



ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์  
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี  
ข้อสอบปลายภาคการศึกษาที่ 2 / 2550

วิชา CPE 311 Microprocessor Based System Design

นศ.วิศวกรรมคอมพิวเตอร์

วันอังคารที่ 4 มีนาคม 2551

เวลา 09.00-12.00

เงื่อนไข

1. อนุญาตให้นำเครื่องคิดเลข และ ไม้บรรทัดที่มีสูตรเข้าห้องสอบได้
2. ไม่อนุญาตให้นำเอกสารใดๆเข้าห้องสอบได้
3. ไม่อนุญาตให้นำข้อสอบออกจากห้องสอบ.
4. ข้อสอบมีทั้งหมด 25 หน้า ไม่รวมปก (2 คำถาม, 120 คะแนน).
5. ถ้ามีข้อสงสัยเกี่ยวกับข้อสอบให้เขียนและแก้ไขโจทย์ตามแต่พิจารณา

นักศึกษาที่ทุจริตจะถูกลงโทษขั้นสูงสุดตามระเบียบของมหาวิทยาลัย

ออกข้อสอบโดย

ผศ. สุรพันธ์ ดุมนาค

อ. ไกรกร เศรษฐไกรกุล

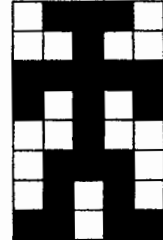
Tel. 0-2470-9083

ชื่อ \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_

1. คำถามทุกข้อมาจาก การทดสอบทางปฏิบัติ ตอบคำถามให้ถูกต้อง. (100 คะแนน)

- a. จากรูปลักษณะตามข้างล่าง , เขียนชุดข้อมูล (8 ไบท์) สำหรับรูปแบบอักษร ขนาด 5x7 ของ CGRAM (10 คะแนน)

DATA = {0x\_\_\_\_, 0x\_\_\_\_, 0x\_\_\_\_, 0x\_\_\_\_, 0x\_\_\_\_, 0x\_\_\_\_, 0x\_\_\_\_, 0x\_\_\_\_};



- b. UART (40 คะแนน)

- i. จากตารางรีจิสเตอร์ที่ให้, เติมข้อมูลให้ถูกต้อง (15 คะแนน)

```
// USART initialization
// Communication Parameters: 9 Data, 1 Stop, Even Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 115200
```

```
UCSRA=0x____;
UCSRB=0x____;
UCSRC=0x____;
UBRRH=0x____;
UBRRL=0x____;
```

- ii. มี AVR chips 2 ตัวติดต่อกันด้วยการสื่อสารแบบ RS232 ที่มีการตั้งค่าที่ไม่ตรงกันดังตารางข้างล่าง  
ให้หาค่าของข้อมูลที่ฝ่ายรับจะได้รับ (25 คะแนน)

	AVR Chip 1	AVR Chip 2
CONFIG	2400, 7, E, 1	2400, 8, N, 1
DATA1	0x64	0x____
DATA2	0x____	0x78
DATA3	0x79	0x____

