TÀI LIỆU THAM KHẢO CHO PIC16F877A

CHƯƠNG1
GIỚI THIỆU TỔNG QUAN VỀ HỌ VI ĐIỀU KHIỂN PIC

Hiện nay trong các máy móc công nghiệp và các thiết bị phục vụ sinh hoạt cho cuộc Sống hầu hết đều ứng dụng rộng rãi các thiết bị điện tử ,mà bộ xử lý trung tâm là các con Chip vi điều khiển hết sức thông minh đặc biệt các Chip này có thể lập trình được Bởi con người! Vì vậy chúng ta cần phải nghiên cứu và phát triển nó.

Trên thị trường hiện nay phổ biến rất nhiều loại vi điều khiển phong phú về chủng loại và giá cả thì tương đối rẻ phù hợp với điều khiện ở Việt Nam trong đó phổ biến có các loại như : MCS51 ; AVR của ATMEL , PIC của MICROCHIP , PSOC của CYPRESS MICRO SYSTEM…

Hiện nay với sự đa dạng và nhiều chủng loại khác nhau của PIC đặc biệt là tính ổn định của chúng đã làm cho nhiều người thích thú và ưa chuộng vì vậy chúng đã được ứng dụng rộng rãi trên toàn thế giới.

Cụm từ PIC được viết tắt từ cụm từ : PERIPHERAL INTERFACE CONTROLLER (Bộ Điều Khiển giao tiếp các thiết bị ngoại vi).Khác với các bộ vi xử ,bộ vi điều khiển được tích hợp toàn bộ như RAM , ROM , các PORTS truy xuất ,giao tiếp ngoại vi trực tiếp trên một con chíp hết sức nhỏ gọn.

PIC16F877A là một vi điều khiển có kiến trúc HARVARD (bộ nhớ chương trình và bộ nhớ dữ liệu được truy xuất độc lập với nhau) sử dụng 14 bit cho các lệnh , và tập lệnh của nó chỉ hầu hết chỉ có một WORD.

CHƯƠNG2
CẤU TRÚC PHẦN CỨNG CỦA PIC16F877A

I)BỘ NHỚ CHƯƠNG TRÌNH CỦA PIC

Không gian bộ nhớ chương trình của PIC khác nhau tuỳ thuộc vào từng loại
Sau đây là một số ví dụ:

-16C711,16F84 có 1024(1K)
-16F877A có 8192(8K)
-17C766 có 16384(16K)

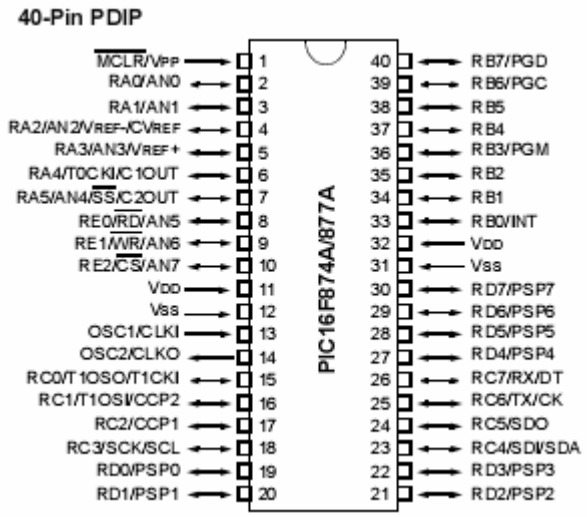
II)BỘ NHỚ DỮ LIỆU CỦA PIC

Các thanh ghi đa mục đích cho người dùng của PIC là các ô nhớ RAM . Mỗi thanh ghi này có độ rộng 8 bít cho tất cả các PIC

Sau đây lả một vài ví dụ:

-12C508 có 25 Bytes RAM
-16C71C có 36 Bytes RAM
-16F877A có 368 Bytes(plus 256 Bytes of nonvolatile EEPROM)

