Računske vežbe iz Projektovanja Elektronskih Sistema

Doc.dr Borisav Jovanović

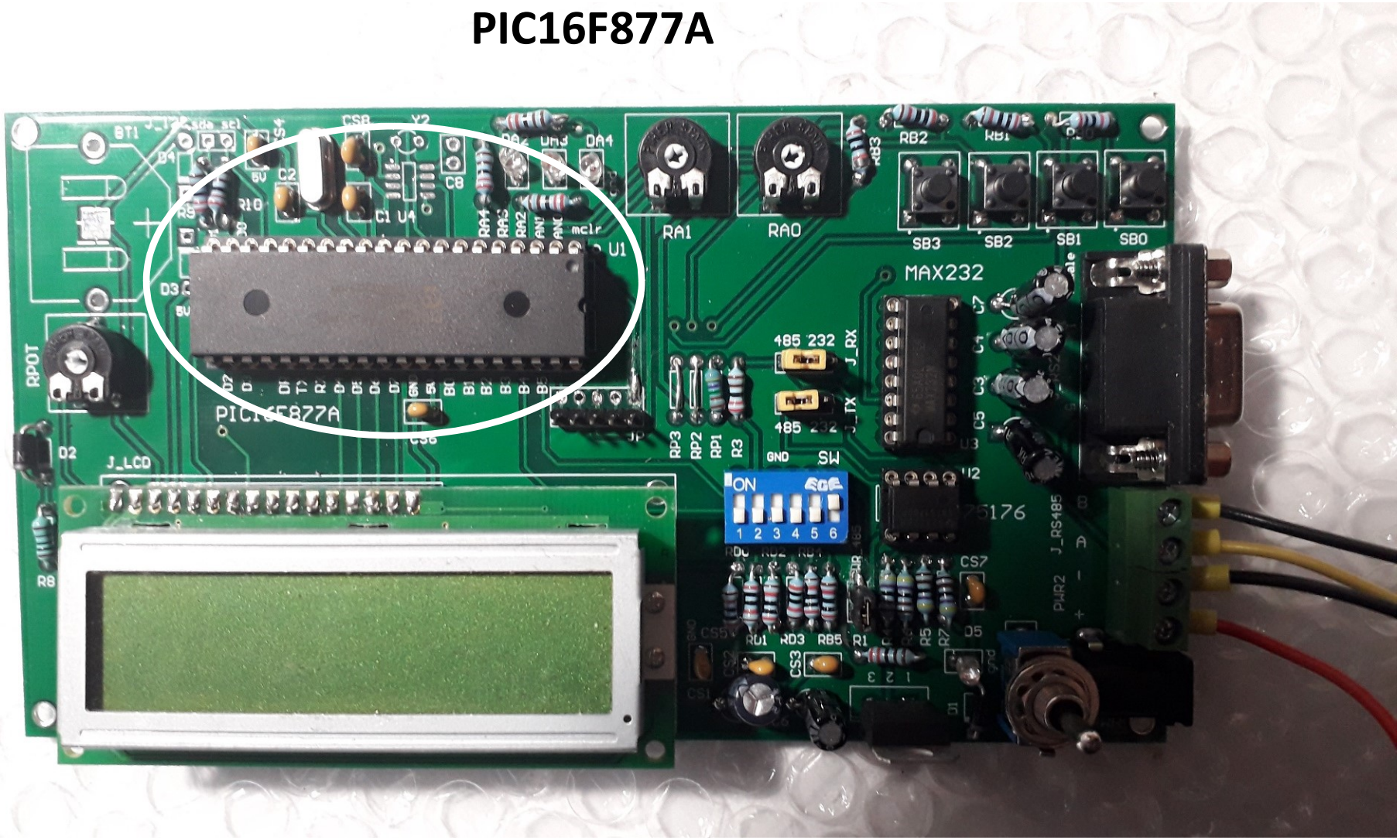
- **Ocenu iz predmeta Projektovanja Elektronskih Sistema čine:**

- **Kompletnost I originalnost uradjenog projekta.** (max. 15 poena)
- **Seminarski rad** (max. 15 poena) ***(treba da bude originanalan, kompletan, lak za čitanje, koncizan. Pisan tehnički i gramatički ispravno).***
- Odbrana praktičnog rada – pismeni deo ispita (max. 45 poena)
- Teorijski deo (max. 30 poena)