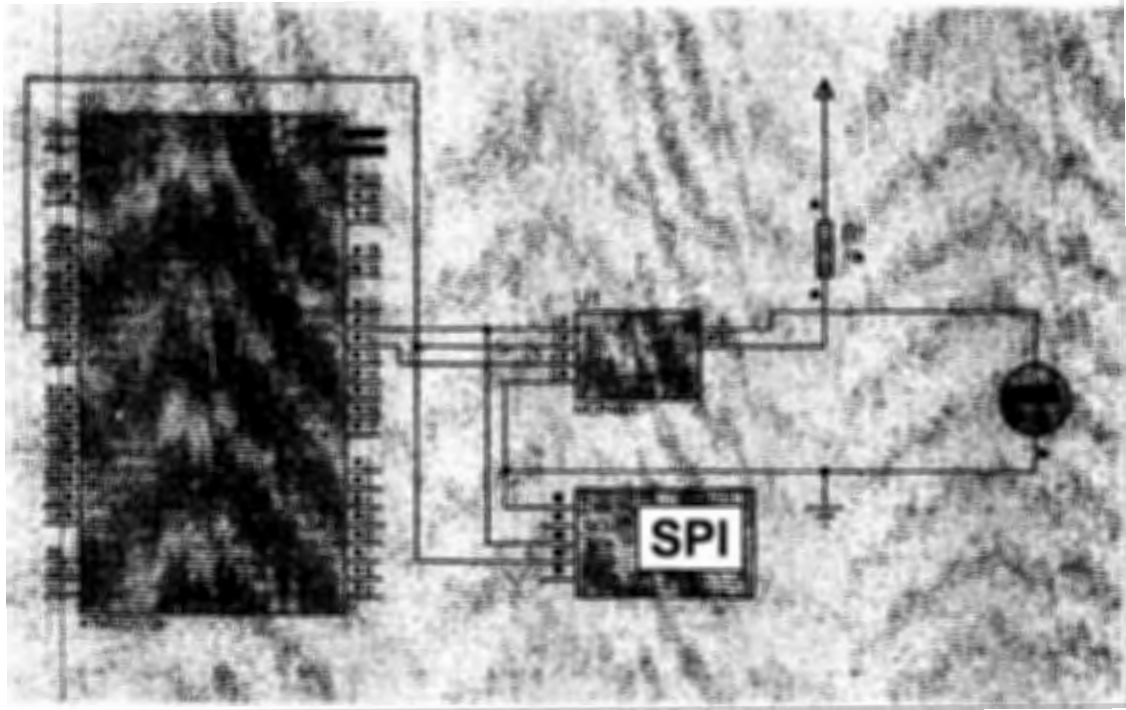
ชื่อ \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_

c. จากวงจรข้างล่าง ATMELE AVR ไมโครคอนโทรลเลอร์ เชื่อมต่อกับ MCP4921 DAC ผ่าน SPI

โวลต์เดจเอาต์พุต ( $V_{out}$ ) มีค่าเท่ากับ 6.166 Vdc ที่  $V_{ref} = 5$  Vdc

ข้อมูลของ SPI เป็นดังนี้ 0b CCCC DDDD DDDD DDDD

CCCC is 4 bit command and DDDD DDDD DDDD is 12bit data



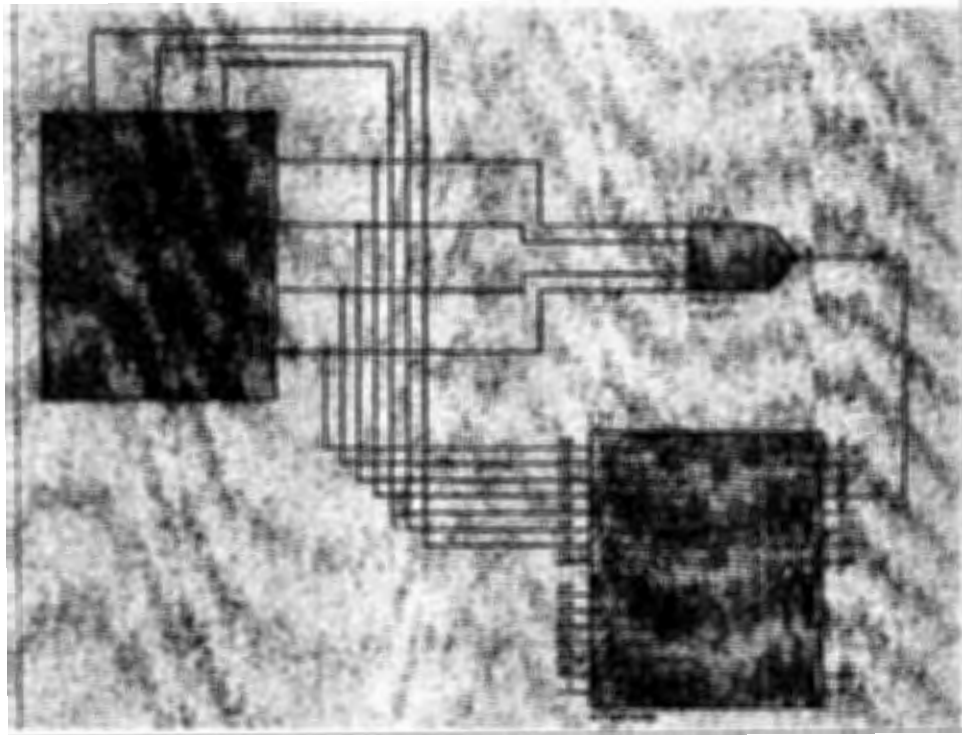
หาค่าของข้อมูลบน SPI bus ที่สร้าง แรงดันขาออกดังกล่าวข้างต้น (ปัดเศษทศนิยมเช่น 1001.2 เป็น 1002) ( 0x แสดงถึงเลขฐาน 16, 0d แสดงถึงเลขฐาน 10)

