Programarea Microcontrolerelor utilizând Limbajul C

Utilizarea programului uVision2 - Keil Software - Crearea unui Proiect

Programul folosit în laborator este o variantă gratis oferită de compania Keil Foftware, având limitare de asamblare (compilare) a programelor ce depăşesc 2K octeţi. Acest program este un simulator de microcontrolere, ajutând programatorul să-şi testeze programul, în timp real sau pas cu pas, înainte de a-l înscrie în microcontroler.

Utilizarea programului:

Se porneşte de la crearea unui „proiect" : „project" → „new project" . Se dă nume proiectului şi se salvează. În acest moment apare „Select device for target" , şi se alege din librărie microcontrolerul pentru care se editează programul. Apoi se creează o pagină nouă „file" → „new". Se salvează această pagină cu extensia: „ .c", apoi din „target 1" → „source group 1" , se dă click dreapta la mouse şi se alege „Add files to Group 'source group 1' ", se deschide o nouă fereastră de unde se alege fişierul mai devreme salvat. Atenţie la extensia fişierului! Odată ales fişierul, se apasă butonul „close" pentru a închide fereastra de dialog.

Începând din acest moment, se poate trece la editarea programului. Programul oferă o interfaţă uşoară de lucru cu utilizatorul, de exp. mnemonicele din program se vor colora în albastru.

După ce s-a scris programul, se va trece la compilarea acestuia, apăsând „F7" (build target). În acest moment se compilează programul şi dacă nu sunt erori se poate trece la simularea lui.

În Laborator, se vor edita, compila şi simula următoarele programe:

Program 1 - BLINK LED - LABORATOR ASC

```
/*Program 1 - BLINK LED  - LABORATOR ASC*/

#include<reg513a.H>          // foloseste Registri SFR ai microcontrolerului C513 (Infineon/Siemens)
sbit LED = P3 ^ 4;           // etichetez pinul P3.4=LED
unsigned int i=0;            // definesc o variabila (fara semn) iia valori de la (0-65535) (2 octetzi)

void main (void)             // bucla principala de program (nu returneaza nimic)
{
        while (1)            // cat timp conditia din paranteze este adevarata, ramaii in bucla!
                {
                        for (i=0; i<30000; i++);        // stai in acest "for" cat timp i<30000
                        LED = ~LED;                     // in acest moment i=30000, si acum
                                                        //complementeaza bitul (LED-ul)!
                }                                  // datorita buclei while, se revine la "for"-ul de mai sus
}
```

Program 2 - BLINK LED - LABORATOR ASC

```
/*Program 2 - BLINK LED  - LABORATOR ASC*/

#include<reg513a.H>          // foloseste Registri SFR ai microcontrolerului C513 (Infineon/Siemens)
sbit LED = P3 ^ 4;           // etichetez pinul P3.4=LED
unsigned int a;              // definesc o variabila (fara semn) iia valori de la (0-65535) (2 octetzi)

void main (void)
{
        while (~0)
                {
                        for (a=0; a<10000; a++);
                        LED = ~LED;
                        for (a=0; a<50000; a++);
                        LED = ~LED;
                }
}
```

Program 3 - BLINK LED - LABORATOR ASC

```
/*Program 3 - BLINK LED  - LABORATOR ASC*/

#include<reg513a.H>          // foloseste Registri SFR ai microcontrolerului C513 (Infineon/Siemens)
sbit LED = P3 ^ 4;           // etichetez pinul P3.4=LED
unsigned int a;              // definesc o variabila (fara semn) iia valori de la (0-65535) (2 octetzi)
void salt(void);

void main (void)
{
        while (1)
                {
                        for (a=0; a<10000; a++);
                        LED = ~LED;
                        salt();          // salt la eticheta "salt"
                }
}

void salt (void)
{
        for (a=0; a<40000; a++);
        LED = ~LED;
}
```

Program 4 - BLINK LED - LABORATOR ASC

```
/*Program 4 - BLINK LED  - LABORATOR ASC*/
#include<reg513a.H>          // foloseste Registri SFR ai microcontrolerului C513 (Infineon/Siemens)
#define CONST       120      // declar constanta CONST = 120
sbit LED = P3 ^ 4;           // etichetez pinul P3.4=LED
unsigned int t;              // definesc o variabila globala (fara semn)
                                        //care ia valori de la (0-65535); este pe (2 octetzi)
void salt1 (unsigned int fifi);         //declaratie de functie
unsigned int salt2 (unsigned char q);   // primeste variabila "q" si returneaza qq

void main (void)                        // bucla principala de program
{
        P3 = 0;                         // "sting" LED-urile conectate la portul P3
        while (1)                       // conditie adevarata totdeauna
                {
                        unsigned int a;             // variabila locala
                        for (a=0; a<35000; a++);    //intarziere
                        LED = ~LED;                 //complementez valoarea logica a pinului de port
                        t = salt2 (CONST);          // salt la eticheta "salt2"
                        salt1 (t);                  // salt la eticheta "salt1"
                }
}

unsigned int salt2 (unsigned char q)    // functia salt2 ~ returneaza qq
{
        unsigned int qq;                // variabila locala
        qq = q*q;                       // calcul banal ...
        return qq;                      // returneaza valoarea lui qq
}

void salt1 (unsigned int fifi)          // functia salt1 ~ nu returneaza nimic!
{                                       //este folosita aceasta functie doar pentru intarziere
        unsigned int ji;                // variabila locala
        for (ji=0; ji<fifi; ji++);      //intarziere
        LED = ~LED;                     //complementez valoarea logica a pinului de port
}
```

Program 5 - BLINK LED - LABORATOR ASC

```
/*Program 5 - BLINK LED  - LABORATOR ASC*/
#include<reg513a.h>          // foloseste Registri SFR ai microcontrolerului C513 (Infineon/Siemens)
#define hp 0x75302710        // declar „hp” (0x7530=30000 zecimal, 0x2710=10000); acolo unde se
                             //intalneste „hp”, compilatorul inlocuieste direct VALOAREA acesteia
sbit LED = P3 ^ 4;           // etichetez pinul P3.4=LED

typedef union                // uniune
{
        struct               //se realizeaza o structura intre a shi b
                {
                        unsigned int a;     // primul integer
                        unsigned int b;     // al doilea integer
                }c;         //
        unsigned long d;     // se declara "d" pentru a se realiza ulterior uniunea
} gigi;                      // uniune de tipul "gigi"
gigi pwm;                    // declar pwm sa fie de tipul "gigi"

void main (void)
{
        unsigned int i;              // declaratie de variabila locala
        unsigned long x;             // declaratie de variabila locala
        x = hp;                      // transfer valoarea constantei hp, variabilei "x"
        P3 = 0;                      // daca am LED-uri conectate la portul P3, acestea se vor stinge
        while(1)                     // bucla infinita
          {
                pwm.d = x++; // intreaga valoare a lui x, se copie (apoi este incrementata) in pwm.d
                for (i=0; i<pwm.c.a; i++);          // stai in acest "for"
                LED = ~LED;                         // complementeaza bitul!
                for (i=0; i<pwm.c.b; i++);          // stai in acest "for"
                LED = ~LED;                         // complementeaza bitul!
          }
}
```