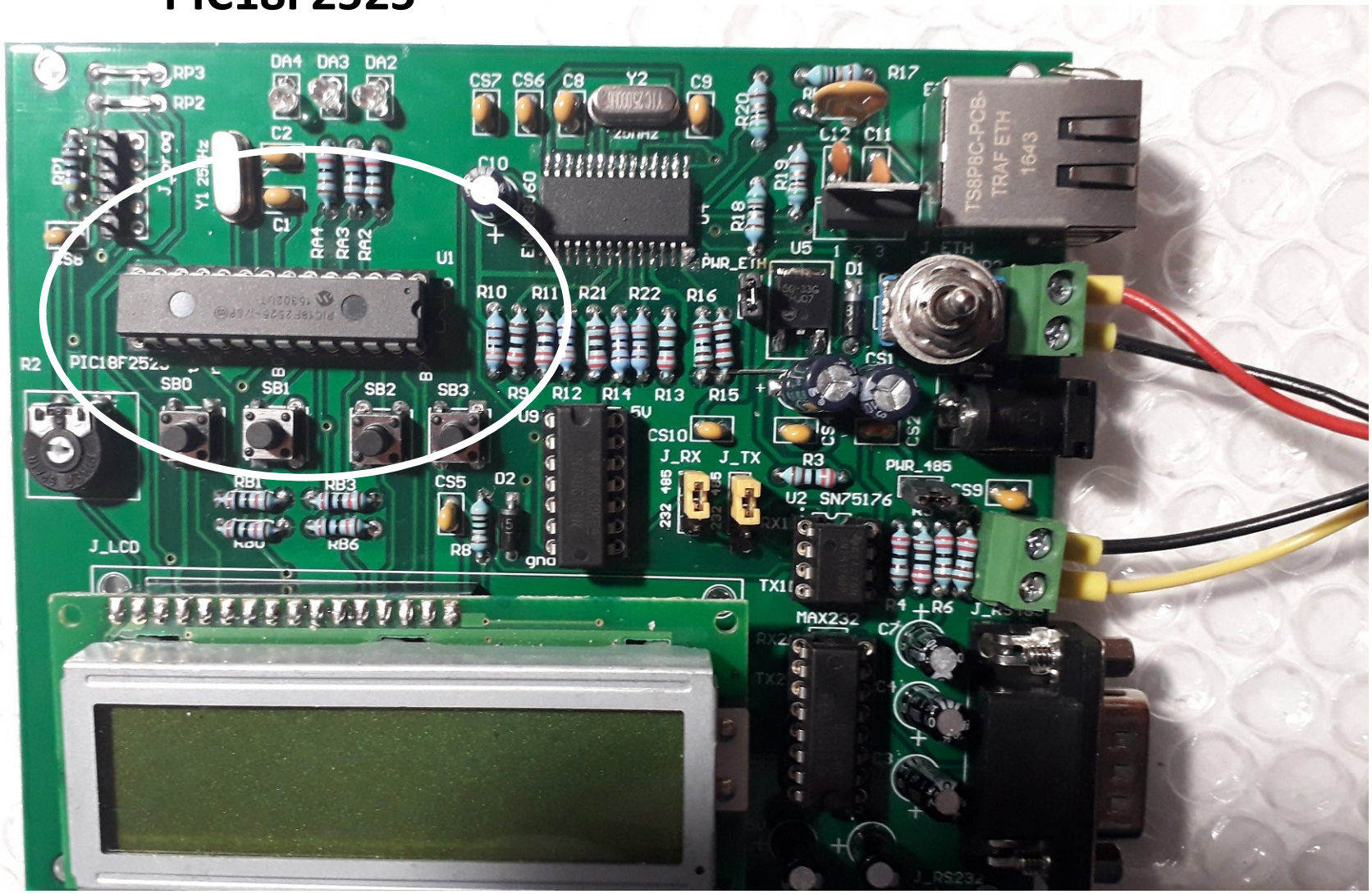
- Odbrana praktičnog rada sastoji se iz zadataka iz oblasti projekta koji je rađen na časovima (računske i laboratorijske vežbe).
- Praktični rad se brani zajedno sa teorijskim delom onog dana kada je zvanično ispit zakazan.

- U projektima studenti koriste napravljene razvojne sisteme bazirane na PIC mikrokontrolerima.
- Svaki projekat ima dva razvojna sistema, tj. dve štampane ploče - Master i Slejv.