III)CÁC CHÂN CỦA PIC 16F877A



1)CÁC CHÂN NGUỒN

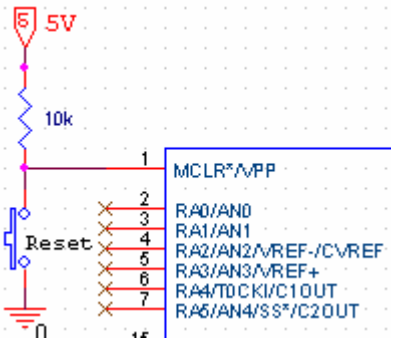
Trong các sơ đồ của mạch 8051 thường kí hiệu chân cấp nguồn là VCC , còn chân nối mass là GND . Còn đối với PIC thì ngược lại thay VCC = VDD còn chân GND = VSS
Trong PIC 16F877A trên hình vẽ ta có thể thấy có tất cả 4 chân cấp nguồn như sau:

- Chân 11 , 32 là các chân VDD (+5v)
- Chân 12 , 31 là các chân VSS (0v)

2)CHÂN RESET

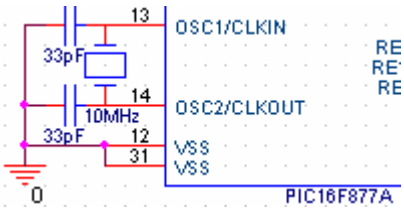
Trên hình ta thấy chân số 1(MCLR) chính là chân RESET của PIC , chân này có nhiệm vụ khởi động lại chip khi chân này được tích cực.

Chân RESET của PIC tích cực ở mức thấp đều này trái ngược hoàn toàn với họ 8051



3)MẠCH DAO ĐỘNG

Trên hình vẽ ta thấy 2 chân 13(OSC1) và chân 14(OSC2) là 2 chân dao động. Tốc độ dao động được xác định thông qua tần số dao động của bộ tạo dao động
Sơ đồ mạch dao động như hình vẽ sau:



4)CỔNG XUẤT NHẬP

+PORT A và thanh ghi TRIS A:
Cổng A có 6 bit thực hiện chức năng vào ra theo 2 chiều việc xác định hướng xuất nhập dược thực hiện thông qua thanh ghi TRIS A.
Việc đưa 1 bit trong thanh ghi TRIS A lên 1 cũng đồng nghĩa với việc đặt chân tương ứng của cổng A là chân nhập dữ liệu.
Việc xoá 1 bit trong thanh ghi TRIS A xuống 0 cũng đồng nghĩa với việc đặt chân tương ứng của cổng A là chân xuất dữ liệu.
Chân RA4/TOCKI là chân đa mục đích với việc vừa là chân xuất nhập vừa là đầu vào của bộ đếm TIMER0 .Đầu vào của chân RA4 là một trigger schmitt
nên có cực máng hở trong chế độ nhập chúng ta cần gắn thêm điện trở kéo dương cho nó.
Các chân khác trong PORT A còn là đầu vào của tín hiệu tương tự trong bộ chuyển đổi ADC . Sự hoạt động của các chân trong chế độ này là việc điều kiển thích hợp các bít trong thanh ghi ADCON1 và CMCON.

TABLE 4-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-/CVREF	bit 2	TTL	Input/output or analog input or VREF- or CVREF.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI/C1OUT	bit 4	ST	Input/output or external clock input for Timer0 or comparator output. Output is open-drain type.
RA5/AN4/$\overline{SS}$/C2OUT	bit 5	TTL	Input/output or analog input or slave select input for synchronous serial port or comparator output.

Legend: TTL = TTL input, ST = Schmitt Trigger input

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: POR, BOR	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

+PORT B và thanh ghi TRIS B:

