MECE336 Microprocessors I
Sleep, Watchdog Timer, EEPROM

Dr. Kurtuluş Erinç Akdoğan

kurtuluserinc@cankaya.edu.tr

Course Webpage: http://MECE336.cankaya.edu.tr

ÇANKAYA ÜNİVERSİTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

Sleep

- Sleep mode is an important way of saving power.
- The microcontroller can be put into this mode by executing the instruction SLEEP.
- SLEEP: Go into standby mode.
- Once in Sleep mode, the microcontroller almost goes into suspended animation.
  - The clock oscillator is switched off,
  - WDT is cleared,
  - program execution is suspended,
  - all ports retain their current settings
  - PD and TO bits in the Status register are cleared and set respectively.
  - If enabled, the WDT continues running.
  - Under these conditions, power consumption falls to a negligible amount a typical value of 1 μA, under specific ideal operating conditions.

SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

Mnemonic, Operands		Description	Cycles	Status Affected
BYTE-ORIENTED FILE REGISTER OPERATIONS				
ADDWF	f, d	Add W and f	1	C,DC,Z
ANDWF	f, d	AND W with f	1	Z
CLRF	f	Clear f	1	Z
CLRW	-	Clear W	1	Z
COMF	f, d	Complement f	1	Z
DECF	f, d	Decrement f	1	Z
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	
INCF	f, d	Increment f	1	Z
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	
IORWF	f, d	Inclusive OR W with f	1	Z
MOVF	f, d	Move f	1	Z
MOVWF	f	Move W to f	1	
NOP	-	No Operation	1	
RLF	f, d	Rotate Left f through Carry	1	C
RRF	f, d	Rotate Right f through Carry	1	C
SUBWF	f, d	Subtract W from f	1	C,DC,Z
SWAPF	f, d	Swap nibbles in f	1	
XORWF	f, d	Exclusive OR W with f	1	Z
BIT-ORIENTED FILE REGISTER OPERATIONS				
BCF	f, b	Bit Clear f	1	
BSF	f, b	Bit Set f	1	
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	
LITERAL AND CONTROL OPERATIONS				
ADDLW	k	Add literal and W	1	C,DC,Z
ANDLW	k	AND literal with W	1	Z
CALL	k	Call subroutine	2	
CLRWDT	-	Clear Watchdog Timer	1	$\overline{TO}$,$\overline{PD}$
GOTO	k	Go to address	2	
IORLW	k	Inclusive OR literal with W	1	Z
MOVLW	k	Move literal to W	1	
RETFIE	-	Return from interrupt	2	
RETLW	k	Return with literal in W	2	
RETURN	-	Return from Subroutine	2	
SLEEP	-	Go into standby mode	1	$\overline{TO}$,$\overline{PD}$
SUBLW	k	Subtract W from literal	1	C,DC,Z
XORLW	k	Exclusive OR literal with W	1	Z

Status Register

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0**: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

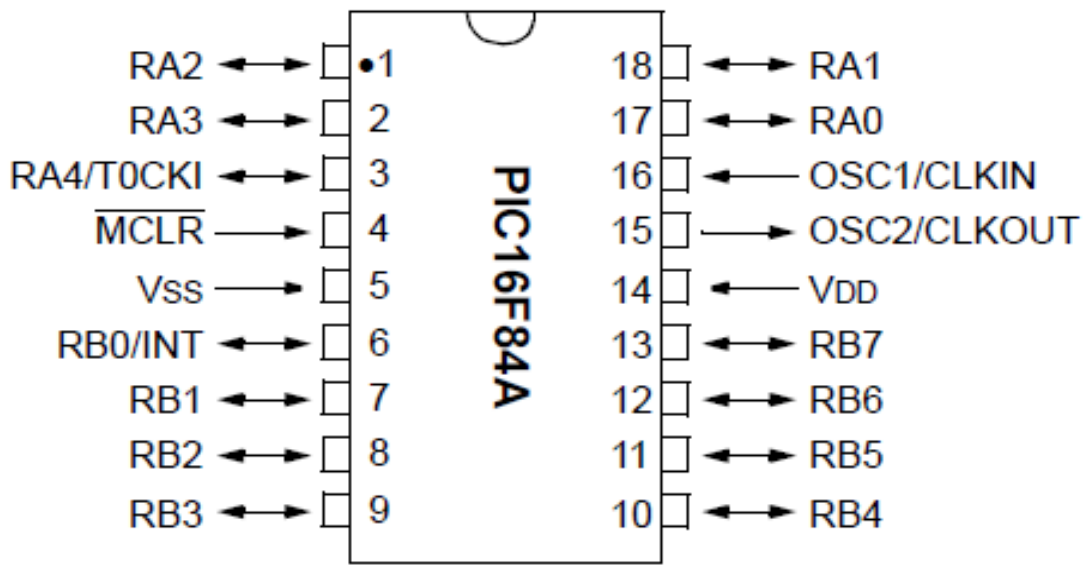
bit 1 **DC**: Digit Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the Most Significant Bit of the result occurred
0 = No carry-out from the Most Significant Bit of the result occurred

