EIE3373: Serial Port Programming (Chapter 12)

Dr. Lawrence Cheung

Semester 1, 2021/22

Topics

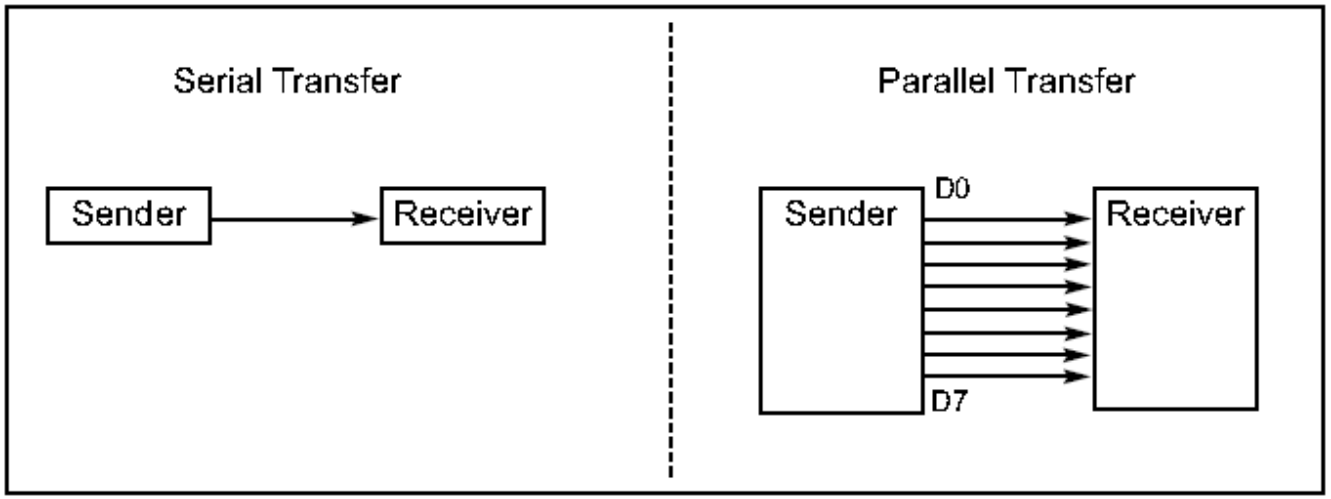
- Parallel vs. serial
- Simplex vs. duplex vs. full duplex
- Synchronous vs. asynchronous
- Data framing
- Data transfer rate
- RS232 standard
- RS232 connections to AVR and MAX232

Topics

- AVR serial port programming
  - UBRR and baud rate
  - Baud rate error calculation
  - UDR register
  - UCSR registers
  - Double the baud rate
  - C programming
  - Using interrupts

