EMBEDDED SYSTEMS CMPE-453

Department of Computer Engineering

Serial Communication-2

HOW TO USE UART

I. Setting the baud rate
II. Enabling serial transmission/reception of data.
III. For transmission
    I. wait until UART is ready
    II. load data into data-register of UART (UDR0)
IV. For reception
    I. Check if data has arrived
    II. and then read it out from UDR0

HOW TO SET BAUD RATE

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous double speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous master mode	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{osc}	System oscillator clock frequency
UBRRn	Contents of the UBRRnH and UBRRnL registers, (0-4095)

USART BAUD RATE REGISTRATION

CALCULATING UBRR0 VALUES

Desired Baud Rate	Normal mode UBRR0	Rounded value	Double speed mode UBRR0	Rounded Value
9600	5.5	6	12.02	12
4800	12.02	12	25.04	25
2400	25.0	25	51.08	51

Baud calculation normal mode:
Baud=clock_freq/(16x(UBRR0+1))
=1M/(16x(6 + 1)) = 8928

Expected baud rate was 9600.

Error= (9600-8928) x 100 /9600= 7%

Baud calculation double speed mode:
Baud=clock_freq/(8x(UBRR0+1))
=1M/(8x(12 + 1)) = 9615

Expected baud rate was 9600.

Error= (9600-9615) x 100 /9600= 0.15%

SETTING THE BAUD RATE

- USART Baud rate register: UBRR0L and UBRR0H
- Bit 12:15 are reserved for future usage
- UBBR0[11:0] 12 bit register holding the baud rate.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

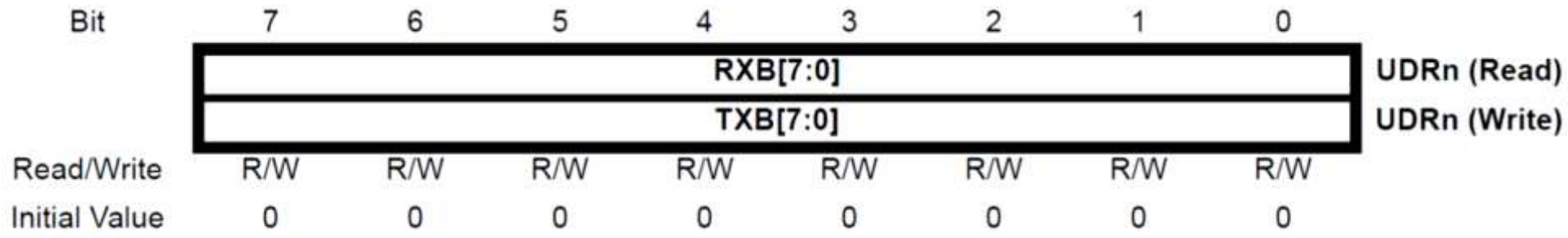
UCSR0A (USART CONTROL AND STATUS REGISTER0 A)

- Bit 1: U2X0 (double the USART transmisson speed):
  - 1 → double speed mode, 0 → Normal mode
- Bit 5: UDRE (USART Data Register Empty)
  - 1 → USART Data Register (UDR0) is empty, i.e., new data can be loaded in UDR0 for transmission.
  - 0 → Data transmisson is in progress. Do not load new data into UDR0.
- Bit 7: RXC0 (Receive Complete)
  - 1→ new data has been received in UDR and ready to be used.
  - 0 → Data reception is in progress

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UDR0 (USART DATA REGISTER 0)

- Transmit and receive buffer share same I/O address.
- Writing UDR0 loads data in TXB.
- Reading UDR0 returns data from RXB.



UCSR0B (USART CONTROL AND STATUS REGISTER0 B)

- Bit 4: RXEN (Receiver Enable n)
  - 1→USART receiver is enable, overwrites normal port operation of PD0 for RxD pin
  - 0→USART receiver is disabled
- Bit 3: TXEN (Transmitter Enable n)
  - 1 → USART transmitter is enabled, overwrites normal operation of PD1 for TxD
  - 0 → USART transmitter is disabled.

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	**TXCIEn**	**UDRIEn**	**RXENn**	**TXENn**	**UCSZn2**	**RXB8n**	**TXB8n**	**UCSRnB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

USCR0C (USART CONTROL AND STATUS REGISTER0 C)

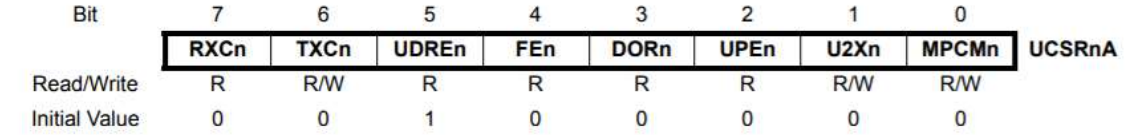
- USART Character Size
- UCSZn1:0 (in UCSR0C) and UCSZn2 (in UCSR0B) define number of data bits in a frame.

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
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	1	1	0	

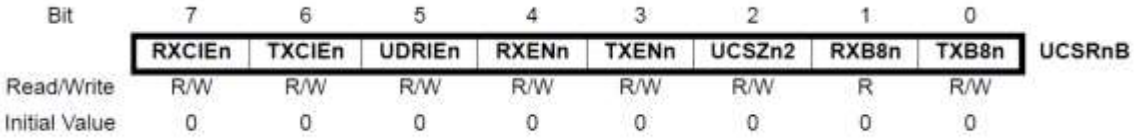
UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

C-FUNCTION TO INITIALIZE UART

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

#include "USART.h"

#include <util/setbaud.h>
void initUSART(void)
        {

        /* requires BAUD, defined in USART.h */

        UBRR0H = UBRRH_VALUE;          /* defined in setbaud.h */
        UBRR0L = UBRRL_VALUE;

         /* select normal or double speed mode */
        #if USE_2X

                UCSR0A |= (1 << U2X0);

        #else

                UCSR0A &= ~(1 << U2X0);

        #endif

        UCSR0B = (1 << TXEN0) | (1 << RXEN0); /* Enable USART transmitter/receiver */

        UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); /* 8 data bits*/

        }
```



Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	



Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

C-FUNCTIONS FOR TRANSMITTING AND RECEIVING UART DATA

```
void transmitByte(uint8_t data)
        {
        /* Wait for empty transmit buffer */
        /* macro defined in io.h*/
        loop_until_bit_is_set(UCSR0A, UDRE0);
        UDR0 = data; /* send data */
        }
```

```
uint8_t receiveByte(void)
        {
        /* Wait for incoming data */
        /* macro defined in io.h*/
        loop_until_bit_is_set(UCSR0A, RXC0);

        /* return register value */
        return UDR0;


        }
```

IN CASE OF ARDUINO

```
// Arduino UART

void setup() {
  pinMode(8, INPUT_PULLUP); // set push button pin as input
  pinMode(13, OUTPUT);      // set LED pin as output
  digitalWrite(13, LOW);    // switch off LED pin

  Serial.begin(9600);       // initialize UART with baud rate of 9600 bps
}

void loop() {
  if (Serial.available()) {
    char data_rcvd = Serial.read();   // read one byte from serial buffer and save to data_rcvd

    if (data_rcvd == '1') digitalWrite(13, HIGH); // switch LED On
    if (data_rcvd == '0') digitalWrite(13, LOW);  // switch LED Off
  }

  if (digitalRead(8) == HIGH)
                Serial.write('0');    // send the char '0' to serial if button is not pressed.
  else
                Serial.write('1');                  // send the char '1' to serial if button is pressed.
}
```