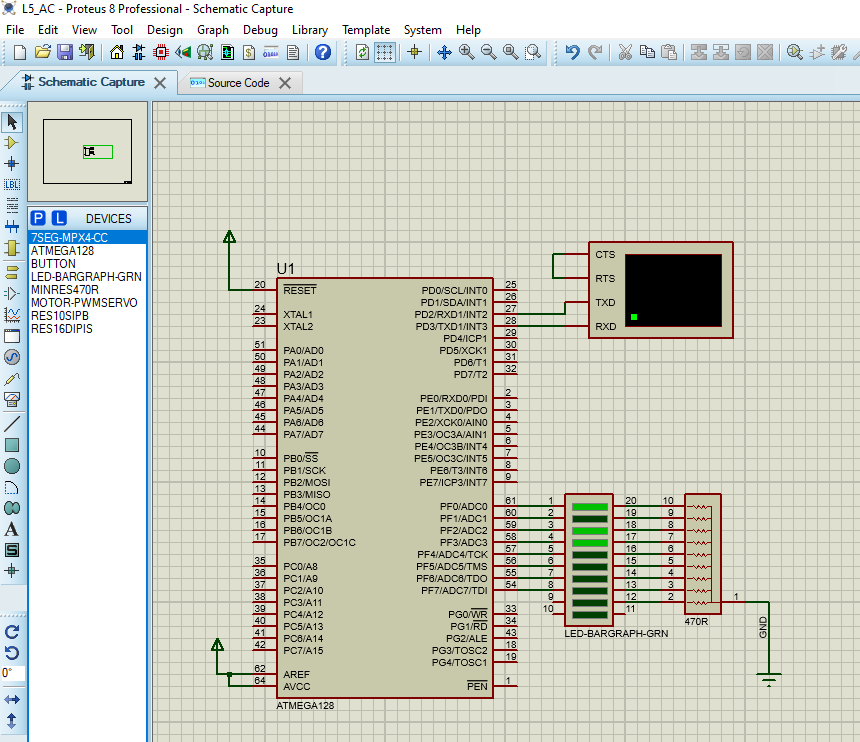
# Laboratorul nr. 5

## Obiective. În urma efectuării lucrării de laborator se învață:

* desenarea circuitelor utilizând programul Proteus;
* plasarea componentelor pe foaia de desenare;
* editarea componentelor (nume, valoare și simbol);
* trasarea liniilor de conexiuni;
* configurarea / accesarea portului D, E, F a microcontroler-lui ATmega128
* configurarea / utilizarea modulului / interfetei seriale asincrone USART1 (modul de interogare ciclica a bit-lui de stare – pooling method )

### **Tema 1**

Să se deseneze cu ajutorul programului *Proteus* circuitul din fig. L1-1.



**Modul de lucru :**

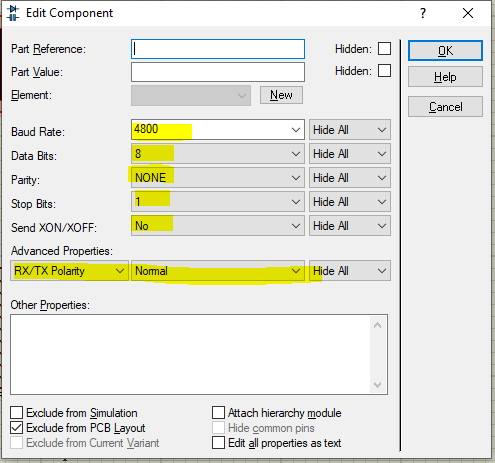
**Pașii pentru lansarea programului *Proteus, crearea proiectului, plasarea componentelor pe planșa de lucru au fost prezentați in cadrul laboratorului cu numărul 1.***

## Adăugarea terminalului virtual pe planșa de lucru:

## 

## Ajustarea parametrilor terminalului virtual:

Poziționam mouse-ul deasupra componentei pe care dorim sa o re-parametrizam, apăsam tastele CTRL+E.