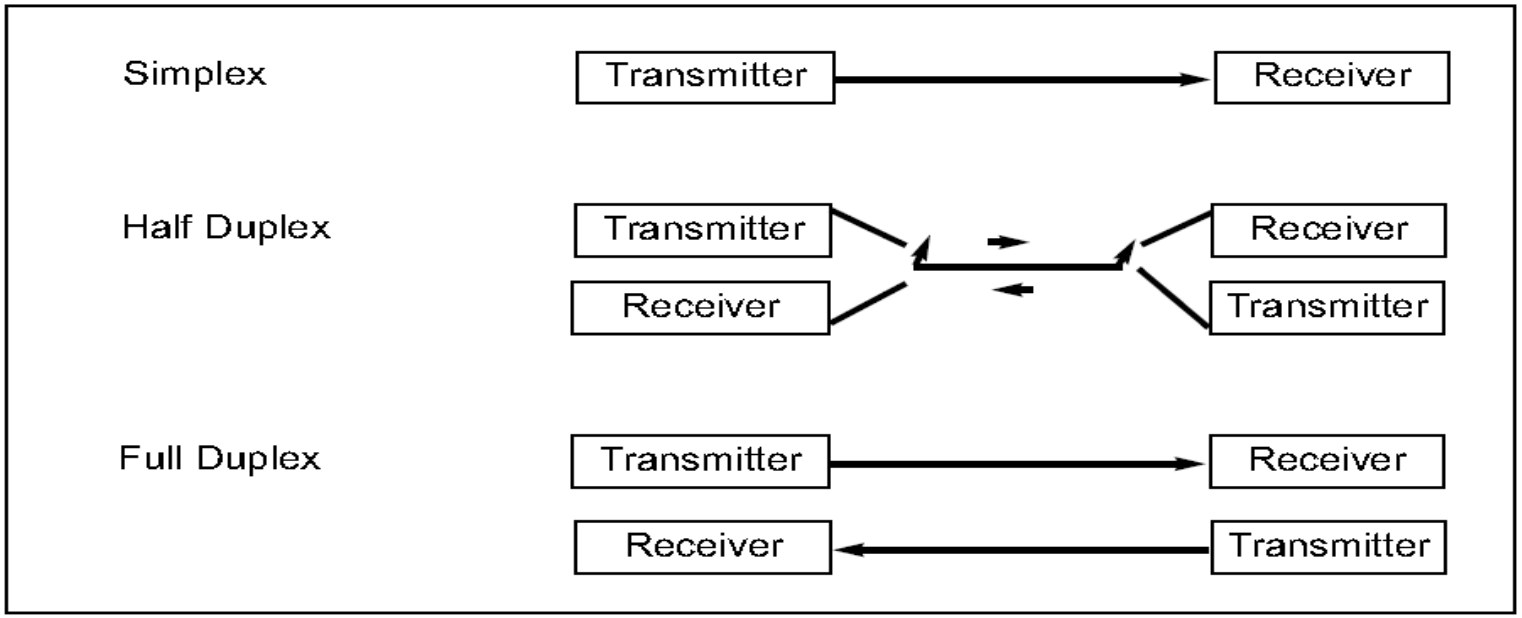
Parallel vs. serial

- 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)



Simplex vs. duplex vs. full duplex

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



Synchronous vs. asynchronous

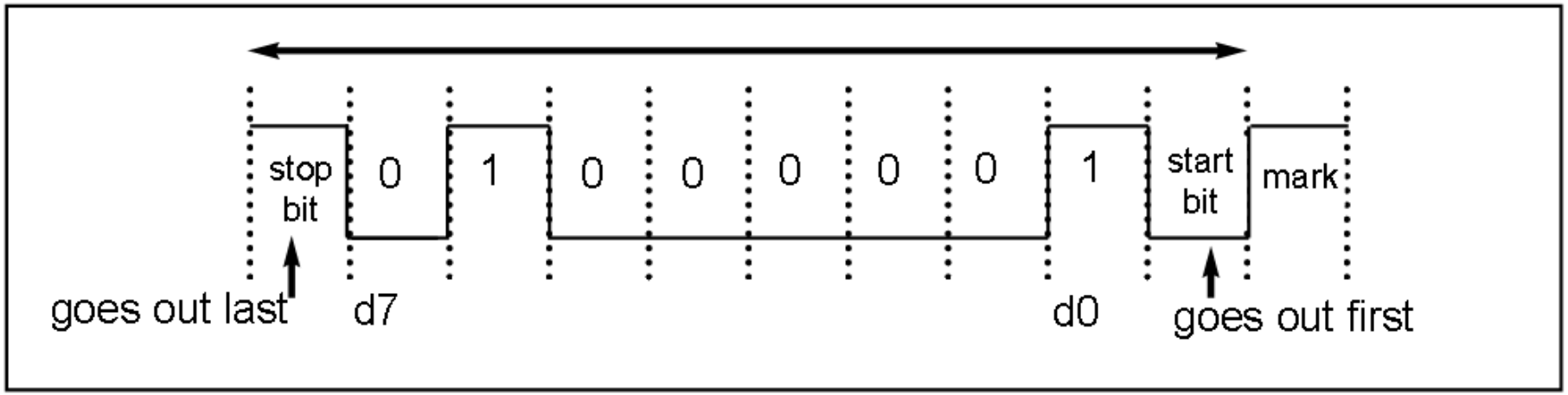
- Synchronous
  - Clock pulses should be transmitted during data transmission.
  - Only one side generates clock at the same time.
- Asynchronous
  - Clock pulses are not transmitted.
  - The two sides should generate clock pulses.
  - There should be a way to synchronize the two sides.

