Serial Comm

/\*

\* comm\_myk\_13apr.c

\*

\* Created: 13-04-2023 13:22:01

\* Author : DSP LAB

\*/

#define *F\_CPU* 8000000UL // 8 MHz

/\*Very Important - change F\_CPU to match target clock

Note: default AVR CLKSEL is 1MHz internal RC

This program transmits continously on USART. Interrupt is used for

Receive charactor, which is then transmitted instead. LEDs are used

as a test. Normal RX routine is included but not used.

Change USART\_BAUDRATE constant to change Baud Rate

\*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

// Define baud rate

#define USART\_BAUDRATE 9600

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

volatile unsigned char value;

/\* This variable is volatile so both main and RX interrupt can use it.

It could also be a uint8\_t type \*/

/\* Interrupt Service Routine for Receive Complete

NOTE: vector name changes with different AVRs see AVRStudio -

Help - AVR-Libc reference - Library Reference - <avr/interrupt.h>: Interrupts

for vector names other than USART\_RXC\_vect for ATmega32 \*/

ISR(USART\_RXC\_vect){

value = UDR; //read UART register into value

PORTB = ~value; // output inverted value on LEDs (0=on)

}

void USART\_Init(void){

// Set baud rate

UBRRL = BAUD\_PRESCALE;// Load lower 8-bits into the low byte of the UBRR register

UBRRH = (BAUD\_PRESCALE >> 8);

/\* Load upper 8-bits into the high byte of the UBRR register

Default frame format is 8 data bits, no parity, 1 stop bit

to change use UCSRC, see AVR datasheet\*/

// Enable receiver and transmitter and receive complete interrupt

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

}

void USART\_SendByte(*uint8\_t* u8Data){

// Wait until last byte has been transmitted

while((UCSRA &(1<<UDRE)) == 0);

// Transmit data

UDR = u8Data;

}

// not being used but here for completeness

// Wait until a byte has been received and return received data

*uint8\_t* USART\_ReceiveByte(){

while((UCSRA &(1<<RXC)) == 0);

return UDR;

}

void Led\_init(void){

//outputs, all off

DDRB =0xFF;

PORTB = 0xFF;

}

int main(void)

{

int i;

USART\_Init(); // Initialise USART

sei(); // enable all interrupts

Led\_init(); // init LEDs for testing

//value = 'A'; //0x41;

//PORTB = ~value; // 0 = LED on

char value[]={'w','e','l','c','o','m','e',' ','t','o',' ','s','s','g','m','c','e'};

for(i=0;i<=16;i++)

{ // Repeat indefinitely

// value= USART\_ReceiveByte();