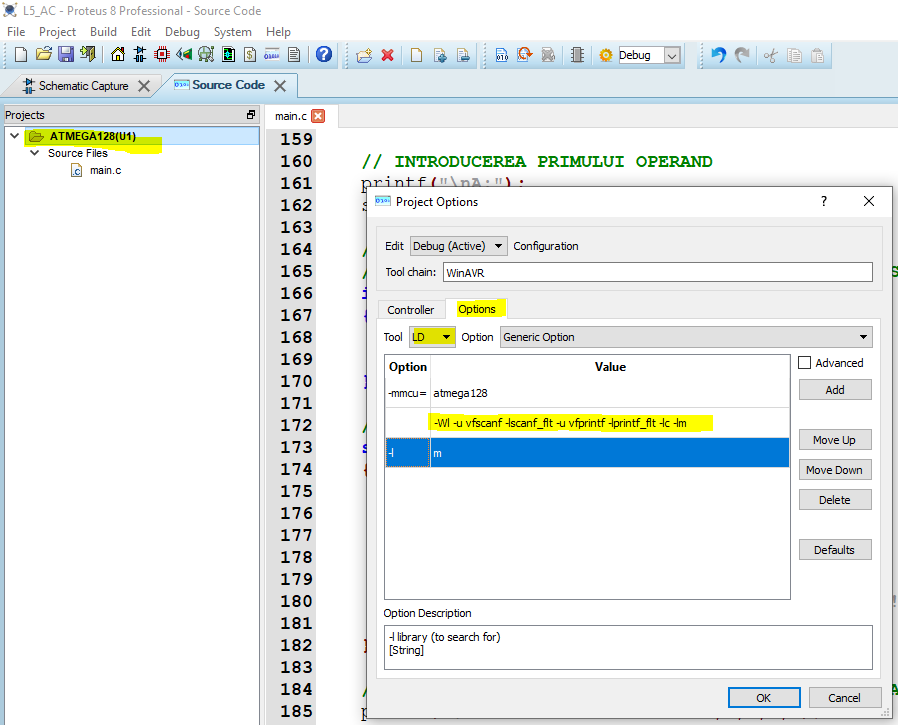
Următorii pașii necesari au fost descriși in cadrul laboratorului cu numărul 1

## Interconectarea componentelor.

## Rularea / oprirea aplicației de tip firmware:

## Modificarea codului sursa (Source Code)

Configurarea compilatorului (linker-lui) pentru a o feri suport pentru printf / scanf formatul standard de lucru.



Opțiunea pentru LD: **-Wl -u vfscanf -lscanf\_flt -u vfprintf -lprintf\_flt -lc -lm**

Pornind de la următorul cod sursa rezolvați următoarele doua teme (codul poate fi descărcat de pe platforma elearning):

#include <stdio.h>

#include <ctype.h>

//#include <inttypes.h>

#include <string.h>

#include <stddef.h>

#include <stdlib.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/sleep.h>

#include <math.h>

// TESTAREA VALORII BITULUI DE PE POZITIA [7..0]

// RETURNEAZA 0 DACA BITUL DE PE POZITIA INDICATA ESTE 0

// RETURNEAZA O VALOARE DIFERITA DE 0 DACA BITUL DE PE POZITIA INDICATA ESTE 1

#define TBIT(var,pos) ((var) &   (1<<(pos)))

// SETAREA IN 1 A BITULUI DE PE POZITIA [7..0]

#define SBIT(var,pos) ((var) |=  (1<<(pos)))

// SETAREA IN 0 A BITULUI DE PE POZITIA [7..0]

#define CBIT(var,pos) ((var) &= ~(1<<(pos)))

// NEGAREA BITULUI DE PE POZITIA [7..0]

#define NBIT(var,pos) ((var) ^=  (1<<(pos)))

// DEFINIREA RUTINELOR PENTRU COMUNICATIA SERIALA ASINCRONA

// 'uart1\_putchar' -> TRANSMISIE DATE DE PE INTERFATA USART

// 'uart1\_getchar' <- RECEPTIE DATE DE PE INTERFATA USART

static int uart1\_putchar(char c);

static int uart1\_getchar(void);