Data framing

- 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 the start bit.
  - [a bit of parity can be transmitted after the character] (optional). In an odd-parity bit system the total number of bits, including the parity bit, is odd.
  - Each frame is ended by one or two marks (1) which is called stop bit(s).

Data framing

- Each character is placed in between start and stop bits.
- This is called framing.
- Framing ASCII ‘A’ (41H)



Data transfer rate

- Three methods to describe the speed:
  - **Baud rate** is defined as the number of signal changes per second.
    - The rate of data transfer is stated in Hz (used in modem).
  - **Data rate** is defined as the number of bits transferred per second.
    - Each signal has several voltage levels.
    - The rate of data transfer is stated in bps (bits per second).

Data transfer rate

- **Effective data rate** is defined as the number of actual data bits transferred per second.
  - Redundant bits must be removed.

Data transfer rate

- Example: Assume that data is sent in the following asynchronous mode:
  - 2400 baud rate
  - Each signal has 4 voltage levels (-5V, -3V, 3V, 5V)
  - 1 start bit, 8-bit data, 1 odd-parity bit, 1 stop bit

Data transfer rate

- Baud rate
  - 2400 baud = 2400 signals per second = 2400 Hz
- Data rate
  - 4 voltage levels
  - $\text{Log}_2 4 = 2$, 2 bits are sent in every signal change.
  - Data rate = 2 bits $\times$ 2400 Hz = 4800 bps

Data transfer rate

- Effective data rate
  - Effective ratio = 8 / (1 + 8 + 2) = 8 / 11
  - Effective data rate

    = data rate $\times$ effective ratio

    = 4800 $\times$ 8 /11 = 3491 bps

Class Exercise 1

- Calculate the effective data rate with the following settings:
  - Assume that data is sent in the asynchronous mode.
  - 4000 baud rate
  - Each signal has 2 voltage levels.
  - 2 start bits, 16-bit data, 2 stop bits (no parity bit)

Class Exercise 1 (Your work)

Class Exercise 1 (Answer)

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.