Note: A subtraction is executed by adding the twos complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

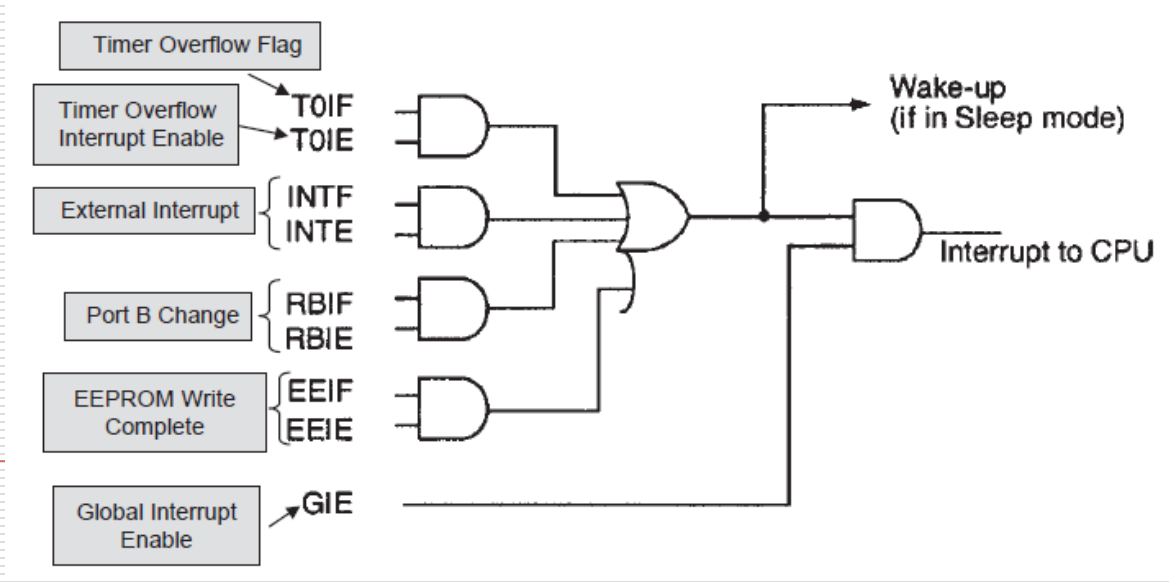
Sleep: Awakening

- **External reset through MCLR pin**. While this causes a wake-up, it also resets the microcontroller; therefore, its use seems limited to complete program restarts. It is possible, however, to detect that the microcontroller has just been in Sleep mode, due to the state of the PD pin in the Status register.



Sleep: Awakening

- **WDT wake-up**. WDT is blocked from causing a reset when in Sleep. Instead, on overflow it just causes a wake-up from Sleep, and the microcontroller continues program execution from the instruction following the Sleep mode.
- **Occurrence of interrupt**. If any interrupt was enabled prior to going into SLEEP, upon receipt of interrupt input, wake-up from Sleep occurs regardless of the state of the Global Interrupt Enable. Timer 0 cannot, however, generate an interrupt, as the internal clock is disabled.



Example

- Write a program that counts the number of positive transitions on input RB.0/INT and displays the current "count" on four LEDs on outputs RB.4 - RB.7.
- Note that the INTEDG bit in the OPTION_REG may be set such that the external interrupt occurs on the positive edge of a signal on RB.0

INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER FILE MAP - PIC16F84A

File Address	Bank 0	Bank 1	File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch
4Fh			CFh
50h			D0h
7Fh			FFh

Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

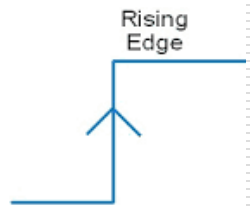
External interrupts and OPTION register

For external interrupts,

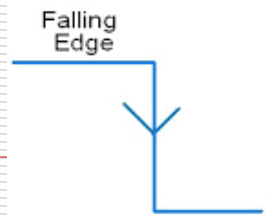
- 1. RB0 must be input.
- 2. INTE must be 1.
- 3. Bit_6 of the OPTION register (INTEDG) is the interrupt edge select bit;

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

- If INTEDG= 1, interrupt occurs rising edge of the signal.



