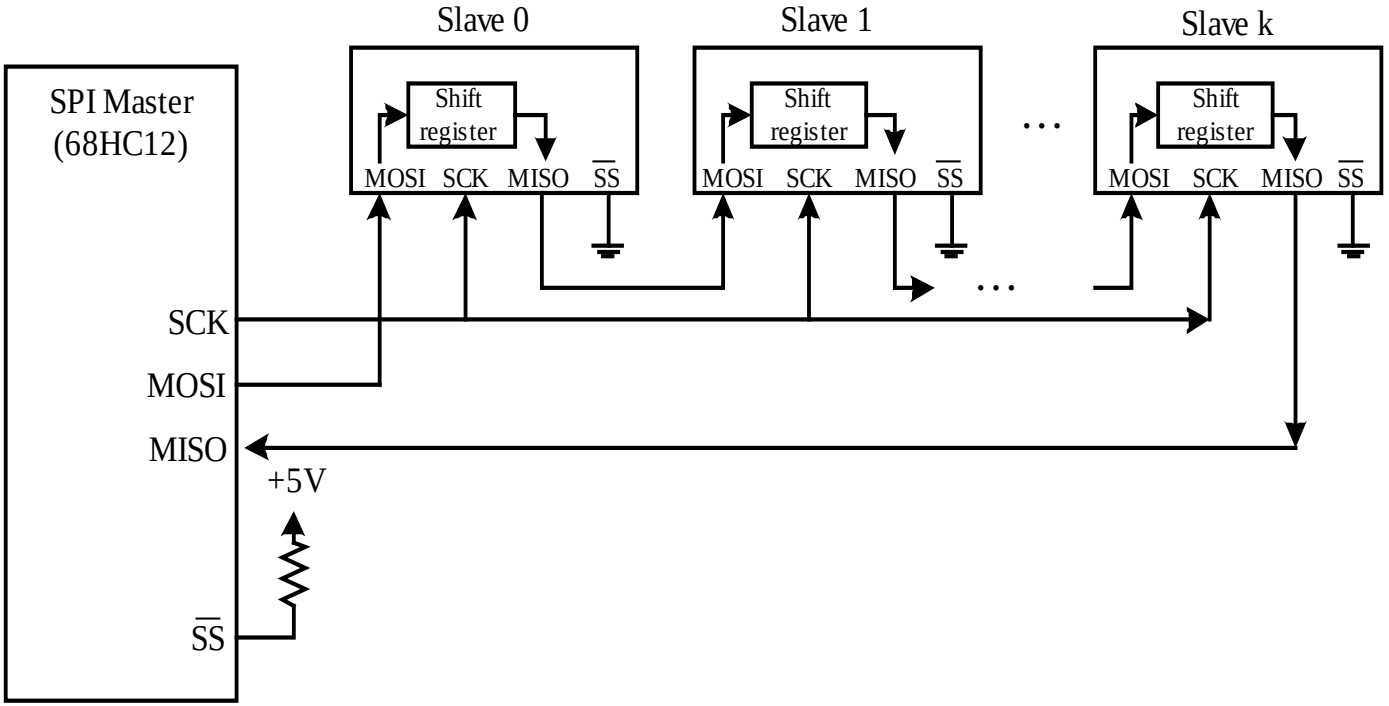


Figure 9.29 Single-master and multiple-slave device connection (method 2)

The HC595 Shift Register

- The HC595 consists of an 8-bit shift register and a D-type latch with three-state parallel output.
- The shift register provides parallel data to the latch.
- The maximum data shift rate is 6 MHz.