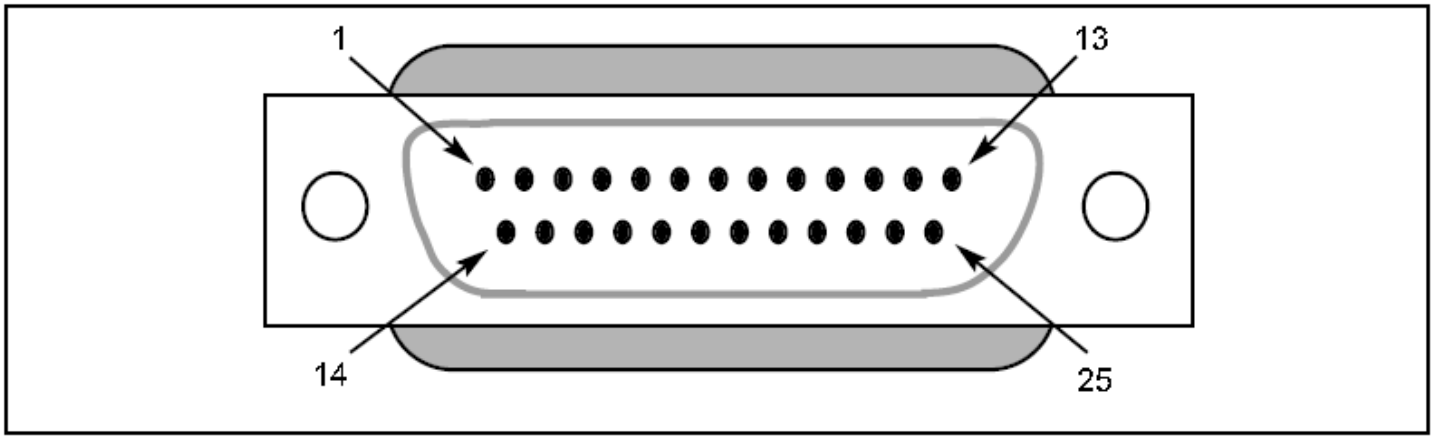


Figure 11-4. RS232 Connector DB-25

RS232 pins

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

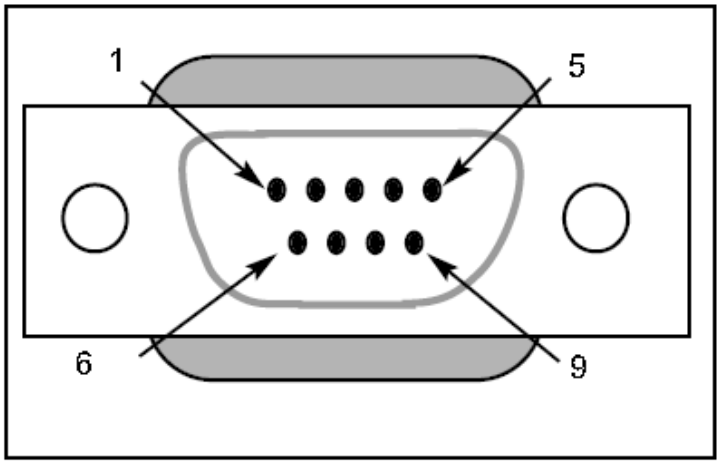
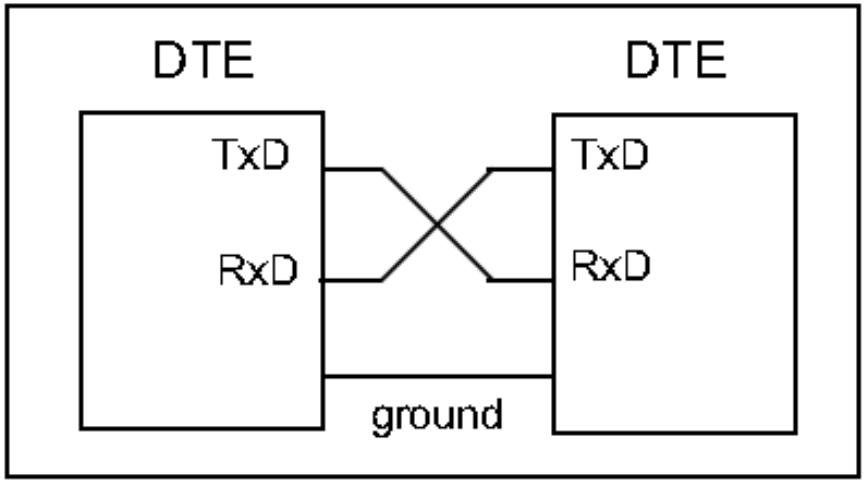


Figure 11-5. DB-9 9-Pin Connector

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 DTEs, we connect TxD of one device to RxD of the other and vise versa. It is called Null Modem Connection.