// DEFINIREA DESCRIPTORUL ATASAT FLUXURILOR STANDARD DE COMUNICATIE

// INTERCONECTAREA FLUXURILOR STANDARD DE COMUNICATIE LA RUTINELE

// CARE IMPLEMENTEAZA / GESTIONEAZA COMUNICATIA SERIALA ASINCRONA

// CELE TREI CONEXIUNI DE INTRARE / IESIRE (I / O) SE NUMESC:

//  - INTRARE STANDARD ( STDIN )

//  - IESIRE  STANDARD ( STDOUT )

//  - EROARE  STANDARD ( STDERR )

static FILE strm\_uart1 = FDEV\_SETUP\_STREAM(uart1\_putchar, uart1\_getchar, \_FDEV\_SETUP\_RW);

// == TEMA2 ==

// INITIALIZAREA MODULULUI UART1, CERINTE:

// IDENTIFICAREA REGISTRILOR RESPONSABILI CU, COMUNICATIA PE INTERFATA USART1

// ACTIVARE MODULUI DE LUCRU: DOAR RECEPTIE TRANSMISIE DATE  (REG.: UCSR1B - pg. 189)

// ( MECANISMUL DE COMUNICATIE FARA INTRERUPERE - 'POOLING METHOD')

// BAUD RATE:4800 (BIT U2X=0, FOSC=1.00MHZ, REG.: UBRR1L, TABEL: 82)

// FORMULA DE CALCUL A BAUD RATE-LUI (TABEL: 74)

// DIMENSIUNE PACHET DATA: 8 BIT (TABEL: 80)

// COMUNICATIE SERIALA ASINCRONA: (REG.: UCSR1C  TABEL: 77)

// FARA PARITATE (REG.: UCSR1C  TABEL: 78)

// 1 BIT STOP: (REG.: UCSR1C  TABEL: 79)

// CONTROL TRASFER DATE: FARA CONTROL (SW/HW)

void uart1\_init(void)

{

 UCSR1B = 0x00; //DEZACTIVAREA INTERFETEI SERIALE PENTRU CONFIGURAREA ACESTEIA

 UCSR1A = 0x00;

 UCSR1C = 0x00;

 UBRR1L = 0x00;

 UBRR1H = 0x00;

 UCSR1B = 0x00;

}

// == TEMA3A ==

// IMPLEMENTAREA FUNCTIONALITATII RUTINEI 'uart1\_putchar', CERINTE:

// IDENTIFICAREA REGISTRULUI RESPONSABIL CU RECEPTIA / TRANSMISIA PE INTERFATA USART1 (PG. 188)

// IDENTIFICAREA REGISTRULUI DE STAREA PENTRU INTERFATA USART1 (PG. 188)

// DACA BYTE-UL (CARACTERUL) DE TRANSMIS ESTE CARRIAGE RETURN '\n' = 0x0A,

// PRIN APELARE RECURSIV A ACESTE RUTINE SA SE TRANSMITA BYTE-UL LINE FEED '\r' = 0x0D

//

// TRANSMITEREA BYTE-LUI (CARACTER-LUI) PE SERIALA SE VA REALIZA DUPA CE IN PREALABIL

// S-A TESTAT CA SERIALA ESTE DISPONIBILA PENTRU A REALIZA ACEST LUCRU (METODA POOLING)

// IN CAZ CONTRAR SE VA ASTEPTA PANA CAND ACEASTA INTERFATA DEVINE DISPONIBILA (VEZI PG. 177)

static int uart1\_putchar(char c)

{

  // TO DO

  return 0;

}

// == TEMA3B ==

// IMPLEMENTAREA FUNCTIONALITATII RUTINEI 'uart1\_getchar', CERINTE:

//

// PRELUAREA BYTE-LUI DIN REGISTRUL ALOCAT INTERFETEI SERIALE UART1 SE VA REALIZA DUPA CE IN PREALABIL

// S-A TESTAT CA PE INTERFATA SERIALA ESTE DISPONIBIL BYTE-UL RECEPTIONAT

// IN CAZ CONTRAR SE VA ASTEPTA PANA CAND BYTE-UL RECEPTIONAT DEVINE DISPONIBIL (VEZI PG. 180)

