
Synchronous Serial Peripheral Interface (SPI)

Chapter 15.3

Web reference:

<http://www.embedded.com/story/OEG20020124S0116>

http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

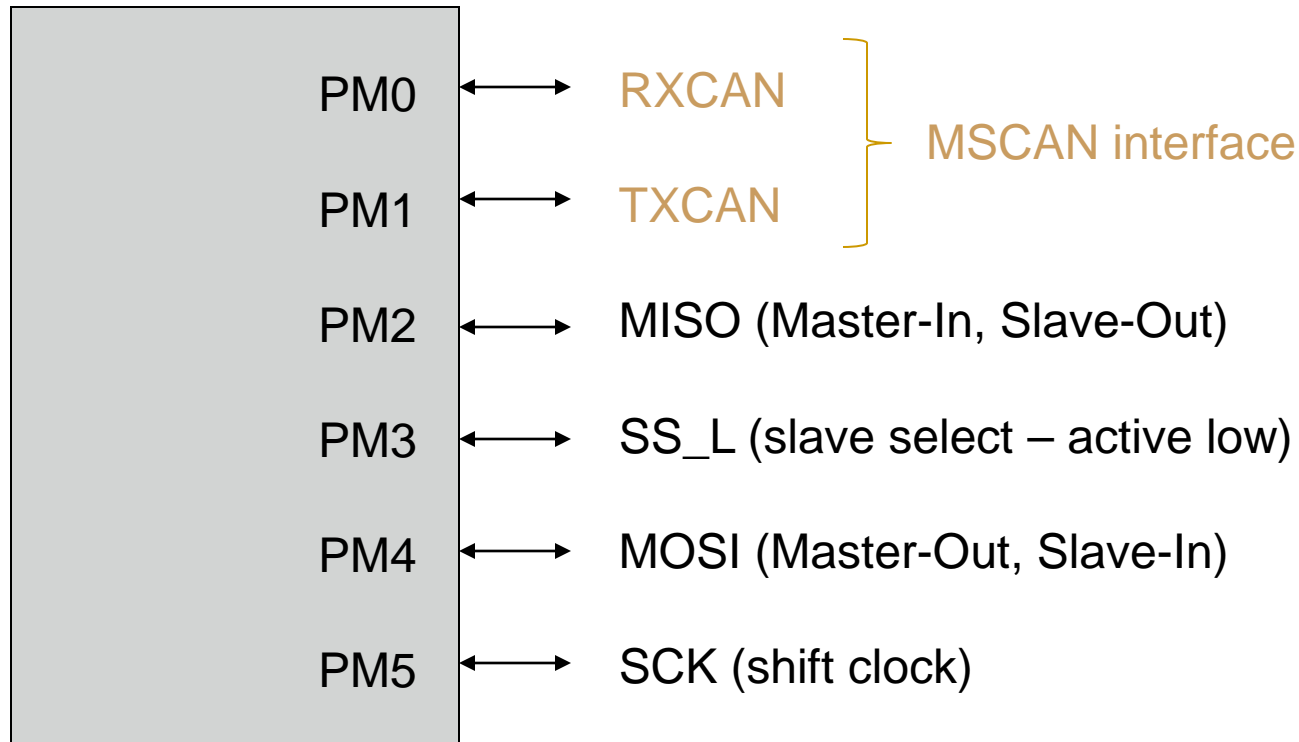
Synchronous Serial Peripheral Interface (SPI)

- Low cost interface between devices
 - CPUs and peripheral functions
 - Good for data streaming at high data rates
 - Typical applications
 - real-time clocks
 - serial EPROMs
 - data converters (A/D, D/A)
 - additional I/O ports
 - display drivers (LCD, LED)
-

SPI features

- Transmitter & receiver synchronized to a common clock
 - supports high data rates
 - Hardware based on shift registers
 - output of one SPI module connected to input of the other – creating a single shift register
 - Master/slave operation
 - master generates clock & “selects” slave device(s)
 - master may control multiple slave devices
-

HCS12 Port M (PORTM \$0250, DDRM=\$0252)



