**\*\*\* AVR atmega32p USART (IoT Army showing prgramm):             date: 06/11/2017**

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#define BAUDRATE 19200

#define BAUD\_PRESCALLER (((*F\_CPU* / (BAUDRATE \* 16UL))) - 1)

//Declaration of our functions

void USART\_init(void);

unsigned char USART\_receive(void);

void USART\_send( unsigned char data);

void USART\_putstring(char\* StringPtr);

char String[]="IoT ARMY";    //String[] is in fact an array but when we put the text between the " " symbols the compiler threats it as a String and automatically puts the null termination character in the end of the text

int main(void){

      USART\_init();        //Call the USART initialization code

      while(1){        //Infinite loop

             USART\_putstring(String);    //Pass the string to the USART\_putstring function and sends it over the serial

*\_delay\_ms*(8660066);        //Delay for 5 seconds so it will re-send the string every 5 seconds

      }

      return 0;

}

void USART\_init(void){

      UBRR0H = (*uint8\_t*)(BAUD\_PRESCALLER>>8);

      UBRR0L = (*uint8\_t*)(BAUD\_PRESCALLER);

      UCSR0B = (1<<RXEN0)|(1<<TXEN0);

      UCSR0C = (3<<UCSZ00);

}

unsigned char USART\_receive(void){

      while(!(UCSR0A & (1<<RXC0)));

      return UDR0;

}

void USART\_send( unsigned char data){

      while(!(UCSR0A & (1<<UDRE0)));

      UDR0 = data;

}

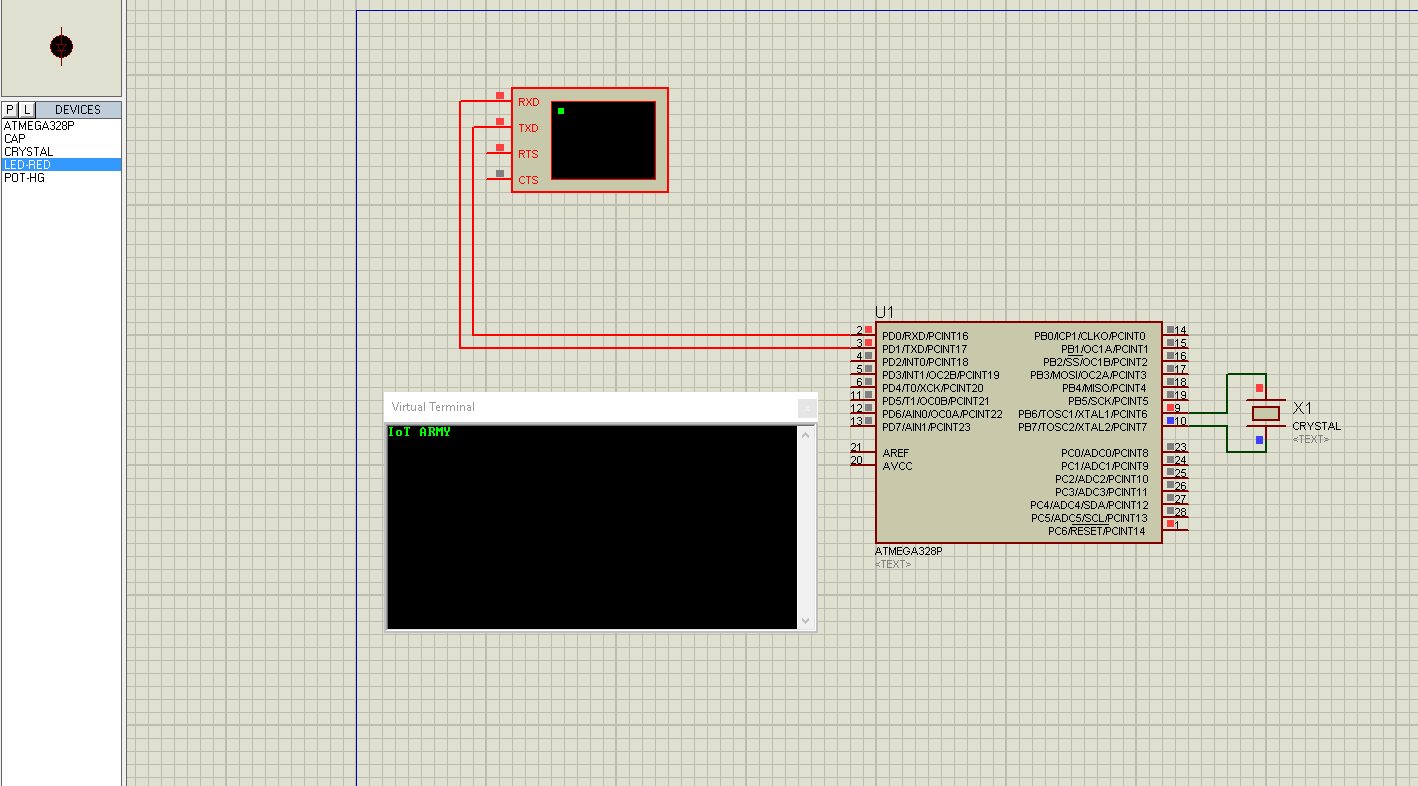
void USART\_putstring(char\* StringPtr){