PIC16F877A


PIC16F877A
MAX232
RA1
RA0
SB3
SB2
SB1
SB0
RB2
RB1
J_LCD
RPOT
SW
ON
1 2 3 4 5 6
J_RX
J_TX
485 232
GND

PIC18F2525


RP3
RP2
DA4
DA3
DA2
CS7
CS6
C8
Y2
C9
R17
C12
C11
C10
C2
C1
RA4
RA3
RA2
U1
R10
R11
R21
R22
R16
R18
R19
R20
U5
PWR_ETH
D1
CS1
SB0
SB1
SB2
SB3
R9
R12
R14
R13
R15
U9
5V
CS10
J_RX
J_TX
R3
PWR_485
CS9
U2 SN75176
RB1
RB3
CS5
D2
R8
J_LCD
MAX232
R4
R6
C7
TS8P8C-PCB-
TRAF ETH
1643

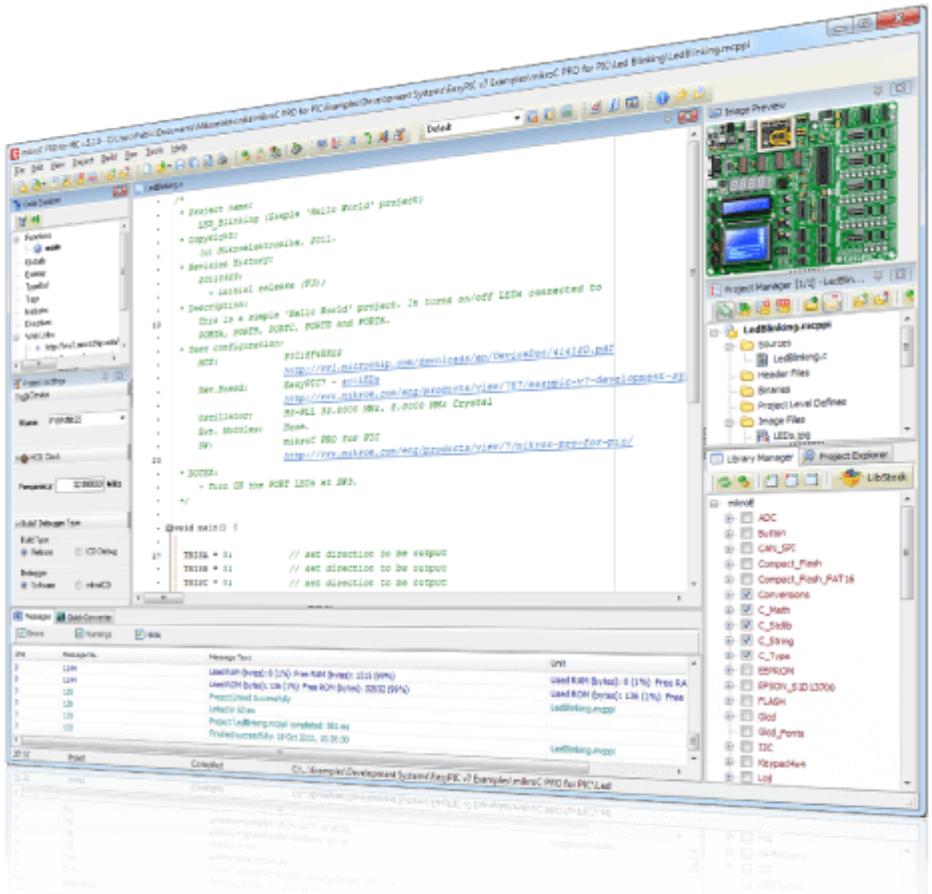
- Svi projektni zadaci imaju istu osnovu kako se rešavaju koja je detaljno objašnjena u primeru na računskim vežbama.
- Svaki tim se sastoji od četiri studenta, od kojih dva realizuju Master deo projekta a dva studenta Slejv podsistem.
- Na kraju, kada se sistem implementira na datim razvojnim sistemima, studenti treba da napišu seminarski rad.
- U laboratoriji, u jednom dvočasovnom terminu, biće dve grupe od po 4 studenta (Master i Slejv) koji rešavaju dva različita zadatka.

Projekti su:

1. Sistem za nadzor šahtova
2. Sistem za automatizovano zalivanje
3. Sistem za merenje koncentracije štetnih gasova
4. Crpne pumpe
5. Upravljanje i nadzor objekta
6. Sistem za upravljanje osvetljenjem u zgradi
7. Sistem za naplatu karata u Aqua Parku
8. Hotelski informacioni sistem
9. Sistem za zaštitu od požara

itd.

mikroC everywhere™ PRO for PIC



Programski jezik C jezik primenjen na PIC mikrokontrolere

Sadržaj:

- Programski jezik C namenjen razvoju firmvera mikrokontrolera.
- Opis osnovnih funkcija.
- Tipovi podataka,
- aritmetički operatori,
- unarni aritmetički operatori, relacioni operatori, logički operatori, operatori nad bitovima. Naredbe grananja,
- petlje *for, while, do while*.
- Rad sa nizovima.