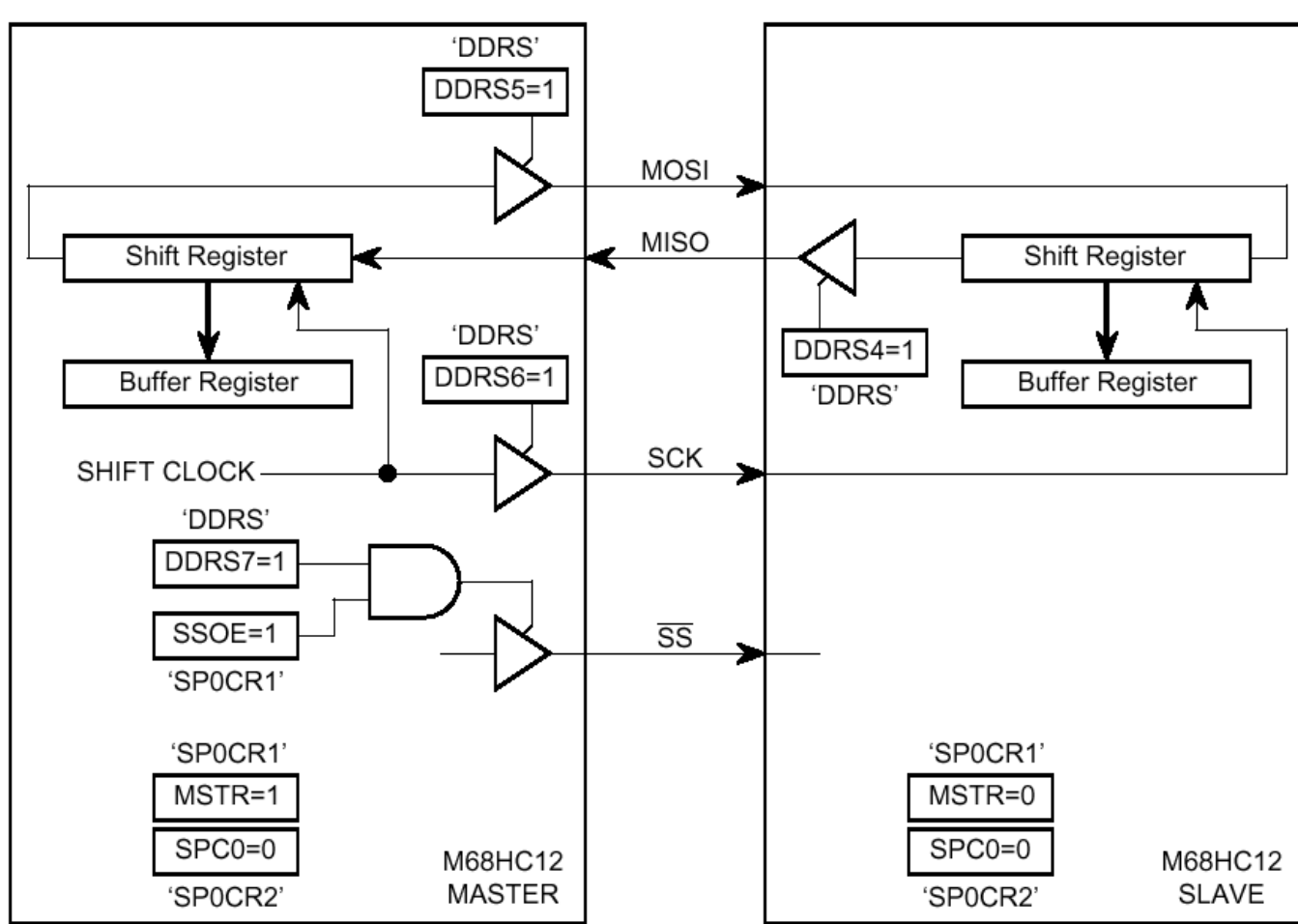
SPI Data Register = \$00DD

SPI Interrupt vector stored at \$FFD8:FFD9

interrupt if **SPF & SPIE = 1**

SPI master/slave normal two-wire operation

(Figure 11-7)



SPI Control Register 1

(SPICR1 = \$00D8)