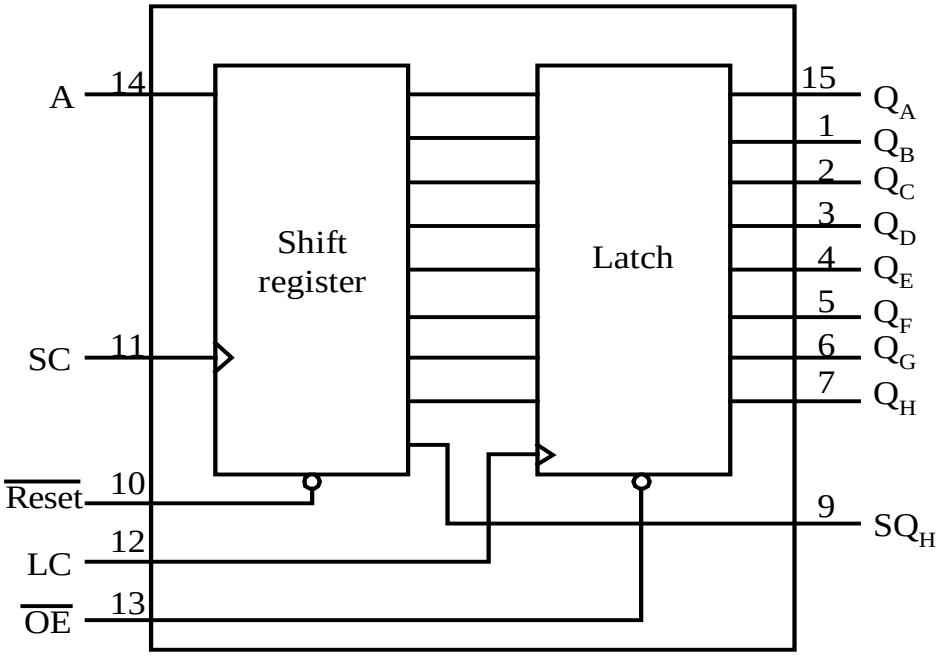


Figure 9.30 HC595 block diagram and pin assignment

Signal Pins of the HC595

- A: serial data input
- SC: shift clock. A low-to-high transition on this pin causes the data at the serial input pin to be shifted into the 8-bit shift register.
- Reset. A low on this pin resets the shift register portion of this device.
- LC: latch clock. A low-to-high transition on this pin loads the contents of the shift register into the output latch.
- OE: output enable. A low on this pin allows the data from the latches to be presented at the outputs.
- Q_A to Q_H: tri-state latch output
- SQ_H: the output of the eight stage of the shift register

Applications of the HC595

- The HC595 is often used to add parallel ports to the microcontroller.
- One possible connection is shown in Figure 9.31.

Output multiple bytes

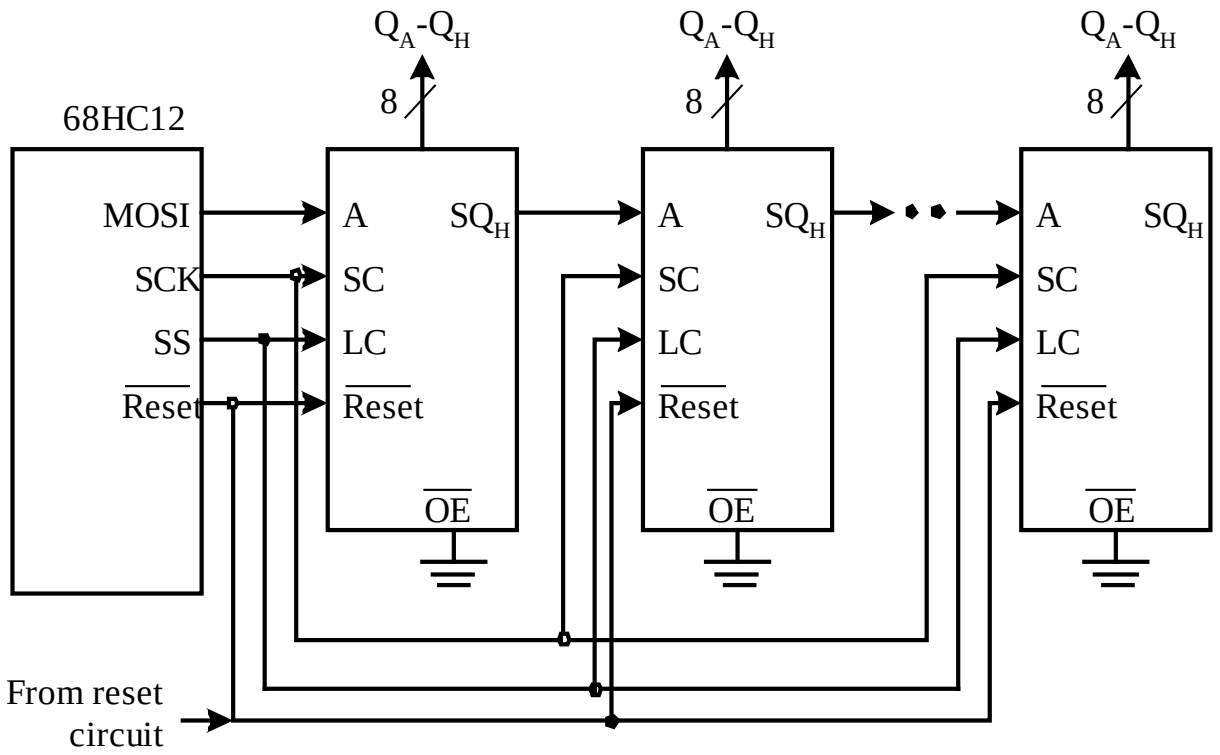


Figure 9.31 Serial connection of multiple HC595s to the SPI

Procedure for Outputting Multiple Bytes in Figure 9.31

Step 1

Program the DDRS register to configure each SPI pin and the TxD and RxD pins properly. Write the value $E2 into the DDRS register.

Step 2

Program the SP0CR1 to enable SPI function, using the rising edge of SCK to shift data in and out, select master mode, normal port S pins, and set the data rate to 4 Mbits/sec, configure the SS pin as a general-purpose output pin, shift data out most significant bit first. Write the value $50 into the SP0CR1 register and write the value $00 into the SP0BR register. Write the value $00 into the SP0CR2 register.

