Documentatie SOC-Proiect

Sipoteanu Octavian-Ioan

March 7, 2024

Abstract

Proiectul dezvoltă o soluție embedded în IAR Embedded Workbench, combinând limbajul C și asamblare pentru a controla un LED în funcție de datele unui senzor. Intensitatea luminii variază în concordanță cu distanța la care se află obiectul detectat.

1 Introduction

Proiectul propus reprezintă o soluție embedded dezvoltată în mediul IAR Embedded Workbench, în care limbajul de programare C și asamblare sunt utilizate pentru a controla un LED în funcție de datele primite de la un senzor. Scopul principal al proiectului constă în reglarea intensității luminii LED-ului în funcție de distanța la care se află un obiect detectat de senzor. Implementarea este structurată într-o arhitectură hibridă, combinând funcții scrise în limbajul de asamblare.
Codul în limbaj C se ocupă de interacțiunea cu senzorul, iar în funcție de datele primite, apelează funcții scrise în limbaj de asamblare pentru manipularea hardware-ului specific al LED-ului și pentru a ajusta parametrii de control în concordanță. Această abordare permite optimizarea performanței și o manipulare mai eficientă a resurselor hardware disponibile, oferind astfel o soluție integrată de control al iluminării bazată pe datele primite de la senzor.
Essențialmente, proiectul îmbină tehnologiile hardware și software pentru a furniza o soluție de control al iluminării. Experiența vizuală este concepută să fie intuitivă, reflectând distanța obiectului detectat prin variații ale intensității LED-ului. Astfel, proiectul demonstrează o integrare eficientă între limbajul C și asamblare, contribuind la realizarea unei aplicații embedded cu funcționalități avansate.

2 Implementarea codului

2.1 Includerea variabilelor si initializarea variabilelor si functiilor externe

2.2

In partea aceasta de cod sunt incluse librariile si inițializarea variabilelor și funcțiilor externe.

```
1 #include <inavr.h>
2 #include <iom16.h>
3
4 unsigned char intrare_automat=0;
5 unsigned short iesire_automat=0;
6 unsigned char stare_automat=0;
7
8 extern void initializare(void);/* Prototipul functiei ASM */
9 extern void c_intrerupere();
10 extern void ext_intrerupere();
11 extern void beculet(unsigned char luminozitate);
```

2.3

In aceasta parte de cod este începutul programului în C, și se inițializează porturile de Input și Output, respectiv DDRD porturile de input și DDRB de output.**********
Pe urmă se apelează funcția de initializare ASM și se activează intreruperile.

```
12
13 void main(void)
14 {
15   DDRD = 0x80;/* Initializeaza porturile de I/O*/
16   DDRB = 0xFF;
17   initializare();
18
19   __enable_interrupt();
```

2.4

In aceasta functie de initializare , incarcam in registrul R16 o valoare ce reprezinta o combinatie de biti specifici care configureaza modul de functionare al unui timmer, acesta fiind FAST PWM.
In r17 incarcam valoarea 1 , aceasta reprezinta Duty Cycle-ul pentru modulul fast PWM
In out TCCR0, r16 si out OCR0, r17 transmitem valoare din registru r16 respectiv r17 in registrii de control al timerului 0 (TCCR0) si In registrul de comparatie al timerului 0 (OCR0).
La urmatoare instructiune de ldi , se incarca in registrul r16 o valoare care configureaza modul de functionare al timerului 1 si activeaza intruruperea la schimbarea de semnal (ICES1)
La urmatoarea , incarca in r17 o valoare care activaza intreruperea pentru capturarea evenimentului la timerul 1
Pe urma , in r18 se incarca o valoare care elibereaza starea de flag pentru evenimentul de captura la timerul 1 (ICF1)
Apoi sunt transmise pe rand valorile din registrii r16,r17 si r18 in registrii corespunzatori.
In r16 se incarca o valoare care configureaza modul de functionare al timerului 2 , in r17 255 apoi se transmir valorile in registrii corespunzatori

```
51
52 NAME initializare
53         #include <ioavr.h>
54         PUBLIC initializare
55
56 RSEG CODE
57
58   initializare:
59   ldi r16,(1<<WGM01)|(1<<WGM00)|(1<<CS02)|(1<<CS00)|(1<<COM01)//IL PUNE PE MOD
60   ldi r17,1 //DUTY CYCLE
61
62   out TCCR0,r16
63   out OCR0, r17
64
65
66   ldi r16,(1<<CS11)|(1<<CS10)|(1<<ICES1)//ASTEAPTA SA PRIMEASCA 1
67   ldi r17,(1<<TICIE1) // PORNESTE INTRURUPEREA LA SCHIMBAREA DE SEMNAL
68   ldi r18,(1<<ICF1)
69
70   out TCCR1B,r16
71   out TIMSK,r17
72   out TIFR,r18
73
74
75   ldi r16, (1<<WGM21)|(1<<WGM20)|(1<<COM21)|(1<<CS22)
76   ldi r17, 255
77   out TCCR2, r16
78   out OCR2, r17
79
80   ret
81 END
82
```