7	6	5	4	3	2	1	0
SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE

SPIE : SPI interrupt enable

SPE: SPI system enable (Port S, bits 4-7)

SPTIE: SPI transmitter interrupt enable

MSTR: 1=master mode, 0=slave mode

CPOL,CPHA: SPI clock polarity & clock phase

select when slave samples data (rise/fall lead/trail edge)

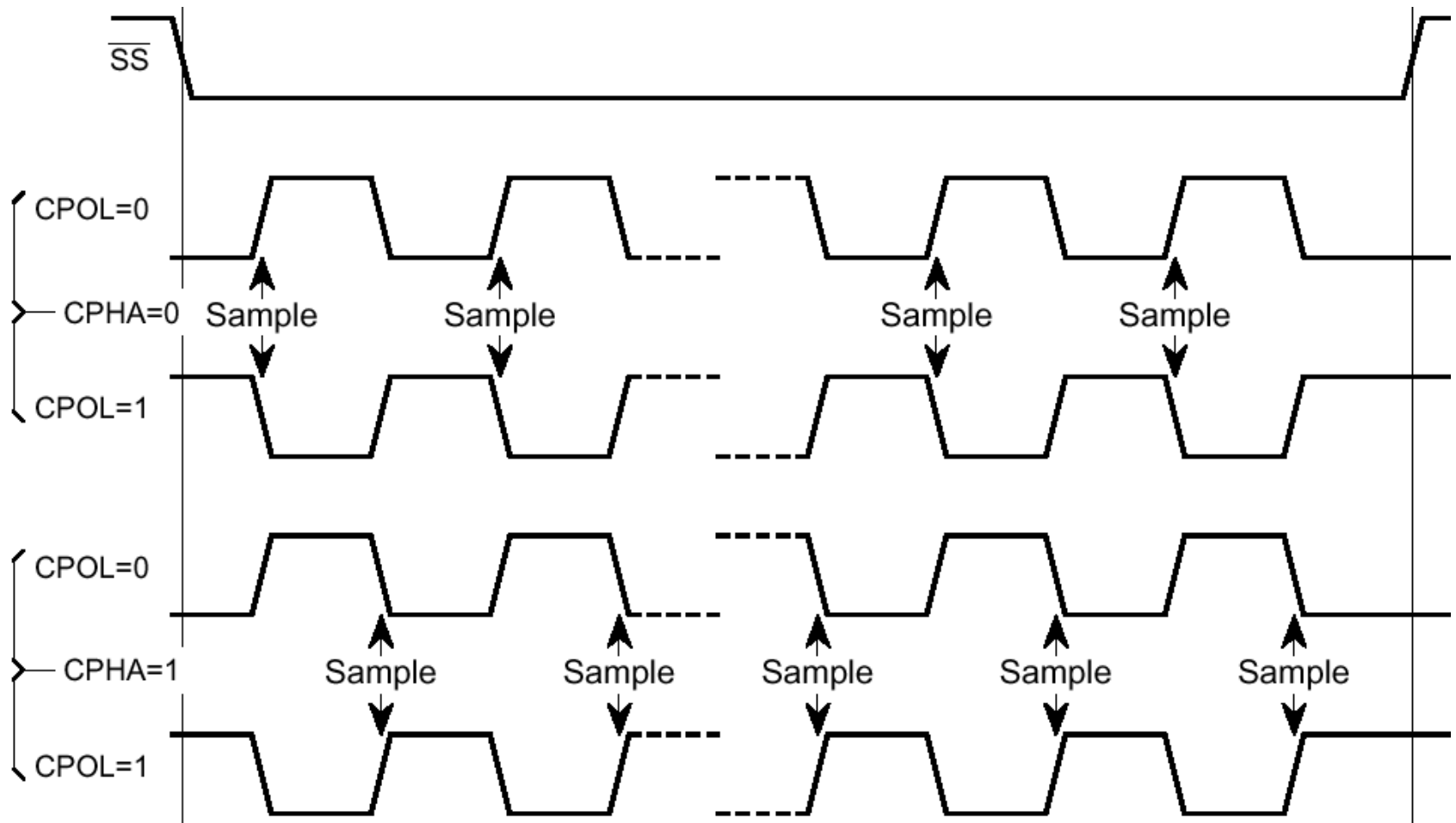
SSOE: slave select output enable

assert SS* when SSOE=1 and DDRS7=1

LSBFE: least significant bit first enable

1=LSB first, 0=MSB first (default)