AVR connection

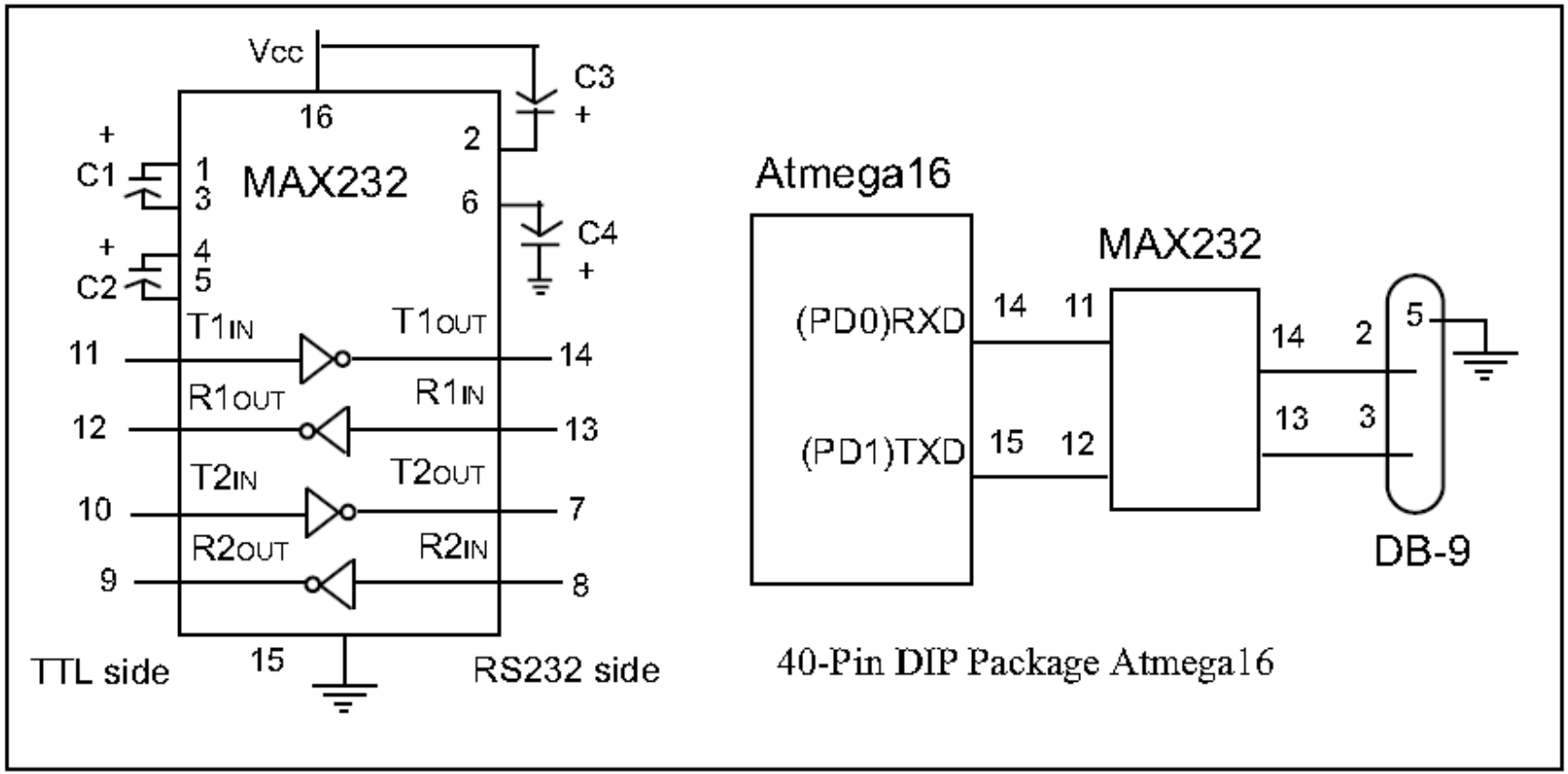


Figure 11-7. (a) Inside MAX232 and (b) its Connection to the Atmega16 (Null Modem)