// PORTUL 'F' SE VA SETA CU VALOAREA BYTE-UL RECEPTIONAT

// FUNCTIA 'uart1\_getchar' VA RETURNA BYTE-UL RECEPTIONAT

static int uart1\_getchar(void)

{

  // TO DO

}

int main()

 {

  // CONTOR OPERATII REALIZATE

  int i=1;

  // DECLARAREA / INITIALIZAREA VARIABILELOR NECESARE EXECUTARII DIFERITELOR OPERATII ARITMETICE

  double a=12.34f, b=56.78f, r=0.0f;

  char o='+';

  // DEZACTIVAREA TUTUROR INTRERUPERILOR PENTRU PROGRAMAREA INTERFETELOR

  // ( ARE ROLUL DE A PREVINE PROGRAMAREA DEFECTUOASA SAU PARTIALA A MODULELOR / INTERFETELOR )

  // SE REALIZEAZA PRIN SETAREA IN 0 A BITULUI 7 IN CADRUL REGISTRULUI SREG

  // A BITUL 7 DIN SREG ESTE RESPONSABIL CU ACTIVAREA / DEZACTIVAREA GLOBALA A INTRERUPERILOR

  cli();

  // APELAREA RUTINEI RESPONSABILE CU INITIALIZAREA INTERFETEI SERIALE ASINCRONE USAR1

  uart1\_init();

  // ASIGNAREA PRIN REFERINTA A DESCRIPTORUL ATASAT LA FLUXURILE STANDARD DE COMUNICATIE

  //

  // ACCESAREA FLUXURILOR DE IESIRE  STANDARD ( STDOUT ) SAU EROARE  STANDARD ( STDERR )

  // VA DETRMINA APELAREA RUTINEI 'uart1\_putchar' (DE EX. putchar(), puts(), printf())

  //

  // ACCESAREA FLUXULUI DE INTRARE  STANDARD ( STDIN )

  // VA DETRMINA APELAREA RUTINEI 'uart1\_getchar' (DE EX. getchar(), gets(), scanf())

  stdout = stdin = stderr = &strm\_uart1;

  // SETAREA PORTULUI 'F' CA PORT DE IESIRE ( COMANDA LEDURILOR BARGRAPHULUI ATASAT uCONTROLERULUI )

  // PE PORTUL 'F' SE VA AFISA CODUL ASCII AL ULTIMEI TASTE APASATE IN TERMINALUL VIRTUAL

  DDRF=0xFF;

  // ACTIVAREA TUTUROR INTRERUPERILOR (SETAREA IN 1 IN CADRUL REGISTRULUI SREG

  // A BITULUI RESPONSABIL CU ACTIVAREA / DEZACTIVAREA GLOBALA A INTRERUPERILOR

  sei();

  // DIFERITE MODURI DE SCRIERE LA CONSOLA

  // CARACTER CU CARACTER / STRING

  // TOATE VARIANTELE DE MAI JOS VOR APELA RUTINA 'uart1\_putchar'

  uart1\_putchar('A');

  uart1\_putchar('T');

  uart1\_putchar('M');

  putchar('e');

  putchar('g');

  putchar('a');

  printf(" Calculator!");

  // BUCLA INFINITA

  while (1)

   {

    // MESAJ UTIL PENTRU DEPANAREA PROGRAMULUI

    printf("\n\nMath op.: %05u   ",i++);

    // SELECTAREA OPERATIEI ARITMETICE DORITA

    //  CODUL ASCII PENTRU: '/'=47, CODUL ASCII PENTRU '^'=0x5E

    printf("\nOP (+,-,\*,%c,%c):", 47, 0x5E);

    do

    {

      o=uart1\_getchar();

      // SAU VARIANTA GETCHAR,

      // o=getchar();

    }

    // (TRECEREA LA PASUL URMATOR ESTE CONDITIONATA DE SELECTAREA UNUIA DIN OPERATORII: +,-,\*,/,^ )

    while(! (o =='+' || o =='-' || o=='\*' || o=='/' || o=='^'));

    // INTRODUCEREA PRIMULUI OPERAND

    printf("\nA:");

    scanf("%lf",&a);