Cổng B có 8 bít xuất nhập theo 2 chiều ,việc chọn chức năng xuất hoặc nhập được điều khiển thông qua thanh ghi TRIS B cũng tương tự như với PORTS A
Ba chân của PORT B là các chân đa chức năng(RB3/PGM,RB6/PGC/RB7/PGD) với các ứng dụng như trong mạch gỡ rối và chương trình điện áp thấp
Mỗi chân của PORT B đều có các điện trở kéo dương ở bên trong có giá trị khoảng 47K có thể cho phép hoạt động ở chế độ này thông qua việc set bit RBPU trong thanh ghi OPTION
Việc điện trở kéo sẽ bị khoá ngay khi PORT B chuyển sang chế độ xuất dữ liệu hoặc khi VĐK mới khởi động
Bốn chân của PORT B là các chân từ RB4 đến RB7 còn là các chân phục vụ ngắt, nếu 1 trong các chân đó được định hình là đầu vào thì nó có thể là nguyên nhân cho 1 ngắt phát sinh
Khi một ngắt được tạo ra cũng đồng thời cờ RBIF(INTCON.0) được set lên 1, và nó có thể đánh thức VĐK đang ở chế độ ngủ(SLEEP)

TABLE 4-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit 0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit 1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit 2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit 3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit 5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit 6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit 7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode or in-circuit debugger.

3: Low-Voltage ICSP Programming (LVP) is enabled by default which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

+ PORT C và thanh ghi TRIS C:

PORTC có tất cả 8 chân đa mục đích với các chức năng như : xuất nhập dữ liệu, đặc biệt 2 chân 18(SCL) và 23(SDA) là 2 chân thực hiện chức năng giao tiếp với ngoại vi thông qua chuẩn I2C
Thanh ghi TRISC cũng tương tự như trên làm nhiệm vụ định nghĩa các chân tương ứng là cổng vào hay cổng ra

TABLE 4-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit 0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit 1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit 2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit 4	ST	RC4 can also be the SPI data in (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit 5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit 6	ST	Input/output port pin or USART asynchronous transmit or synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin or USART asynchronous receive or synchronous data.

Legend: ST = Schmitt Trigger input

+PORT D và thanh ghi TRIS D - PORT E và thanh ghi TRIS E:

Hai PORT này đều có 8 chân đa mục đích nhưng chủ yếu vẫn là để xuất nhập đữ liệu còn các ứng dụng khác chung ta sẽ không xét ở đây
Các bạn có thể tham khảo thêm trong các hình dưới đây:

TABLE 4-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit 0	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 0.
RD1/PSP1	bit 1	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 2.
RD3/PSP3	bit 3	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 3.
RD4/PSP4	bit 4	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 4.
RD5/PSP5	bit 5	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 5.
RD6/PSP6	bit 6	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 6.
RD7/PSP7	bit 7	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit 0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected).
RE1/$\overline{WR}$/AN6	bit 1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected).
RE2/$\overline{CS}$/AN7	bit 2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

CHƯƠNG 3
CÁC THANH GHI CÓ CHỨC NĂNG ĐẶC BIỆT

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
accesses 70h-7Fh	7Fh	accesses 70h-7Fh	F0h FFh	accesses 70h-7Fh	170h 17Fh	accesses 70h - 7Fh	1F0h 1FFh

Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.

2: These registers are reserved; maintain these registers clear.

Các thanh ghi này có chức năng điều khiển các hoạt động và các khối giao tiếp ngoại vi của vi điều khiển

I) THANH GHI STATUS

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

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 0 : là một cờ báo tràn mỗi khi có nhớ từ bit 7 trong phép cộng hoặc có mượn trong phép trừ

Bit 1 : là cờ nhớ phụ bị tác động khi thực hiện phép toán vượt quá 4 bít thấp

Bit 2 :

Z=1 nếu kết quả phép toán bằng 0

Z=0 nếu kết quả phép toán khác 0

Bit 3 :

PD=1 sau khi bật nguồn hoặc bởi lệnh CLRWDT

PD=0 khi lệnh SLEEP được thực thi

Bit 4 :

TO=1 nếu có lệnh SLEEP thực thi hoặc lệnh CLRWDT hoặc sau khi bật nguồn

T0=0 nếu bộ WDT bị chàn

Bit 6-5:

00: chọn bank 0

01: chọn bank 1

10: chọn bank 2

11: chọn bank 3

Bit 7:

1: chọn bank 2,3

0: chọn bank 0,1

II) THANH GHI OPTION_REG:

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Bit 0-1-2 : dùng để chọn giá trị cho bộ chia tần cho TIMER0 hoặc WDT

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

Bit 3 :

