

Laborator 4.

Porturi I/O la PIC16F84A

PIC16F84 are doua porturi numite PORTA si PORTB. Acestea pot fi folosite atat ca intrari cat si ca iesiri. Fiecare pin poate fi configurat independent. PORTA dispune de 5 pini asociati cu 5 biti . PORTB dispune de 8 pini asociati cu 8 biti. Cei 5 respectiv 8 pini sunt asociati cu grupuri de biti numite TRISA si TRISB. Pentru a folosi un bit din PORTA ca iesire configuram bitul corespunzator din TRISA astfel: pentru ca pinul sa fie iesire il setam 0; pentru a fi intrare il setam 1. Analog pentru PORTB prin TRISB.

TABLE 4-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

TABLE 4-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxxx xxxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
0Bh,8Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Pentru a putea modifica porturile de intrare si de iesire este nevoie urmatoorii pasi :

- 1 . Se trece in bank-ul 1 de memorie
- 2 . Se initializeaza bitii TRISA sau TRISB , cu 1 pentru intrate si cu 0 pentru iesire
3. Se trece inapoi in bank-ul 0 de memorie
4. Se atribuie valorile necesare bitilor din PORTA sau PORTB

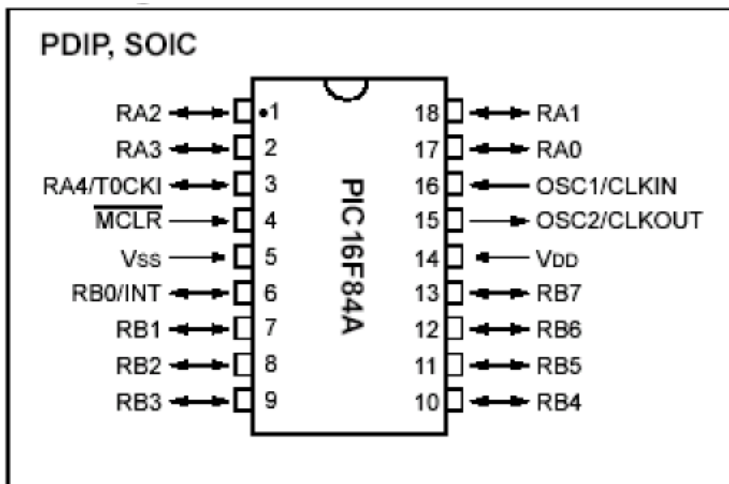
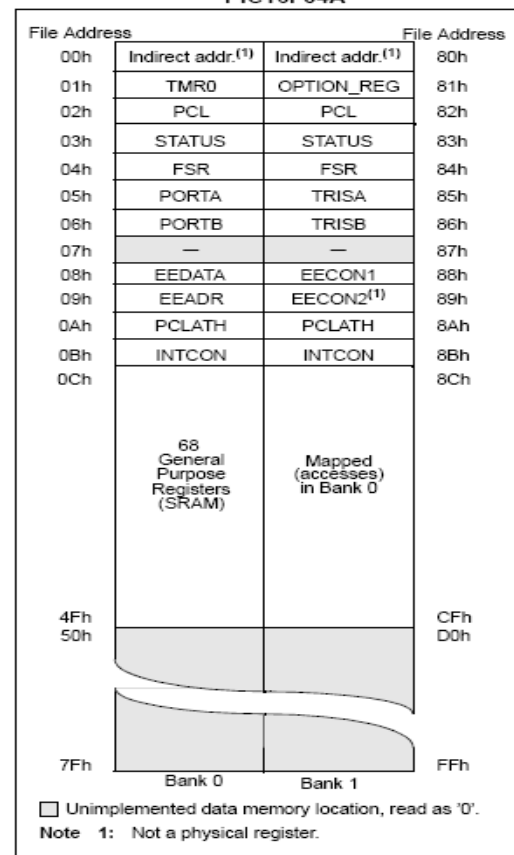


FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



Citirea/scrierea la porturi

Aplicatie:

Vom vedea cum putem citi si scrie la porturile PIC-ului. Putem conecta un circuit extern la pinii microcontrollerului si putem sa folosim actionarile externe pentru a modifica functionarea programului.