    // INTRODUCEREA CELUI DE-AL  OPERAND

    // DACA OPERATIA SELECTATA ESTE '^', ACEST PAS ESTE SARIT

    if(o!='^')

    {

      printf("\nB:");

      scanf("%lf",&b);

    }

    // IDENTIFICAREA OPERATIEI ARITMETICE SOLICITATA

    switch(o)

    {

        case '+': r=a+b; break;

        case '-': r=a-b; break;

        case '\*': r=a\*b; break;

        case '/': r=a/b; break;

        case '^': r=pow(a, 0.5); b=0.5; break;

        default: printf("\n %c unsupported operation!!",o);

    }

    // AFISARE REZULTAT / OPERATIE ARITMETICA EXECUTATA

    printf("\n%.5lf %c %.5lf=%.5lf ",a,o,b,r);

   }

 }

## Tema 2 (vezi codul sursa de mai sus, secțiunea // = TEMA 2 =  )

**== TEMA2 ==**

Initializarea modulului UART1, cerinte:

* identificarea registrilor responsabili cu, comunicatia pe interfata USART1
* activare modului de lucru: doar receptie transmisie date (reg.: UCSR1B - pg. 189)

( mecanismul de comunicatie fara intrerupere - 'pooling method')

* BAUD RATE:4800 (bit U2X=0, FOSC=1.00MHZ, reg.: UBRR1L, TABEL: 82)

formula de calcul a baud rate-lui (tabel: 74)

* dimensiunea pachetului de date: 8 bit (tabel: 80)
* comunicatie seriala asincrona: (reg.: UCSR1C tabel: 77)
* fara paritate (reg.: UCSR1C tabel: 78)
* 1 bit de STOP: (reg.: UCSR1C tabel: 79)
* control trasfer date: fara control (SW/HW)

## **Tema 3 (**vezi codul sursa de mai sus, secțiunea // = TEMA 3 =  )

**== TEMA3A ==**

* implementarea funcționalității rutinei 'uart1\_putchar', cerințe:
* identificarea registrului responsabil cu recepția / transmisia pe interfața USART1 (pg. 188)
* identificarea registrului de starea pentru interfața USART1 (pg. 188)
* daca byte-ul (caracterul) de transmis este CARRIAGE RETURN '\n' = 0x0A,

prin apelare recursiv a aceste rutine sa se transmită byte-ul LINE FEED '\r' = 0x0D

* transmiterea byte-lui (caracter-lui) pe seriala se va realiza după ce in prealabil s-a testat ca seriala este disponibila pentru a realiza acest lucru (metoda poloni) in caz contrar se va aștepta pana când aceasta interfața devine disponibila (vezi pg. 177)

**== TEMA3B ==**

* implementarea funcționalității rutinei 'uart1\_getchar', cerințe:
* preluarea byte-lui din registrul alocat interfeței seriale uart1 se va realiza după ce in prealabil s-a testat ca pe interfața seriala este disponibil byte-ul recepționat in caz contrar se va aștepta pana când byte-ul recepționat devine disponibil (vezi pg. 180)
* portul 'F' se va seta cu valoarea byte-ul recepționat funcția 'uart1\_getchar' va returna byte-ul recepționat