PSA =1: bộ chia tần dùng cho WDT

PSA =0: bộ chia tần dùng cho TIMER0

Bit 4 :

TOSE =1: chọn sường xuống là sường tác động lên chân RA4

TOSE =0: chọn sường lên là sường tác động lên chân RA4

Bit 5 :

TOCS =1: chọn xung đếm trong TIMER0 là xung trên chân RA4

TOCS =0: chọn xung đếm trong TIMER0 là xung nội

Bit 6 :

INTEDG =1: xảy ra ngắt khi chân RB0 có sườn lên

INTEDG =0: xảy ra ngắt khi chân RB0 có sườn xuống

Bit 7 :

RBPU =1: cấm cho phép điện trở kéo dương PORTB

RBPU =0: cho phép điện trở kéo dương PORTB

III) THANH GHI INTCON:

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

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	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

Bit 0: Cờ báo ngắt cho các chân RB4-RB7

RBIF =1: xuất hiện ít nhất một trong các ngắt tại các chân RB4-RB7

RBIF =0: không xuất hiện ngắt tại các chân RB4-RB7

Bit 1: Cờ ngắt cho chân RB0

INTF =1: xuất hiện ngắt trên chân ngắt ngoài RB0

INTF =0: không xuất hiện ngắt trên chân ngắt ngoài RB0

Bit 2: Cờ ngắt cho bộ TIMER0

TMR0IF =1: xảy ra tràn trong thanh ghi TMR0

TMR0IF =0: chưa xảy ra tràn trong thanh ghi TMR0

Bit 3:

RBIE =1: cho phép ngắt trên các chân RB4-RB7
RBIE =0: cấm ngắt trên các chân RB4-RB7

Bit 4:

INTE =1: cho phép ngắt trên chân RB0
INTE =0: cấm ngắt trên chân RB0

Bit 5:

TMR0IE =1: cho phép ngắt bằng bộ TIMER0
TMR0IE =0: cấm ngắt bằng bộ TIMER0

Bit 6:

PEIE =1: cho phép ngắt phục vụ cho thiết bị ngoại vi
PEIE =0: cấm các ngắt phục vụ cho thiết bị ngoại vi

Bit 7:

GIE =1: cho phép tất cả các ngắt được thực hiện
GIE =0: cấm tất cả các ngắt không được thực hiện

CHÚ Ý:

Vị trí của bộ nhớ dữ liệu được chia thành 4 BANK thanh ghi(các khối , các vùng) ở mỗi thời điểm , chúng ta chỉ có thể truy xuất trên 1 BANK thanh ghi nào đó mà thôi .Việc
chọn BANK nào thông qua việc điều khiển các bit 5-6-7 của thanh ghi STATUS

Chúng ta thấy rằng trong PIC còn rất nhiều các thanh ghi chức năng khác nhưng chúng ta sẽ không bàn đến nó ở đây.Nếu các bạn cần mở rộng kiến thức thì có thể tham khảo thêm trong DATASHEET của 16F877A

CHƯƠNG 4
CÁC ỨNG DỤNG CƠ BẢN CỦA PIC 16F877A

I) ĐẾM VÀ ĐỊNH THỜI:

1)BỘ ĐỊNH THỜI TIMER0

Timer0 là một trong 3 bộ định thời của PIC16F877A , mỗi một định thời thì sử dụng
các thanh ghi chức năng khác nhau với nhiệm vụ và cách thức hoạt động cũng khác nhau

+CÁC THANH GHI DÙNG TRONG TIMER0

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
01h,101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