      while(\*StringPtr != 0x00){

             USART\_send(\*StringPtr);

      StringPtr++;}

}



\*\*\* **Integer print (100 to 1 ) using usart Register       date: 8-11-2017**

**#define F\_CPU 16000000UL**

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

**#include <util/delay.h>**

**#define BAUDRATE 19200**

**#define BAUD\_PRESCALLER (((F\_CPU / (BAUDRATE \* 16UL))) - 1)**

**//Declaration of our functions**

**void USART\_init(void);**

**unsigned char USART\_receive(void);**

**void USART\_send( unsigned char data);**

**void USART\_putstring(char\* StringPtr);**

**char String[]="IoT ARMY";    //String[] is in fact an array but when we put the text between the " " symbols the compiler threats it as a String and automatically puts the null termination character in the end of the text**

**int main(void){**

**USART\_init();        //Call the USART initialization code**

**while(1){        //Infinite loop**

**USART\_putstring(String);    //Pass the string to the USART\_putstring function and sends it over the serial**

**\_delay\_ms(5000000);        //Delay for 5 seconds so it will re-send the string every 5 seconds**

**}**

**return 0;**

**}**

**void USART\_init(void){**

**UBRR0H = (uint8\_t)(BAUD\_PRESCALLER>>8);**

**UBRR0L = (uint8\_t)(BAUD\_PRESCALLER);**

**UCSR0B = (1<<RXEN0)|(1<<TXEN0);**

**UCSR0C = (3<<UCSZ00);**

**}**

**unsigned char USART\_receive(void){**

**while(!(UCSR0A & (1<<RXC0)));**

**return UDR0;**

**}**

**void USART\_send( unsigned char data){**

**while(!(UCSR0A & (1<<UDRE0)));**

**UDR0 = data;**

**}**

**void USART\_putstring(char\* StringPtr)**

**{**

**while(\*StringPtr != 0x00){**

**USART\_send(\*StringPtr);**

**StringPtr++;**

**}**

**USART\_send(13);**

**USART\_send(49);**

**USART\_send(48);**

**USART\_send(48);**

**USART\_send(13);**

**for(int j=57;j>47;j--)**

**{**

**for(int i=57;i>47;i--)**

**{**

**USART\_send(j);**

**USART\_send(i);**

**USART\_send(13);**

**}**

**}**

**\_delay\_ms(5000000);**

**}**

**ADC Conversion ATMega 328**

[**http://www.embedds.com/adc-on-atmega328-part-1/**](http://www.embedds.com/adc-on-atmega328-part-1/)

