AVR UART has five registers associated to it

- UBBR
- UCSRA
- UCSRB
- UCSRC
- UDR

UDR (Stands for USART data register)

7	6	5	4	3	2	1	0

- There are two shift registers which share same I/O
    - Transmit shift register
        - It has a buffer connected to it directly called transmit shift buffer register
        - When data is written to UDR , it gets redirected called transmit shift buffer register
    - Receive shift register
        - It has a buffer connected to it directly called receive shift buffer register
        - When data is received , on reading UDR it returns the content of receive shift buffer register

UBRR (stands for USART baud rate register)

7	6	5	4	3	2	1	0

- Decides baud rate for uart communication
- Holds the prescaler
- UBRR = (Fosc/(16 (desired baud rate)))-1 , where Fosc is cpu frequency , and desired baud rate is the baud rate we want to set set for uart communication

UCSRA(Stands for USART control and status register A)

7	6	5	4	3	2	1	0
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

- RXC
  - This flag is set when there is new data in receive buffer that is not yet read
  - It is cleared when receive buffer is empty
  - Can be used to generate receive complete interrupt

- TXC
  - This is set when entire frame in transmit shift register has been transmitted and there is no data available in transmit data buffer register
  - It can be cleared writing 1 to its bit location
  - Gets automatically cleared when transmit complete interrupt is executed
  - Can be used to generate transmit complete interrupt

- UDRE
  - Stands for usart data register empty
  - This flag is set when transmit data buffer is empty and is ready to receive data
  - If this bit is cleared, nothing shall be written in UDR , as doing so will override your last data
  - This can be used to generate data register empty interrupt

- FE
  - Stands for frame error
  - This bit is set when a frame error occurs while receiving
  - When the first start byte is zero

- DOR
  - Stands for data overrun
  - This bit is set when if data overrun is detected
  - When receive data buffer and receive shift register is full and a new byte is detected
- PE
  - Stands for parity error
  - If parity checking is enabled , and the newly received byte has parity error

- U2X
  - Stands for double the transmission speed
  - Setting this bit to one will double transfer rate for asynchronous communication

- MPCM
  - Stands for multiprocessor communication mode
  - When set 1 , it enables the multiprocessor communication mode

UCSRB (Stands for USART control and status register B):

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

- RXCIE
  - Stands for receive complete interrupt enable
  - set this bit to 1 to enable interrupt on receive flag (RXC) in UCSRA.

- TXCIE
  - Stands for transmission complete interrupt enable
  - Set this to 1 to enable interrupt on receive flag , (TXC) in UCSRA.

- UDRIE
  - Stands for uart data register empty flag
  - set this bit to 1 o enable interrupt on UDRE flag in UCSRA

- RXEN
  - Stands for receive enable
  - Set this bit to 1 to enable usart receiver

- TXEN
  - Stands for transmitter enable
  - Set this bit to 1 to enable usart transmitter

- UCSZ2
  - Msb in setting character size
  - Character size is derived from UCSZ2 , UCSZ1, UCSZ0 as per table shown after UCSRC register explanation.
- RXB8
  - Stands for receive data bit 8
  - This is the ninth data bit of received character when using serial frames with nine data bits.
- TXB8
  - Stands for transmit data bit 8
  - This is the ninth data bit of transmitted character when using serial frames with nine data bits.
  -

UCSRC (Stands for USART control and status register C)

7	6	5	4	3	2	1	0
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

- URSEL
  - Stands for register select
  - This bit selects to access wither UCSRC or UBRRH register , as they share the same I/O locations
  - While writing to either of these registers, this bit controls which of the two registers will be targeted to
  - Setting this bit to 1 selects UCSRC
  - Resetting this bit to 0 selects UBBR

- UMSEL
  - Stands for uart select mode
  - Setting this bit to 1 selects synchronous mode of operation
  - Resetting this bit to 0 selects asynchronous mode of operation

- UPM1
  - Stands for parity mode
  - Acts as msb while setting parity mode
  - Parity mode is set in this sequence UPM1,UPM0
- UPM0
  - Stands for parity mode
  - Acts as lsb while setting parity mode
  - Parity mode is set in this sequence UPM1,UPM0
  - Check for table to set parity mode after this section

- USBS
  - Stands for stop bit selection
  - Selects number of stop bit to be transmitted per frame
  - Setting this bit to zero selects 1 stop bits per frame
  - Setting this bit to one selects 2 stop bits per frame

- UCSZ1
  - This bit combined with UCSZ2 and UCSZ0 sets the character size in frame
  - Refer the table after this section

- UCSZ0
  - This bit combined with UCSZ2 and UCSZ1 sets the character size in frame
  - Refer the table after this section

- UCPOL
  - Stands for clock polarity
  - This bit is used only for synchronous mode

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

UPM1	UPM0	Parity
0	0	disabled
0	1	reserved
1	0	Even parity
1	1	Odd parity

****

Source code: receive data serially using receive complete interrupt:

```
#include<avr/io.h>
#include<avr/interrupt.h>
Uint8_t data ;
ISR(USART_RXE_VECT)
{
data = UDR ;
}
Int main(void)
{
USCRB = (1<<RXEN) | (1<< RXCIE);
USCRC = (1<<UCSZ2) | (1<< UCSZ1) | (1<< URSEL);
UBBRL = 0X33;
sei();
return 0;
}
```

****

Source code: receive data serially using UDR complete interrupt:

```
#include<avr/io.h>
#include<avr/interrupt.h>
Uint8_t data ;
ISR(USART_UDRE_VECT)
{
Data = 125;
UDR =  data;;
}

Int main(void)
{
USCRB = (1<<TXEN) | (1<< UDRIE);
USCRC = (1<<UCSZ2) | (1<< UCSZ1) | (1<< URSEL);
UBBRL = 0X33;
sei();
return 0;
}
```

****

Source code:

```
#include<avr/io.h>
int main(void)
{
DDRA = 0xFF;
USCRB = (1<< RXEN);
USCRC = (1<<UCSZ1) | (1<< UCSZ0) | (1<< URSEL);
UBRRL = 0x33;
while(USCRA & (1<<RXE)
        {
        PORTA = UDR;
        }
return 0;
}
```

****

Source code:

```
#include<avr/io.h>
Void usart_init()
{
        USCRB = (1<<TXEN) ;
        USCRC = (1<<UCSZ1) | (1<< UCSZ0) | (1<< URSEL);
        UBBRL = 0X33;

}

Void uart_send(unsigned char data)
{
while(USCRA & (1<< UDRE))
        {
        UDR = data;
        }
}

int main(void)
{
Unsigned char str[15] = " sameer teaches ";
Uint8_t length = 15;
Uint8_t i =0;
usart_init();
while(1)
uart_send(str[i++]);
If (i > length)
        i=0;
return 0;
}
```

****