STATUS equ 03h ;Adresa registrului STATUS

TRISA equ 85h ; Adresa registrului TRISA

PORTA equ 05h ; Adresa portului A

bsf STATUS,5 ;Trecem in bancul 1

Urmeaza sa specificam folosirea portului A ca iesire. Pentru aceasta trimitem 0 la registrul TRISA. Pentru a face ca un pin sa fie intrare, ii trimitem valoarea 1 .

EXAMPLE 4-1: INITIALIZING PORTA

```
BCF    STATUS, RP0 ;
CLRF   PORTA      ; Initialize PORTA by
                ; clearing output
                ; data latches
BSF    STATUS, RP0 ; Select Bank 1
MOVLW  0x0F      ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISA     ; Set RA<3:0> as inputs
                ; RA4 as output
                ; TRISA<7:5> are always
                ; read as '0'.
```

EXAMPLE 4-2: INITIALIZING PORTB

```
BCF    STATUS, RP0 ;
CLRF   PORTB     ; Initialize PORTB by
                ; clearing output
                ; data latches
BSF    STATUS, RP0 ; Select Bank 1
MOVLW  0xCF      ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISB    ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

movlw 00h ;Setare pini penru Port A

movwf TRISA ; toti bitii sunt iesire

bcf STATUS,5 ;Trecem in bancul 0

Acum am setat primul bit (RA0) de la PORTA ca fiind intrare. Acum urmeaza sa aflam daca pinul este pe una din cele doua valori HIGH (1) sau LOW(0). Putem testa folosind functiile BTFSC sau BTFSS .

Urmatorul program face un LED sa se aprinda cu o anumita viteza. Daca se da semnal de comanda la pinul RA0 atunci LED-ul se va aprinde un timp T si se va stinge un timp T/2. Conectam un comutator intre RA0 si borna de tensiune pozitiva de la alimentare. Cand RA0 va fi 1 se va da semnalul de inchiderea a intreruptorului si deci de aprindere a becului. Cand RA0 va fi 0 se va da semnalul de deschidere a intreruptorului si deci de stingere a becului.

Codul sursa:

```
STATUS equ 03h; locatia din memorie a registrului STATUS
TRISA equ 85h ; locatia din memorie a registrului TRISA
PORTA equ 05h ; locatia din memorie a portului I/O PORTA
COUNT1 equ 08h ;Primul counter folosit ca delay
COUNT2 equ 09h ;Al doilea counter folosit ca delay
```

```
;Setarea portului
bsf STATUS,5 ;trecem in bancul 1 de memorie
movlw 00h
movwf TRISA ;bit 1 este iesire , bit 0 este intrare.
bcf STATUS,5 ;trecem in bancul 0 de memorie
```

Start

```
;Aprinderea LED-ului
movlw 01h
movwf PORTA
```

```
;****Verificam daca intrerupatorul este inchis
BTFSC PORTA,0 ;citeste bitul 0 al PORTA
call Delay ;daca este 1, apelam de doua ori delay pentru ca LED-ul sa stea aprins
o doua perioade
call Delay ;daca este 0, apelam o intarziere pentru ca LED-ul sa stea aprins o
perioada
```

```
;****Intarziere terminata, stingem LED-ul****
movlw 00h ;Stingem LEDul
movwf PORTA
;****Verificam daca intrerupatorul s-a deschis
BTFSC PORTA,0 ;citim PORT A BIT 0. daca este 0, delay normal, daca este 1
de doua ori delay
call Delay
call Delay
```

```
;****Inapoi la inceputul programului
goto Start ; inapoi la start si aprindem LED-ul
```

```
;****subrutina Delay
Delay
movlw 01h
movwf COUNT1
```

```
movlw 01h
movwf COUNT2
Loop1
decfsz COUNT1,1 ; LED ul ramane stins/aprins suficient cat sa putem observa
goto Loop1
Loop2
decfsz COUNT2,1
goto Loop2 ;
return

;****Sfarsitul programului****
End
```

Urmariti cu ajutorul ferestrei Watch urmatoarele register : PORTA, TRISTA, COUNT1, COUNT2 si registrul special WREG