- If INTEDG= 0, interrupt occurs falling edge of the signal.



- Depends on the hardware, INTEDG must be 0 or 1.

REGISTER FILE MAP - PIC16F84A

File Address			File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch
4Fh			CFh
50h			D0h
7Fh	Bank 0	Bank 1	FFh

Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

```
     LIST P=16F84A
     INCLUDE "P16f84A.INC"
     __config _CP_OFF&_WDT_OFF&_XT_OSC

     ORG 00
     GOTO MAIN

     ORG 04
     GOTO INT_SERV

MAIN:
     BSF STATUS, RP0      ; bank 1
     MOVLW 1
     MOVWF TRISB
     BCF STATUS, RP0      ; back to bank 0

     COUNTER EQU 20
     CLRF COUNTER         ; zero the counter
     BCF PORTB, 4         ; zero the LEDs
     BCF PORTB, 5
     BCF PORTB, 6
     BCF PORTB, 7

     BSF OPTION_REG, INTEDG   ; interrupt on
                              ;positive
     BCF INTCON, INTF     ; clear interrupt flag
     BSF INTCON, INTE     ; mask for external
                          ; interrupts
     BSF INTCON, GIE      ; enable interrupts
```

```
PT1: SLEEP
     GOTO PT1
INT_SERV:
     INCF COUNTER, F

     BTFSS COUNTER, 0     ; light the
                          ; appropriate LEDs
     BCF PORTB, 4
     BTFSC COUNTER, 0
     BSF PORTB, 4
     BTFSS COUNTER, 1
     BCF PORTB, 5
     BTFSC COUNTER, 1
     BSF PORTB, 5
     BTFSS COUNTER, 2
     BCF PORTB, 6
     BTFSC COUNTER, 2
     BSF PORTB, 6
     BTFSS COUNTER, 3
     BCF PORTB, 7
     BTFSC COUNTER, 3
     BSF PORTB, 7

     BCF INTCON, INTF     ; clear the
                          ; appropriate flag
     RETFIE               ; this also set global
                          ; interrupt enable

     END
```

