Programming Technologies: Serial Communication

Due on Octubre 2016

José A. Aviña - UART Programming

Índice

Assignment 1

Listing 1: Serial Writing with AVR Libc

```
#include <avr/io.h>
#include <util/delay.h>
/*
PD0 -> rx
PD1 -> tx


UART baud rate
UBRR0L

UART baud rate
UBRR0H

UART Control &
UCSR0A

UART Control &
UCSR0B

UART Control &
UCSR0C
UDR0
*/

#ifndef F_CPU
#define F_CPU 16000000 //MhZ
#endif
#define UART_BAUD_RATE 9600 //Baud

void uart_init()
{
  uint16_t ubbr = F_CPU/((UART_BAUD_RATE)*16L) - 1;
  // asynchronous 8N1
  UCSR0C= 3<<UCSZ00;
  // set baud rate
  UBRR0L = ubbr; // low byte
  UBRR0H = (ubbr >> 8); // high byte
  // enable rx and tx
  UCSR0B = (1<<RXEN0) | (1<<TXEN0);
}

static unsigned char uart_read_char(void)
{
  while( !(UCSR0A & (1 << RXC0)));
  return UDR0;
}

void uart_write_char(char c)
{
  while (!(UCSR0A & (1<<UDRE0))); // wait until buffer is ready
```

Assignment 1 continued on next page...

```
  UDR0 = c;
}

void uart_write_string(char *c)
{
  while (*c != '\0') {
    uart_write_char(*c);
    c++;
  }
}


int main(void)
{
  uart_init();
  unsigned char byte;

  byte =  uart_read_char();

  if (byte == '1') {
    while (1)  {
     uart_write_string("Hello from Arduino!\n");
    }
  }
 }
```

Assignment 2

Listing 2: Serial Reading with AVR Libc

```
#include <avr/io.h>
#include <util/delay.h>
/*
PD0 -> rx
PD1 -> tx


UART baud rate
UBRR0L
UBRR0H

UART Control &
UCSR0A

UART Control &
UCSR0B

UART Control &
UCSR0C
UDR0
*/
```

```
#ifndef F_CPU
#define F_CPU 16000000 //MhZ
#endif
#define UART_BAUD_RATE 9600 //Baud

void uart_init()
{
  uint16_t ubbr = F_CPU/((UART_BAUD_RATE)*16L) - 1;
  // asynchronous 8N1
  UCSR0C= 3<<UCSZ00;
  // set baud rate
  UBRR0L = ubbr; // low byte
  UBRR0H = (ubbr >> 8); // high byte
  // enable rx and tx
  UCSR0B = (1<<RXEN0) | (1<<TXEN0);

}

static unsigned char uart_read_char(void)
{
  while( !(UCSR0A & (1 << RXC0)));
  return UDR0;
}

void uart_write_char(char c)
{
  while (!(UCSR0A & (1<<UDRE0))); // wait until buffer is ready
  UDR0 = c;
}

void uart_write_string(char *c)
{
  while (*c != '\0') {
    uart_write_char(*c);
    c++;
  }
}


int main(void)
{
  uart_init();
 // DDRB = 0x00;
  PORTB = 0x00;
  DDRB = 0b00100000;

  unsigned char byte = '0';

  byte = uart_read_char();
 // DDRB = 0b00100000;
 // PORTB = 0b00000000;

  byte = uart_read_char();
```

```
_delay_ms(500);

 if (byte == '1') {
    while (1)  {

      PORTB |= (1<<PB5);  //arduino digital pin 5 -> 5V
      _delay_ms(1000);

      PORTB &= ~(1<<PB5);  //arduino digital pin 5 -> GND
      _delay_ms(1000);
   }
 }
}
```

Assignment 3

Implementar una aplicación *Cliente-Servidor* para comunicar vía serial el ATmegaX (Arduino) con Linux OS, haciedo recurso a los códigos fuentes de los Listados 1, 2 y 3:

- Servidor Linux/Cliente Arduino.
- Servidor Arduino/Cliente Linux.

Listing 3: Serial Communication in Linux

```
#include <sys/types.h>
     #include <sys/stat.h>
     #include <fcntl.h>
     #include <termios.h>
     #include <stdio.h>

     #define BAUDRATE B9600
     #define MODEMDEVICE "/dev/ttyACM0"
     #define _POSIX_SOURCE 1 /* POSIX compliant source */


 int   main()
     {
       int fd,c, res;
       struct termios oldtio,newtio;
       char buf[20];

       fd = open(MODEMDEVICE, O_RDWR | O_NOCTTY );
       if (fd <0)
         perror(MODEMDEVICE);

       tcgetattr(fd,&oldtio); /* save current port settings */

       //bzero(&newtio, sizeof(newtio));
       newtio.c_cflag = BAUDRATE | CRTSCTS | CS8 | CLOCAL | CREAD;
       newtio.c_iflag = IGNPAR;
       newtio.c_oflag = 0;
```

```
        /* set input mode (non-canonical, no echo,...) */
30      newtio.c_lflag = 0;

        newtio.c_cc[VTIME]    = 1;   /* inter-character timer unused */
        newtio.c_cc[VMIN]     = 19;   /* blocking read until n chars received */

35      //tcflush(fd, TCIFLUSH);
        tcsetattr(fd,TCSANOW,&newtio);

        write(fd, "1", 1);

40      while (1) {       /* loop for input */
          res = read(fd,buf,19);   /* returns after n chars have been input */
          buf[res]=0;               /* so we can printf... */
          printf(": %s\n", buf);
          //tcflush(fd, TCIFLUSH);
45
        }
        tcsetattr(fd,TCSANOW,&oldtio);

        return 0;
50    }
```