AVR Serial Port Programming in Assembly and C

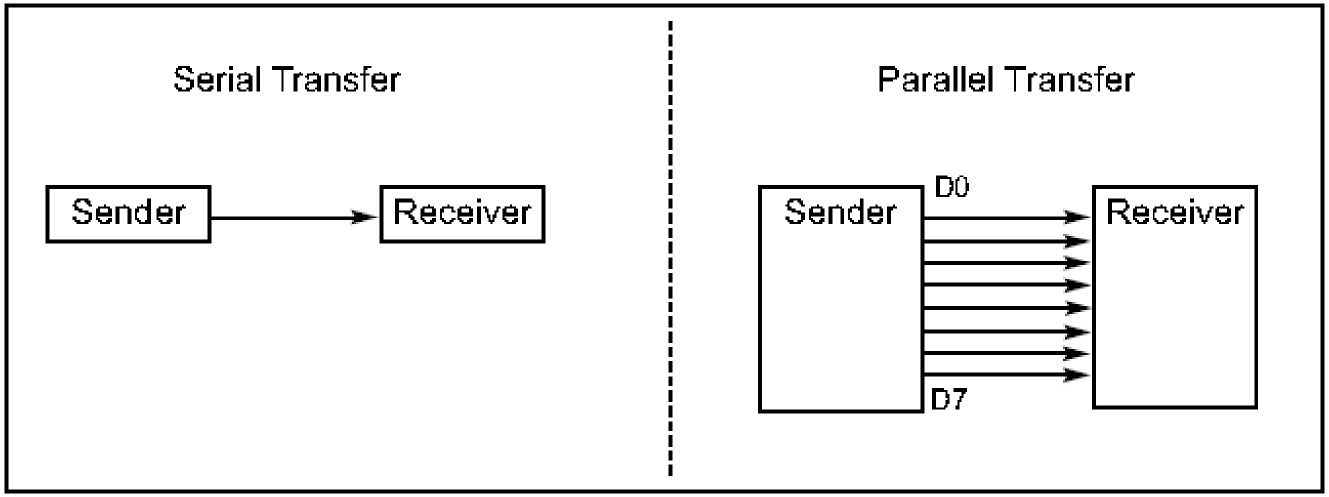
The AVR microcontroller and embedded systems using assembly and c

Topics

- Parallel vs. Serial
- Simplex vs. duplex vs. full duplex
- Synchronous vs. asynchronous
- Data framing and bps
- RS232 Standard
- RS232 connections to AVR and MAX232
- AVR Serial Port Programming
  - UBRR and baud rate
  - Baud rate error calculation
  - UDR register
  - UCR registers
  - Writing a program in assembly and C

Parallel vs. Serial

- Parallel
  - Transfer a byte of data at a time -> faster, easier
- Serial
  - Transfers a bit after another -> cheaper, ideal for long distance through phone line (modem is needed)



Direction

- Simplex: data can moves only in one directions
- Half Duplex: data can moves in two direction but not at the same time
- Full Duplex: data can moves in two direction at the same time

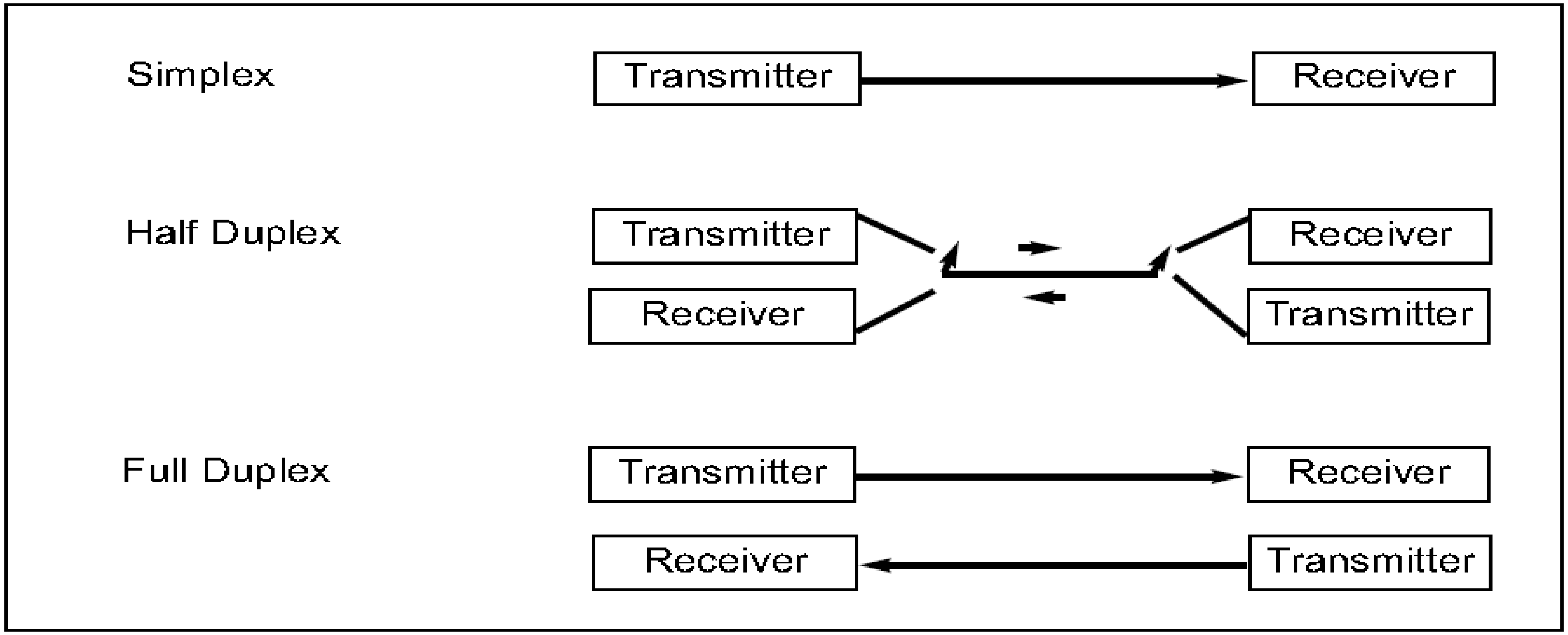


Figure 11-2. Simplex, Half-, and Full-Duplex Transfers

Synchronous vs. asynchronous

- Synchronous
  - Clock pulse should be transmitted during data transmission.
  - Only one side generates clock at the same time.
- Asynchronous
  - Clock pulse is not transmitted.
  - The two sides should generate clock pulse.
  - There should be a way to synchronize the two sides.