CCCC = 0b \_\_\_\_\_ or 0x \_\_\_\_\_

DDDD DDDD DDDD = 0b \_\_\_\_\_ or 0x \_\_\_\_\_ or 0d \_\_\_\_\_

ชื่อ \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_

2. จากวงจรข้างล่าง ตอบคำถามให้ถูกต้อง เขียน โปรแกรมภาษาแอสเซมบลีที่ใช้ในการกำหนดค่าเริ่มต้นเฉพาะ  
รีจิสเตอร์ที่ต้องการและ โปรแกรมย่อย (subroutines) ที่จำเป็น ( 20 คะแนน)



- a. Scan keypad ด้วยวิธี polling ( 5 คะแนน)

ชื่อ \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_

**b. Scan keypad ด้วยการใช้ interrupt ( 5 คะแนน)**

ชื่อ \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_

- c. อ่านค่าจาก keypad เพื่อที่จะใช้งาน ADC ปุ่มเลข “0” เพื่อเปิดการใช้งาน ADC0 ปุ่มเลข “1” เพื่อเปิดการใช้งาน ADC1 แสดงโปรแกรมย่อยที่จำเป็น อธิบายคำตอบให้ชัดเจน ( 10 คะแนน)

The transmit buffer can only be written when the UDRE Flag in the UCSRA Register is set. Data written to UDR when the UDRE Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

## USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

### • Bit 7 – RXC: USART Receive Complete

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e. does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

### • Bit 6 – TXC: USART Transmit Complete

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

### • Bit 5 – UDRE: USART Data Register Empty

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register Empty interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the Transmitter is ready.

### • Bit 4 – FE: Frame Error

This bit is set if the next character in the receive buffer had a Frame Error when received (i.e., when the first stop bit of the next character in the receive buffer is zero). This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

### • Bit 3 – DOR: Data OverRun

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

### • Bit 2 – PE: Parity Error

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

### • Bit 1 – U2X: Double the USART transmission speed



This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

• **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 148.

**USART Control and Status Register B – UCSRB**

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

• **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

• **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

• **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR and PE Flags.

• **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed (i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted). When disabled, the Transmitter will no longer override the TxD port.

• **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

• **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

• **Bit 0 – TXB8: Transmit Data Bit 8**



TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

## USART Control and Status Register C – UCSRC

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

The UCSRC Register shares the same I/O location as the UBRRH Register. See the "Accessing UBRRH/UCSRC Registers" on page 149 section which describes how to access this register.

- **Bit 7 – URSEL: Register Select**

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

- **Bit 6 – UMSEL: USART Mode Select**

This bit selects between Asynchronous and Synchronous mode of operation.