https://phanderson.com/PIC/16C84/interrupts/interrupt_1.html

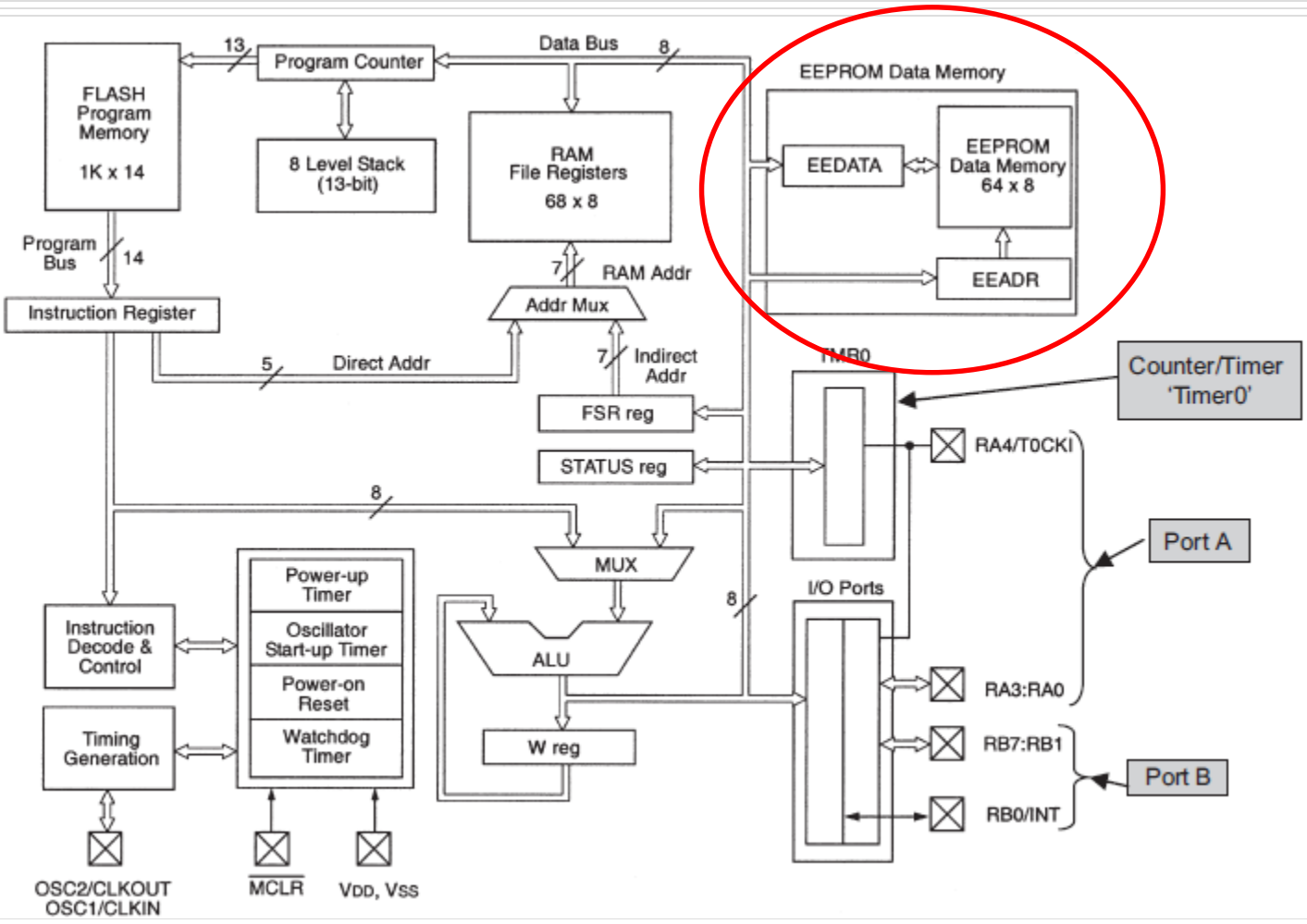
EEPROM: Basics

PIC16F84A devices have 64 bytes of data EEPROM with an address range from 00h to 3Fh.

- The EEPROM is non-volatile and is particularly useful for holding data variables that can be changed but are likely to be needed for the medium to long term.

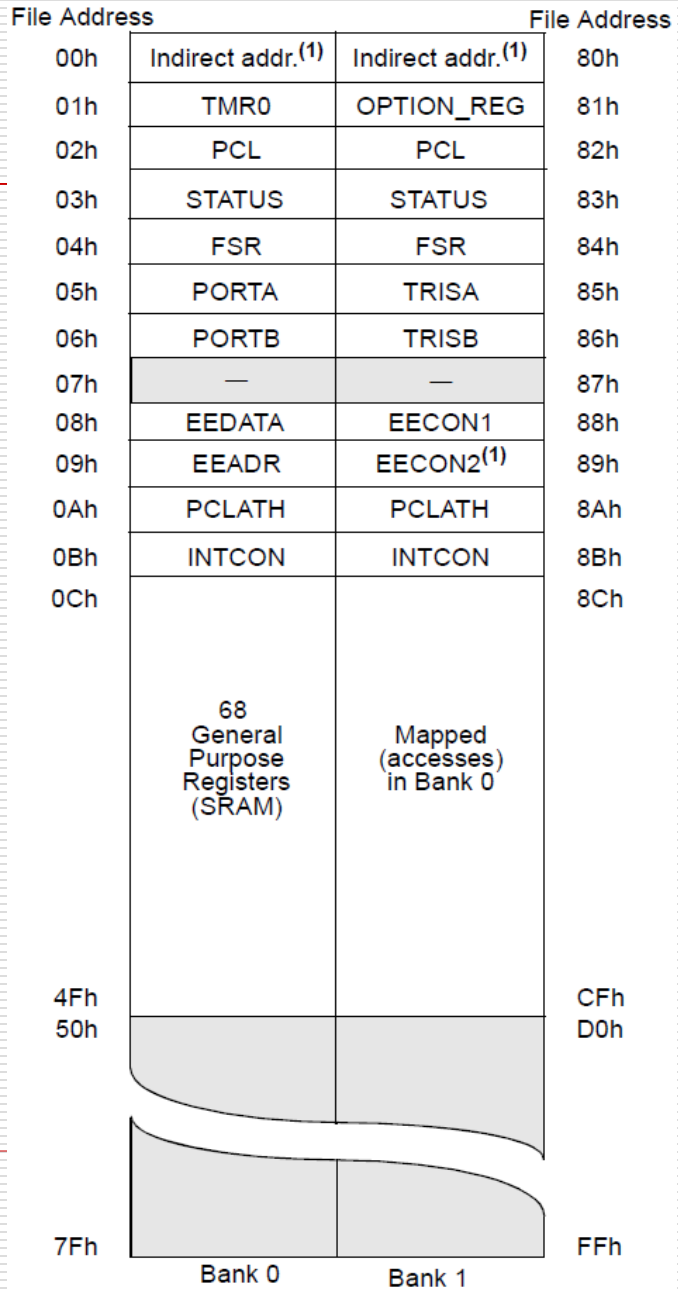
- Examples include TV tuner settings, phone numbers stored in a cell phone or calibration settings on a measuring instrument.

