5. Întreruperi

5.1 Introducere

Pentru a înțelege mai bine facilitățile oferite de întreruperi vom începe acest capitol cu exemplul poştaşului. Poştaşul este însărcinat cu distribuția scrisorilor la adresele destinatarilor. O serie de scrisori şi colete sunt aduse personal la uşa fiecăruia, însă de cele mai multe ori, poştaşul va utiliza căsuța poştală corespunzătoare fiecărei locuințe. Această metodă este mult mai convenabilă poştaşului pentru distribuția scrisorilor decât livrarea la uşă dacă luăm în considerare faptul că într-un singur cartier pot exista mai multe mii de locuințe. În plus, cutia poştală permite primirea scrisorilor chiar dacă destinatarul nu este acasă.

Cu toate aceste facilități oferite de existența cutiei poştale, fiecare dintre noi trebuie să verifice la anumite intervale de timp dacă s-a primit vre-o scrisoare. Această metodă poate fi destul de frustrantă dacă se aşteaptă primirea unei scrisori importante, sau dacă locuința respectivului se află la un etaj nu tocmai aproape de cutia poştală (e.g. etajul X). Viața ar fi poate mult mai simplă dacă poştaşul ne-ar putea anunța (e.g. prin interfon) că s-a primit o scrisoare.

Lăsând deoparte poştaşul, să ne concentrăm puțin atenția către domeniul microcontrollerelor. Până acum, ne-am obişnuit că pentru a verifica starea unui periferic, trebuie testat în mod repetat perifericul respectiv (i.e. un anumit bit sau registru întreg). Din această cauză, implementarea unor programe cu o complexitate ridicată poate fi problematică. Poate ar fi mult mai uşor dacă programul ar putea fi „anunțat" de modificarea stării perifericului, ceea ce ar face ca între timp programul în cauză să se poată ocupa de o altă parte a problemei. Procesul de „anunțare" se realizează prin întreruperi.

Mecanismul de suspendare a firului de execuție la apariția unui eveniment, se numeşte *întrerupere*. Microcontrollere au deja implementat un mecanism de *întrerupere*, prin care se permite execuția unei *rutine de tratare a întreruperii*. Pentru tratarea întreruperilor, în cazul seriei 16F există un singur vector (0x04) iar în cazul seriei 18F există doi vectori (0x08 şi 0x18). Vectorul de întrerupere reprezintă de fapt o adresă de program la care se face salt în momentul apariției unei întreruperi. Astfel, singurul vector pentru seria 16F este utilizat pentru deservirea tuturor întreruperilor. Față de această serie, microcontrollere din seria 18F permit setarea unei priorități pentru fiecare întrerupere, vectorul 0x08 având o prioritate mare (en. „high"), pe când vectorul 0x18 are o prioritate mică (en. „low").

În continuare vom vorbi despre configurarea şi utilizarea întreruperilor valabile doar pentru seria 18F. Aplicarea acestor concepte se va face similar în cazul celorlalte serii.

5.2 Activarea/dezactivarea întreruperilor

Controlul operației fiecărei întreruperi se realizează prin teri biți:

- Un bit de flag pentru a indica apariția unui eveniment întrerupere;
- Un bit de activare care permite selectarea individuală a surselor de întrerupere utilizate;
- Un bit de prioritate.

Dacă nu se utilizează priorităţi, activarea/dezactivarea tuturor surselor de întrerupere se realizează prin setarea/resetarea bitului GIE al registrului INTCON.

Dacă priorităţile sunt utilizate, întreruperile cu prioritatea „high" se pot activa independent de întreruperile cu prioritatea „low" prin bitul GIEH al registrului INTCON. Aceeaşi afirmaţie este valabilă şi în cazul priorităţilor „low", care se pot controla prin bitul GIEL al aceluiaşi registru.

Această serie de microcontrollere permite dezactivarea utilizării priorităţilor prin resetarea bitului IPEN al registrului RCON. În cazul acesta, vectorul de întrerupere va fi 0x08 pentru toate sursele.

Întrucât sursele de întrerupere includ atât perifericele microcontrollerelor cât şi o serie de componente interne, seria 18F oferă posibilitatea activării/dezactivării întreruperilor provenite de la periferice prin bitul PEIE al registrului INTCON (valabil doar în cazul în care priorităţile nu sunt folosite).