SPI clock phases. (Figure 15-6)



Clock generated by SPI master

SPI Baud Rate Register

(SPIBR = \$00DA)

7	6	5	4	3	2	1	0
0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0

SPPR[2:0] = baud rate preselection

SPR[2:0] = baud rate selection

Define SCK generated by SPI master

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \times 2^{(\text{SPR} + 1)}$$

$$\text{BaudRate} = \text{BusClock} / \text{BaudRateDivisor}$$

Example:

Assume 8 MHz bus clock

`movb #$41, SP0BR ;SPPR=4, SPR=1`

SCK frequency = 8MHz / (5x2²) = 400KHz

TABLE 15-6 SPI Baud Rate Selection (8-MHz Bus Clock)

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	SPIBR Value ₁₀	Baud Rate Divisor	Baud Rate
0	0	0	0	0	0	0	2	4.00 MHz
0	0	0	0	0	1	1	4	2.00 MHz
0	1	0	0	0	0	32	6	1.33 MHz
0	0	0	0	1	0	2	8	1.00 MHz
1	0	0	0	0	0	64	10	800.00 kHz
0	1	0	0	0	1	33	12	666.67 kHz
1	1	0	0	0	0	96	14	571.43 kHz
0	0	0	0	1	1	3	16	500.00 kHz
1	0	0	0	0	1	65	20	400.00 kHz
0	1	0	0	1	0	34	24	333.33 kHz
1	1	0	0	0	1	97	28	285.71 kHz
0	0	0	1	0	0	4	32	250.00 kHz
1	0	0	0	1	0	66	40	200.00 kHz
0	1	0	0	1	1	35	48	166.67 kHz
1	1	0	0	1	0	98	56	142.86 kHz
0	0	0	1	0	1	5	64	125.00 kHz
1	0	0	0	1	1	67	80	100.00 kHz
0	1	0	1	0	0	36	96	83.33 kHz
1	1	0	0	1	1	99	112	71.43 kHz
0	0	0	1	1	0	6	128	62.50 kHz
1	0	0	1	0	0	68	160	50.00 kHz
0	1	0	1	0	1	37	192	41.67 kHz
1	1	0	1	0	0	100	224	35.71 kHz
0	0	0	1	1	1	7	256	31.25 kHz
1	0	0	1	0	1	69	320	25.00 kHz
0	1	0	1	1	0	38	384	20.83 kHz
1	1	0	1	0	1	101	448	17.86 kHz
0	0	1	1	1	1	23	512	15.63 kHz
1	0	0	1	1	0	70	640	12.50 kHz
0	1	0	1	1	1	39	768	10.42 kHz
1	1	0	1	1	0	102	896	8.93 kHz
0	1	1	1	1	1	55	1024	7.81 kHz
1	0	0	1	1	1	71	1280	6.25 kHz
1	0	1	1	1	1	87	1536	5.21 kHz
1	1	0	1	1	1	103	1792	4.46 kHz
1	0	1	1	1	1	119	2048	3.91 kHz

SPI Control Register 2

(SPICR2 = \$00D8)