**Table 55. UMSEL Bit Settings**

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation



• **Bit 5:4 – UPM1:0: Parity Mode**

These bits enable and set type of Parity Generation and Check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE Flag in UCSRA will be set.

**Table 56. UPM Bits Settings**

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

• **Bit 3 – USBS: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the transmitter. The Receiver ignores this setting.

**Table 57. USBS Bit Settings**

USBS	Stop Bit(s)
0	1-bit
1	2-bit

• **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

**Table 58. UCSZ Bits Settings**

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

• **Bit 0 – UCPOL: Clock Polarity**

This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

**Table 59. UCPOL Bit Settings**

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

### USART Baud Rate Registers – UBRRH and UBRRHs

Bit	15	14	13	12	11	10	9	8																		
	<table><tr><td>URSEL</td><td>–</td><td>–</td><td>–</td><td colspan="4">UBRR[11:8]</td></tr><tr><td colspan="8">UBRR[7:0]</td><td></td></tr></table>								URSEL	–	–	–	UBRR[11:8]				UBRR[7:0]									UBRRH UBRRL
URSEL	–	–	–	UBRR[11:8]																						
UBRR[7:0]																										
	7	6	5	4	3	2	1	0																		
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W																		
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W																		
Initial Value	0	0	0	0	0	0	0	0																		
	0	0	0	0	0	0	0	0																		

The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/UCSRC Registers" on page 149 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.



## Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 60. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 146). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left( \frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1 \right) \cdot 100\%$$

**Table 60.** Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	$f_{\text{osc}} = 1.0000 \text{ MHz}$				$f_{\text{osc}} = 1.8432 \text{ MHz}$				$f_{\text{osc}} = 2.0000 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	<b>25</b>	<b>0.2%</b>	<b>51</b>	<b>0.2%</b>	<b>47</b>	<b>0.0%</b>	<b>95</b>	<b>0.0%</b>	<b>51</b>	<b>0.2%</b>	<b>103</b>	<b>0.2%</b>
4800	<b>12</b>	<b>0.2%</b>	<b>25</b>	<b>0.2%</b>	<b>23</b>	<b>0.0%</b>	<b>47</b>	<b>0.0%</b>	<b>25</b>	<b>0.2%</b>	<b>51</b>	<b>0.2%</b>
9600	<b>6</b>	<b>-7.0%</b>	<b>12</b>	<b>0.2%</b>	<b>11</b>	<b>0.0%</b>	<b>23</b>	<b>0.0%</b>	<b>12</b>	<b>0.2%</b>	<b>25</b>	<b>0.2%</b>
14.4k	<b>3</b>	<b>8.5%</b>	<b>8</b>	<b>-3.5%</b>	<b>7</b>	<b>0.0%</b>	<b>15</b>	<b>0.0%</b>	<b>8</b>	<b>-3.5%</b>	<b>16</b>	<b>2.1%</b>
19.2k	<b>2</b>	<b>8.5%</b>	<b>6</b>	<b>-7.0%</b>	<b>5</b>	<b>0.0%</b>	<b>11</b>	<b>0.0%</b>	<b>6</b>	<b>-7.0%</b>	<b>12</b>	<b>0.2%</b>
28.8k	<b>1</b>	<b>8.5%</b>	<b>3</b>	<b>8.5%</b>	<b>3</b>	<b>0.0%</b>	<b>7</b>	<b>0.0%</b>	<b>3</b>	<b>8.5%</b>	<b>8</b>	<b>-3.5%</b>
38.4k	<b>1</b>	<b>-18.6%</b>	<b>2</b>	<b>8.5%</b>	<b>2</b>	<b>0.0%</b>	<b>5</b>	<b>0.0%</b>	<b>2</b>	<b>8.5%</b>	<b>6</b>	<b>-7.0%</b>
57.6k	<b>0</b>	<b>8.5%</b>	<b>1</b>	<b>8.5%</b>	<b>1</b>	<b>0.0%</b>	<b>3</b>	<b>0.0%</b>	<b>1</b>	<b>8.5%</b>	<b>3</b>	<b>8.5%</b>
76.8k	—	—	<b>1</b>	<b>-18.6%</b>	<b>1</b>	<b>-25.0%</b>	<b>2</b>	<b>0.0%</b>	<b>1</b>	<b>-18.6%</b>	<b>2</b>	<b>8.5%</b>
115.2k	—	—	<b>0</b>	<b>8.5%</b>	<b>0</b>	<b>0.0%</b>	<b>1</b>	<b>0.0%</b>	<b>0</b>	<b>8.5%</b>	<b>1</b>	<b>8.5%</b>
230.4k	—	—	—	—	—	—	<b>0</b>	<b>0.0%</b>	—	—	—	—
250k	—	—	—	—	—	—	—	—	—	—	<b>0</b>	<b>0.0%</b>
Max <sup>(1)</sup>	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