**IMPORTANT**

**Buna practică inginerească cere ca desenul să fie foarte clar,**

**să nu existe suprapuneri între înscrisuri şi elementele de circuit.**

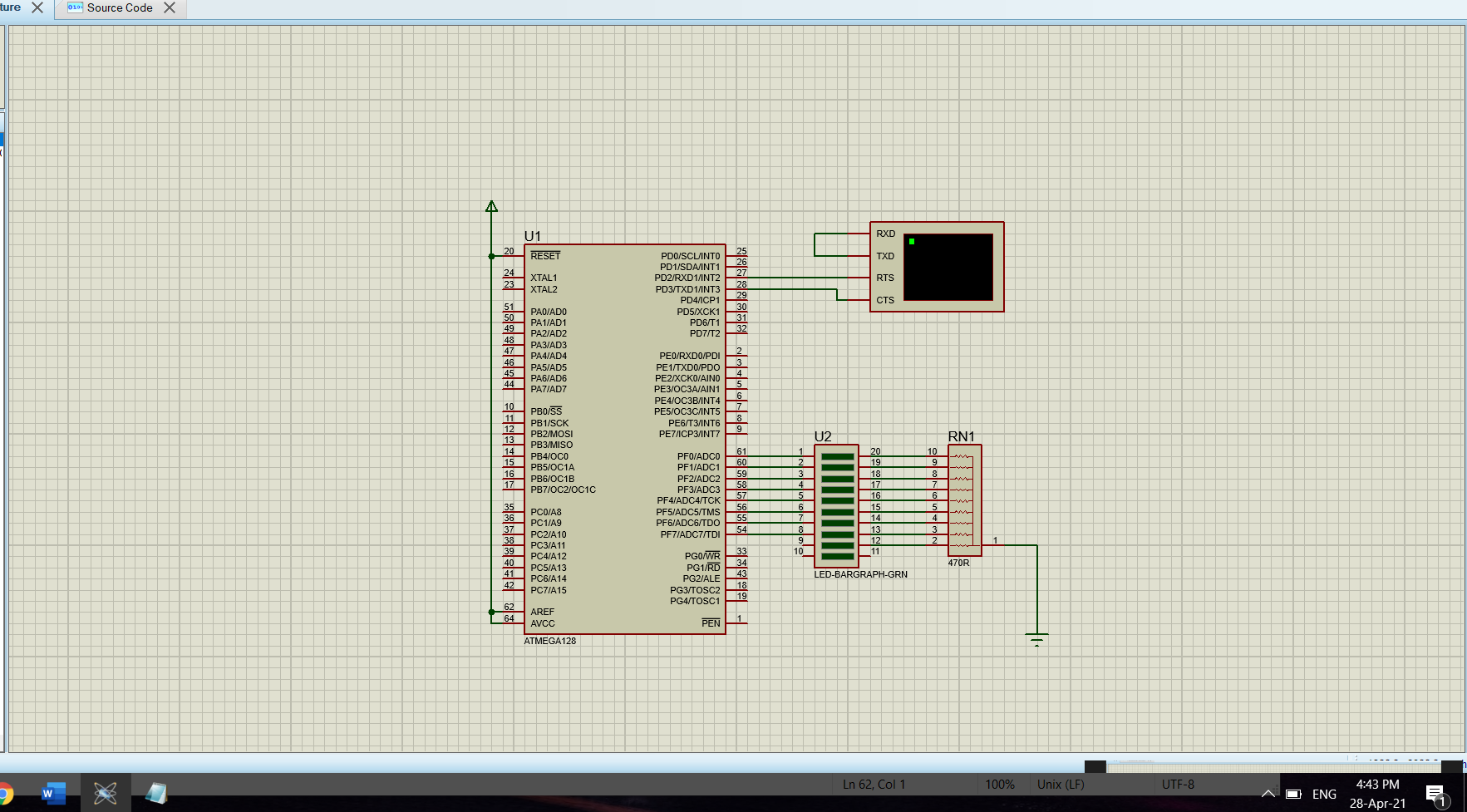
**Toate înscrisurile (nume, valori, parametri) se deplasează până când se văd clar atât componentele cât şi înscrisurile.**