Data framing and bps

- To synchronize the two sides, framing is used:
  - Each frame starts with a space (0) which is called Start bit
  - A character of 7-9 bits is transmitted after start bit
  - [a bit of parity can be transmitted after the character] (optional)
  - Each frame is ended by one or two mark (1) which is called stop bit(s).
- Numbers of bit transmitted in a second is called bps

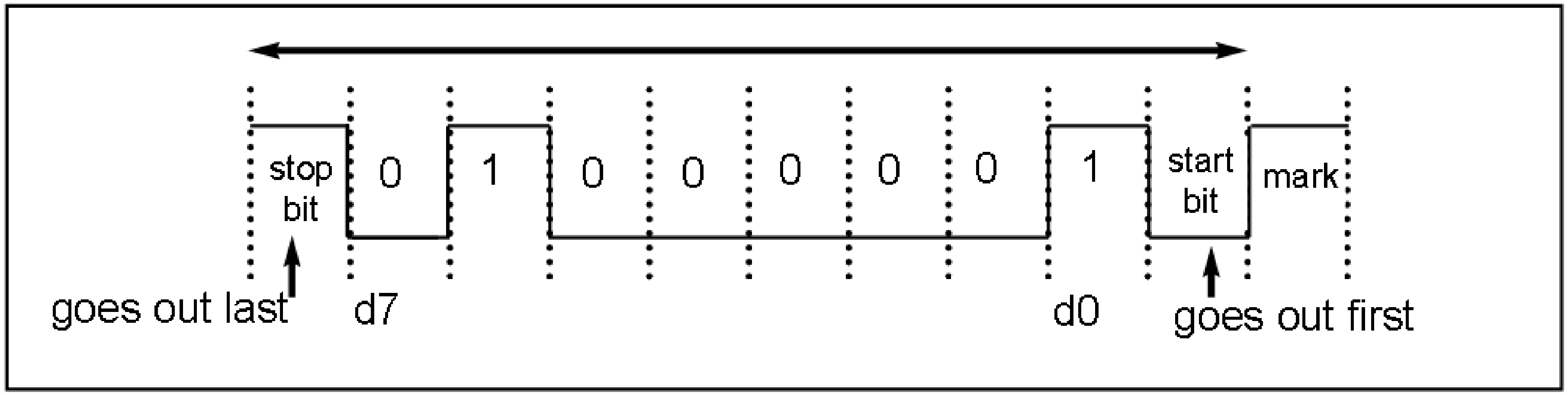


Figure 11-3. Framing ASCII 'A' (41H)

RS232 Standard

- RS232 was set by the Electronics Industries Association (EIA) in 1960.
- Because the standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible.
- A 1 is represented by −3 to −25 V
- A 0 is represented by +3 to +25 V

RS232 Pins

- DB 25 used to be the standard connector of RS232
- Now DB9 is used instead of DB25

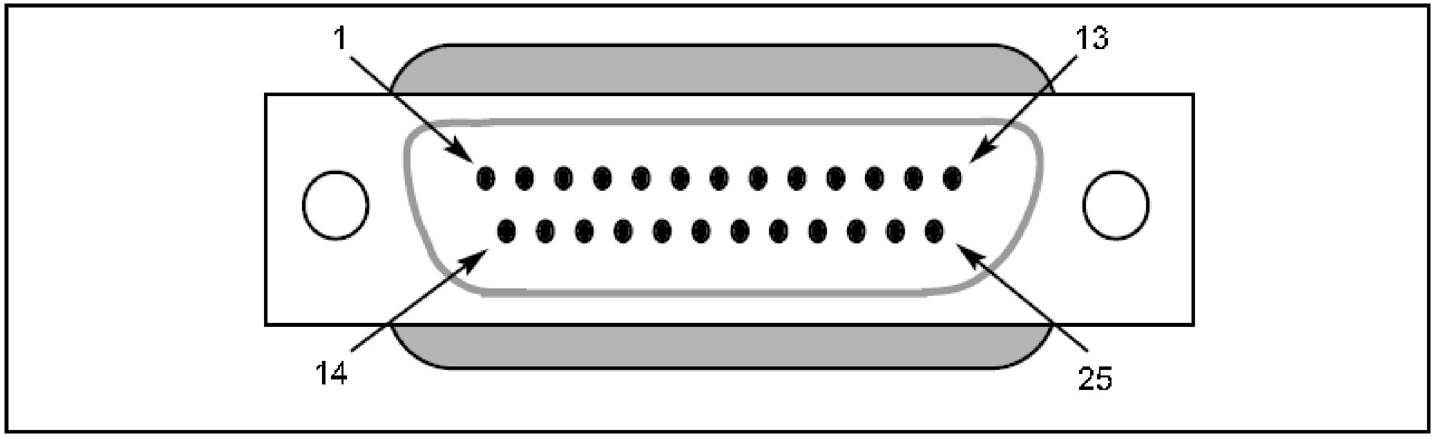


Figure 11-4. RS232 Connector DB-25

RS232 Pins

- DB 25 used to be the standard connector of RS232
- Now DB9 is used instead of DB25

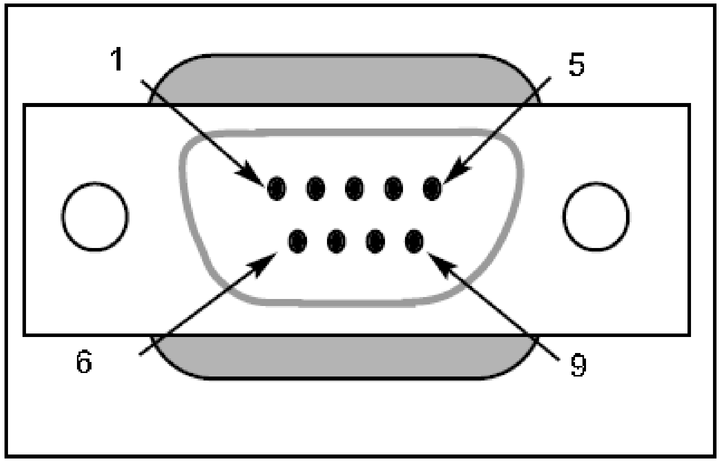


Figure 11-5. DB-9 9-Pin Connector

Pin	Description
1	Data carrier detect (DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (DSR)
7	Request to send (RTS)
8	Clear to send (CTS)
9	Ring indicator (RI)

Null modem connection

- DTE (Data Terminal Equipment) refers to terminals and computers that send and receive data.
- To connect two DTE, we connect TXD of one device to RXD of the other and vise versa. It is called Null Modem Connection