1. UBRR = 0, Error = 0.0%

**Table 61.** Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	$f_{osc} = 3.6864 \text{ MHz}$				$f_{osc} = 4.0000 \text{ MHz}$				$f_{osc} = 7.3728 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	—	—	0	-7.8%	—	—	0	0.0%	0	-7.8%	1	-7.8%
1M	—	—	—	—	—	—	—	—	—	—	0	-7.8%
Max <sup>(1)</sup>	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

1. UBRR = 0, Error = 0.0%

**Table 62.** Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	$f_{osc} = 8.0000 \text{ MHz}$				$f_{osc} = 11.0592 \text{ MHz}$				$f_{osc} = 14.7456 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	—	—	2	-7.8%	1	-7.8%	3	-7.8%
1M	—	—	0	0.0%	—	—	—	—	0	-7.8%	1	-7.8%
Max <sup>(1)</sup>	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

**Table 63.** Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	$f_{osc} = 16.0000 \text{ MHz}$				$f_{osc} = 18.4320 \text{ MHz}$				$f_{osc} = 20.0000 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	—	—	4	-7.8%	—	—	4	0.0%
1M	0	0.0%	1	0.0%	—	—	—	—	—	—	—	—
Max <sup>(1)</sup>	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRR = 0, Error = 0.0%



## External Interrupts

The external interrupts are triggered by the INT0, and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..1 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register – MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 and INT1 requires the presence of an I/O clock, described in “Clock Systems and their Distribution” on page 23. Low level interrupts on INT0/INT1 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. This makes the MCU less sensitive to noise. The changed level is sampled twice by the Watchdog Oscillator clock. The period of the Watchdog Oscillator is 1  $\mu$ s (nominal) at 5.0V and 25°C. The frequency of the Watchdog Oscillator is voltage dependent as shown in “Electrical Characteristics” on page 239. The MCU will wake up if the input has the required level during this sampling or if it is held until the end of the start-up time. The start-up time is defined by the SUT Fuses as described in “System Clock and Clock Options” on page 23. If the level is sampled twice by the Watchdog Oscillator clock but disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The required level must be held long enough for the MCU to complete the wake up to trigger the level interrupt.

### MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

#### • Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-bit and the corresponding interrupt mask in the GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 31. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

**Table 31.** Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.



- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 32. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

