AVR and UART Configuration – Tutorial #15

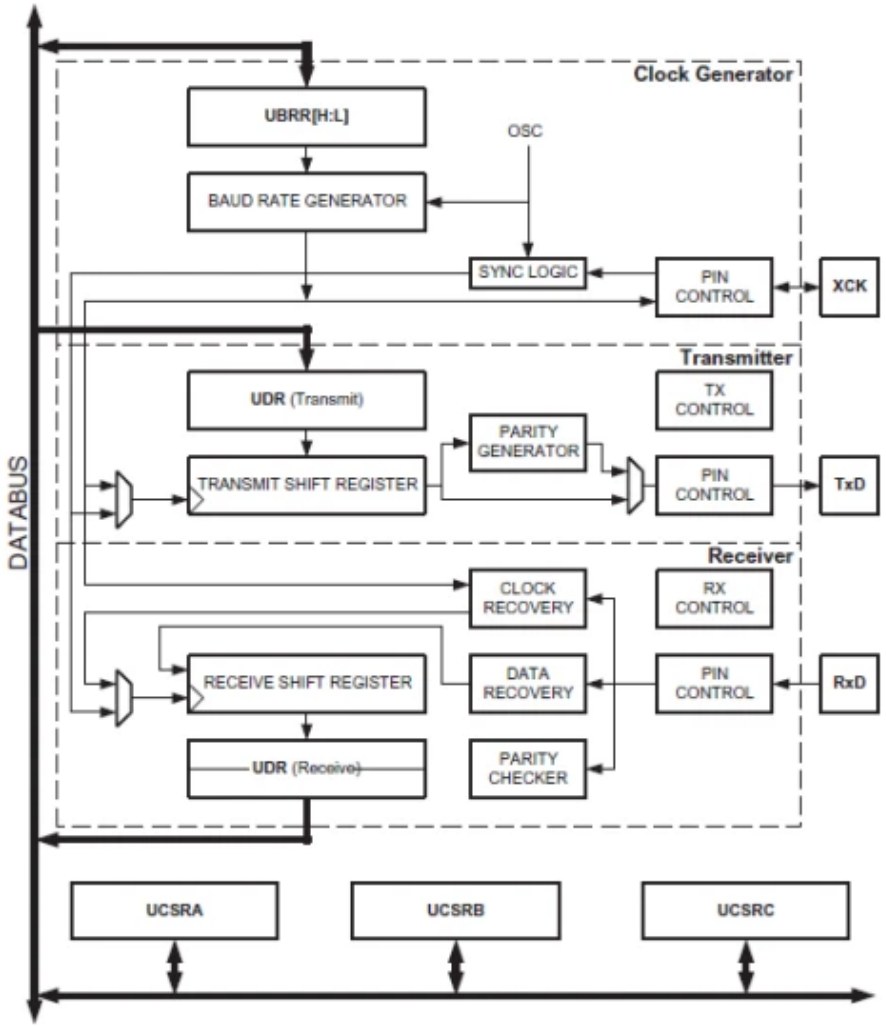
T.K. HAREENDRAN

AVR tutorial

Share this:

When it comes to AVR UART configuration, it is required to define the packet format a transmitter is going to transmit, and packet format is defined by character size, parity bits and stop bits. Another one needs configuration is the baud rate by putting up some egresses in the UBRR (USART Band Rate Register). From the previous chapter we learned that USART band registers are responsible for configuring baud rate of UART, and the value in this register will define the exact baud rate.

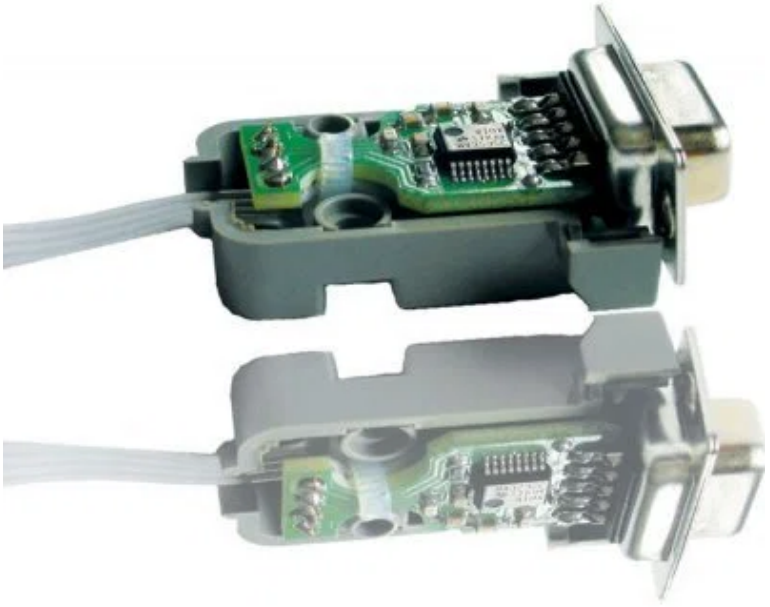
The AVR datasheet contains the table of values for a particular baud rate at a particular crystal frequency. The relation between baud rate and UBBR values is given by the formula: UBRR = Fosc/16xBaud – 1. For instance, with 16Mhz crystal frequency and UBRR contains rounded off value of 103 in it, baud rate will be 9600 bps. In the formula, UBRR is the UBRR register value in integer, Fosc is the crystal frequency in Hertz, and the baud rate is the required baud rate in bps. Note that only 12 bits of UBRR are used for defining baud rates, ie, 8 bits of UBRRL and 4 low order bits of UBRRH.



The Atmega controller has a so called Universal Asynchronous Receiver Transmitter (UART) which can be used to communicate with any other RS-232 device like a PC. A simplified block diagram of the USART Transmitter (from the Atmega8L datasheet) is shown here. CPU accessible I/O Registers and I/O pins are shown in bold lines.

The dashed boxes in the figure separate the three main parts of the USART. Clock generator, Transmitter and Receiver. Control Registers are shared by all units.The clock generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCK (transfer clock) pin is only used by synchronous transfer mode.

The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a parity checker, control logic, a Shift Register and a two levelreceive buffer (UDR). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun and Parity Errors.



For handling UART, first the baud has to be specified in UART. After the initialization, UART can perform reading and writing tasks. Here, both reading and writing data buffer registers share the same input-output (I/O) address referred as UDR (USART Data Request). The transmit data buffer register (TXB) will be the destinaion for data written to the UDR register location. Reading the UDR register location will return the contents of the receive data buffer register (RXB).

Here is a USART initialization fnction in C language:

```
void USART_init (int ubbr)
{
UBRRL = ubrr;
UBRRH = (ubrr>>8);
USRC | = (1<
```

Here we are passing 'ubbr' value (which we want to set) in UBBR. Lower byte is setted by directly storing u

```
unsigned char USART_ReadChar()
{
   while (! (UCSRA & (1<
```

And for writing:

```
void USART_WriteChar (char data)
{
   while (! (UCSRA & (1<
```

→ Part 16: **AVR EEPROM Handling**

← Part 14: **AVR ADC**

Share this:

More