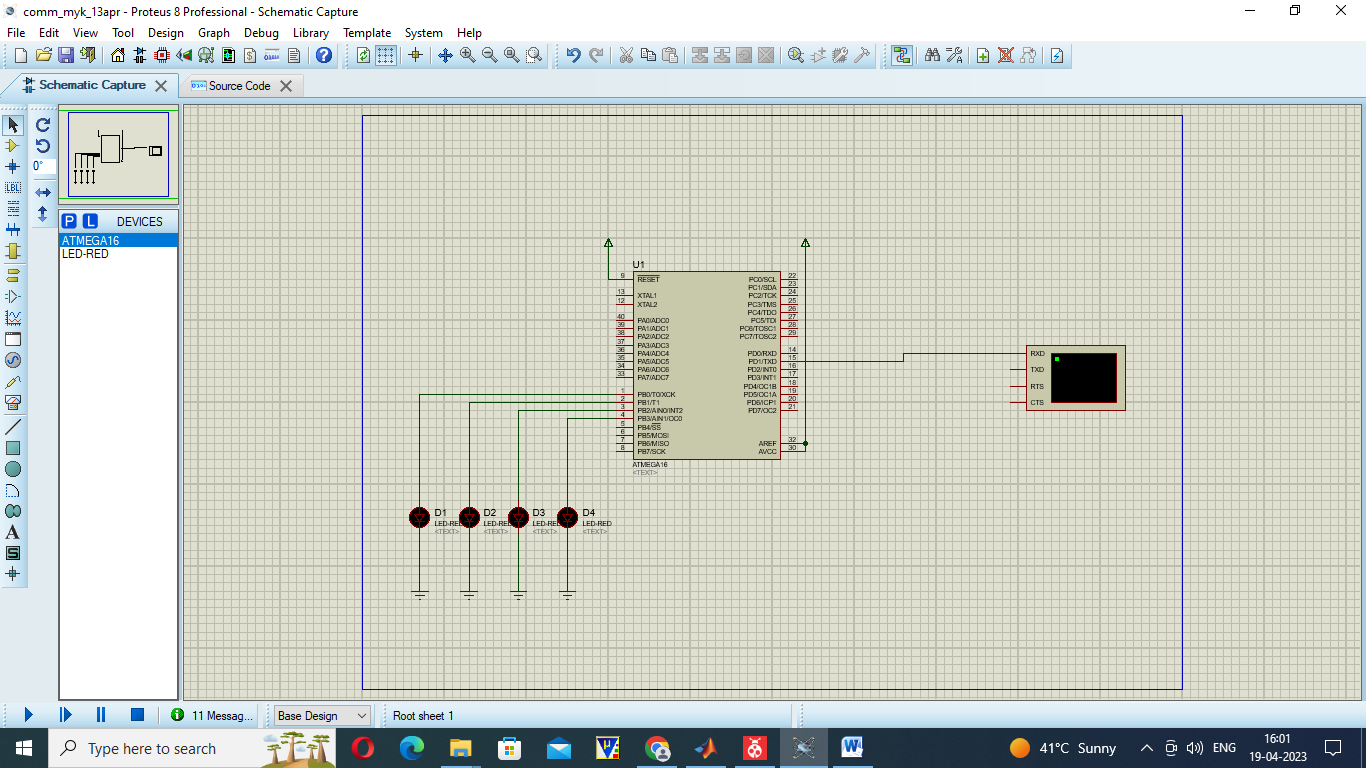
USART\_SendByte(value[i]); // send value

*\_delay\_ms*(250);

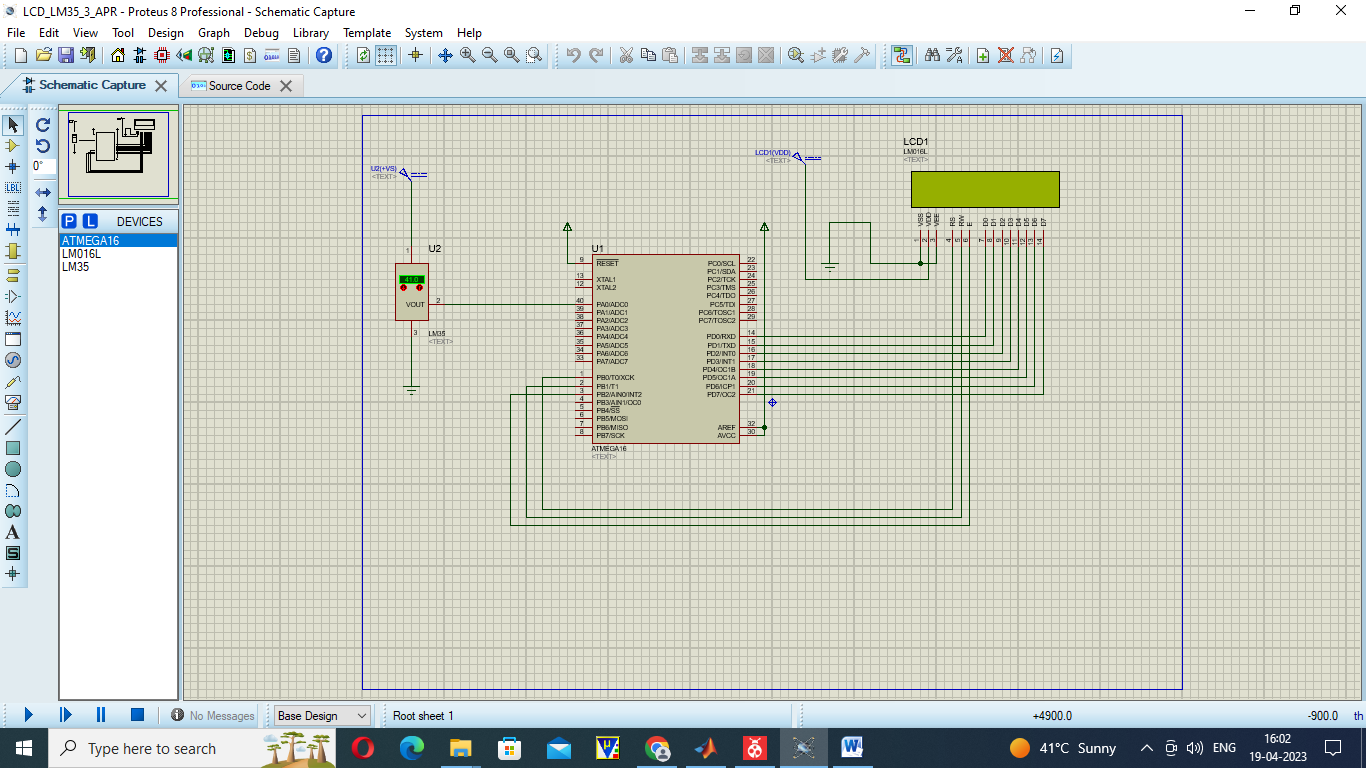
// delay just to stop Hyperterminal screen cluttering up