7	6	5	4	3	2	1	0
0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0

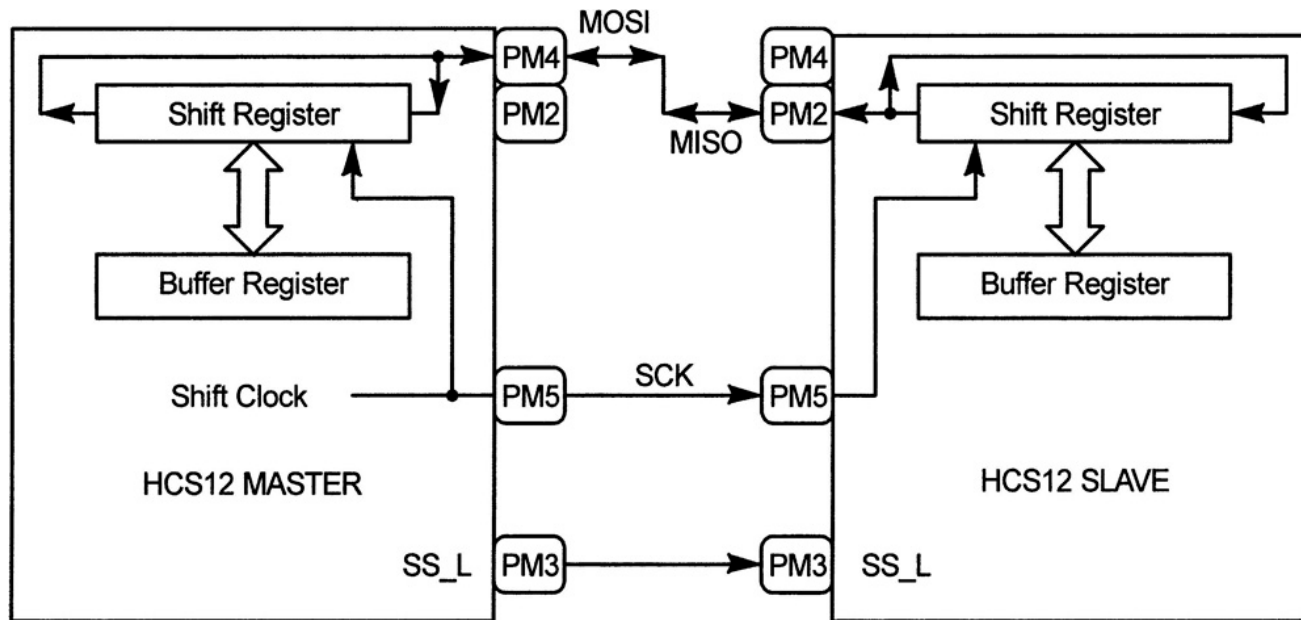
MODFEN: Mode fault enable (Use SS_L pin with mode fault feature)

BIDIROE: Bidirectional mode output buffer enable (default=disabled)

SPISWAI: Stop in “wait mode” (default is not to stop)

SPC0: serial pin control 0

1 = bidirectional mode, 0 = normal 2-wire mode



Master Sending

SPICR1:SPE = 1
 SPICR1:MSTR = 1
 SPICR1:SSOE = 0
 SPICR2:MODFEN = 1
 SPICR2:BIDIROE = 1
 SPICR2:SPC0 = 1

Slave Receiving

SPICR1:SPE = 1
 SPICR1:MSTR = 0
 SPICR1:SSOE = x
 SPICR2:MODFEN = x
 SPICR2:BIDIROE = 0
 SPICR2:SPC0 = 1

Master Receiving

SPICR1:SPE = 1
 SPICR1:MSTR = 1
 SPICR1:SSOE = 0
 SPICR2:MODFEN = 1
 SPICR2:BIDIROE = 0
 SPICR2:SPC0 = 1

Slave Sending

SPICR1:SPE = 1
 SPICR1:MSTR = 0
 SPICR1:SSOE = x
 SPICR2:MODFEN = x
 SPICR2:BIDIROE = 1
 SPICR2:SPC0 = 1

Figure 15-4 SPI bidirectional mode.

SPI Status Register

(SPISR = \$00DB)

7	6	5	4	3	2	1	0
SPIF	0	SPTEF	MODF	0	0	0	0

SPIF : SPI interrupt request flag

- set at end of SPI transfer
- clear by reading SP0SR when SPIF set & read/write SP0DR
- interrupt if SPIE=1 in control register 1