## Cerințe

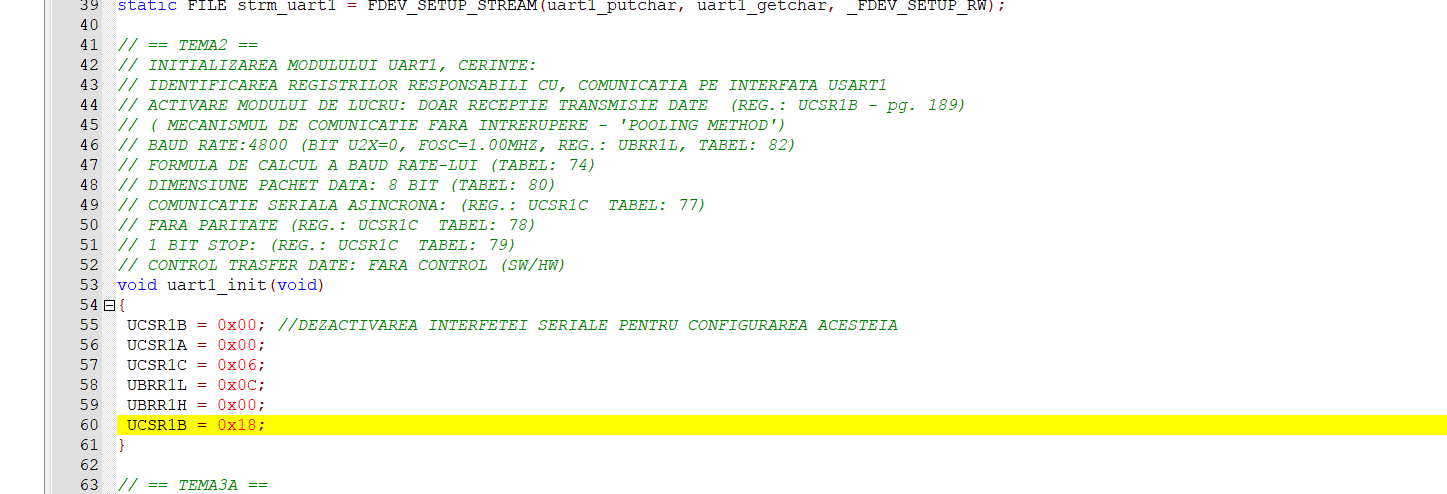
Lucrarea trebuie sa cuprindă rezolvarea temelor:

( **funcționalitatea descrisa mai sus poate fi vizualizata in videoclipul L4\_AC.mp4** )

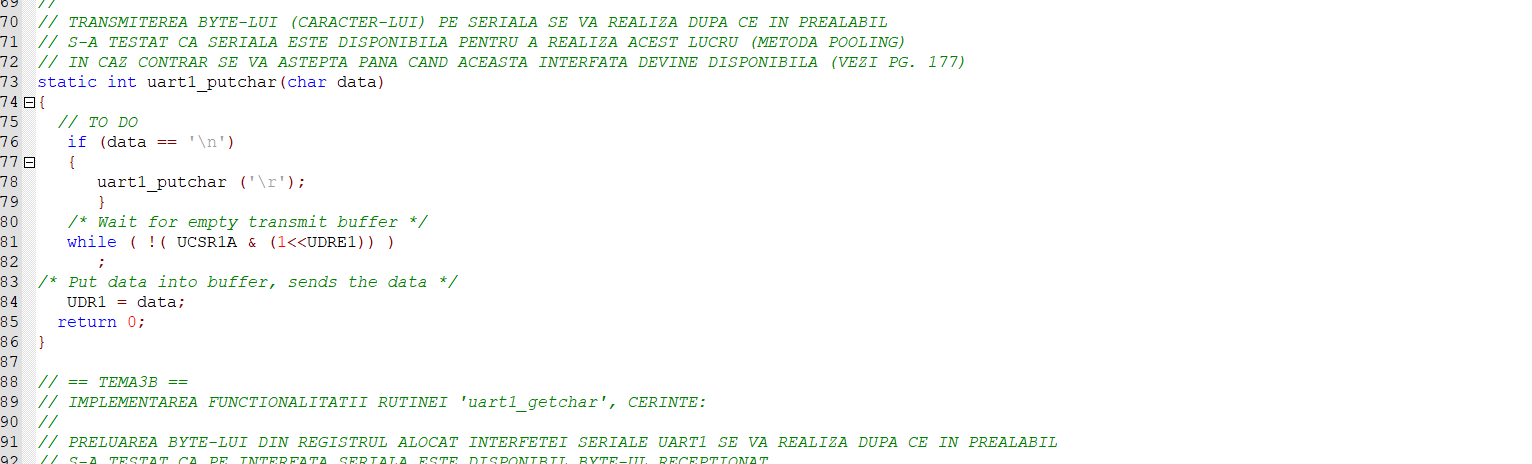
* Rezolvarea temei 1;