USART in AVR

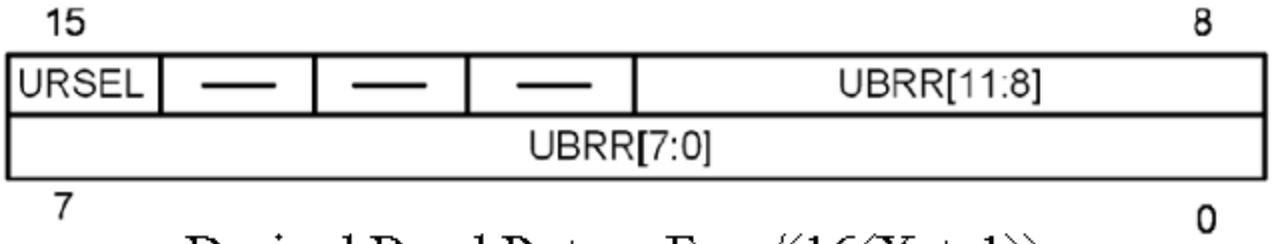
- AVR supports: Normal Asynchronous, Double Speed Asynchronous, Master Synchronous and Slave synchronous mode features.
- We concentrate on Asynchronous mode to connect the AVR to a PC.

USART in AVR

- In AVR 5 registers are associated with USART (Universal Synchronous Asynchronous Receiver/Transmitter).
  - UBRR (USART Baud Rate Register)
  - 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.



$$\text{Desired Baud Rate} = Fosc/(16(X + 1))$$

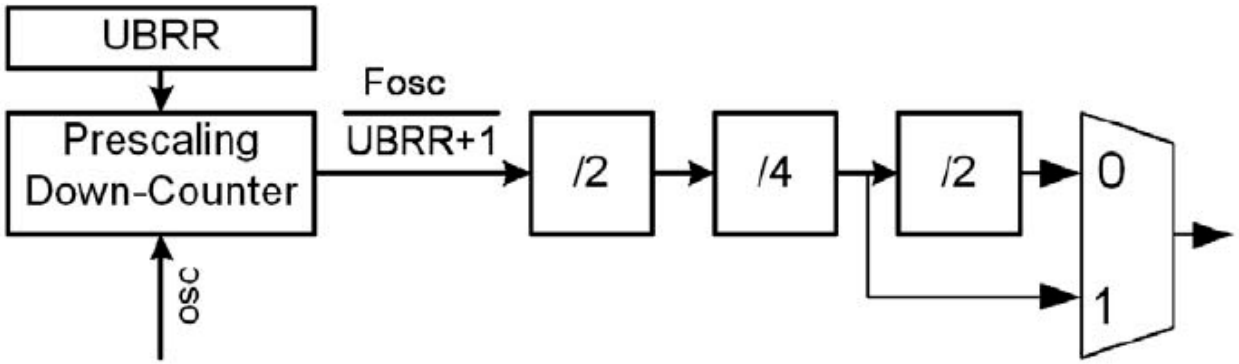


Table 11-3: Some PC Baud Rates in HyperTerminal

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

Baud rate calculation

- Fosc = Frequency of oscillator connected to the XTAL1 and XTAL2 pins
- Desired Baud Rate = Fosc / (16(X + 1))
- X is the value we loaded into the UBRR register.
- X = (Fosc / (16 $\times$ (Desired Baud Rate))) – 1
- Example:

  Fosc = 8 MHz

  Desired Baud Rate = 38400

  X = 12 (decimal) = C (hex)

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

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

  - Example: X = 8MHz / (16 × 38400) – 1 = 12.0208
  - Error = (12.0208 – 12) / 12 = 0.001736 (0.1736%)
  - Baud Rate Error = (38462 – 38400) / 38400 = 0.001603 (0.1603%)

UDR

- UDR = USART Data Register
- 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 = USART Control and Status Register A
- UCSRB = USART Control and Status Register B

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 discussed 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

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

Serial port programming (Receive)

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.
4. The RXC flag bit of the UCSRA register is monitored for a HIGH to see if an entire character has been received yet.
5. When RXC is raised, the UDR register has the byte. Its contents are moved into a safe place.
6. To receive the next character, go to Step 5.

