Robotech Labs

USART of AVR Microcontrollers.

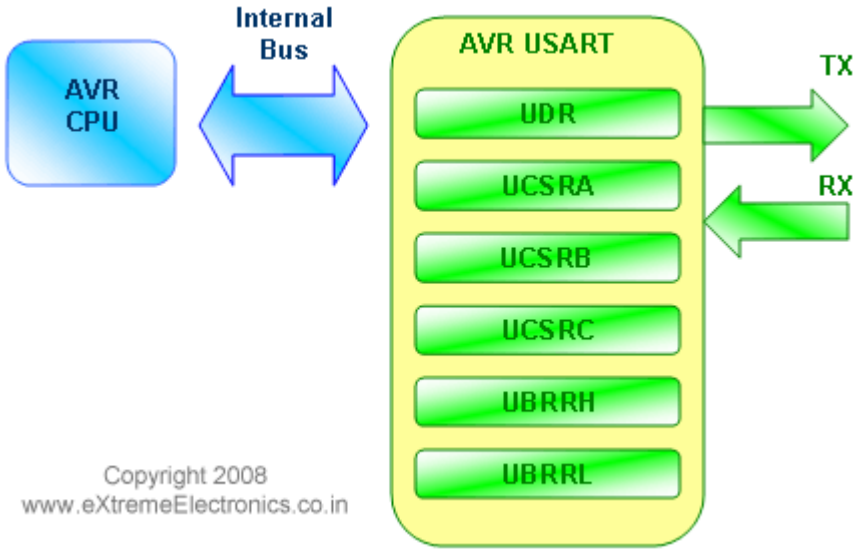
The USART of the AVR is connected to the CPU by the following six registers.

- **UDR** - USART Data Register : Actually this is not one but two register but when you read it you will get the data stored in receive buffer and when you write data to it goes into the transmitters buffer. **This important to remember it.**

- **UCSRA** - USART Control and status Register A : As the name suggests it is used to configure the USART and it also stores some status about the USART. There are two more of this kind the **UCSRB and UCSRC**.

- **UBRRH** and **UBRRH** : This is the USART Baud rate register, it is 16BIT wide so UBRRH is the High Byte and UBRRL is Low byte. But as we are using C language it is directly available as UBRR and compiler manages the 16BIT access.

So the connection of AVR and its internal USART can be visualized as follows.



Registers Explained

UDR: Explained above.
UCSRA: USART Control and Status Register A

Bit No	7	6	5	4	3	2	1	0
Name	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

Initial Val	0	0	1	0	0	0	0	0

RXC this bit is set when the USART has completed receiving a byte from the host (may be your PC) and the program should read it from **UDR**
TXC This bit is set (1) when the USART has completed transmitting a byte to the host and your program can write new data to USART via UDR

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

Bit No	7	6	5	4	3	2	1	0
Name	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Initial Val	0	0	0	0	0	0	0	0

RXCIE: Receive Complete Interrupt Enable - When this bit is written one the the associated interrupt is enabled.
TXCIE: Transmit Complete Interrupt Enable - When this bit is written one the the associated interrupt is enabled.
RXEN: Receiver Enable - When you write this bit to 1 the USART receiver is enabled. **The normal port functionality of RX pin will be overridden.** So you see that the associated I/O pin now switch to its secondary function,i.e. RX for USART.
TXEN: Transmitter Enable - As the name says!
UCSZ2: USART Character Size - Discussed later.
For our first example we won't be using interrupts so we set UCSRB as follows

```
UCSRB=(1<<RXEN)|(1<<TXEN);
```

UCSRC: USART Control And Status Register C
****

Bit No	7	6	5	4	3	2	1	0
Name	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Initial Val	0	0	0	0	0	0	0	0

IMPORTANT : The UCSRC and the UBRRH (discussed below) register shares same address so to determine which register user want to write is decided with the 7th(last) bit of data if its 1 then the data is written to UCSRC else it goes to UBRRH. This seventh bit is called the
URSEL: USART register select.
UMSEL: USART Mode Select - This bit selects between asynchronous and synchronous mode. As asynchronous mode is more popular with USART we will be using that.

UMSEL	Mode
0	Asynchronous
1	Synchronous

USBS: USART Stop Bit Select - This bit selects the number of stop bits in the data transfer.

USBS	Stop Bit(s)
0	1 BIT

1	2 BIT

UCSZ: USART Character size - These three bits (one in the UCSRB) selects the number of bits of data that is transmited in each frame. Normally the unit of data in MCU is 8BIT (C type "char") and this is most widely used so we will go for this. Otherwise you can select 5,6,7,8 or 9 bit frames!

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

So we set UCSRC as follows

```
UCSRC=(1<<URSEL)|(3<<UCSZ0);
```

UBRR: USART Baud Rate Register:

****

This is the USART Baud rate register, it is 16BIT wide so **UBRRH** is the High Byte and **UBRRL** is Low byte. But as we are using C language it is directly available as UBRR and compiler manages the 16BIT access. This register is used by the USART to generate the data transmission at specified speed (say 9600Bps). UBRR value is calculated according to following formula.

$$UBRR = \frac{f_{osc}}{16 \times \text{Baud Rate}} - 1$$

Where fosc is your CPU frequency say 16MHz

Baud Rate is the required communication speed say 19200 bps

Example:

For above configuration our UBRR value comes to be 51.083 so we have to set

```
UBRR=51;
```

in our initialization section. Note UBRR can hold only integer value. So it is better to use the baud rates that give UBRR value that are purely integer or very close to it. So if your UBRR value comes to be 7.68 and you decided to use UBRR=8 then it has high error percentage, and**communication is unreliable**!

Initialization of USART

Before using the USART it must be initialized properly according to need. Having the knowledge of RS232 communication and Internal USART of AVR you can do that easily. We will create a function that will initialize the USART for us.

```
#include <avr/io.h>
#include <inttypes.h>

void USARTInit(uint16_t ubrr_value)
{

   //Set Baud rate
   UBRR= ubrr_value;

   /*Set Frame Format


   >> Asynchronous mode
   >> No Parity
   >> 1 StopBit
   >> char size 8

   */

   UCSRC=(1<<URSEL)|(3<<UCSZ0);


   //Enable The receiver and transmitter
   UCSRB=(1<<RXEN)|(1<<TXEN);


}
```

Now we have a function that initializes the USART with a given UBRR value.