Temat ćwiczenia: Wyświetlanie za pomocą linijki LED, generacja fali PWM

Wykonujący ćwiczenie: Górka Paweł, Wesołowski Krzysztof

Daty: 23.10.2009, 30.10.2009 **Grupa:** Piątek godzina 12.00

Laboratorium z dnia 23.10.2009

Nie wykonywaliśmy ćwiczenia, ze względu na obecność na seminarium dotyczącym automatycznej generacji kodu z MATLAB'a organizowanego przez ONT

Laboratorium z dnia 30.10.2008: *Wyświetlanie na linijce LED z wykorzystaniem Timer'a*

```
MAINLOOP: ;pusta, główna petla programu - calosc pracy procesora oparta jest o przerwania
          ;Jego istota jest przesuwanie 4 swiecacych 'bitow' po linijce LED w prawo i lewo
    SJMP MAINLOOP
ISR_T0:
    push PSW      ;umieszczenie na stosie psw i acc  - dla bezpieczenstwa - jezeli
    push ACC      ;obsluga przerwania zmienilaby zawartosc tych 2 rejestrow,
                  ;mogloby to po zakonczeiu obslugi zaklocic dalsza prace programu
    DJNZ R7, DALEJ ;zmniejsz wartosc w R7 o 1, skocz do DALEJ jezeli wartosc R7=/= 0
    LCALL PRZESUN ;jezeli wartosc R7 = 0 wywolywania jest procedura PRZESUN
    mov R7, 0x30 ;oraz do R7 przypisywana jest wartosc z pod komorki pamieci o adresie 0x30
    DALEJ:
    pop ACC ;zdjecie 'starych' wartosci acc i psw ze stosu
    pop PSW
RETI ;koniec obslugi przerwania
PRZESUN: ;procedura zmieiajaca sposob swiecenia linijki LED
    ;sprawdzenie czy jest juz jedynka na skraju po lewej stronie linijki
    JNB 15, NIEWPRAWO
    ;Jest jedynka na skrajnej lewej pozycji- odbijam w prawo
    setb 0 ; bit 0 z obszaru pamieci bitowej, jest uzywany tutaj jako bit kontrolny
           ;kierunku poruszania sie, gdy jest rowny 1 to swiecenie porusza sie w prawo
    NIEWPRAWO: ;Nie odbijam w prawo
    ;Sprawdzenia czy jest juz jedynka na skraju po prawej stronie linijki LED
    JNB 8, NIELEWO
    ;Jest jedynka na skrajnej prawej pozycji- odbijam w lewo
    clr 0 ;zerowanie bitu zerowego z obszaru pamieci adresowaej bitowo - czyli ustalenie
          ;kierunku ruchu jako w lewo
    NIELEWO:
    ;Normalne przesuwanie
    JNB 0, IDEWLEWO
    ;ide w prawo
        mov A, 0x21
        RR A ;przesuniecie wyswietlania w prawo
        mov 0x21, A
    LJMP JUZPOSZEDLEM ;jezeli przesuwalem w prawo, to omijam przesuwanie w lewo wykonujac
                      ;skok
    IDEWLEWO:
        mov A, 0x21
        RL A ;przesuniecie wyswietlania w lewo
        mov 0x21, A
    JUZPOSZEDLEM:
        mov A, 0x21
        CPL A ;negacja bitow akumulatora, wynikajaca z tego, ze diody aktywowane sa stanem
              ;niskim
        mov P1, A ;zaswiecenie odpowiedich diod, za pomoca wystawienia wartosci akumulatora
                  ;na port P1
RET ;koniec wywolania procedury
INITPROC:
    mov 0x21, #0x04 ;zapisanie wartosci 4 pod komorke pamieci 0x21. Adres ten posluzy do
                    ;poruszania zaswiecona dioda po linijce LED na zmiane w prawo i lewo.
                   ;Celowo wykorzystano tutaj adresowany bitowo obszar pamieci procesora
    mov 0x30, #0x08 ;wartosc wykorzystywana do 'programowego' zmniejszania częstotliwości
;przesuwania wyswietlania (podzial czestotliwosci przez 8), oczywiscie używanie pamieci
;RAM na trzymanie stalej wartosci jest tutaj marnotrawstwem, jednak w planie było
```

```
;zmienianie tego dzielnika za pomoca zewnetrznych buttonow.
    mov SP, #0x40       ;ustawienie SP w 'bezpieczny' obszar pamieci, konkretnie wysuniecie
                        ;go poza uzywane przez nas zmienne
    mov R7, 0x30
    ;konfiguracja pracy timera:
    setb TR0 ;wlaczenie timer0
    setb EA ;aktywacja przerwan
    setb ET0 ;aktywacja przerwania od timer0
RET
```

Laboratorium z dnia 30.10.2008: *Generacja PWM*

```
MAINLOOP:
;pusta nieskonczona petla programowa. Calosc pracy programu oparta o obsluge przerwan.
;Celem programu bylo generacja fali PWM o zmiennym wypelnieniu na port sterujacy praca
;linijki diodowej,co skutkowało róznym natezeniem swiecenia linijki
    SJMP MAINLOOP
LJMP START
ISR_T0:
    push PSW ;umieszczenie PSW i ACC na stosie dla 'bezpieczenstwa'
    push ACC ;w tym programie niekonieczne ale czsto petla glowal uzywa ACC
    ;Sterowanie wypelnieniem
      mov A, 0x21
      SUBB A, #1  ;zmniejszenie rejestru odpowiedzialnego za szerekogsc stanu '1' w fali
                  ;PWM o jeden, zrezygnowalismy z DEC aby latwiej zmieniac skok,
                  ;wartosci/czestosc jego wykonania
      mov 0x21, A
      mov A, 0x22
      ADD A, #1 ;zwiekszenie rejestru odpowiedzialnego za szer. stanu '0' w fali PWM o 1
      mov 0x22, A
        ;z kazdym wywolaniem obslugi przerwania, wypelnienie fali zmienia sie
    ;PWM - jezeli byl stan niski, przelacz na stan wysoki, i vice-versa
    JNB 0, NISKI ;jezeli nie jest ustawiony bit 0 z przestrzeni adresowania bitowego, to
                 ;skocz do etykiety NISKI
      clr 0 ;aktualnie stan wysoki zatem zmien na niski
      mov p1, #0xFF ;wystaw na wszystkie piny portu '1'
      mov TH0, 0x21 ;przypisz do TH0 dekrementowana wczesiej wartosc spod adres 0x21;
      SJMP KONIEC
    NISKI:
      setb ;0; aktualnie generowany jest stan niski zatem zmien go na wysoki
      mov p1, #0x00 ;wystaw na wszystkie piny portu '0'
      mov TH0, 0x22 ;przypisz do TH0 incremetowana wczesiej wartosc spod adresu 0x22
    KONIEC:
    pop ACC ;zdjecie ze stosu
    pop PSW
RETI
INITPROC:
    mov SP, #0x40 ;ustawienie stosu w 'bezpiecznej' przestrzeni adresowej
    setb EA ;wlacz przerwania
    setb ET0 ;wlacz timer0 irq
    setb TR0 ;wlacz timer0
    mov 0x21, #(0xFF/2) ;wysoki - poczatkowe ustawienie wypelnienia fali na 50%.
    mov 0x22, #(0xFF/2);niski - wartosci 0x21 i 0x22 poprzez ich wpisywanie do rejestru TH0
                       ;sluza okresleniu wypelnienia PWM
    ;Tryb 0 - rozdzielczosc 8 bitów i 5bitowy preskaler, daje nam czestotliwosc PWM w
    ;okolicy 112Hz
RET
```