IPEN	GIEH	GIEL	GIE	PEIE
1	1/0	1/0	x	x
0	x	X	1/0	1/0

Tabelul 5.1. Biţii care trebuie configuraţi dacă se utilizează sau nu priorităţi de întrerupere

5.3 Configurarea întreruperilor

De regulă, paşii care trebuie urmaţi pentru configurarea întreruperilor sunt următorii:

- Activarea/Dezactivarea priorităţilor (IPEN=1/0, registrul RCON);
- Activarea/Dezactivarea întreruperilor provenite de la periferice (PEIE=1/0, registrul INTCON);
- Setarea priorităţii pentru sursa de întrerupere utilizată (dacă IPEN=1);
- Ştergerea flag-ului corespunzător sursei de întrerupere configurate;
- Activarea întreruperii corespunzătoare sursei configurate;
- Setarea bitului global de întrerupere (GIE=1 sau GIEH/GIEL=1, registrul INTCON).

5.4 Tratarea întreruperilor

În momentul apariţiei unei întreruperi, bitul global de configurare este resetat pentru a dezactiva alte întreruperi. Dacă IPEN=0 (i.e. nu există mai multe priorităţi de întrerupere) acest bit este GIE. Dacă se utilizează şi priorităţi de întrerupere, acesta va fi GIEH sau GIEL. Întreruperile „low" pot fi întrerupte de alte întreruperi cu priorităţi „high", iar întreruperile „low" nu sunt tratate atâta timp cât există întreruperi „high" netratate.

La apariţia unei întreruperi, adresa de revenire este pusă pe stivă iar numărătorul de program este încărcat cu 0x08 sau 0x18. În cazul în care se utilizează mai multe întreruperi, sursa care a generat întreruperea se poate determina prin testarea flagurilor. La revenirea din întrerupere flag-ul corespunzător trebuie resetat.

Revenirea dintr-o întrerupere se realizează prin instrucţiunea RETFIE, care setează bitul GIE (GIEH sau GIEL în cazul în care se utilizează şi priorităţi).

Exemplul 5.1. Pentru a exemplifica modalitatea de configurare şi implementare a întreruperilor se va utiliza secvența următoare de program în cadrul căruia s-a considerat generarea unei întreruperi corespunzătoare Timerului 0 (având flag-ul T0IF):

```
     #include “P18F4455.INC”

     ORG    0x00

     GOTO   MAIN

     ; 0x008 dacă nu se utilizează BOOT-LOADER
     ; 0x808 dacă se utilizează BOOT-LOADER
     ORG    0x008

     ; Salt la rutina de tratare a întreruperii
     GOTO   Interrupt_Service_Routine

     ; Alte rutine
     ; …

Interrupt_Service_Routine:

     ; Se testează dacă s-a generat întreruperea care
     ; ne interesează
     BTFSS  INTCON, TMR0IF
     RETFIE

     ; Alte instrucțiuni
     ; …

     ; Resetarea flagului
     BCF    INTCON, TMR0IF

     ; Revenire din rutină
     RETFIE

MAIN:
     ; Resetare flag Timer 0
     BCF    INTCON, TMR0IF

     ; Dezactivarea priorităților
     BCF    RCON, IPEN

     ; Activare întrerupere Timer 0
     BSF    INTCON, TMR0IE

     ; Activarea întreruperilor
     BSF    INTCON, GIE

     ; …

     END
```

În secvența anterioară de program, s-a considerat o întrerupere provenită de la un overflow al Timerului 0. Configurarea propriuzisă a Timerului a fost lăsată la latitudinea cititorului.

Exemplul 5.2. Secvența următoare de program exemplifică utilizarea vectorilor de întrerupere cu mai multe priorități.

```
     #include "P18F4455.INC"

     ORG    0x00

     GOTO   MAIN

     ; ISR-high
     ; 0x008 dacă nu se utilizează BOOT-LOADER
     ; 0x808 dacă se utilizează BOOT-LOADER
     ORG    0x08

     ; Salt la rutina de tratare a întreruperii
     GOTO   Interrupt_Service_Routine_HIGH

     ; ISR-low
     ; 0x018  dacă nu se utilizează BOOT-LOADER
     ; 0x818 dacă se utilizează BOOT-LOADER
     ORG    0x018

     ; Salt la rutina de tratare a întreruperii
     GOTO   Interrupt_Service_Routine_LOW
Interrupt_Service_Routine_HIGH:
     ; …
     RETFIE

Interrupt_Service_Routine_LOW:
     ; …
     RETFIE

MAIN:
     ; …

     END
```