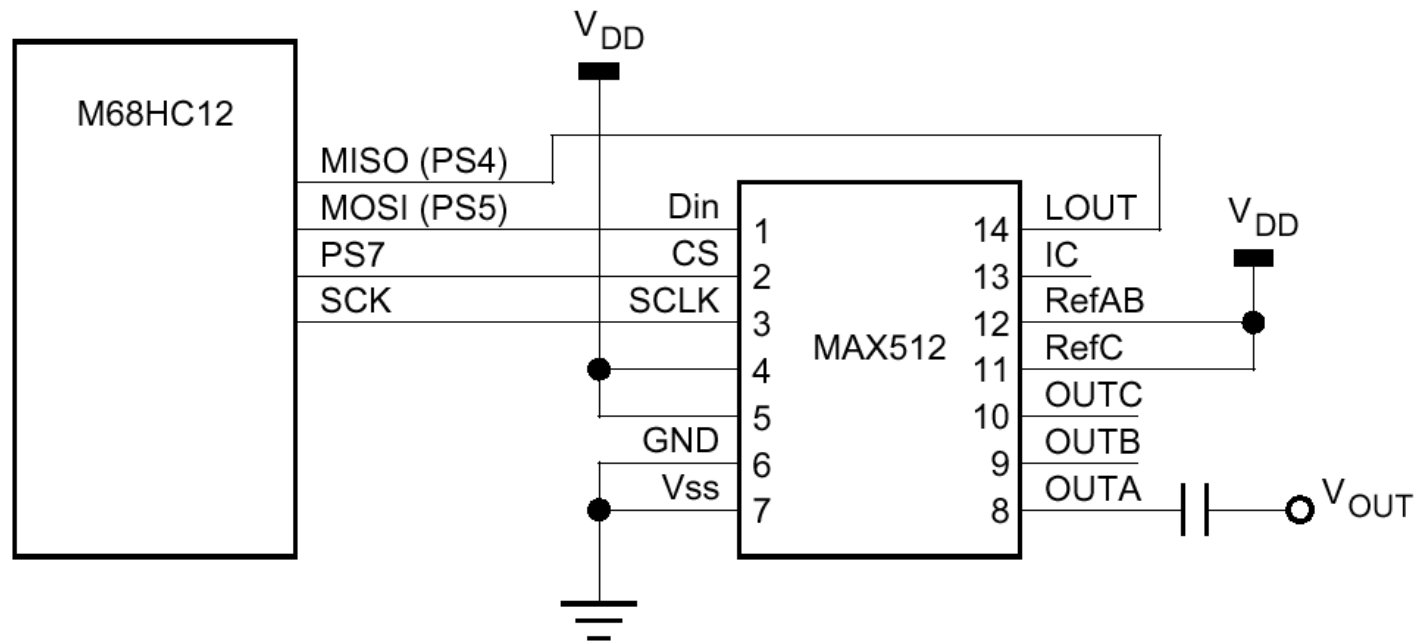
SPTEF: SPE Transmit empty interrupt flag

- set if SPE data register empty
- clear by reading SPISR followed by write to SPIDR

MODF: mode error flag

- set if SS* pulled low while SPI is in master mode
(another SPI device trying to be master)

SPI Example: Serial D/A



- MAX512 is a serial 8-bit D/A converter
 - Din = serial digital data input, LOUT = serial output
 - SCLK = serial clock
 - \overline{CS} = chip select (active low – to start receiving)
 - RefAB, RefC = reference voltages
 - OUTA, OUTB = analog output

HCS12 SPI I/O Example (1)

- ; Continuously output sawtooth wave to SPI port.
- ; Using MAX512 SPI D/A converter to produce analog output

;; Bits for SPI control/status registers

```
SPIF EQU %10000000 ; SPI interrupt flag (SR)
SPE EQU %01000000 ; SPI enable (CR1)
MSTR EQU %00010000 ; Master select (CR1)
SS EQU %10000001 ; Slave select (CR1)
SMASK EQU %11100000 ; Enable Port S bits 5,6,7 as outputs
SPISETUP EQU SPE|MSTR ; Enable SPI as master device
DASETUP EQU %10000111 ; Maxim D/A control – enable all outputs
```

```
ORG PROG
lds #STACK
movb #SMASK,DDRS ;Port S bits 7-5 outputs
movb #SPISETUP,SPICR1 ;Enable SPI in master mode
clr SPIBR ;Set SCLK to 4 MHz
clr Sawtooth ;Clear variable
```