**Table 32. Interrupt 0 Sense Control**

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

## General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.



## General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	–	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

### • Bit 7 – INTF1: External Interrupt Flag 1

When an event on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

### • Bit 6 – INTF0: External Interrupt Flag 0

When an event on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

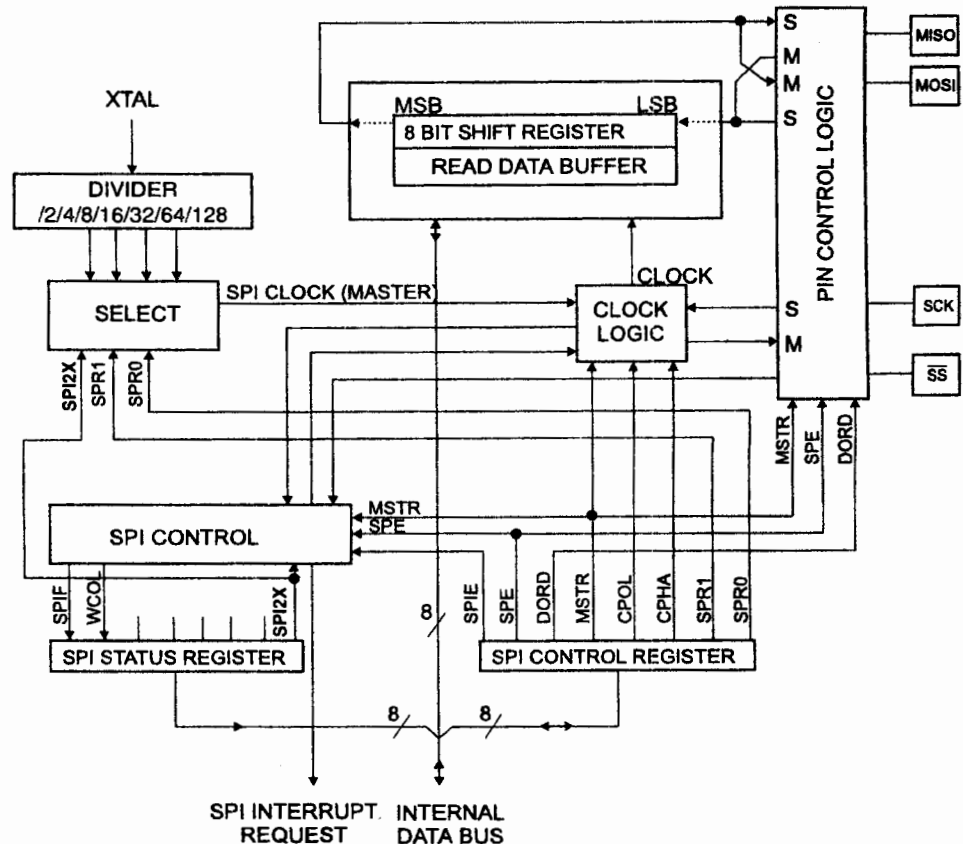


## Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega128 and peripheral devices or between several AVR devices. The ATmega128 SPI includes the following features:

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

Figure 75. SPI Block Diagram



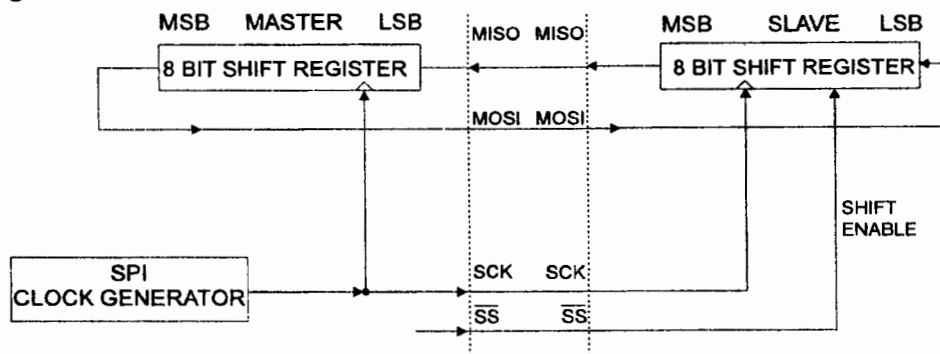
Note: Refer to Figure 1 on page 2 and Table 30 on page 71 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 76. The system consists of two Shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select  $\overline{SS}$  pin of the desired Slave. Master and Slave prepare the data to be sent in their respective Shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select,  $\overline{SS}$ , line.