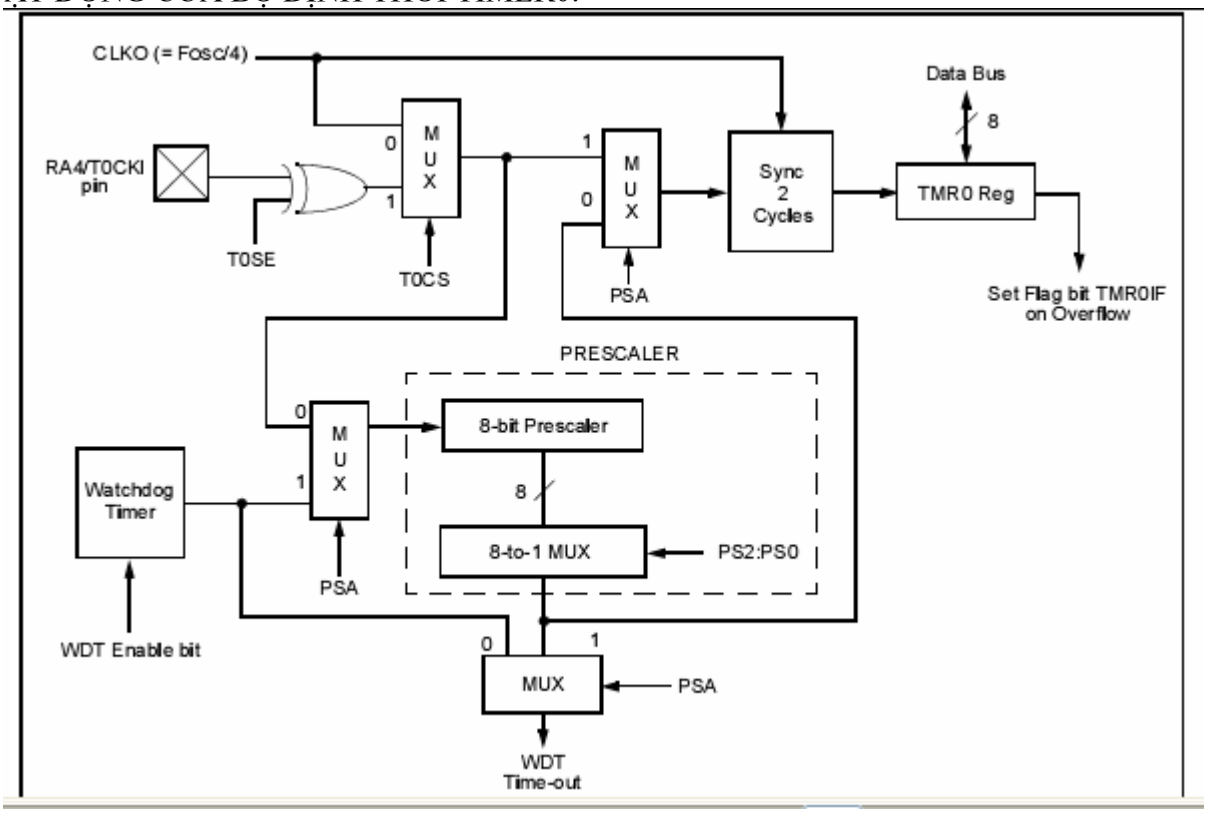
Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

THANH GHI OPTION: Là thanh ghi cho phép đọc ghi dùng để điều khiển thiết lập cấu hình cho Timer0

THANH GHI INTCON:Là thanh ghi chứa cờ ngắt của Timer0

THANH GHI TMR0: Là thanh ghi 8 bit ,mỗi lần có xung tác động thì giá trị của thanh ghi sẽ tăng lên 1 đơn vị cho đến khi tràn thì thanh ghi sẽ chở về 0

+HOẠT ĐỘNG CỦA BỘ ĐỊNH THỜI TIMER0:



Nhìn sơ đồ khối của TIMER0 ta có thể thấy nó hoạt động ở 2 chế độ

- Chế độ định thời: ở chế độ này chúng ta cần chọn xung tác động là xung nội(TOCS =0) lúc này xung tạo ra bởi bộ giao động sau khi được chia 4 sẽ đi qua bộ chia tần cung cấp cho Timer0 đếm .Sau khi một xung được đếm giá trị của thanh ghi TMR0 sẽ tăng lên 1 đơn vị , khi xảy ra tràn thì cờ TMR0IF sẽ được set lên 1.
- Chế độ đếm: ở chế độ này chúng ta cần chọn xung tác động là xung ngoài(TOCS =1) Timer0 sẽ lấy xung từ bên ngoài thông qua chân RA4 thông qua bộ chia tần sẽ cung cấp cho Timer0 tương tự như trên.Việc chọn kiểu xung tác động thông qua việc điều khiển bit T0SE.
- Chế độ WDT: chúng ta sẽ không tìm hiểu vấn đề này….