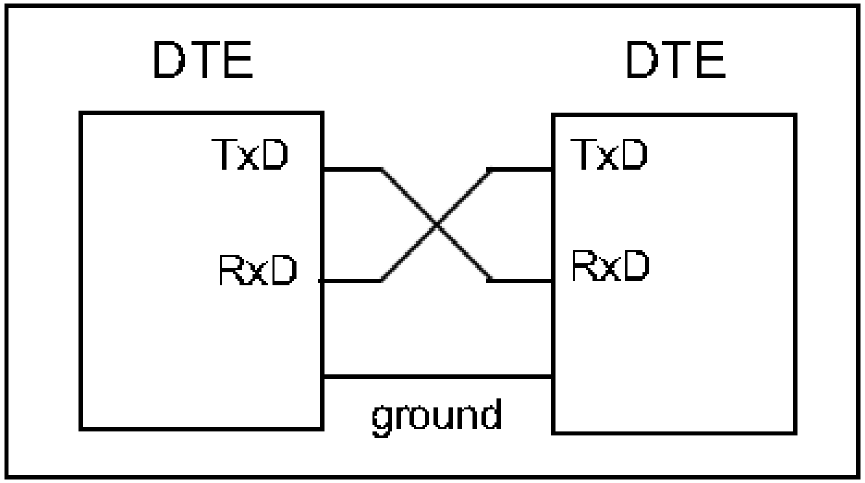


Figure 11-6. Null Modem Connection

AVR Connection

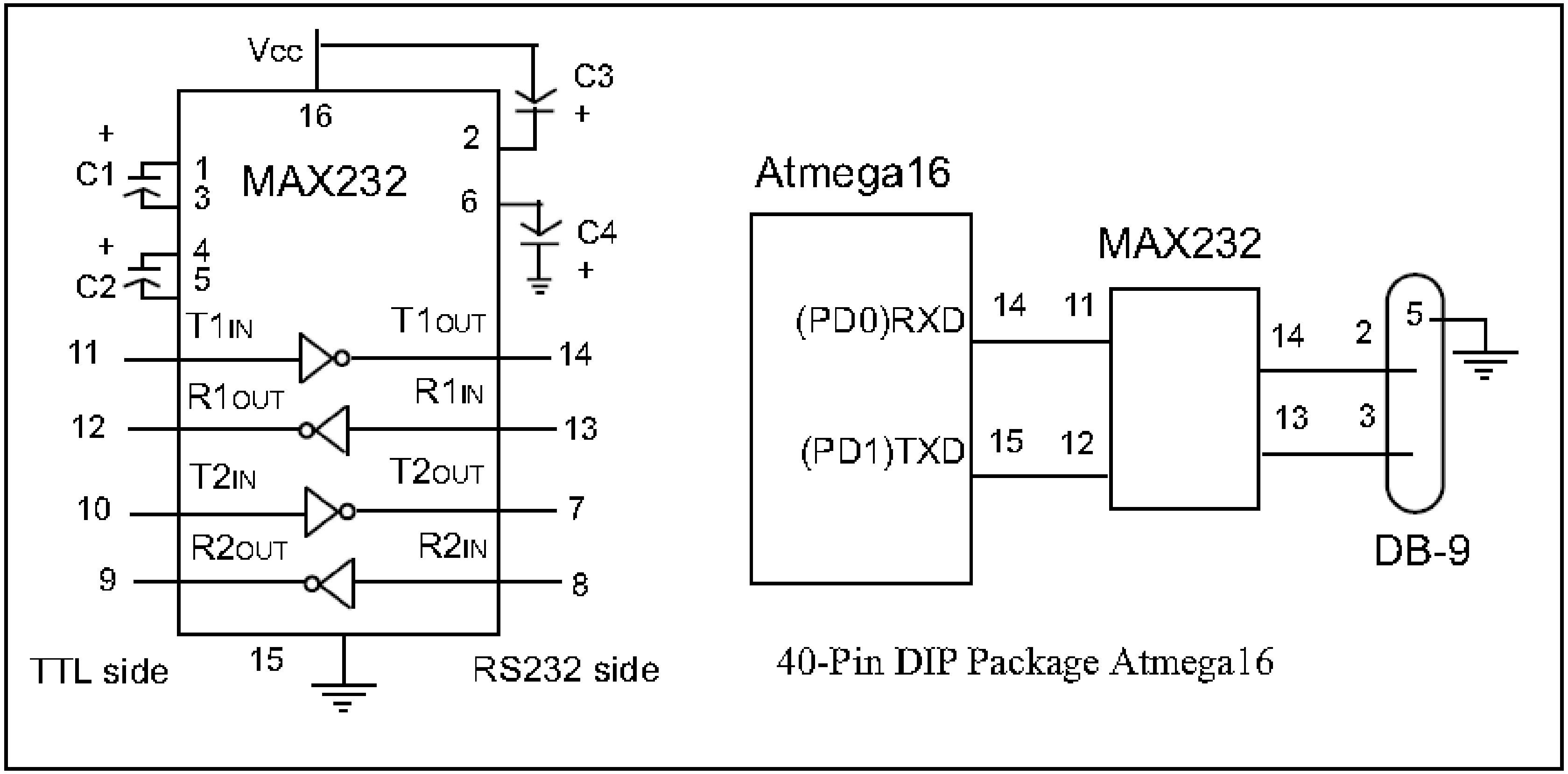


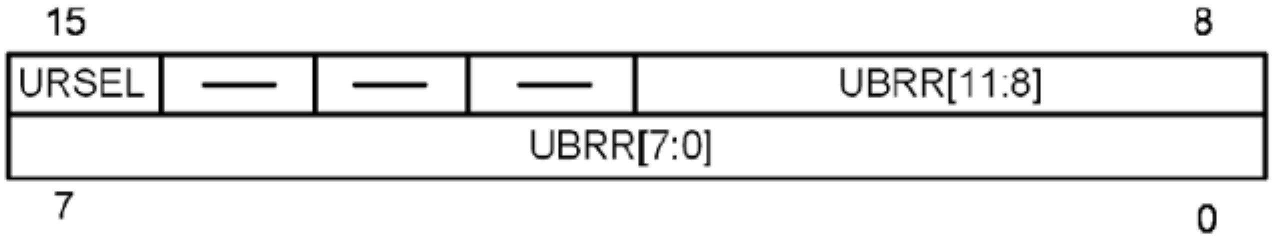
Figure 11-7. (a) Inside MAX232 and (b) its Connection to the Atmega16 (Null Modem)

USART in AVR

- AVR supports: normal asynchronous, double speed asynchronous, master synchronous and Slave synchronous mode features.
- We concentrate on Asynchronous mode to connect the AVR to a PC
- In AVR 5 Registers are associated with USART
  - UBRR (USART Baud Rate Reegister)
  - UDR (USART Data Register)
  - UCSRA (USART Control and Status Register A)
  - UCSRB (USART Control and Status Register B)
  - UCSRC (USART Control and Status Register C)

UBRR register and baud rate

- The two sides have to agree on a baud rate
- Table 11-3 shows standard baud rates of PC
- To set the baud rate in AVR you should set the value of UBRR



Desired Baud Rate = Fosc/(16(X + 1))

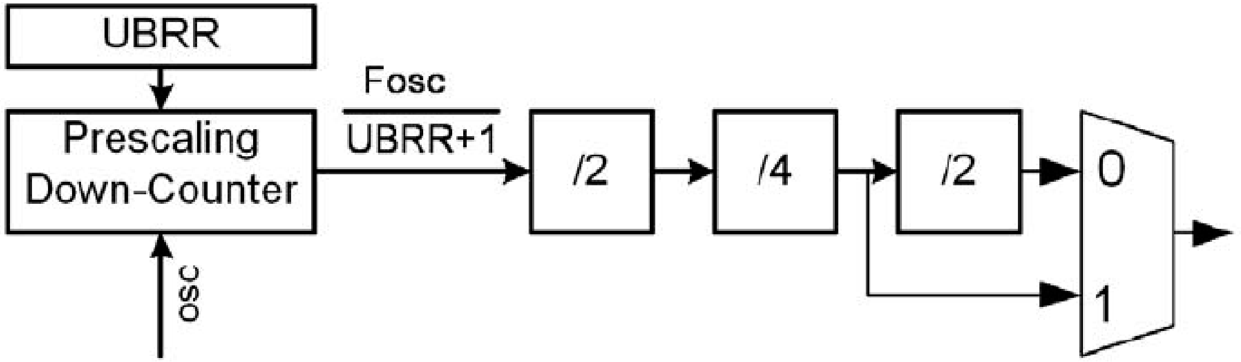


Table 11-3: Some PC Baud Rates in HyperTerminal

1,200
2,400
4,800
9,600
19,200
38,400
57,600
115,200

Baud Rate Error Calculation

- When we calculate the value of UBRR, the result may not be integer value but we can load only integer values in UBRR register -> baud rate error