When configured as a Master, the SPI interface has no automatic control of the  $\overline{SS}$  line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the 8 bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI interrupt enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select,  $\overline{SS}$  line. The last incoming byte will be kept in the buffer register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the  $\overline{SS}$  pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the  $\overline{SS}$  pin is driven low. As one byte has been completely shifted, the end of transmission flag, SPIF is set. If the SPI interrupt enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the buffer register for later use.

Figure 76. SPI Master-Slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed  $f_{osc}/4$ .

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and  $\overline{SS}$  pins is overridden according to Table 69. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 68.

Table 69. SPI Pin Overrides<sup>(1)</sup>

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input



Note: 1. See "Alternate Functions of Port B" on page 71 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. `DDR_SPI` in the examples must be replaced by the actual data direction register controlling the SPI pins. `DD_MOSI`, `DD_MISO` and `DD_SCK` must be replaced by the actual data direction bits for these pins. For example, if MOSI is placed on pin PB5, replace `DD_MOSI` with `DDB5` and `DDR_SPI` with `DDRB`.

#### Assembly Code Example<sup>(1)</sup>

```

SPI_MasterInit:
    ; Set MOSI and SCK output, all others input
    ldi r17, (1<<DD_MOSI) | (1<<DD_SCK)
    out DDR_SPI, r17
    ; Enable SPI, Master, set clock rate fck/16
    ldi r17, (1<<SPE) | (1<<MSTR) | (1<<SPR0)
    out SPCR, r17
    ret

SPI_MasterTransmit:
    ; Start transmission of data (r16)
    out SPDR, r16
Wait_Transmit:
    ; Wait for transmission complete
    sbis SPSR, SPIF
    rjmp Wait_Transmit
    ret
    
```

#### C Code Example<sup>(1)</sup>

```

void SPI_MasterInit(void)
{
    /* Set MOSI and SCK output, all others input */
    DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK);
    /* Enable SPI, Master, set clock rate fck/16 */
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
    /* Start transmission */
    SPDR = cData;
    /* Wait for transmission complete */
    while(!(SPSR & (1<<SPIF)))
    ;
}
    
```

Note: 1. The example code assumes that the part specific header file is included.

The following code examples show how to initialize the SPI as a slave and how to perform a simple reception.

#### Assembly Code Example<sup>(1)</sup>

```

SPI_SlaveInit:
    ; Set MISO output, all others input
    ldi r17, (1<<DD_MISO)
    out DDR_SPI, r17
    ; Enable SPI
    ldi r17, (1<<SPE)
    out SPCR, r17
    ret

SPI_SlaveReceive:
    ; Wait for reception complete
    sbis SPSR, SPIF
    rjmp SPI_SlaveReceive
    ; Read received data and return
    in r16, SPDR
    ret

```

#### C Code Example<sup>(1)</sup>

```

void SPI_SlaveInit(void)
{
    /* Set MISO output, all others input */
    DDR_SPI = (1<<DD_MISO);
    /* Enable SPI */
    SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
    /* Wait for reception complete */
    while(!(SPSR & (1<<SPIF)))
    ;
    /* Return data register */
    return SPDR;
}

```

Note: 1. The example code assumes that the part specific header file is included.



## **$\overline{SS}$ Pin Functionality**

### **Slave Mode**

When the SPI is configured as a slave, the Slave Select ( $\overline{SS}$ ) pin is always input. When  $\overline{SS}$  is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When  $\overline{SS}$  is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the  $\overline{SS}$  pin is driven high.

The  $\overline{SS}$  pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the  $\overline{SS}$  pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

### **Master Mode**

When the SPI is configured as a master (MSTR in SPCR is set), the user can determine the direction of the  $\overline{SS}$  pin.

If  $\overline{SS}$  is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the  $\overline{SS}$  pin of the SPI slave.

If  $\overline{SS}$  is configured as an input, it must be held high to ensure Master SPI operation. If the  $\overline{SS}$  pin is driven low by peripheral circuitry when the SPI is configured as a master with the  $\overline{SS}$  pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a slave. As a result of the SPI becoming a slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in master mode, and there exists a possibility that  $\overline{SS}$  is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI master mode.