- In the 16F84A ,the EEPROM is not placed in the main data memory map. Instead it is addressed through the EEADR register and data is transferred through EEDATA register. These are both SFRs



- The EEPROM data memory allows byte read and write. A byte write automatically erases the location and writes the new data (erase before write).
- EEPROM is addressed through the EEADR register and data is transferred through the EEDATA register.
- Reading from EEPROM is a simple process but writing to it is not. The latter takes significant time in electronic terms (i.e. milliseconds) and care must be taken to avoid accidental writes. A set of controls is therefore required to start the process and (for write) to detect when it is ended.
- These are found in the bits of the EECON1 register;
- To read an EEPROM location, the required address must be placed in EEADR and the RD bit set in EECON1.
- The data in that memory location is then copied to the EEDATA register and can be read immediately.

REGISTER FILE MAP



EECON1

REGISTER 3-1: EECON1 REGISTER (ADDRESS 88h)

U-0	U-0	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
—	—	—	EEIF	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **EEIF:** EEPROM Write Operation Interrupt Flag bit

1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **WRERR:** EEPROM Error Flag bit

1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ Reset or any WDT Reset during normal operation)
0 = The write operation completed

bit 2 **WREN:** EEPROM Write Enable bit

1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR:** Write Control bit

1 = Initiates a write cycle. The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit

1 = Initiates an EEPROM read RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
0 = Does not initiate an EEPROM read

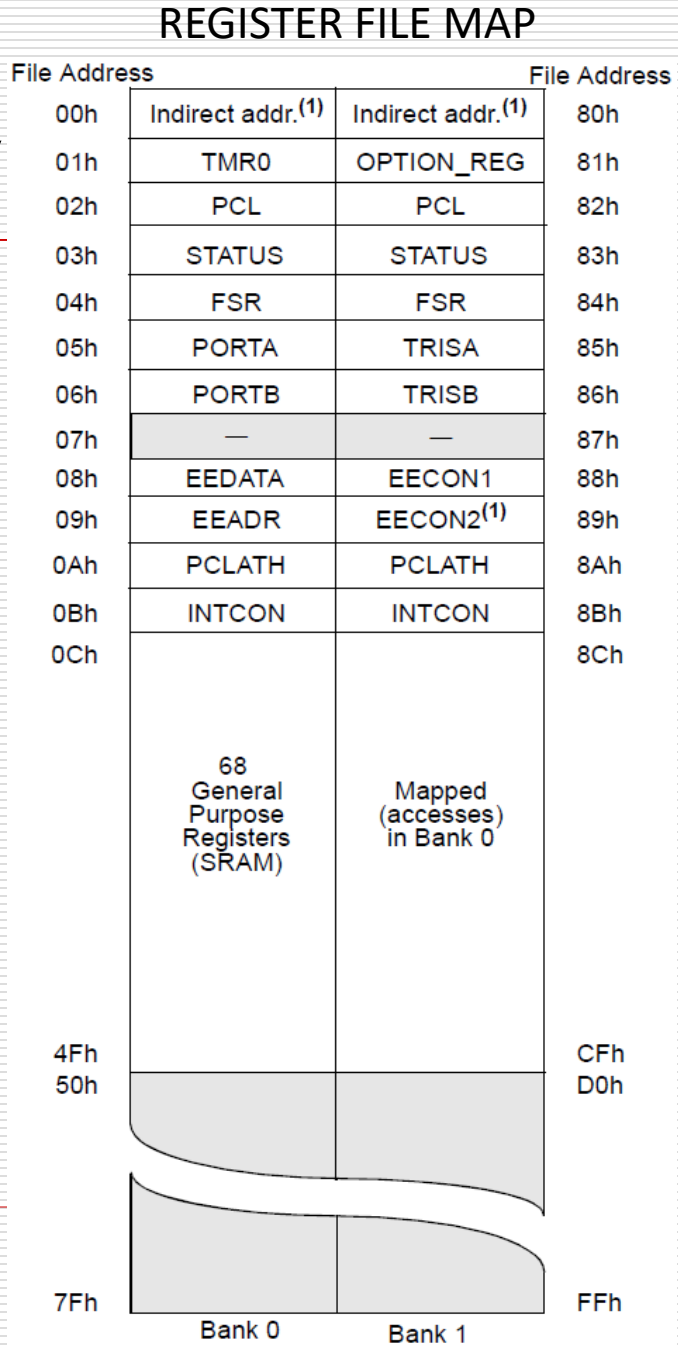
Reading the EEPROM Data Memory

- To read a data memory location, the user must write the address to the EEADR register and then set control bit RD (EECON1<0>). The data is available, in the very next cycle, in the EEDATA register; therefore, it can be read in the next instruction. EEDATA will hold this value until another read or until it is written to by the user (during a write operation).

```
BCF     STATUS, RP0     ; Bank 0
MOVLW   CONFIG_ADDR ;
MOVWF   EEADR           ; Address to read
BSF     STATUS, RP0     ; Bank 1
BSF     EECON1, RD      ; EE Read
BCF     STATUS, RP0     ; Bank 0
MOVF    EEDATA, W       ; W = EEDATA
MOVWF   PORTB           ; PORTB = EEDATA
```