**##adc with ldr :**

**#define F\_CPU 16000000**

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

**#include <util/delay.h>**

**#define BUAD 19200**

**#define BUAD\_RATE\_CALC ((F\_CPU/(16\*BUAD)) - 1)**

**//int main(void){**

**//**

**////High and low bits**

**//UBRR0H = (BUAD\_RATE\_CALC >> 8);**

**//UBRR0L = BUAD\_RATE\_CALC;**

**//////////////////**

**////transimit and recieve enable**

**//UCSR0B = (1 << TXEN0)| (1 << TXCIE0) | (1 << RXEN0) | (1 << RXCIE0);**

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

**//////////////////////////////////////////////////////////////////**

**//int counter = 100;**

**//DDRC =0xff;**

**//DDRB =0xff;**

**//while (1){**

**//send\_usart( counter);**

**//rcv\_usart();**

**//counter--;**

**//\_delay\_ms(5000);**

**//if(counter == 0){**

**//return 0;**

**//}**

**//}**

**//}**

**//void send\_usart( unsigned char data )**

**//{**

**//while (( UCSR0A & (1<<UDRE0))  == 0){};**

**//UDR0 = data;**

**//}**

**//**

**//void rcv\_usart()**

**//{**

**//while (!(UCSR0A & (1<<RXC0)) ){};**

**//**

**//PORTC = UDR0;**

**//}**

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

**#define PORT\_ON(port,pin) port |= (1<<pin)**

**#define PORT\_OFF(port,pin) port &= ~(1<<pin)**

**int main(void)**

**{**

**unsigned int adc\_value; // Variable to hold ADC result**

**DDRB=0xff; // Set Port D as Output**

**PORTB = 0x00;**

**ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);**

**// ADEN: Set to turn on ADC , by default it is turned off**

**//ADPS2: ADPS2 and ADPS0 set to make division factor 32**

**ADMUX=0b01000000; // ADC input channel set to PC5**

**while (1)**

**{**

**ADCSRA |= (1<<ADSC); // Start conversion**

**while (ADCSRA & (1<<ADSC)); // wait for conversion to complete**

**adc\_value = ADCW; //Store ADC value**

**if (adc\_value < 512)**

**{**

**PORT\_OFF(PORTB,5); // Toggle LEDs**

**PORT\_ON (PORTB,4);**

**}**

**else if(adc\_value > 512)**

**{**

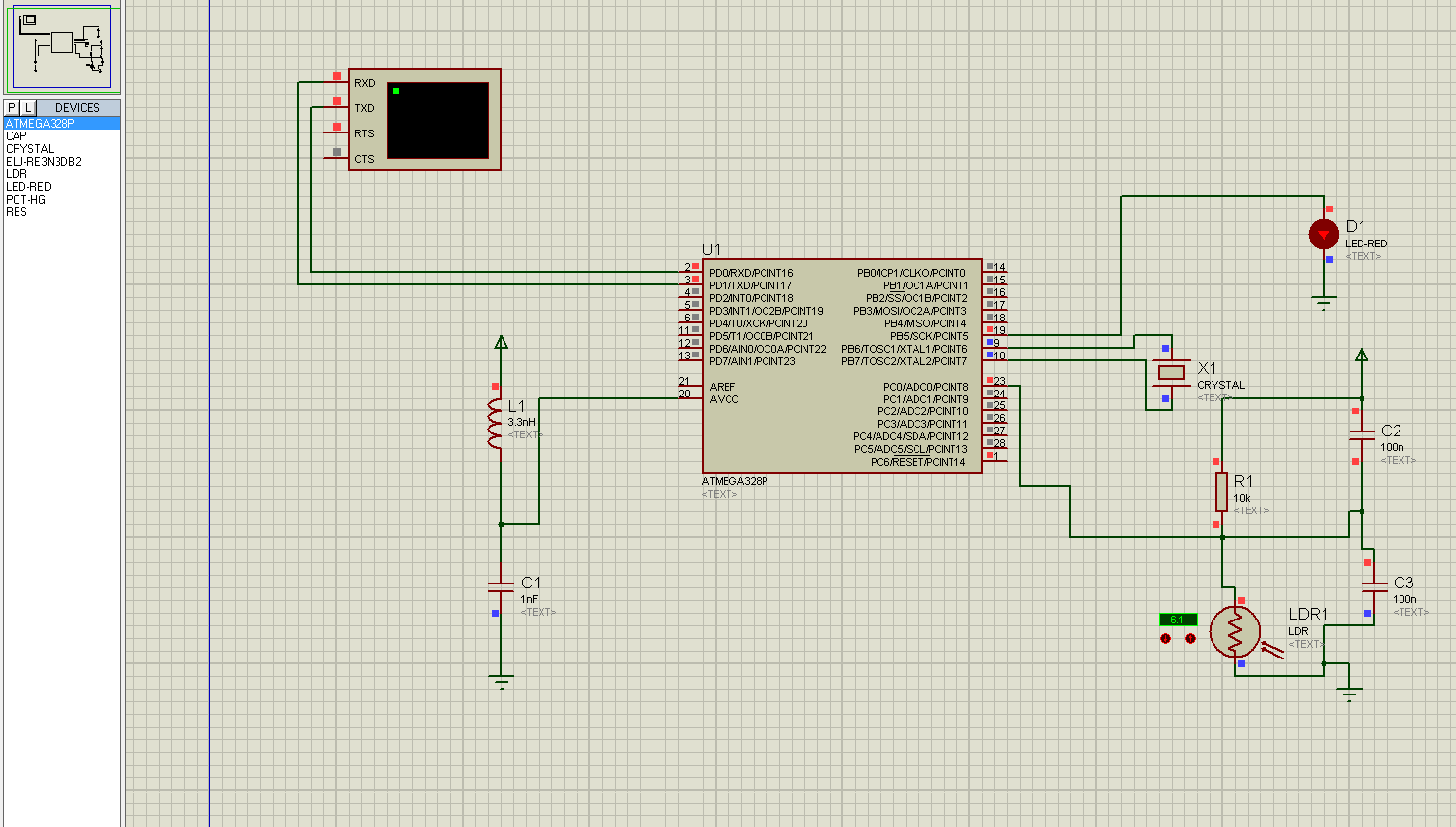
**PORT\_ON (PORTB,5); // Toggle LEDs**

**PORT\_OFF(PORTB,4);**

**}**

**}**

**}**

****

**Usart continuous data reader:**

**#define F\_CPU 16000000UL  // 16 MHz**

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

**#include <util/delay.h>**

**#include <avr/interrupt.h>**

**//#define PORT\_ON(port,pin) port |= (1<<pin)**

**//#define PORT\_OFF(port,pin) port &= ~(1<<pin)**

**//Define Baud**

**#define USART\_BAUDRATE 9600**

**#define BAUD\_PRESCALE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)**

**#pragma region Global Variables**

**volatile uint8\_t value = 0;// volatile so both main and RX interrupt can use it**

**volatile uint8\_t newData = 0;**

**volatile uint8\_t adcHData = 0;**

**volatile uint8\_t adcLData = 