Serial port programming (Send)

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.
5. Monitor the UDRE bit of the UCSRA register to make sure UDR is ready for next byte.
6. To transfer the next character, go to Step 5.

C programming

Write a program to send the message "The Earth is but One Country. " to the serial port continuously. Using the settings in the last example.

Solution:

```
#include <avr/io.h>                          //standard AVR header

void usart_init (void)
{  UCSRB = (1<<TXEN);
   UCSRC = (1<< UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x33;
}
void usart_send (unsigned char ch)
{  while (! (UCSRA & (1<<UDRE)));
   UDR = ch;
}
int main (void)
{  unsigned char str[ 30] = "The Earth is but One Country. ";
   unsigned char strLenght = 30;
   unsigned char i = 0;
   usart_init();
   while(1)
   {
      usart_send(str[ i++] );
      if (i >= strLenght)
         i = 0;
   }
   return 0;
}
```

Double the baud rate

- Two ways to increase the baud rate
  1. Use a higher frequency crystal.
  2. Change a bit in the UCSRA register (software way)
     - When the AVR is powered up, the U2X bit of the UCSRA is zero. We can set it high to double the baud rate.
- Desired Baud Rate = Fosc / (8(X + 1))
- X is the value we loaded into the UBRR register.
- X = (Fosc / (8 $\times$ (Desired Baud Rate))) – 1

C programming

Write an AVR C program to receive a character from the serial port. If it is 'a' – 'z' change it to capital letters and transmit it back. Use the settings in the last example.

Solution:

```
#include <avr/io.h>                          //standard AVR header

void transmit (unsigned char data);

int main (void)
{
                  // initialize USART transmitter and receiver
   UCSRB = (1<<TXEN)|(1<<RXEN);

   UCSRC = (1<< UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x33;

   unsigned char ch;

   while(1)
   {
      while(!(UCSRA&(1<<RXC)));              //while new data received
         ch = UDR;
      if (ch >= 'a' && ch<='z')
      {
         ch+=('A'-'a');
         while (! (UCSRA & (1<<UDRE)));
           UDR = ch;
      }
   }
   return 0;
}
```

Using interrupts

Write a C program to receive bytes of data serially and put them on Port B. Use Receive Complete Interrupt instead of the polling method.

Solution:

```
#include <avr\io.h>
#include <avr\interrupt.h>

ISR(USART_RXC_vect)
{
   PORTB = UDR;
}

int main (void)
{
   DDRB = 0xFF;                          //make Port B an output
   UCSRB = (1<<RXEN)|(1<<RXCIE);         //enable receive and RXC int.
   UCSRC = (1<< UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x33;
   sei();                                //enable interrupts
   while (1);                            //wait forever
   return 0;
}
```

Using interrupts

Write a C program to transmit the letter 'G' serially at 9600 baud, continuously. Assume XTAL = 8 MHz. Use interrupts instead of the polling method.

Solution:

```
#include <avr\io.h>
#include <avr\interrupt.h>
ISR(USART_UDRE_vect)
{
    UDR = 'G';
}

int main (void)
{
   UCSRB = (1<<TXEN)|(1<<UDRIE);
   UCSRC = (1<< UCSZ1) (1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x33;

   sei();                          //enable interrupts
   while (1);                      //wait forever
   return 0;
}
```

ATmega328p

- ATmega32
  - UCSRA, UCSRB, UCSRC
- ATmega328p
  - UCSR0A, UCSR0B, UCSR0C
- The contents of three registers in ATmega328p are different to that in ATmega32. Check the datasheet.

Reference Readings

- Chapter 12 – *The AVR Microcontroller and Embedded Systems : Using Assembly and C*, M. A. Mazidi, S. Naimi, and S. Naimi, Pearson, 2014.

End