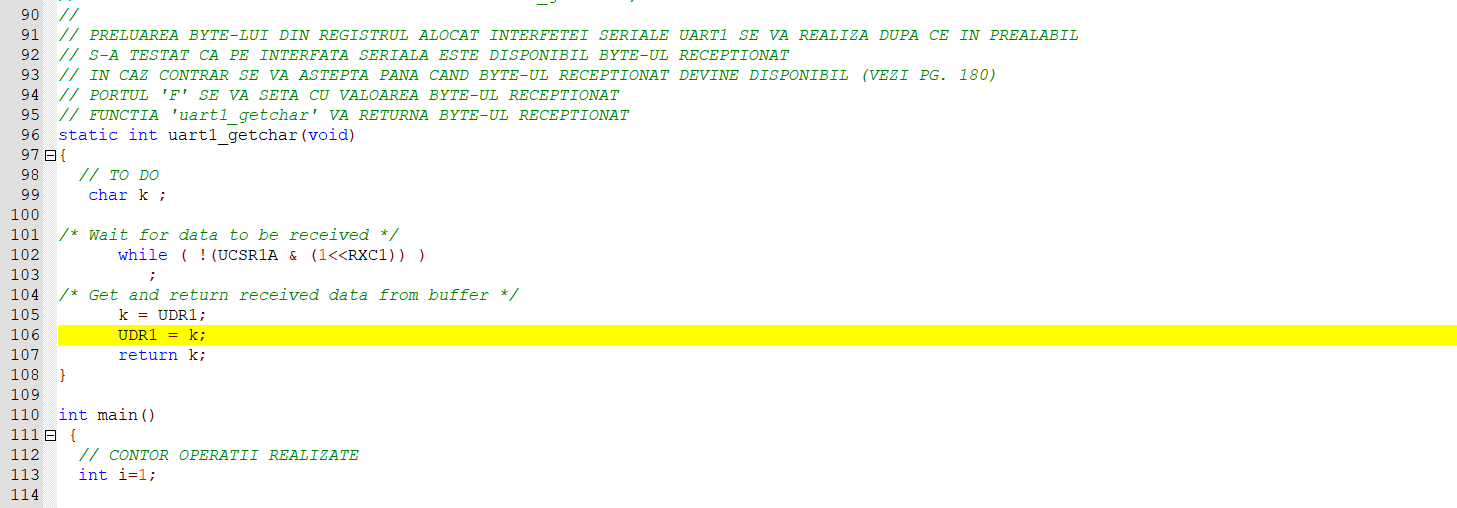


* Rezolvarea temei 2;



* Rezolvarea temei 3;



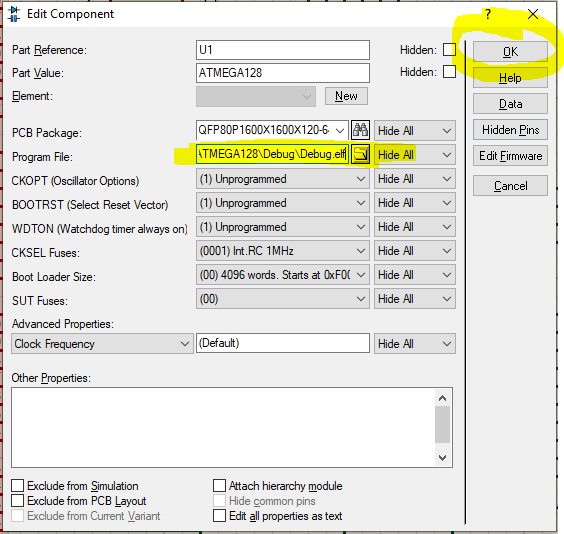


## CODUL HEX DEMO:

Încărcarea codului HEX DEMO presupune executarea următorilor pași:

* salvarea într-un fișier cu extensia hex a codului in format Intel HEX (<https://en.wikipedia.org/wiki/Intel_HEX>) de mai jos.
* încărcarea in microcontroler a codului hex ( poziționam mouse-ul deasupra microcontrolerului,

apăsam tastele CTRL+E, selectam fișierul cu extensia .hex salvat )



Codul HEX DEMO poate fi descarcat de pe platforma: https://elearning.unitbv.ro/