Funkcije

- Funkcije su osnovni gradivni blokovi svakog C programa.
- Svi programi sadrže bar jednu funkciju – **main()**
- Mnogi programi koje ćete pisati sadržaće mnoge funkcije.
- Primer funkcija je:

```
void main( ) {
}

void function1( ) {
}

void function2( ) {
}
```

Funkcije

Primeri funkcija

```
int volume(int s1, int s2, int s3);

void main()
{
      int vol;
      vol = volume(5,7,12);
      printf("volume: %d\n",vol);

}

int volume(int s1, int s2, int s3)
{
      return s1*s2*s3;
}
```

Tipovi podataka

C programski jezik podržava nekoliko osnovnih tipova.

Sledeća tabela prikazuje osnovne tipove podataka koje možete koristiti pri pisanju C programa za PIC mikrokontrolere:

Type	Size in bytes	Range
(unsigned) char	1	0 .. 255
signed char	1	- 128 .. 127
(signed) short (int)	1	- 128 .. 127
unsigned short (int)	1	0 .. 255
(signed) int	2	-32768 .. 32767
unsigned (int)	2	0 .. 65535
(signed) long (int)	4	-2147483648 .. 2147483647
unsigned long (int)	4	0 .. 4294967295

tip bit

```
unsigned char RAMP_ID = 0x00;
unsigned char Category = 0x00;

#define PinTaster PORTB.F0
// ulaz; povezan na taster

#define PinEvent PORTA.F4
// izlaz; povezan na LED diodu

bit TMP_Taster;
bit TMP_Taster2;
bit Event;
```

Sledeći tipovi su podržani od strane C programskog jezika ali ih nećete koristiti zbog ograničenih resursa PIC mikrokontrolera, koji ne poseduju *Floating point* aritmetičku jedinicu.

Type	Size in bytes	Range
`float`	4	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$
`double`	4	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$
`long double`	4	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$

Aritmetički operatori

Operator	Operation
	Binary Operators
+	addition
–	subtraction
*	multiplication
/	division
%	modulus operator returns the remainder of integer division (cannot be used with floating points)

- C programski jezik definiše 5 aritmetičkih operatora.
- Zbog ogramičenih resursa PIC mikrokontrolera ograničite se sabiranje i oduzimanje.
- Ako treba pomnožiti dva broja ili podeliti, koristiti sabiranje, oduzimanje i šiftovanje (pomeranje) brojeva ulevo i udesno

Unarni aritmetički operatori

Za inkrementiranje i dekrementiranje promenljivih koristite operatore ++ i - -

```
a=a+1;  // ili
a=a-1;

a++; // za inkrement
a -- ; // za dekrement
```

Relacioni operatori

Operator	Operation
==	equal
!=	not equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal

- Relacioni operatori vraćaju vrednost tačno ili netačno zavisno od rezultata poređenja
- Jedna stvar koju treba da uočite jeste da je rezultat uvek 0 ili 1, iako C definiše vrednost **Tačno** kao bilo koju vrednost različitu od nule.
- **Netačno** se definiše kao nula

Logički operatori

- Logički operatori podržavaju osnovne logičke operacije AND, OR NOT
  Ponovo, ovi operatori vraćaju 0 za false ili 1 za true.
- Logički operatori i tabela istinitosti data je ispod.

Operator	Operation
&&	logical AND
||	logical OR
!	logical negation

Naredbe grananja

`If` iskaz je uslovni iskaz. Blok koda pridružen if iskazu se izvršava zavisno od uslova iskaza. Ponovo ponavljamo, svaka vrednost različita od nule se smatra za tačnu (true), a vrednost nula je netačno (false). Najjednostavnija forma je:

```
if (izraz)
     iskaz;
```

```
if (izraz)
     iskaz1;
else
     iskaz2;
```

`If` iskaz se takođe može koristiti za kontrolu izvršavanja blokova koda. Uopšteni format je:

```
if (izraz)

{
     iskaz;
}
```

```
if (izraz)
{
     .
iskaz;
     .
}
else
{
.
iskaz;
.
}
```

```
void ProcessInputs()
{
   TMP_Taster2 = TMP_Taster1;
   if (PinTaster == 1)
            TMP_Taster1 = 1; // aktivan 1
       else
            TMP_Taster1 = 0;
       if ((TMP_Taster2 == 1) && (TMP_Taster1 == 1))
       {
            Event = 1;
            PinEvent = 1;
       }
}
```