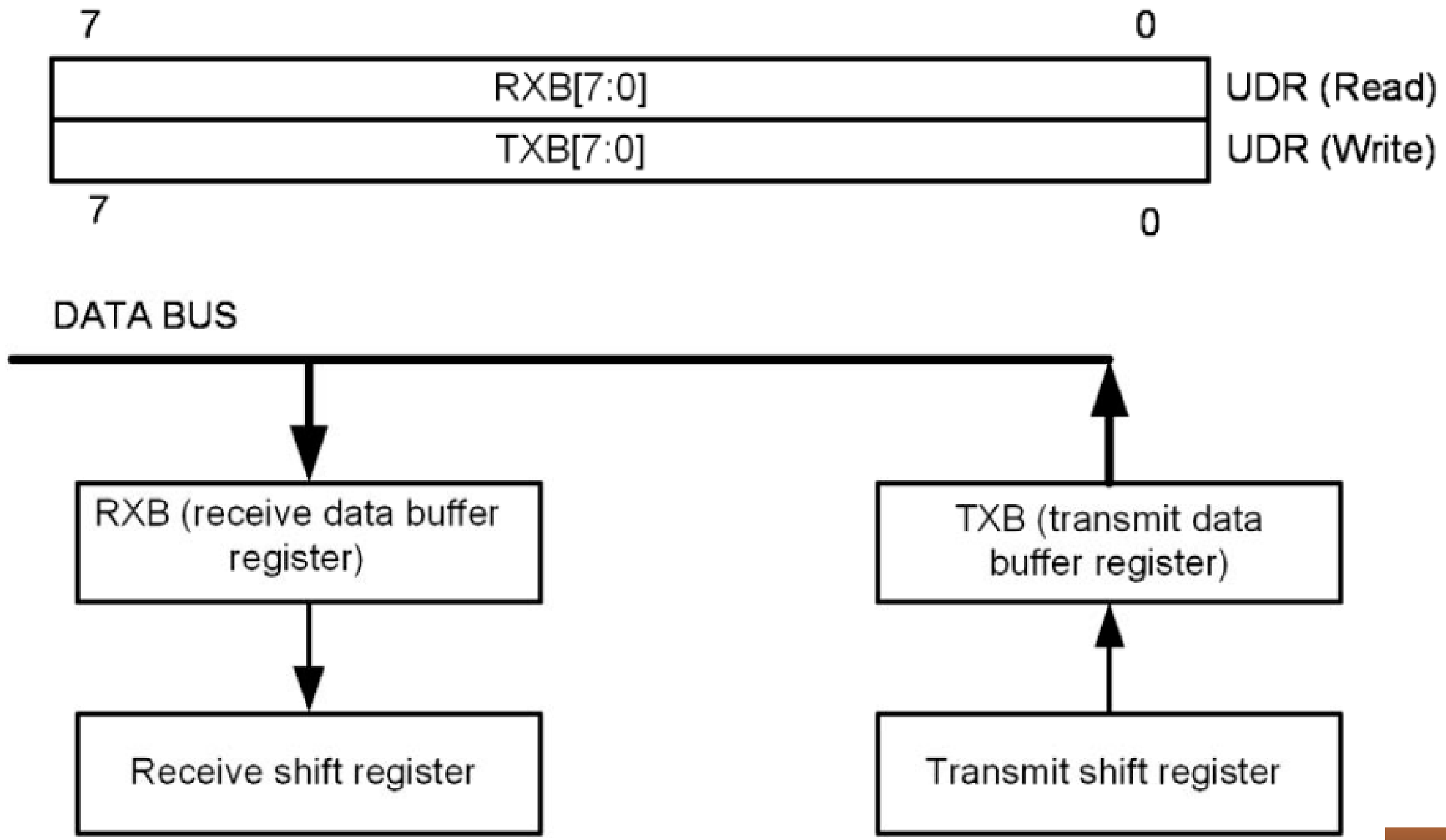
Error = (calculated value – int part)/ int part

Error = (calculated baud rate – desired baud rate) / desired baud rate

- To omit baud rate error some specific Crystals can be used:
  - 11.0592 MHz
  - 7.3728 MHz

UDR

- UDR is a bridge between you and the serial shift register
- If you read UDR, data is read from received shift register
- If you write to UDR, data is written into transmit shift register



UCSRA

RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

RXC- Bit7: USART Receive Complete
This flag bit is cleared when the receive buffer is empty and set when there are unread data in the receive buffer. It can be used to generate a Receive Complete interrupt

TXC- Bit6: 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 available in the transmit data buffer register (TXB). It can be cleared by writing a one to its bit location. Also it is automatically cleared when a transmit complete interrupt is executed. It can be used to generate a Transmit Complete interrupt.

UDRE-Bit5: USART Data Register Empty
This flag is set when the transmit data buffer is empty and it is ready to receive new data. If this bit is cleared you should not write to UDR because it override your last data.The UDRE Flag can generate a Data Register empty Interrupt

FE-Bit4: Frame Error
This bit is set if the next character in the receive buffer had a Frame Error when received. A Frame Error is detected when the first stop bit of the next character in the receive buffer is zero.

UCSRA

RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

DOR-Bit3: Data OverRun
This bit is set if a Data OverRun is detected. A Data OverRun occurs when the receive data buffer and receive shift register are full, and a new start bit is detected.

PE-Bit2: Parity Error
This bit is set if parity checking was enabled (UPM1 = 1) and the next character in the receive buffer had a Parity Error when received.

U2X-Bit1: Double the USART Transmission Speed
Setting this bit will doubl the transfer rate for asynchronous communication.

MPCM-Bit0: Multi-processor Communication Mode
This bit enables the Multi-processor Communication mode. MPCM is not discussed in this book.

Note that FE, DOR and PE are valid until the receive buffer (UDR) is read. Always set these bits to zero when writing to UCSRA.

UCSRB

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

RXCIE-Bit7: Receive Complete Interrupt Enable:
Setting this bit will enables interrupt on the RXC Flag in UCSRA.

TXCIE-Bit6: Transmit Complete Interrupt Enable:
Setting this bit will enables interrupt on the TXC Flag in UCSRA.

UDRIE-Bit5: USART Data Register Empty Interrupt Enable:
Setting this bit will enables interrupt on the UDRE Flag in UCSRA.

RXEN-Bit4: Receive Enable:
Setting this bit will enables the USART Receiver.

TXEN-Bit3: Transmit Enable:
Setting this bit will enables the USART transmitter.

UCSRB

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

UCSZ2-Bit2:Character Size
This bit combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame.

RXB8: Receive data bit 8
It is the ninth data bit of the received character when using serial frames with nine data bits. This bit is not covered in this book

TXB8: Transmit data bit 8
It is the ninth data bit of the transmitted character when using serial frames with nine data bits. This bit is not covered in this book

UCSRC

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

URSEL-Bit7: Register Select:
This bit selects between accessing the UCSRC or the UBRRH Register and will be des-cussed more in this section

UMSEL-Bit6: USART Mode Select:
This bit selects between Asynchronous and Synchronous mode of operation.

0 = Asynchronous Operation
1 = Synchronous Operation

UPM1:0-Bit5:4: Parity Mode:
These bits disable or enable and set type of parity generation and check.

00 = Disabled
01 = Reserved
10 = Even Parity
11 = Odd Parity

UCSRC

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

USBS-Bit3: Stop Bit Select:

This bit selects the number of Stop Bits to be transmitted.

0 = 1 bit

1 = 2 bit

USCZ1:0-Bit2:1: Chacter Size:

Thease 0 bits combined with the UCSZ2 bit in UCSRB sets the Character Size in a frame and will be discussed more in this section.