}

}



LM35 LCD



/\*

\* comm\_myk\_13apr.c

\*

\* Created: 13-04-2023 13:22:01

\* Author : DSP LAB

\*/

#define *F\_CPU* 8000000UL // 8 MHz

/\*Very Important - change F\_CPU to match target clock

Note: default AVR CLKSEL is 1MHz internal RC

This program transmits continously on USART. Interrupt is used for

Receive charactor, which is then transmitted instead. LEDs are used

as a test. Normal RX routine is included but not used.

Change USART\_BAUDRATE constant to change Baud Rate

\*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

// Define baud rate

#define USART\_BAUDRATE 9600

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

volatile unsigned char value;

/\* This variable is volatile so both main and RX interrupt can use it.

It could also be a uint8\_t type \*/

/\* Interrupt Service Routine for Receive Complete

NOTE: vector name changes with different AVRs see AVRStudio -

Help - AVR-Libc reference - Library Reference - <avr/interrupt.h>: Interrupts

for vector names other than USART\_RXC\_vect for ATmega32 \*/

ISR(USART\_RXC\_vect){

value = UDR; //read UART register into value

PORTB = ~value; // output inverted value on LEDs (0=on)

}

void USART\_Init(void){

// Set baud rate

UBRRL = BAUD\_PRESCALE;// Load lower 8-bits into the low byte of the UBRR register

UBRRH = (BAUD\_PRESCALE >> 8);

/\* Load upper 8-bits into the high byte of the UBRR register

Default frame format is 8 data bits, no parity, 1 stop bit

to change use UCSRC, see AVR datasheet\*/

// Enable receiver and transmitter and receive complete interrupt

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

}

void USART\_SendByte(*uint8\_t* u8Data){

// Wait until last byte has been transmitted

while((UCSRA &(1<<UDRE)) == 0);

// Transmit data

UDR = u8Data;

}

// not being used but here for completeness

// Wait until a byte has been received and return received data

*uint8\_t* USART\_ReceiveByte(){

while((UCSRA &(1<<RXC)) == 0);

return UDR;

}

void Led\_init(void){

//outputs, all off

DDRB =0xFF;

PORTB = 0xFF;

}

int main(void)

{

int i;

USART\_Init(); // Initialise USART

sei(); // enable all interrupts

Led\_init(); // init LEDs for testing

//value = 'A'; //0x41;

//PORTB = ~value; // 0 = LED on

char value[]={'w','e','l','c','o','m','e',' ','t','o',' ','s','s','g','m','c','e'};

for(i=0;i<=16;i++)

{ // Repeat indefinitely

// value= USART\_ReceiveByte();

USART\_SendByte(value[i]); // send value

*\_delay\_ms*(250);

// delay just to stop Hyperterminal screen cluttering up

}

}