5.5 Problemă rezolvată

Utilizarea întreruperilor poate simplifica foarte mult rezolvarea unei probleme numai în cazul în care există o proiectare sistematică inițială care să pună în evidență paşii care trebuie urmați de la stabilirea cerințelor şi până la implementare. Vom exemplifica o asemenea proiectare utilizând problema următoare:

Să se modifice viteza de pâlpâire alternativă a două LED-uri legate pe PORTD<2:3> utilizând un buton legat pe PORTD<0>. Întrucât starea pinului la care este conectat butonul este 0 la apăsare, se va considera modificarea vitezei de pâlpâire la fiecare front descrescător detectat. Numărul de viteze maxime permise este 4. Dacă se apasă butonul conectat pe PORTD<1>, viteza se va reseta.

Din specificația informală a problemei se pot distinge următoarele cerințe:

- 2 LED-uri legate pe PORTD<2:3>;
- 2 butoane legate pe PORTD<0:1>;
- Definirea a 4 viteze de pâlpâire;
- Modificarea vitezei la fiecare apăsare a butonului D0;

- Resetarea vitezei la apăsarea butonului D1.

Din această specificație va rezulta diagrama de stare din Figura 5.1.

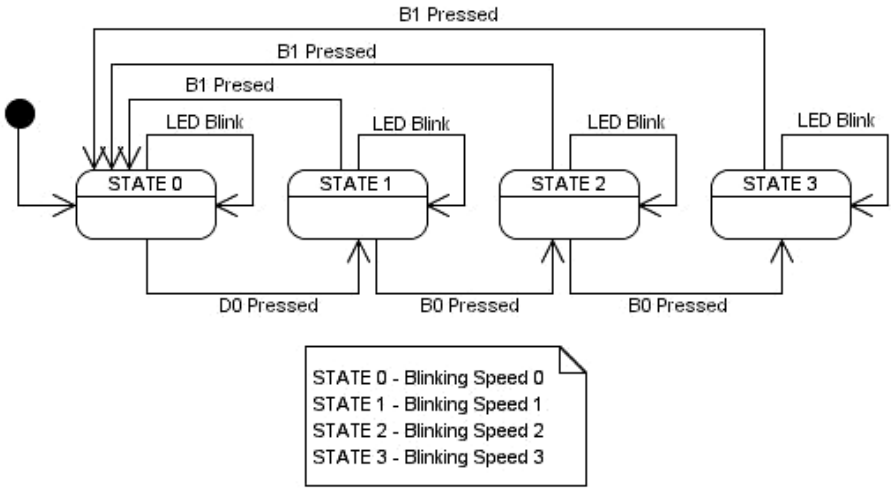


Figura 5.1. Diagrama de stare pentru modificarea vitezei de pâlpâire a LED-urilor utilizând butoanele B0 şi B1

În următorul pas trebuie stabilite temporizările din fiecare stare. Dacă ne reamintim puțin problema prel-ului din capitolul alocat descrierii porturilor, identificăm o temporizare unică pentru toate stările alocate eliminării prel-ului butoanelor. Față de această temporizare, viteza de pâlpâire a LED-urilor diferă însă de la o stare la alta, de unde va rezulta şi o variație a temporizării utilizate.

Eliminarea prel-ului se poate realiza prin utilizarea unei rutine de întârziere bazate pe instrucțiuni sau Timere. Pe de altă parte, problema se simplifică foarte mult dacă temporizarea pâlpâirii LED-urilor se implementează utilizând un Timer şi întreruperi.

Dacă verificăm posibilitățile de configurare a Timerului 0, vom observa că prin simpla modificare a valorii divizorului se poate influența timpul de incrementare. Această observație permite identificarea stărilor în funcție de valoarea divizorului şi trecerea de la o stare la alta prin simpla incrementare a divizorului.