USPOL-Bit2: Clock Polarity

This bit is used for Synchronous mode only and will not be covered in this section.

UCSRC

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

USBS-Bit3: Stop Bit Select:

This bit selects the number of Stop Bits to be transmitted.

Table 11-5: values of UCSZ2:0 for different character size

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

Note: Other values are reserved

UCSRC

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

USBS-Bit3: Stop Bit Select:

This bit selects the number of Stop Bits to be transmitted.

0 = 1 bit	
1 = 2 bit	

USCZ1:0-Bit2:1: Chacter Size:

Thease 0 bits combined with the UCSZ2 bit in UCSRB sets the Character Size in a frame and will be discussed more in this section.

USPOL-Bit2: Clock Polarity

This bit is used for Synchronous mode only and will not be covered in this section.

Serial Port Programming

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

1. The UCSRB register is loaded with the value 10H, enabaling USART receiver. The receiver will override normal port operation for the RxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 8 MHz) to set the baud rate for serial data transfer.
5. The RXC flag bit of the UCSRA register is monitored for a HIGH to see if an entire character has been received yet.
6. When RXC is raised, the UDR register has the byte. Its contents are moved into a safe place.
7. To receive the next character, go to Step 5.

Serial Port Programming

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

1. The UCSRB register is loaded with the value 10H, enabaling USART receiver. The receiver will override normal port operation for the RxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 8 MHz) to set the baud rate for serial data transfer.
5. The RXC flag bit of the UCSRA register is monitored for a HIGH to see if an entire character has been received yet.
6. When RXC is raised, the UDR register has the byte. Its contents are moved into a safe place.
7. To receive the next character, go to Step 5.

Serial Port Programming

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

1. The UCSRB register is loaded with the value 08H, enabaling USART transmitter. The Transmitter will override normal port operation for the TxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 4 MHz) to set the baud rate for serial data transfer.
4. The character byte to be transmitted serially is written into the UDR register.
6. Monitor the UDRE bit of the UCSRA register to make sure UDR is ready for next byte.
7. To transfer the next character, go to Step 5.

Serial Port Programming

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

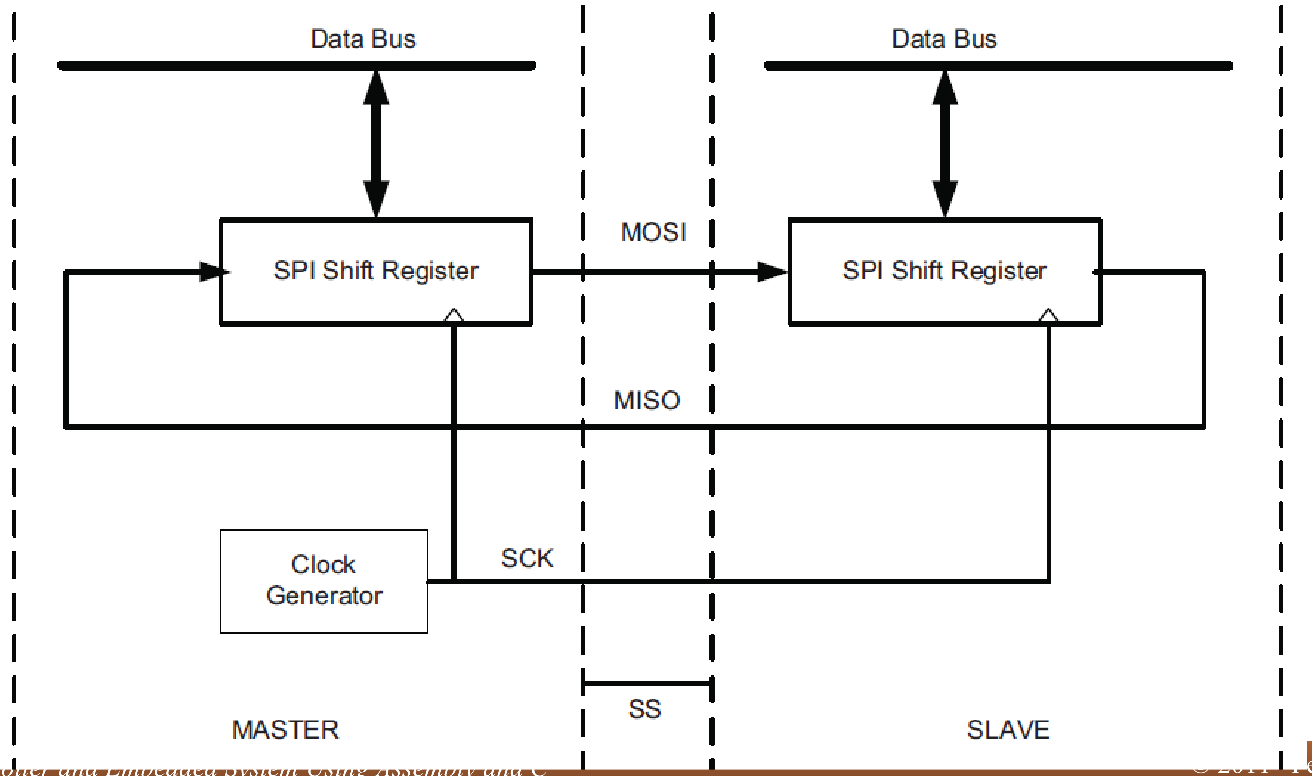
1. The UCSRB register is loaded with the value 08H, enabaling USART transmitter. The Transmitter will override normal port operation for the TxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 4 MHz) to set the baud rate for serial data transfer.
4. The character byte to be transmitted serially is written into the UDR register.
6. Monitor the UDRE bit of the UCSRA register to make sure UDR is ready for next byte.
7. To transfer the next character, go to Step 5.