Writing to the EEPROM Data Memory

- To write to an EEPROM location, the required data and address must be placed in EEDATA and EEADR respectively.
- The write process is enabled by the WREN (Write Enable) bit being set high, followed by the bytes 55H followed by AAH being sent to the EECON2 register.
- The built-in requirement for these codes helps to ensure that accidental writes do not take place, for example on power-up or -down.
- The WR bit is then set high and writing actually commences.
- The write completion is signalled by the setting of bit EEIF in EECON1.
- EECON2 is not a physical register.



Writing to the EEPROM Data Memory

- To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific sequence to initiate the write for each byte.

```
BSF     STATUS, RP0     ; Bank 1
BCF     INTCON, GIE     ; Disable INTs.
BSF     EECON1, WREN    ; Enable Write
MOVLW   55h ;
MOVWF   EECON2          ; Write 55h
MOVLW   AAh ;
MOVWF   EECON2          ; Write AAh
BSF     EECON1,WR       ; Set WR bit begin write
BSF     INTCON, GIE     ; Enable INTs
```

- The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. We strongly recommend that interrupts be disabled during this code segment.

Writing to the EEPROM Data Memory

- WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM. The user should keep the WREN bit clear at all times, except when updating EEPROM. The WREN bit is not cleared by hardware.

```
BSF     STATUS, RP0     ; Bank 1
BCF     INTCON, GIE     ; Disable INTs.
BSF     EECON1, WREN    ; Enable Write
MOVLW   55h ;
MOVWF   EECON2          ; Write 55h
MOVLW   AAh ;
MOVWF   EECON2          ; Write AAh
BSF     EECON1,WR       ; Set WR bit begin write
BSF     INTCON, GIE     ; Enable INTs
```

- At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. EEIF must be cleared by software.

Example

- Write h’E3’ to location 0X03 of EEPROM.

```
MOVLW     0X03
MOVWF     EEADR
MOVLW     h’E3’
MOVWF     EEDATA
BSF       STATUS, RP0      ; Bank 1
BCF       INTCON, GIE      ; Disable INTs.
BSF       EECON1, WREN     ; Enable Write
MOVLW     55h ;
MOVWF     EECON2           ; Write 55h
MOVLW     AAh ;
MOVWF     EECON2           ; Write AAh
BSF       EECON1,WR        ; Set WR bit begin write
BSF       INTCON, GIE      ; Enable INTs
```

DATA EEPROM INTERRUPT

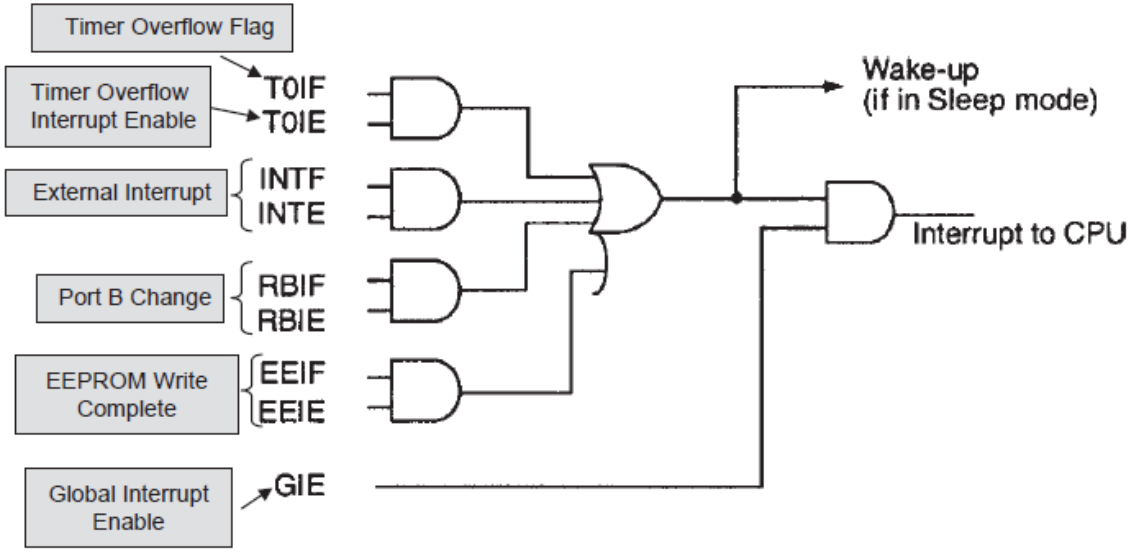
- At the completion of a data EEPROM write cycle, flag bit EEIF (EECON1<4>) will be set. The interrupt can be enabled/disabled by setting/clearing enable bit EEIE (INTCON<6>)