SWITCH naredba

Promenljiva se sukcesivno testira i poredi njena vrednost sa listom konstanti tipa *integer* ili karakter. Kada se naiđe na slaganje, izvršavaju se komande pridružene datoj konstanti sve dok se ne naiđe na **break** iskaz. Ako se ne naiđe na nikakvo slaganje, izvršavaju se komande pridružene *default* uslovu testiranja. Primer korišćenja **switch** iskaza je:

```
switch (variable)
{
case constant1:
     statement(s);
     break;
case constant2:
     statement(s);
     break;
case constantN:
     statement(s);
     break;
default:
     statement(s);
}
```

Operatori nad bitovima

- C sadrži šest specijalnih operatora za bit-po- bit operacije nad brojevima. Ovi tipovi se mogu koristiti nad podacima koji su tipa **char, short, int ili long**.
- Operatori nad bitovima su:
  - **&** - bitsko AND
  - **|** - bitsko OR
  - **^** - bitsko XOR
  - **~** - prvi komplement
  - **<<** pomeranje (šiftovanje ulevo)
  - **>>** pomeranje(šiftovanje udesno)

Primer svih operatora nad bitima:

```
      AND                              OR
      00000101      (5)                00000101      (5)
&     00000110      (6)          |     00000110      (6)
      ----------------                 ----------------
      00000100      (4)                00000111      (7)

      ŠIFT ulevo                       Šift udesno
      00000101      (5)                00000101      (5)
      <<     2                         >> 2
      :----------------                ------------------------
      = 00010100    (20)               00000001      (1)
```

Za setovanje bita neke promenljive koristi se operator nad bitovima | (log. OR), za resetovanje operator & (log. AND) . Neka je **a** promenljiva tipa **unsigned char**. Prikazani su sledeći primeri:

- Setovanje bita na bit poziciji 4

a=a|0b00010000; ili a=a|0x10; ili a=a|16;

- Setovanje bita na bit poziciji 7

a=a|0b10000000; ili a=a|0x80; ili a=a|128;

- Resetovanje bita na bit poziji 0

a=a&0x11111110; ili a=a&0xFE; ili a=a&254;

- Resetovanje bita na bit poziciji 6

a=a&0x10111111; ili a=a&0xDF;

- Provera da li je bit na poziciji 4 setovan

  if ((a&0b00010000)==0b00010000) {…};

ili if ((a&0x10)==0x10) {…}

- Provera da li je bit na poziciji 7 setovan

  if ((a&0b10000000)==0b10000000) {…};

ili if ((a&0x80)==0x80) {…}

- Provera da li je bit na poziciji 0 resetovan

  if ((a|0b11111110)==0b11111110) {…};

ili if ((a|0xFE)==0xFE) {…}

- Provera da li je bit na poziciji 6 resetovan

  if ((a|0b10111111)==0b10111111) {…};

ili if ((a|0xDF)==0xDF) {…}

```
// deo prekidne funkcije za prijem podataka

// ispituje se bajt ch
if ((ch & 0xE0) == 0x20)
{
   // primljeni bajt je bajt prozivke
   BytesToReceive = 0x00;
   CallFlag = 1;
}

else if ((ch & 0xE0) == 0x60)
{
    // primljeni bajt je komanda za
    // podesavanje sata realnog vremena

    BytesToReceive = 0x03;
    Counter2 = 3;
    // 300 ms je vreme tokom kojeg
    // trebaju da stignu preostali bajtovi
}
```

Da li su na bitovima 7, 6 i 5 “001”
Ch(7:5)==“001”

XXXX XXXX
1110 0000
0010 0000

Da li su na bitovima 7, 6 i 5 “011”
Ch(7:5)==“011”

XXXX XXXX
1110 0000
0110 0000

```
if (Error == 1)
{
  // nema kartica
  DR = 1;
  if (Operation == 0x01)
     CommandModified = 0b00010000 | RAMP_ID;
     //CommandModified = 0x10 | RAMP_ID;
     //CommandModified = 16 | RAMP_ID;
  else
     CommandModified = 0b00000000 | RAMP_ID;
     //CommandModified = 0x00 | RAMP_ID;
     //CommandModified =  RAMP_ID;

  transmit(CommandModified);  // slanje jednog bajta preko RS485
  DR = 0;
}
```

PETLJE

1. FOR petlja

```
for (inicijalizacija; test_uslov ; inkrement)
```

2. WHILE petlja

```
while (izraz)
{
     iskaz;
}
```

3. DO WHILE petlja

```
do
{
     iskazi
}

while (izraz)
```

- Primeri for petlje:

  for(i=0; i<N; i++) {…}

- Primeri while petlje:

```
i=0;
while (i<N) {
…
i++;
}
```

Primer bekonačne petlje: while(1) {...}

Rad sa nizovima

Opšta forma deklarisanja jednodimenzionalnih nizova je:

```
type var_name [size];
```

gde **type** predstavlja korektan C tip podataka, **var_name** predstavlja ime niza, a size određuje koliko elemenata ima u nizu. Na primer, ako želimo niz od 50 elemenata možemo da koristimo ovaj iskaz.

```
int  height [50];
```