SPI
Serial Peripheral Interface

The AVR microcontroller and embedded systems
using assembly and c

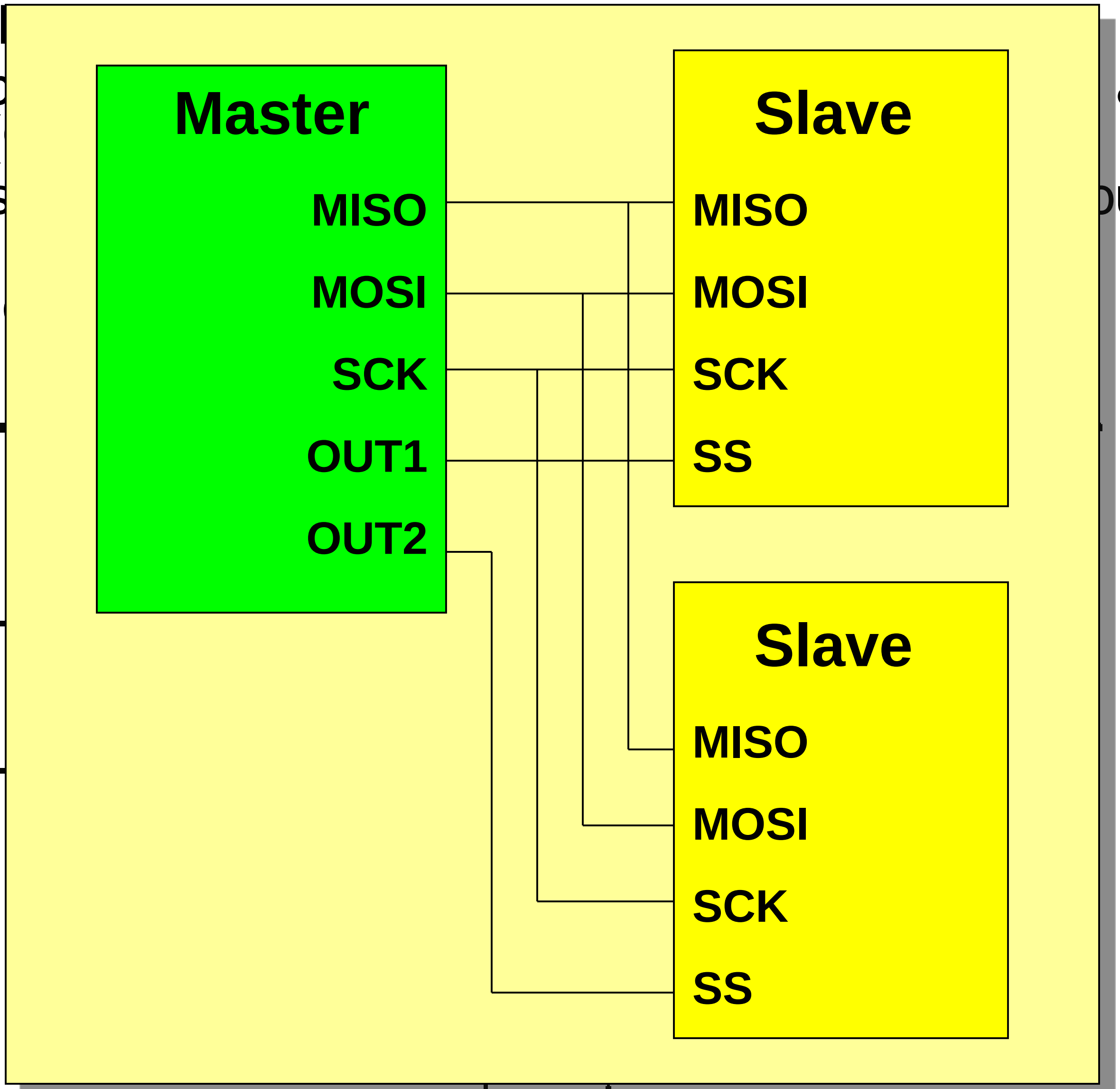
SPI …

- Started by Motorola Corp. (now Freescale)
- Uses 2 pins for data (SDI and SDO) one for clock (SCLK) and chip enable (CE)
- The pins are alternatively named MISO (Master In Slave out) , MOSI (Master Out Slave In) , SS (Slave Select) and SCK
- Read and Write happens at the same time

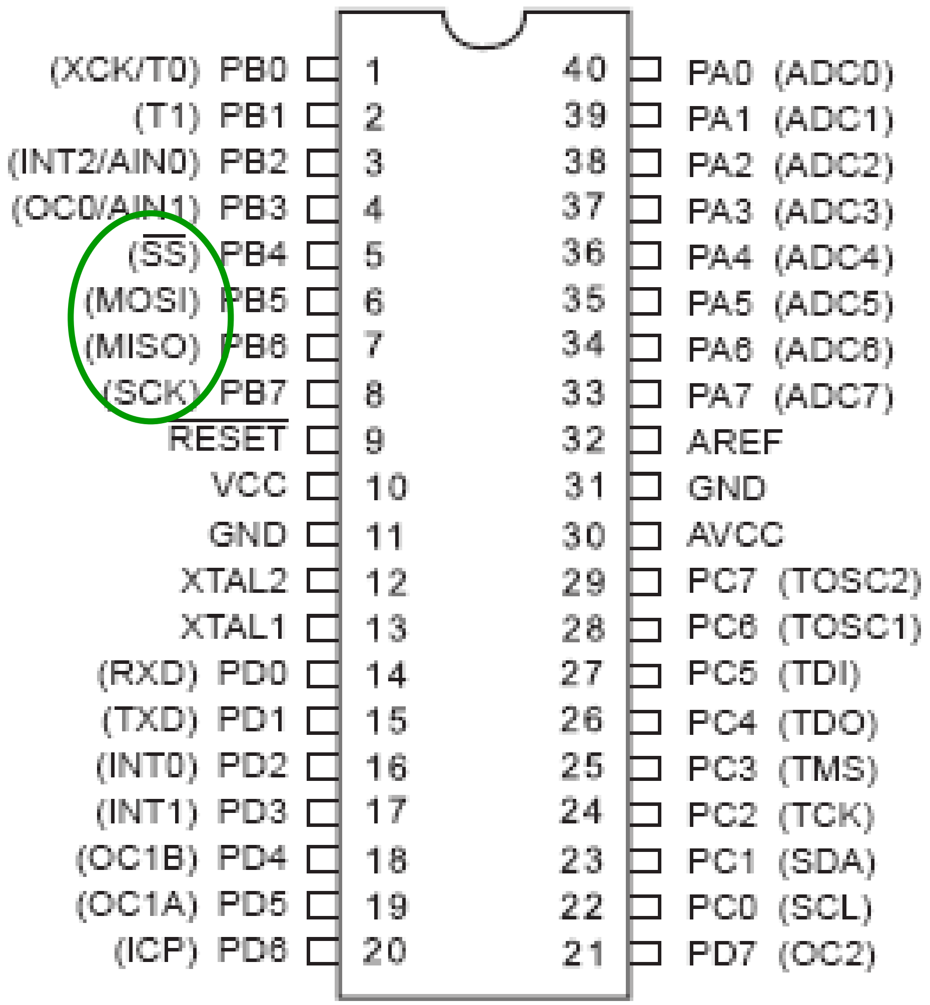


SPI …

- Started
- Uses 2 p and chip enable (
- The pins ut) , MOSI (Master
- Read an



SPI pins in Atmega32

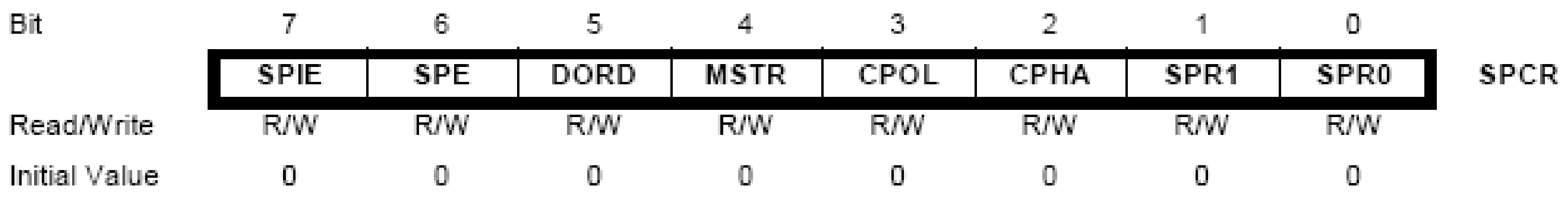


SPSR (SPI Status Register)

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

- SPIF (SPI interrupt Flag)
  - Transmitted/Received
  - Switched to Slave mode
- WCOL (Write COLision Flag)
- SPI2X (Double SPI Speed bit)
  - 1: Double

SPCR (SPI Control Register)



Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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	