2)BỘ ĐỊNH THỜI TIMER1:

Bộ định thời Timer1 là bộ định thời 16 bit cũng với 2 chức năng cơ bản như Timer0

+CÁC THANH GHI DÙNG TRONG TIMER1:

TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

THANH GHI T1CON: Là thanh ghi thiết lập cấu hình hoạt động cho Timer1

Bit 0:

TMR1ON =1: cho phép Timer1 hoạt động

TMR1ON =0: không cho phép Timer1 hoạt động

Bit 1:

TMR1CS =1: dùng nguồn xung từ bên ngoài thông qua chân RC0 (sường dương)

TMR1CS =0: dùng nguồn xung từ bộ tạo dao động

Bit 2:

khi TMR1CS =1:

T1SYNC=1: không sử dụng xung ngoài là xung đồng bộ

T1SYNC=0: cho phép sử dụng xung ngoài là xung đồng bộ

khi TMR1CS =0:

Bit này không được sử dụng

Bit 3:

T1OSCEN=1: cho phép bộ tạo dao động hoạt động

T1OSCEN=0: không cho phép bộ tạo dao động hoạt động

Bit 5 -4 :

Thiết lập giá trị cho bộ chia tần

```
11 = 1:8 prescale value
10 = 1:4 prescale value
01 = 1:2 prescale value
00 = 1:1 prescale value
```

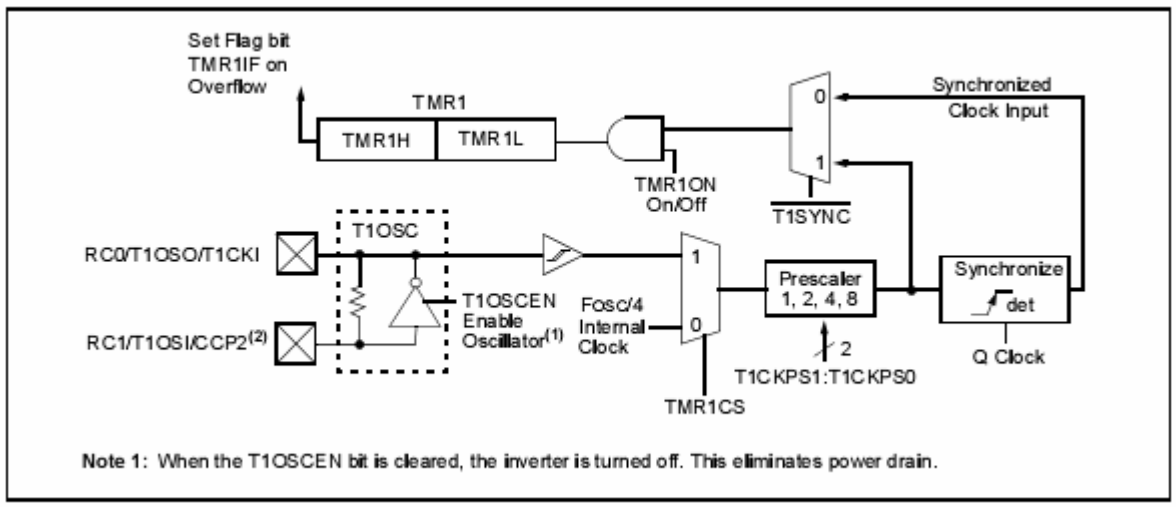
THANH GHI TMR1: Là thanh ghi lưu trữ giá trị định thời 16 bit được tạo thành từ 2 thanh ghi 8 bit TMR1L – TMR1H

THANH GHI PIR1: Là thanh ghi chứa cờ tràn TMR1IF của Timer1

THANH GHI PIE1: Là thanh ghi chứa bit TMR1IE cho phép ngắt Timer1 hoạt động

+HOẠT ĐỘNG CỦA BỘ ĐỊNH THỜI TIMER1

FIGURE 6-2: TIMER1 BLOCK DIAGRAM



Nhìn vào sơ đồ khối ta thấy Timer1 có 2 chức năng cơ bản sau:

- Chế độ định thời: trước hết cần phải cho Timer1 hoạt động bằng cách set bit TMR1ON
  sau đó chọn chế độ sử dụng xung nội(TMR1CS =1).Xung từ bộ tạo dao động sẽ được chia 4 sau đó đưa qua bộ chia tần cung cấp cho Timer1 đếm đồng thời giá trị của thanh ghi TMR1 sẽ tăng lên 1 đơn vị cho đến khi tràn và cờ tràn TMR1IF=1.
- Chế độ đếm: khi sử dụng chế độ này chung ta cần phải set bit TMR1CS =1, nguồn xung từ bên ngoài có thể lấy từ 2 chân RC0 - RC1 thông qua việc thiết lập bit T1OSCEN , nếu bit T1SYNC=0 thì xung tác động từ bên ngoài sẽ đồng bộ với xung dao động bên trong , quá trình đồng bộ xảy ra sau khi xung đi qua bộ chia tần

3)BỘ ĐỊNH THỜI TIMER2:

Timer2 là bộ định thời 8 bit tương tự như Timer1 nhưng lại có tới 2 bộ chia tần có thể được dùng trong ứng dụng để điều chế độ rộng xung (PWM)

CÁC THANH GHI DÙNG TRONG TIMER2:

TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Timer2 Module's Register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.
Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

THANH GHI T2CON: Là thanh ghi thiết lập cấu hình cho Timer2

Bit 1- 0: thiết lập gia trị cho bộ chia tần Prescale

00 = 1:1
01 = 1:4
1x = 1:16

Bit 2:

TMR2ON=1: cho phép sử dụng Timer2
TMR2ON=0: không cho phép sử dụng Timer2

Bit 6- 3: thiết lập giá trị cho bộ chia tần Postcale

0000 = 1:1 postscale
0001 = 1:2 postscale
0010 = 1:3 postscale
•
•
•
1111 = 1:16 postscale

THANH GHI PIR1: chứa cờ tràn TMR2IF của Timer2

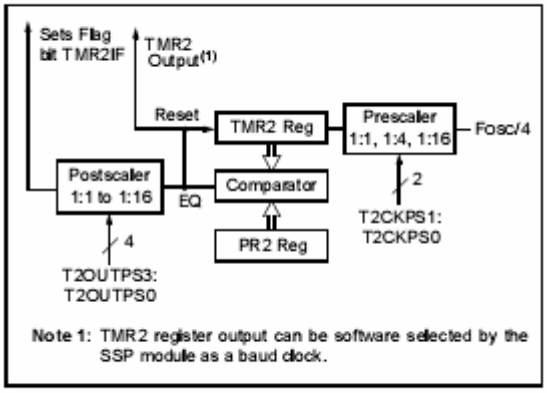
THANH GHI PIE1: chứa cờ cho phép ngắt TMR2IE của Timer2

THANH GHI PR2: ứng dụng trong PWM

THANH GHI TMR2: lưu trữ giá trị định thời 8 bit cho Timer2

+HOẠT ĐỘNG CỦA BỘ ĐỊNH THỜI TIMER2

FIGURE 7-1: TIMER2 BLOCK DIAGRAM



- Chế độ định thời: nguồn xung từ bộ tạo dao động sau khi được chia 4 sẽ được đưa bộ chia tần Prescale nạp vào thanh ghi TMR2 , khi xảy ra tràn bit TMR2IF=1.
- Chế độ PWM: trước hết chúng ta nạp giá trị cho thanh ghi PR2 sau đó khi giá trị của thanh ghi TMR2 sẽ được so sánh với giá trị của thanh ghi PR2 nếu chúng bằng nhau thì thanh ghi TMR2 sẽ được Reset đồng thời giá trị các chân PWM sẽ thay đổi… chúng ta sẽ đề cập đến vấn đề này ở phần sau.