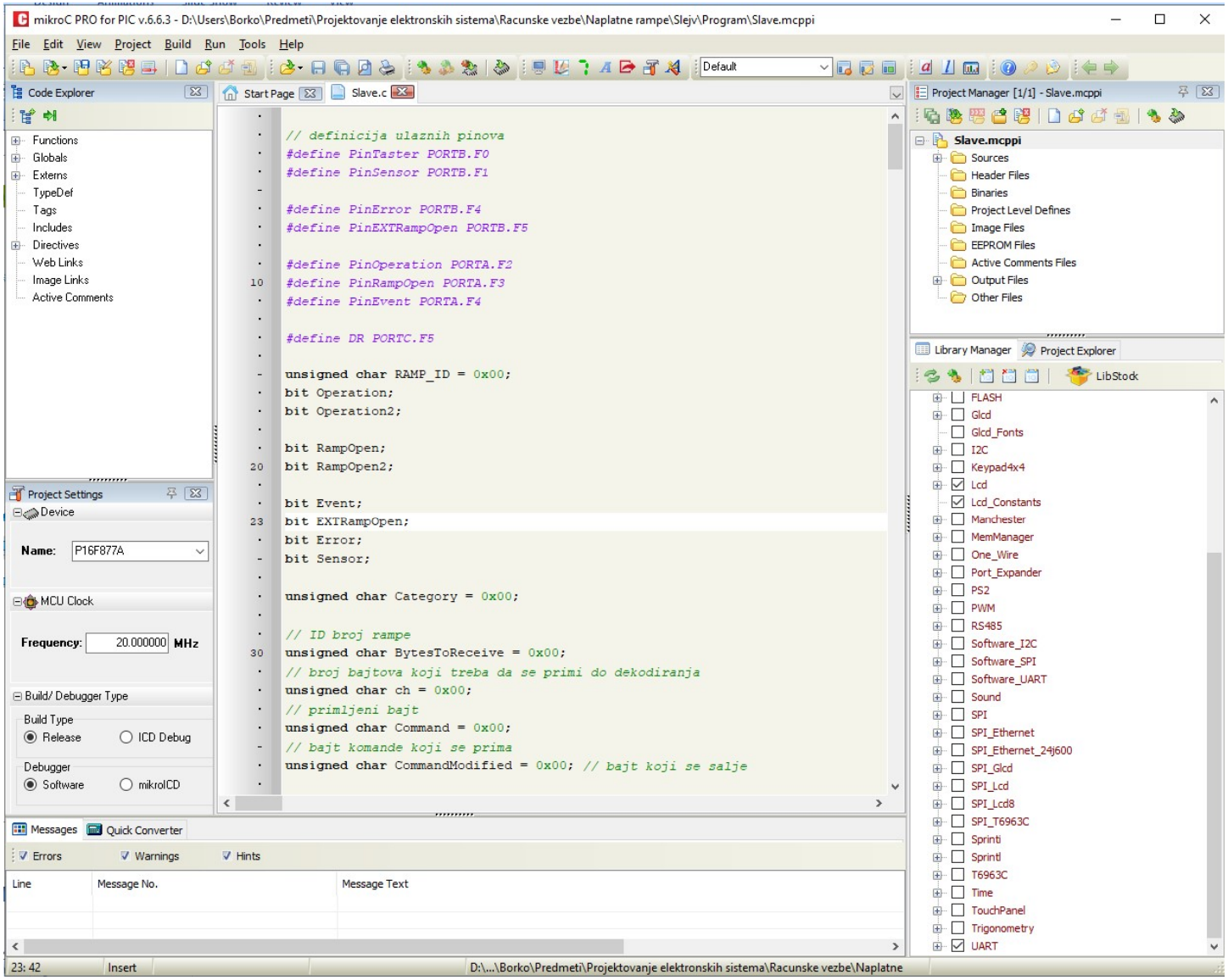
C definiše da, u deklarisanom nizu, prvi elemenat niza ima index 0. Ako niz ima 50 elemenata, poslednji elemenat ima indeks 49. Koristeći prethodni primer, pretpostavimo da želimo da indeksiramo 25-ti elemenat niza *height* i da mu dodelim vrednost 60. Sledeći primer pokazuje kako da se to uradi:

```
height [24] = 60;
```

```
char str[3] = ('a', 'b', 'c');

char name [5] = "John";
```

```
int i[5] = {1,2,3,4,5};
 void  main(void)
 {
      int num[10];
      int i;

      for (i=0,i<10,i++)
            num[i] = i * i;

 for (i=0,i<10,i++)
            printf("%d ",num[i]);
 }
```