Step 3

Choose normal mode for all port S pins. Write the value $01 into the PURDS register.

Step 4

Write a byte into the SP0DR register to trigger eight pulses from the SCK pin.

Step 5

Repeat step 4 as many times as needed.

Step 6

Set the SS pin to low and then pull it to high to load the data in the shift register of each HC595 into the output latch. After this step, the Q_A through Q_H pins of each HC595 contains valid data.

Example 9.10 Write a program to output the contents of 8 bytes that are stored from $800 to the HC595s in Figure 9.31.

Solution: The program is as simple as follows:

```
#include "d:\miniide\hc12.inc"
            org     $800
buf         db      1,2,3,4,5,6,7,8
            org     $1000
            jsr     spi_init        ; initialize the SPI function
            ldx     #$800           ; use X as the array pointer
            ldab    #8              ; use B as the loop counter
loop        ldaa    1,x+            ; read one byte and move the pointer
            staa    SP0DR
            brclr   SP0SR,$80,*     ; wait for the SPI transfer to complete
            dbne    b,loop
            swi
spi_init    movb    #$E2,DDRS
            movb    #$50,SP0CR1
            movb    #$00,SP0CR2
            movb    #$00,SP0BR
            movb    #$01,PURDS
            rts
            end
```

Another Method of Connecting Multiple HC595s

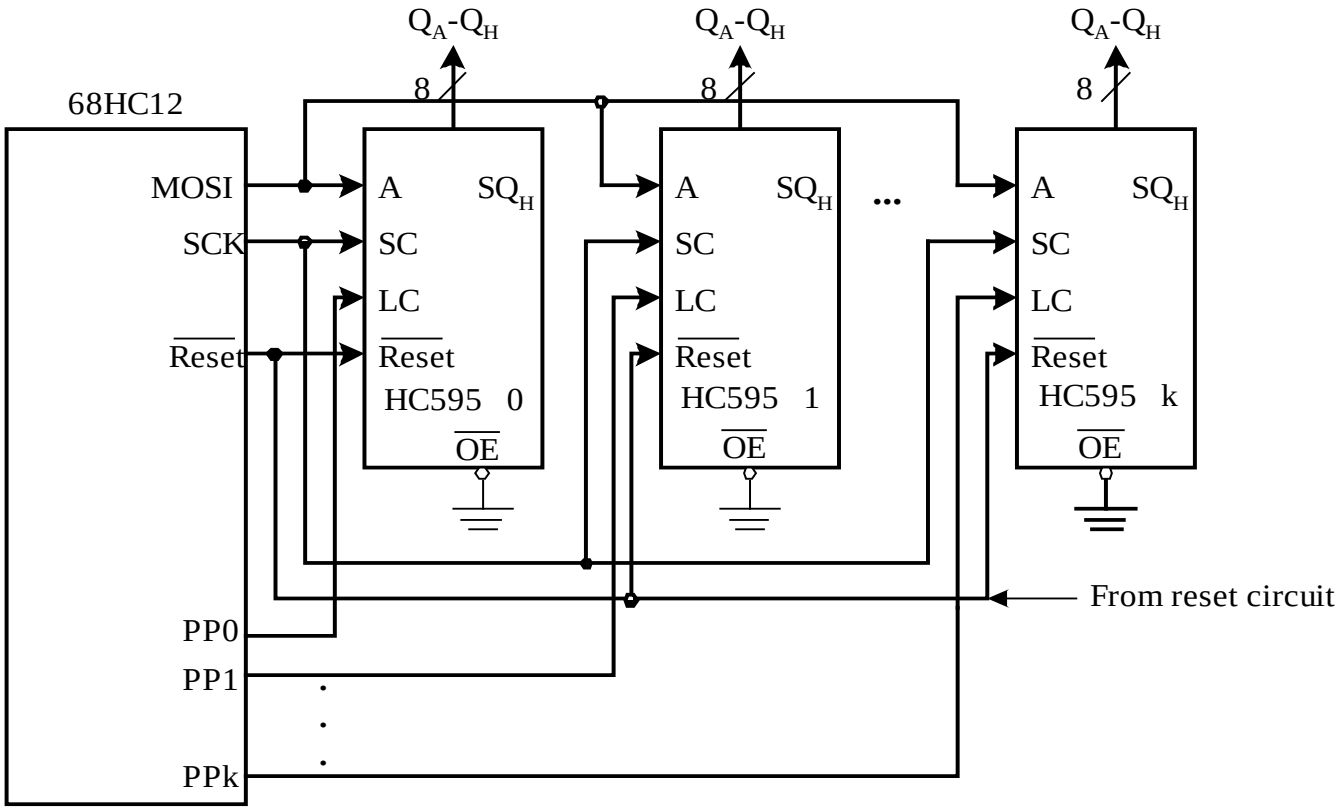


Figure 9.32 Parallel connection of multiple HC595s to the SPI

Homework #10

- See course website: http://390.revan.us
  - click homework tab
- Please submit a hard copy