EECON1

U-0	U-0	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
—	—	—	EEIF	WRERR	WREN	WR	RD
bit 7			bit 4				bit 0

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

Interrupt Logic



INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER FILE MAP - PIC16F84A

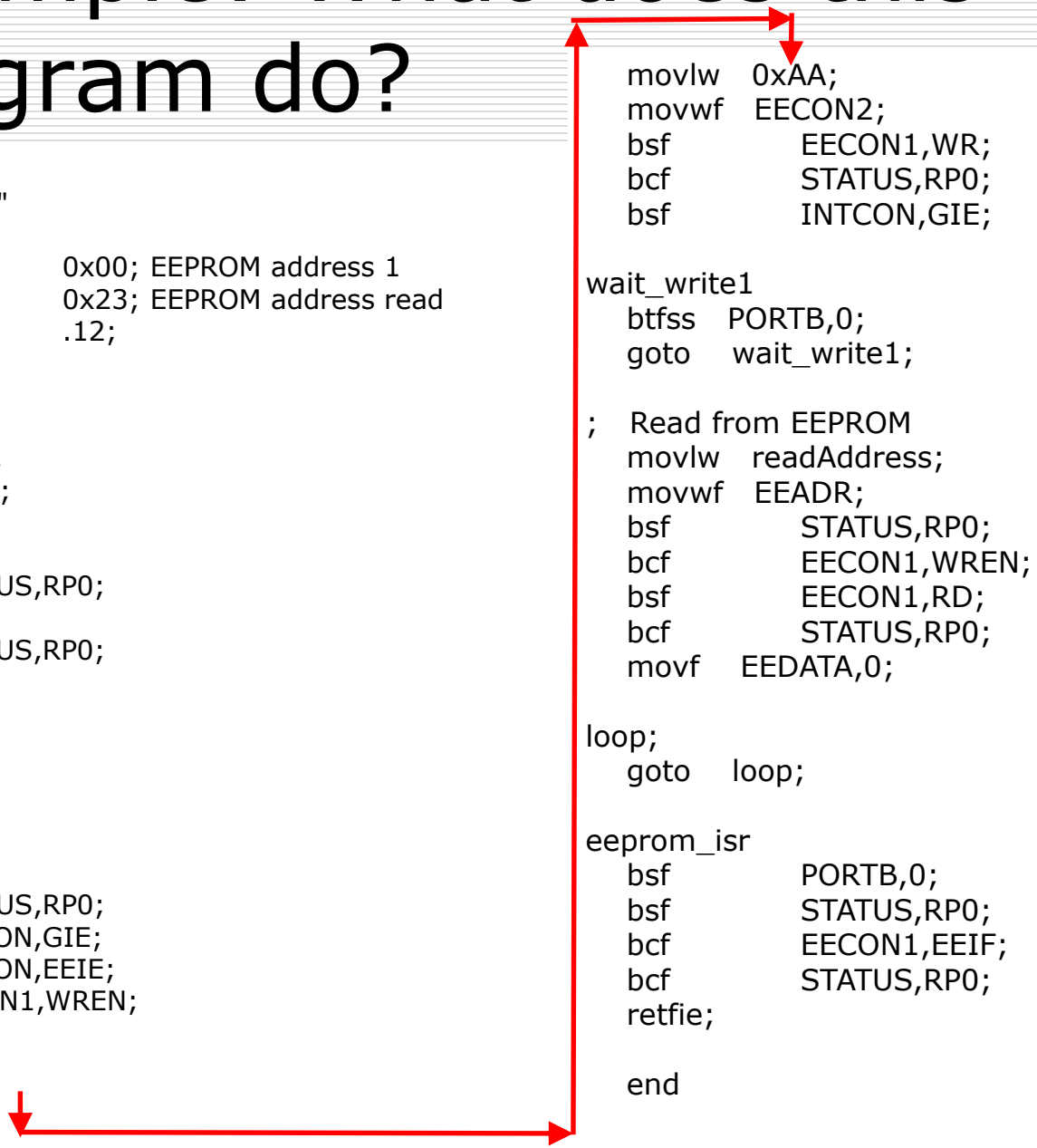
File Address	Bank 0	Bank 1	File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch
4Fh			CFh
50h			D0h
7Fh			FFh

Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

Example: What does this program do?