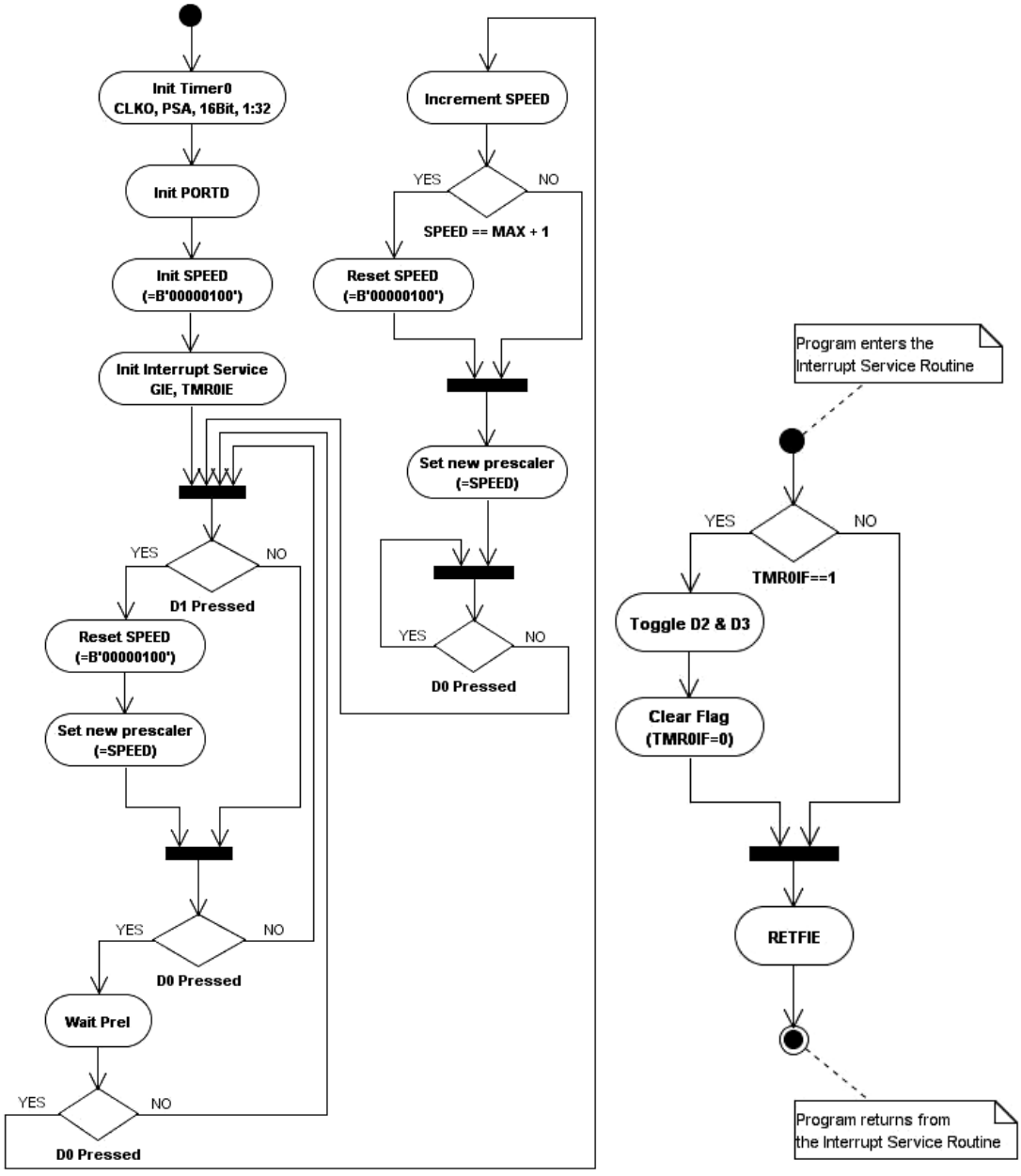
În Figura 5.2 se poate vedea diagrama de activitate a programului principal (a) şi diagrama de activitate a rutinei de tratare a întreruperii (b). Pentru implementarea pâlpâirii LED-urilor se va utiliza Timer 0 pe 16 biți, a cărui inițializare este realizată de secvența următoare de program:

```
BCF     T0CON, TMR0ON
BCF     T0CON, T08BIT
BCF     T0CON, T0CS
BCF     T0CON, PSA
BSF     T0CON, T0PS2
BCF     T0CON, T0PS1
BCF     T0CON, T0PS0
```

După inițializarea porturilor şi a vitezei inițiale (ceea ce reprezintă câteva instrucțiuni MOVxx) configurarea întreruperii generate de Timer 0 la resetarea acestuia:

```
BCF     RCON,    IPEN
```

```
BCF     INTCON, PEIE
BCF     INTCON, TMR0IF
BSF     INTCON, TMR0IE
BSF     INTCON, GIE
```



a) b)