SPI D/A example (2)

; Main loop – continuously output a sawtooth wave

```
Loop:  bclr    PORTS,SS           ;SS low to select D/A
       movb   #DASETUP,SPIDR   ;Write setup byte to D/A

Spin:  brclr   SPISR,SPIF,Spin   ;wait for SPIF (cmd sent on MOSI)
       ldaa   SPISR            ;clear SPIF – read SR
       ldaa   SPIDR            ;           and DR
       movb   Sawtooth,SPIDR   ;data to convert
       inc    Sawtooth

Spin2: brclr   SPISR,SPIF,Spin2  ;wait for SPIF (data sent on MOSI)
       ldaa   SPISR            ;clear SPIF – read SR
       ldaa   SPIDR            ;           and DR
       bset   PORTS,SS         ;SS high to deselect D/A
       bra    Loop             ;repeat
```

Master with multiple slaves

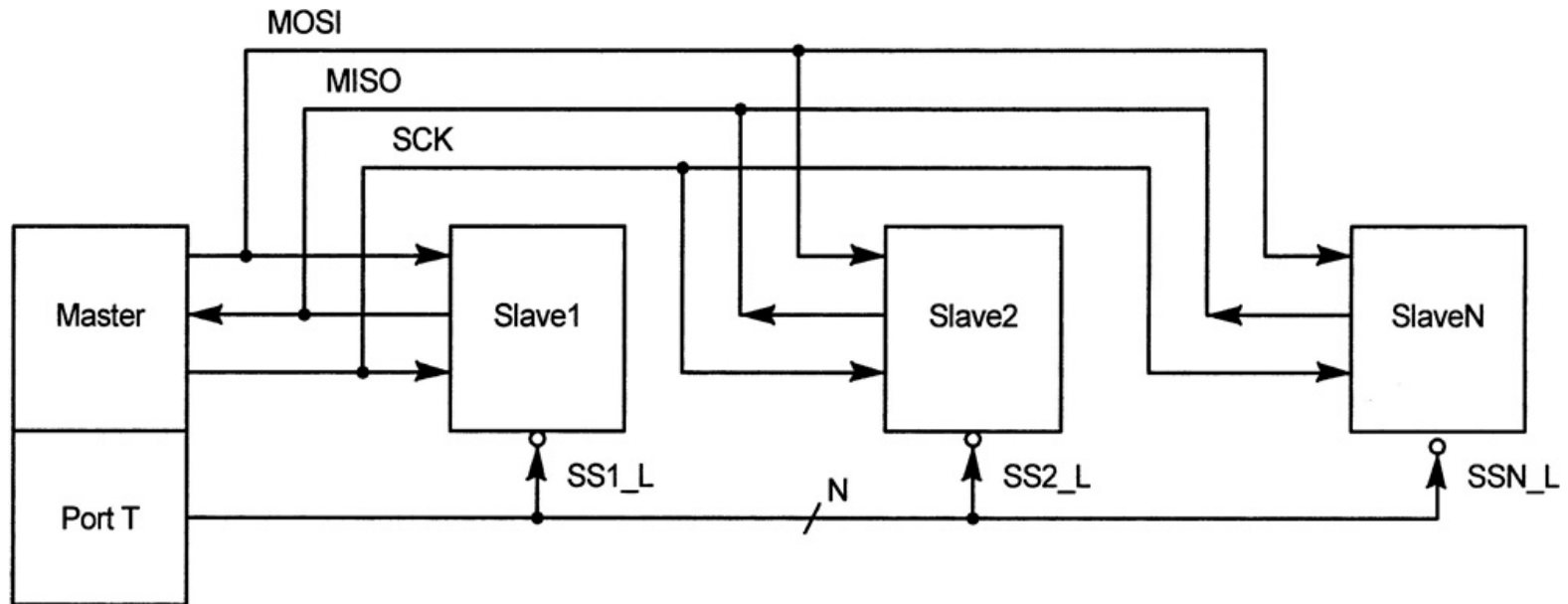


Figure 15-5 Master and multiple slaves.