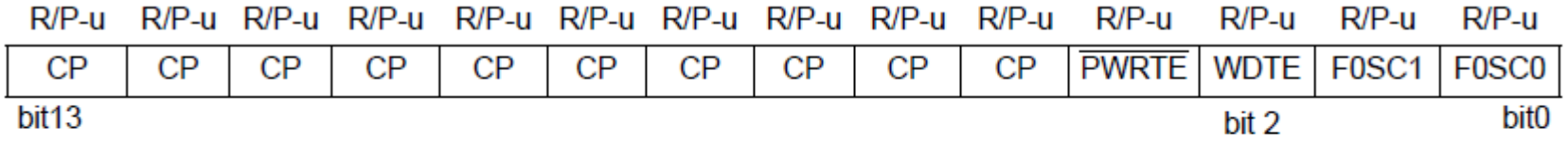
```
 list p=16f84a;
 include "p16f84a.inc"

address1    equ         0x00; EEPROM address 1
readAddress equ         0x23; EEPROM address read
value1      equ         .12;

    org 0;
    goto    main;
    org         0x04;
    goto    eeprom_isr;

main;
    bsf         STATUS,RP0;
    clrf    TRISB;
    bcf         STATUS,RP0;
    clrf    PORTB;

    movlw   address1;
    movwf   EEADR;
    movlw   value1;
    movwf   EEDATA;

    bsf         STATUS,RP0;
    bcf         INTCON,GIE;
    bsf         INTCON,EEIE;
    bsf         EECON1,WREN;

    movlw   0x55;
    movwf   EECON2;
    movlw   0xAA;
    movwf   EECON2;
    bsf         EECON1,WR;
    bcf         STATUS,RP0;
    bsf         INTCON,GIE;

wait_write1
    btfss   PORTB,0;
    goto    wait_write1;

;   Read from EEPROM
    movlw   readAddress;
    movwf   EEADR;
    bsf         STATUS,RP0;
    bcf         EECON1,WREN;
    bsf         EECON1,RD;
    bcf         STATUS,RP0;
    movf    EEDATA,0;

loop;
    goto    loop;

eeprom_isr
    bsf         PORTB,0;
    bsf         STATUS,RP0;
    bcf         EECON1,EEIF;
    bcf         STATUS,RP0;
    retfie;

    end
```



The Watchdog Timer

- The primary function of the Watchdog Timer (WDT) is
  - to reset the microcontroller, in the event of a software malfunction, by resetting the device if it has not been cleared in software.
  - to wake the device from Sleep mode.
- The Watchdog Timer is a free running On-Chip RC Oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin.
- That means that the WDT will run even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction.
- During normal operation, a WDT time-out generates a device RESET.
- If the device is in SLEEP mode, a WDT wake-up causes the device to wake-up and continue with normal operation.
- The WDT can be enabled by programming configuration bit WDTE as a '1' by using the relevant coniguration bit

__config _CP_OFF&_WDT_ON&_XT_OSC



PIC16F84A CONFIGURATION WORD

WDTE: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

WDT OPERATION

- When enabled, the WDT will increment until it overflows or “times out”.
- A WDT time-out will force a device Reset, except during Sleep modes.
- To prevent a WDT Time-out Reset, the user must periodically clear the Watchdog Timer using the instructions, CLRWDT.
- If the WDT times out during Sleep modes, the device will wake-up and continue code execution from where the CLRWDT instruction was executed.
- In either case, the TO_bit (4_bit of STATUS register) will be set to indicate that the device Reset or wake-up event was due to a WDT time-out.
- If the WDT wakes the CPU from Sleep mode, the SLEEP status bit (bit_3 of status register) will also be set to indicate that the device was previously in a Power-Saving mode

STATUS REGISTER

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

WDT PERIOD

- The WDT has a nominal time-out period of 18 ms, (with no prescaler).
- The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs).
- If longer time-out periods are desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION_REG register. Thus, time-out periods up to 2.3 seconds can be realized.
- The CLRWDT and SLEEP instructions clear the WDT and the postscaler (if assigned to the WDT) and prevent it from timing out and generating a device RESET condition.
- The TO bit in the STATUS register will be cleared upon a WDT time-out.

For exp; 1/128 WDT_rate,
time_out period= 128*18ms=2,3 second.

OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Example

- Write a program to increment the value of PORTB from 00. Every 1152 ms, PORTB get started counting again from 00.