Mikro C IDE

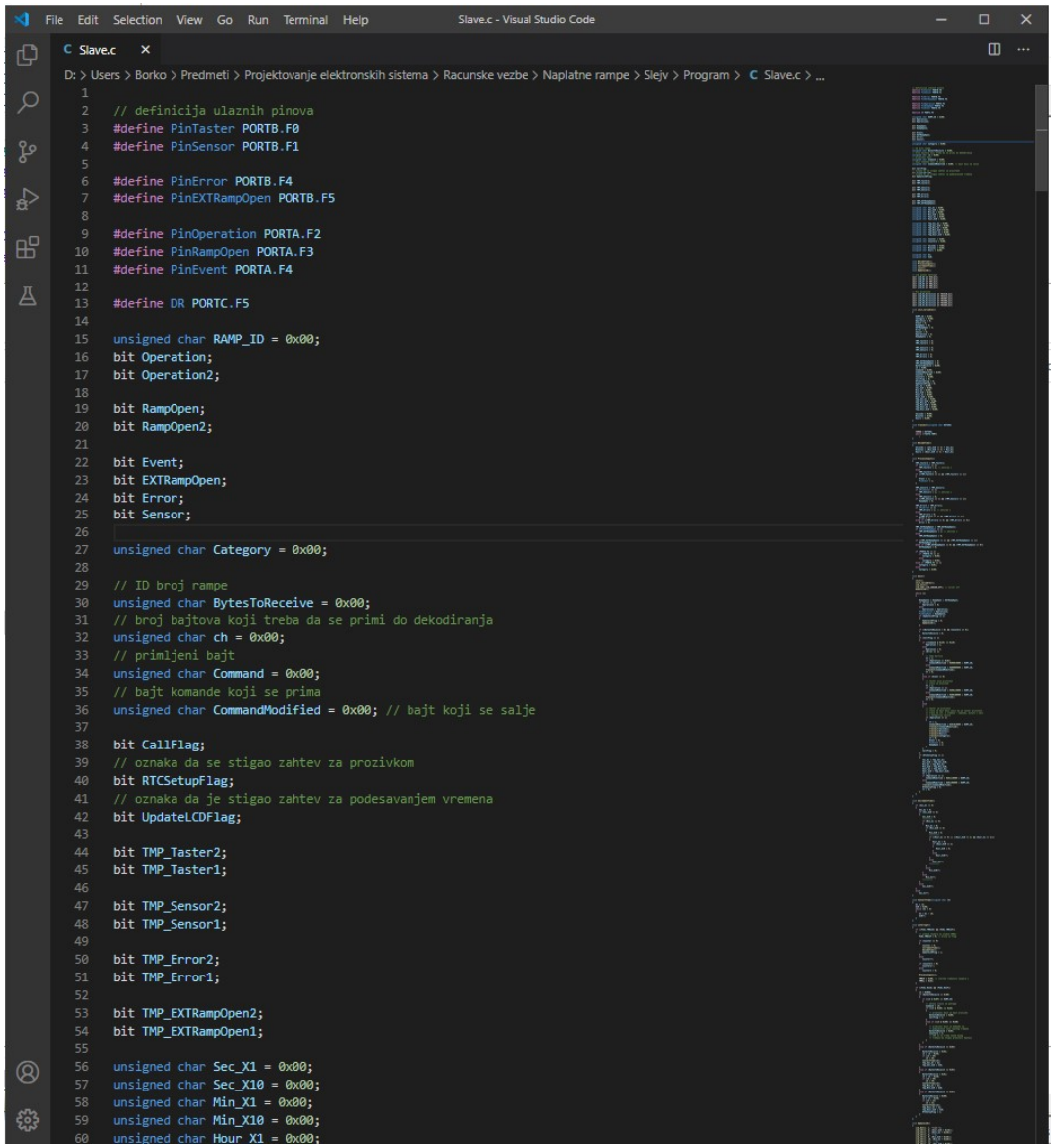

mikroC PRO for PIC v.6.6.3 - D:\Users\Borko\Predmeti\Projektovanje elektronskih sistema\Racunske vezbe\Naplatne rampe\Slejv\Program\Slave.mcppi
File Edit View Project Build Run Tools Help
Code Explorer
Functions
Globals
Externs
TypeDef
Tags
Includes
Directives
Web Links
Image Links
Active Comments
Start Page
Slave.c
// definicija ulaznih pinova
#define PinTaster PORTB.F0
#define PinSensor PORTB.F1
#define PinError PORTB.F4
#define PinEXTRampOpen PORTB.F5
#define PinOperation PORTA.F2
#define PinRampOpen PORTA.F3
#define PinEvent PORTA.F4
#define DR PORTC.F5
unsigned char RAMP_ID = 0x00;
bit Operation;
bit Operation2;
bit RampOpen;
bit RampOpen2;
bit Event;
bit EXTRampOpen;
bit Error;
bit Sensor;
unsigned char Category = 0x00;
// ID broj rampe
unsigned char BytesToReceive = 0x00;
// broj bajtova koji treba da se primi do dekodiranja
unsigned char ch = 0x00;
// primljeni bajt
unsigned char Command = 0x00;
// bajt komande koji se prima
unsigned char CommandModified = 0x00; // bajt koji se salje
Project Manager [1/1] - Slave.mcppi
Slave.mcppi
Sources
Header Files
Binaries
Project Level Defines
Image Files
EEPROM Files
Active Comments Files
Output Files
Other Files
Library Manager
Project Explorer
LibStock
FLASH
Glcd
Glcd_Fonts
I2C
Keypad4x4
Lcd
Lcd_Constants
Manchester
MemManager
One_Wire
Port_Expander
PS2
PWM
RS485
Software_I2C
Software_SPI
Software_UART
Sound
SPI
SPI_Ethernet
SPI_Ethernet_24j600
SPI_Glcd
SPI_Lcd
SPI_Lcd8
SPI_T6963C
Sprinti
SprintI
T6963C
Time
TouchPanel
Trigonometry
UART
Project Settings
Device
Name: P16F877A
MCU Clock
Frequency: 20.000000 MHz
Build/ Debugger Type
Build Type
Release
ICD Debug
Debugger
Software
mikroICD
Messages
Quick Converter
Errors
Warnings
Hints
Line
Message No.