### **SPI Control Register – SPCR**

Bit	7	6	5	4	3	2	1	0	
	<b>SPIE</b>	<b>SPE</b>	<b>DORD</b>	<b>MSTR</b>	<b>CPOL</b>	<b>CPHA</b>	<b>SPR1</b>	<b>SPR0</b>	<b>SPCR</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

#### **• Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the global interrupt enable bit in SREG is set.

#### **• Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

#### **• Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

#### **• Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If  $\overline{SS}$  is configured as an input and is driven low while MSTR is set, MSTR will

be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

• **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 77 and Figure 78 for an example. The CPOL functionality is summarized below:

**Table 70.** CPOL functionality

CPOL	Leading edge	Trailing edge
0	Rising	Falling
1	Falling	Rising

• **Bit 2 – CPHA: Clock Phase**

The settings of the clock phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 77 and Figure 78 for an example. The CPHA functionality is summarized below:

**Table 71.** CPHA functionality

CPHA	Leading edge	Trailing edge
0	Sample	Setup
1	Setup	Sample

• **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the Oscillator Clock frequency  $f_{osc}$  is shown in the following table:

**Table 72.** Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$





## SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	<b>SPIF</b>	<b>WCOL</b>	–	–	–	–	–	<b>SPI2X</b>	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### • Bit 7 – SPIF: SPI Interrupt Flag

When a serial transfer is complete, the SPIF flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If  $\overline{SS}$  is an input and is driven low when the SPI is in Master mode, this will also set the SPIF flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

### • Bit 6 – WCOL: Write COLLision flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

### • Bit 5..1 – Res: Reserved Bits

These bits are reserved bits in the ATmega128 and will always read as zero.

### • Bit 0 – SPI2X: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 72). This means that the minimum SCK period will be 2 CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at  $f_{osc}/4$  or lower.

The SPI interface on the ATmega128 is also used for program memory and EEPROM downloading or uploading. See page 303 for SPI Serial Programming and verification.

## SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	<b>MSB</b>							<b>LSB</b>	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a Read/Write Register used for data transfer between the register file and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

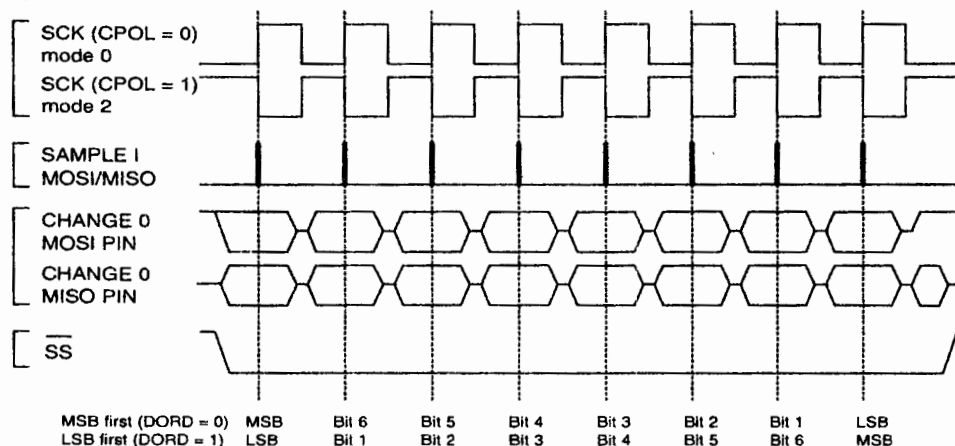
## Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 77 and Figure 78. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 70 and Table 71, as done below:

**Table 73.** CPOL and CPHA Functionality

	Leading edge	Trailing edge	SPI mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

**Figure 77.** SPI Transfer Format with CPHA = 0



**Figure 78.** SPI Transfer Format with CPHA = 1