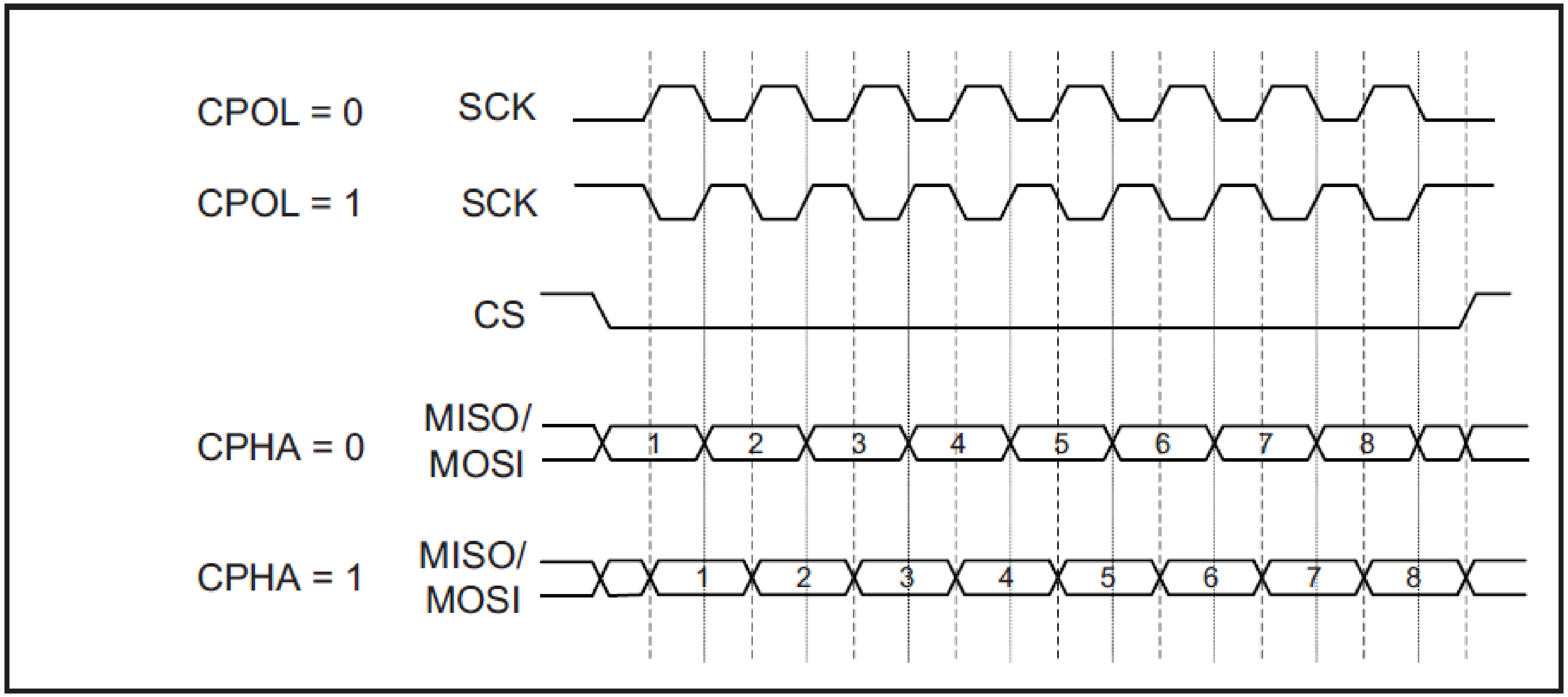
- SPIE (SPI Interrupt Enable)
- SPE (SPI Enable)
- DORD (Data Order): 1:Send LSB 1st, 0:Send MSB 1st
- MSTR (Master/Slave Select): 1:Master, 0: Slave
- CPOL (Clock Polarity): 1: idle SCK = 1, 0: SCK = 0
- CPHA (Clock Phase):
- SPR1 (SPI Clock Rate Select 1)
- SPR0 (SPI Clock Rate Select 0)

SPCR (SPI Control Register)

Table 17-1: SPI Clock Polarity and Phase

CPOL	CPHA	Data Read and Change Time	SPI Mode
0	0	Read on rising edge, changed on a falling edge	0
0	1	Read on falling edge, changed on a rising edge	1
1	0	Read on falling edge, changed on a rising edge	2
1	1	Read on rising edge, changed on a falling edge	3

SPCR



SPCR (SPI Control Register)

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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	

- SPIE (SPI Interrupt Enable)
- SPE (SPI Enable)
- DORD (Data Order): 1:Send LSB 1st, 0:Send MSB 1st
- MSTR (Master/Slave Select): 1:Master, 0: Slave
- CPOL (Clock Polarity): 1: idle SCK = 1, 0: SCK = 0
- CPHA (Clock Phase):
- SPR1 (SPI Clock Rate Select 1)
- SPR0 (SPI Clock Rate Select 0)

SPCR (SPI Control Register)

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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	

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	$f_{osc}/64$

- SPIE (S
- SPE (SP
- DORD (
- MSTR (
- CPOL ((
- CPHA (Clock Phase):
- SPR1 (SPI Clock Rate Select 1)
- SPR0 (SPI Clock Rate Select 0)

SS Pin

- Master Mode
  - You can set the direction to output and SPI will not control the pin
  - If you set the direction to input, It should be externally pulled up
    - if you make it externally low, the SPI module stops working in master mode and switches to slave mode by clearing the MSTR bit in SPCR, and then sets the SPIF bit in SPSR.
- Slave Mode
  - SS pin is always input and you can not control it by software.
  - You should hold it externally low to activate the SPI.
  - When SS is driven high, SPI is disabled and all pins of SPI are input. Also the SPI module will immediately clear any partially received data in the shift register BUT IT WILL NOT BE DISABLED

Master Operating Mode

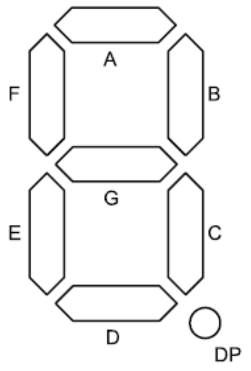
- Set the MSTR bit to one
- Set SCK frequency by setting the values of SPI2X, SPR1, and SPR2
- Set the SPI mode. If not set, it is 0.
- Enable SPI by setting the SPIE bit to one
- Write a byte to the SPI Data Register (SPDR)
- Poll the SPIF flag. Data transfer is finished when it changes to one.
- read the received byte from SPDR before the next byte arrives.
  - Note: After the transmission, the byte in the Master shift register is moved to the Slave Shift register and the Byte in the Slave shift register is moved to the Master shift register. It means that send and received happens at the same time. If you only want to read a byte, you should transmit a dummy byte like 0xff and then read the received data!