2.5

Aici este programul propiu-zis cu bucla while infinita. Aici , avem un bloc switch pentru automatul folosit , acesta gestionand diferitele stari ale variabilei "stare-automat". In starea 0 , acesta verifica daca primeste semnal de la senzor , astfel , va face intrarea automat 1 , care o sa trimita automatul in starea 1. Aici facem timerul TCNT1 0 , pentru a putea masura timpul precis. In starea 1 verifica daca semnalul primit de la senzor se termina , astfel salveaza intr-o variabila timpul din TCNT1 si trimite automatul spre starea 0.

```
19    __enable_interrupt();
20    while(1)/* Bucla infinit?*/
21    {
22      switch(stare_automat)
23      {
24      case 0://starea 0, astept semnal de la senzor
25        {
26          if(intrare_automat==1)//am primit semnalul de la senzor
27          {
28            TCNT1=0;
29            stare_automat=1;
30            TCCR1B = (1<<CS11)|(1<<CS10)|(0<<ICES1);
31          }
32          break;
33        }
34
35
36      case 1://starea 1, sunt in semnal si astept sfarsitul lui
37        {
38          if(intrare_automat==0)//s-a terminat semnalul de la senzor
39          {
40            iesire_automat= TCNT1;//salvez valoarea din timer
41            stare_automat=0;
42            TCCR1B = (1<<CS11)|(1<<CS10)|(1<<ICES1);
43          }
44
45          break;
46
47        }
48      }
49
50      beculet(iesire_automat);
51    }
52  }
```

2.6

Aici specificam utilizarea vectorului de intrerupere INTVEC(1) si ca rutina de intrerupere este asociata lui TIMER1-CAP1-vect.

in.c | functii.asm

```
1  NAME ext_intrerupere
2      #include <ioavr.h>
3      extern c_intrerupere
4      COMMON INTVEC(1)
5      ORG TIMER1_CAPT_vect
6      RJMP c_intrerupere
7  ENDMOD
8
```

2.7

Prima data acesta declara variabilele externe. Se citeste valoarea portului D si se incarca in registrul 16.
cu cbr eliminam bitului 7 din valoarea citita apoi din aceasta eliminam valoarea corespunzatoare ca sa avem 1 in registru

```
 9 NAME c_intrerupere
10     #include <ioavr.h>
11     PUBLIC c_intrerupere
12     EXTERN iesire_automat,intrare_automat,stare_automat
13     RSEG CODE
14     c_intrerupere:
15         st -Y,R16 ; Push regi°tri pe stivã
16         in R16,SREG ; Cite°te status register
17         st -Y,R16 ; Cite°te status register
18
19         in r16,PIND
20         cbr r16,191
21                 cpi r16, 0
22         breq final
23                 subi r16, 63
24         final:
25         sts intrare_automat,r16
26
27         ld R16,Y+ ; Pop status register
28         out SREG,R16 ; Salveazã status register
29         ld R16,Y+ ; Pop Register R16
30
31         reti
32 ENDMOD
```

2.8

Variabila care de indica timpul dat de la senzor se afla in registrii r16(L) si r17(H) deoarece este o valoare care este pe 16 biti.Daca r17 nu este gol ,inseamna ca senzorul a receptionat mai departe decat distanta maxima , astfel vom pune 0 si pe r16 si transmitem pe OCR2 ceea ce va inchide LED-ul. In caz contrar , folosim instructiunea com care complementeaza valoarea registrului r16 si ne ajuta sa trimitem intensitate la LED in functie de distanta.

-h

```
33
34 NAME beculet
35         #include <ioavr.h>
36         PUBLIC beculet
37 RSEG CODE
38         beculet: //in r16, r17 e variabila data ca parametru
39                 cpi r17, 0
40                 brne stinge_bec
41         aprinde_bec:
42                 com r16
43                 out OCR2, r16
44                 jmp final
45         stinge_bec:
46                 ldi r16, 0
47                 out OCR2, r16
48         final:
49                 ret
50 ENDMOD
```

References

[1] Atmel Corporation. *AVR Instruction Set Manual.*

[2] Atmel Corporation. *ATmega16 Datasheet.*

[3] IAR Systems. *IAR Embedded Workbench AVR Compiler Reference Guide.*

[4] IAR Systems. *IAR Embedded Workbench AVR Assembler Reference Guide.*

[5] *SOC final laborator.*