Figura 5.2. Diagrama de activitate pentru: a) Funcționalitatea principală a modificării vitezei; b) Rutina de întrerupere pentru controlul stării LED-urilor

După aceste configurări, Timer 0 trebuie pornit (BSF T0CON, TMR0ON), urmând testarea butoanelor. Testarea butonului B1 este urmată, în cazul în care se detectează o apăsare, de resetarea vitezei de pâlpâire (MAIN_LOOP este o etichetă indicând începutul buclei de test principale):

```
MAIN_LOOP:
     BTFSS  PORTD, 1
     CALL   Reset_Speed
```

unde Reset_Speed este rutina de resetare a vitezei de pâlpâire, din care se apelează rutina de setarea a divizorului (i.e. rutina Set_Prescaler):

```
Reset_Speed:
     MOVLW  B'00000100'
     MOVWF  SPEED
     MOVFF  SPEED, PORTB      ; afişarea vitezei curente
     CALL   Set_Prescaler     ; setarea divizorului
     RETURN

Set_Prescaler:
     BCF    T0CON, TMR0ON ; Oprirea Timerului

     BCF    T0CON, T0PS2  ; Stergerea biţilor divizorului
     BCF    T0CON, T0PS1
     BCF    T0CON, T0PS0

     MOVF   SPEED, W      ; Setarea noului divizor
     IORWF  T0CON, F

     BSF    T0CON, TMR0ON ; Pornirea Timerului
     RETURN
```

Testarea apăsării butonului D0 trebuie să ia în calcul şi prel-ul butonului, rezultând următoarea secvență de cod:

```
     BTFSC  PORTD, 0
     GOTO   MAIN_LOOP

     CALL   Delay_10_Millis
     BTFSC  PORTD, 0
     GOTO   MAIN_LOOP
```

După aceasta, urmează rutina de incrementare a vitezei:

```
Increment_Speed:
     INCF   SPEED, F
     MOVFF  SPEED, PORTB

     BTFSC  SPEED, 3          ; Se testează dacă s-a atins viteza maximă
     CALL   Reset_Speed

     CALL   Set_Prescaler     ; Setarea noului divizor

     RETURN
```

Într-un final, urmează secvența de cod prin care se aşteaptă eliberarea butonului D0:

```
WAIT_RELEASE_BUTTON:
     BTFSS   PORTD, 0
     GOTO    WAIT_RELEASE_BUTTON

     GOTO    MAIN_LOOP
```

În contextul secvențelor de program prezentate anterior, programul va intra în rutina de tratare a întreruperii generate de Timer 0 la fiecare resetare a acestuia. Timpul în care are loc acest reset este dat de configurația Timerului, care se schimbă pe prcursul execuției în funcție de viteza de lucru. Astfel, secvența de tratare a întreruperii, este următoarea:

```
     ORG           0X808          ; S-a considerat existența unui
                                  ; BOOT-LOADER
     GOTO          Interrupt_Service

Interrupt_Service:
     BTFSS         INTCON, TMR0IF ; Se testează dacă este o întrerupere
     RETFIE                       ; generată de Timer 0

     BTG           PORTD, 2       ; Se modifică starea LED-urilor
     BTG           PORTD, 3

     BCF           INTCON, TMR0IF ; Se şterge flag-ul Timerului

     RETFIE
```

Problemă.

Se consideră două butoane PORTD<0:1>, două LED-uri PORTD<2:3> şi un ventilator legat pe PORTB<0>. Să se realizeze diagrama de stare şi de activitate pentru următoarele funcționalități, după care să se implementeze prin program:

- Într-o primă fază, LED-urile pâlpâie cu o modificare a stărilor în fiecare secundă;
- Dacă se apasă D0 (i.e. se detectează un front căzător), să se pornească ventilatorul a cărui rulare trebuie să dureze un timp de 5 secunde;
- Atâta timp cât ventilatorul este pornit, LED-urile pâlpâie la intervale de 100 milisecunde;
- Dacă în timp ce ventilatorul este pornit se detectează o apăsare a butonului D1, ventilatorul se va opri, iar LED-urile vor pâlpâi în fiecare secundă.

NOTĂ: Se recomandă implementarea diferitelor viteze de pâlpâire prin simpla modificare a divizorului timerului. Totodată, funcționalitatea de pâlpâire se va separa de restul programului şi se va implementa în rutina de întrerupere (pentru resetarea Timerului 0).