0;**

**#pragma endregion**

**#pragma region Function**

**ISR(USART\_RX\_vect)**

**{**

**value = UDR0;**

**newData = 1;**

**}**

**void USART\_Init(void)**

**{**

**// Set baud rate**

**UBRR0L = BAUD\_PRESCALE;//lower 8-bits into the the UBRRL register**

**UBRR0H = (BAUD\_PRESCALE >> 8); //upper 8-bits into the UBRRH register**

**UCSR0B = ((1<<TXEN0)|(1<<RXEN0) | (1<<RXCIE0));//enable tx and Rx. Receive interrupt**

**/\* Default frame format is 8 data bits, no parity, 1 stop bit to change use UCSRC, see AVR datasheet\*/**

**}**

**void USART\_SendByte(uint8\_t data){**

**// Wait until last byte has been transmitted**

**while((UCSR0A &(1<<UDRE0)) == 0);**

**// Transmit data**

**UDR0 = data;**

**}**

**void USART\_Sends (const char \*s)**

**{**

**do**

**{**

**USART\_SendByte (\*s);        //send one char of string**

**}**

**while (\*s++);            //next char of string until end of string sign**

**}**

**void USART\_Sendi(const uint16\_t var) {**

**char cache[7];**

**USART\_Sends (utoa(var, cache, 10));**

**}**

**void InitADC()**

**{**

**// Select Vref=AVcc**

**ADMUX |= (1<<REFS0);**

**//set prescaller to 128 and enable ADC**

**ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)|(1<<ADEN);**

**}**

**uint16\_t ReadADC(uint8\_t ADCchannel)**

**{**

**//select ADC channel with safety mask**

**ADMUX = (ADMUX & 0xF0) | (ADCchannel & 0x0F);**

**//single conversion mode**

**ADCSRA |= (1<<ADSC);**

**// wait until ADC conversion is complete**

**while( ADCSRA & (1<<ADSC) );**

**adcHData = ADCH;**

**adcLData = ADCL;**

**return ADC;**

**}**

**void LEDInit(void){**

**DDRB = 0x20;//B5 Output**

**PORTB = 0x00;//All off**

**}**

**#pragma endregion**

**int main(void)**

**{**

**LEDInit();//Init LED**

**//initialize ADC**

**InitADC();**

**uint16\_t adc\_value; // Variable to hold ADC result**

**USART\_Init();**

**sei();**

**#pragma region USART Send "IOT-Army"**

**USART\_SendByte(0x49);**

**USART\_SendByte('O');**

**USART\_SendByte(0b01010100);**

**USART\_SendByte(45);**

**USART\_SendByte(0x41);**

**USART\_SendByte('R');**

**USART\_SendByte(0x4D);**

**USART\_SendByte(0b01011001);**

**#pragma endregion**

**while (1)**

**{**

**adc\_value = ReadADC(0);**

**//uint8\_t adcH = adc\_value >> 8;**

**//uint8\_t adcL = adc\_value & 0x0F;**

**//adc\_value = ADCW; //Store ADC value**

**if (ADC < 512)**

**{**

**PORTB|=(1<<5); // Toggle LEDs**

**PORTB&=~ (1<<4);**

**}**

**else**

**{**

**PORTB |= (1<<5); // Toggle LEDs**

**PORTB&=~(1<<4);**

**}**

**//USART\_SendByte(ADCL);**

**#pragma region Send Adc Data through USART**

**USART\_SendByte(0x0D);**

**USART\_SendByte(0x0A);**

**USART\_Sendi(adc\_value);**

**#pragma endregion**

**\_delay\_ms(200);**

**}**

**}**