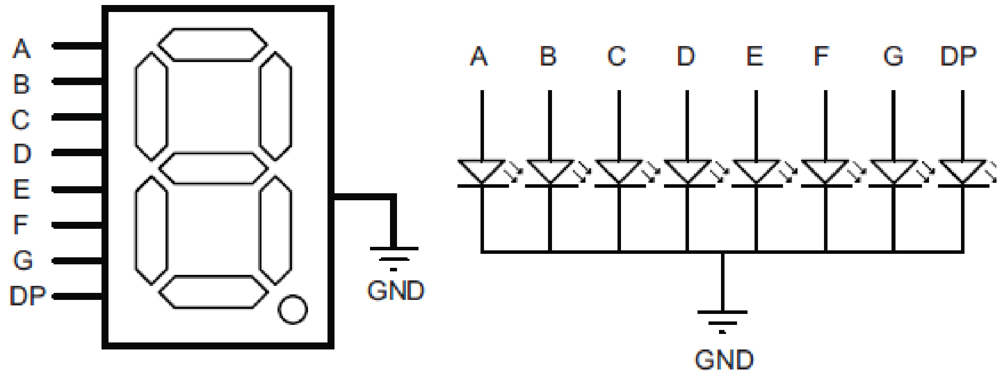
Slave Operating Mode

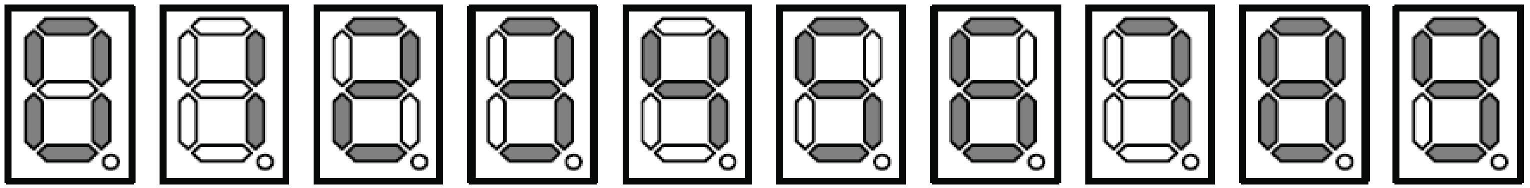
- Set the SPI mode. If not set, it is 0.
- Enable SPI by setting the SPIE bit to one
- Write a byte to the SPI Data Register (SPDR)
- Poll the SPIF flag. Data transfer is finished when it changes to one.
- read the received byte from SPDR.

What is 7-Segment

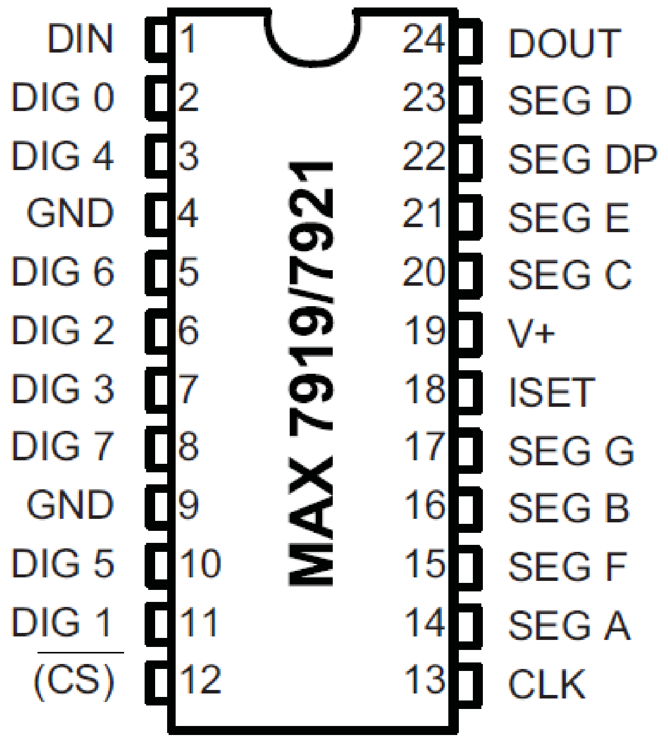
- 7-Segments are made of 7 LEDs to show different numbers plus another LED to display the decimal point.
- There are two types of 7-segments, common anode and common cathode. (below figure)





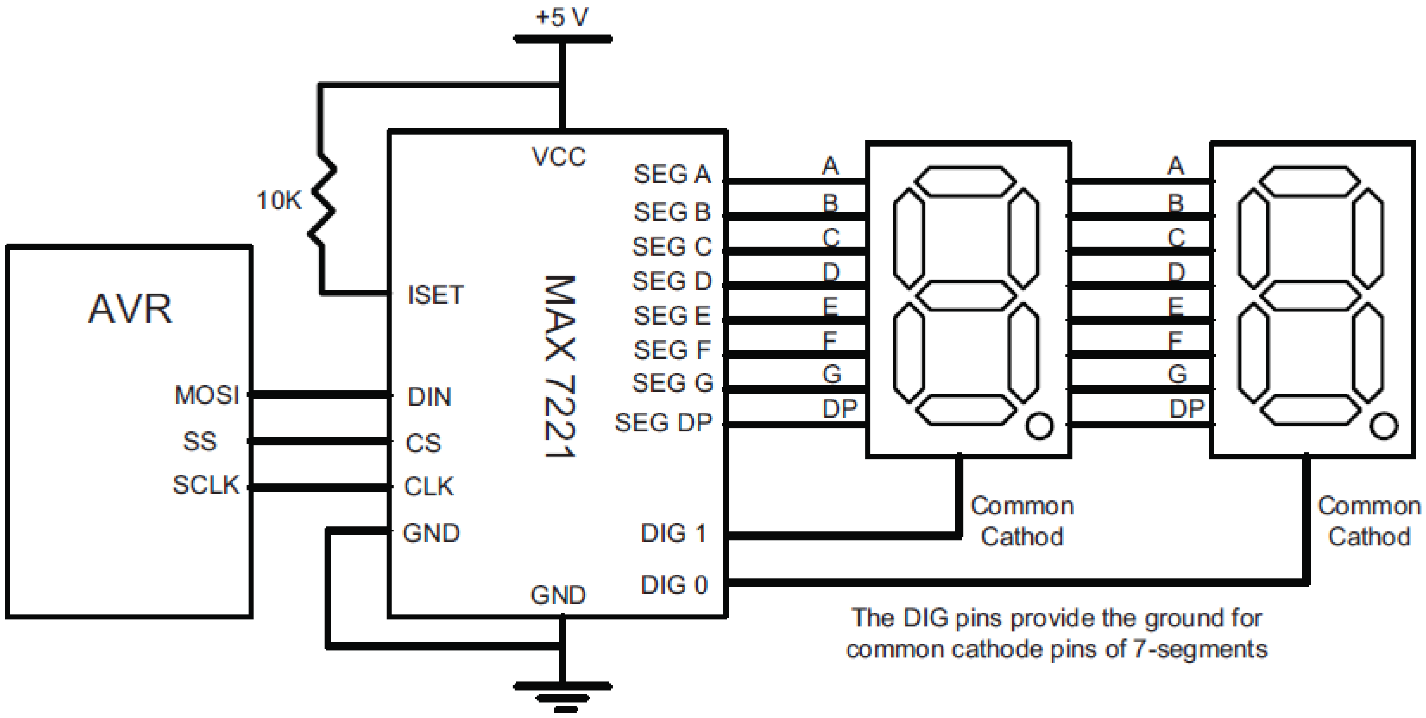


MAX 7921 (SPI 7-Seg Driver)



MAX 7921 Connections to AVR

- Up to 8 7-Segs can be connected to MAX7921
- The ISET resistor control the intensity of light
- The VCC and GND must be able to handle up to 300 mA


+5 V
10K
AVR
MOSI
SS
SCLK
VCC
ISET
DIN
CS
CLK
GND
GND
MAX 7221
SEG A
SEG B
SEG C
SEG D
SEG E
SEG F
SEG G
SEG DP
DIG 1
DIG 0
A
B
C
D
E
F
G
DP
Common Cathod
Common Cathod
The DIG pins provide the ground for common cathode pins of 7-segments