Message Text
23: 42
Insert
D:\...\Borko\Predmeti\Projektovanje elektronskih sistema\Racunske vezbe\Naplatne

Visual Studio Code

Veoma dobar editor teksta za pisanje C/C++ fajlova.

Ima ugradjeno raspoznavanje kljucnih reci I automatsko formatiranje C koda.

Mozete ga slobodno skinuti sa interneta I instalirati



Struktura programa

```
// deklaracija promenljivih
void init() {  ….
}
void init_variables() { ….
}
void main() {
   init();
    init_variables();
    while(1) { ….
        // provera flegova I pozivanje funkcija
    }
}
void interrupt() {
    if ((PIE1.RCIE==1) && (PIR1.RCIF==1))  { ….
    }
     …..
    if ( (PIE1.TMR1IE==1) && (PIR1.TMR1IF==1)) {
    }
}
```

```
void init()
{  // pinovi na portu A su digitalni izlazi
   TRISA = 0x03;
   // pinovi na portu B su digitalni ulazi
   TRISB = 0x3F;
   TRISC = 0xC0;
   // pinovi 6 i 7 su vezani za RS232
   TRISD = 0x0F;
   PORTA = 0x00;
   PORTB = 0x00;
   PORTC = 0x00;
   ADCON0 = 0x00;
// iskljucujemo A/D konverziju
   ADCON1 = 0b00000110;
// svi digitalni pinovi
   INTCON = 0b11000000; // default
   PIE1 = 0b00000000;   // default
   T1CON = 0b00110000;
// konfiguracija za Tajmer 1
// preskaler 1/8
   TMR1H = 0x0B;
// startne vrednosti tajmera 1
   TMR1L = 0xDC;
   T1CON.TMR1ON = 1;

// Fosc=20MHz, Tosc= 50ns
// takt frekv. Fosc/4 dolazi
// na preskaler koji je podesen na 1/8
// izlaz preskalera vodi se na brojace TMR1
    H_TMR1L
// (10000)h- (0BDC)h= (F424)h= (62500)dec
//  62500 x  8 x 4 Tosc= 100ms

PIR1.TMR1IF = 0;
PIE1.TMR1IE = 1;

Uart1_Init(19200);

// konfigurisemo serijski port
// na bitsku brzinu od 19200
// i konfigurisemo dodatne bitove
// za serijski prenos
   TXSTA.TXEN = 1;
   RCSTA.SPEN = 1;
   RCSTA.CREN = 1;
   PIE1.RCIE = 1;
   INTCON.GIE = 1;
   // globalna dozvola prekida
}
```