II) CÁC NGẮT THÔNG DỤNG:

Như chúng ta đã biết, vi điều khiển tại mỗi thời điểm nó chỉ có thể làm một công việc nhất định. Nhưng trong thực tế thì lại khác, người lập trình lại muốn vi điều khiển đang làm công việc này lại tự động chuyển sang làm công viec khác ,vậy làm thế nào để vi điều khiển làm được đó? đơn giản là nó sẽ sử dụng cơ chế gọi là : Ngắt

1)CÁC NGẮT CỦA PIC 16F877A

+Ngắt tràn Timer0
+Ngắt do có thay đổi trạng thái trên các chân từ RB4- RB7
+Ngắt ngoài trên chân RB0
+Ngắt chuyển đổi ADC đã hoàn tất
+Ngắt bộ chuyển đệm RS 232 chống
+Ngắt do dữ liệu nhận từ RS232 đã sẵn sàng
+Ngắt tràn Timer1
+Ngắt tràn Timer2
+Ngắt do có capture hay compare trên chân CCP1
+Ngắt do có capture hay compare trên chân CCP2
+Ngắt do có hoạt động SPI hay I2C
+Ngắt do có dữ liệu vào cổng parallel slave
+Ngắt do ghi vào EPROM hoàn tất
+Ngắt do xung đột BUS
+Ngắt do kiểm tra bằng nhau comparator

Ta thấy rằng Pic có rất nhiều ngắt ứng dụng trong nhiều chức năng khác nhau nhưng ở đây chúng ta chỉ đề cập đến một số ngắt cơ bản sau:

+NGẮT DO CÁC TIMER HOẶC NGẮT NGOÀI :

Về cơ bản hoạt động của các ngắt Timer hoặc ngắt ngoài hoạt động như sau:

- Xung tạo ra do bộ tạo dao động hoặcnguồn xung bên ngoài sẽ được cung cấp cho các thanh ghi định thời tương ứng của các bộ định thời , khi các bộ định thời xảy ra tràn cờ ngắt tương ứng được bật và một yêu cầu ngắt được phục vụ lúc này vi điều khiển sẽ tạm ngừng công việc hiện tại, hoàn thành lệnh hiện thời ngay tức khắc để nhảy vào chương trình phục vụ ngắt ISR. Khi đó bộ đếm chương trình PC sẽ được đẩy vào ngăn xếp STACK và đồng thời bit GIE =0 chương trình rẽ nhánh đến địa chỉ vectơ ngắt 0x04 ,tại đây vi điều khiển sẽ thực hiện các yêu cầu mà ngắt đòi hỏi .

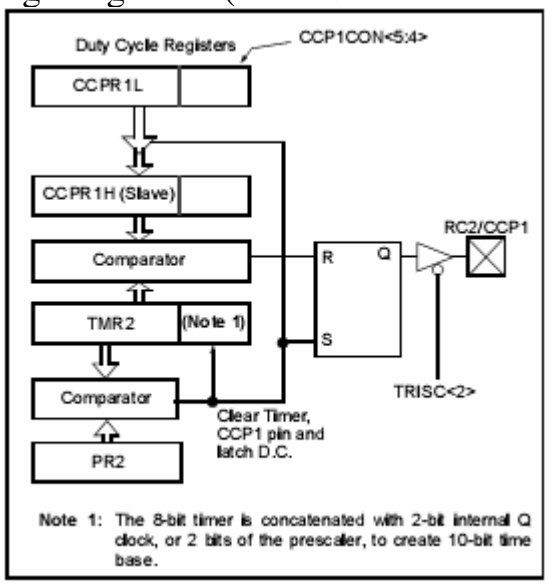
- Việc thiết lập cấu hình cho các ngắt sẽ thông qua các bit của các thanh ghi chức năng như sau:

+Đối với Timer0 : Bít điều khiển là bit TMR0IE(INTCON.5)
+Đối với Timer1 : Bít điều khiển là bit TMR1IE(PIE.0)
+Đối với Timer2 : Bít điều khiển là bit TMR2IE(PIE.1)
+Đối với ngắt ngoài: Bít điều khiển là bit INTE(INTCON.4)
+Đối với ngắt do các chân RB4 – RB7: Bít điều khiển là bit RBIE(INTCON.3)

Chú ý: Trước khi thiết lập các ngắt chúng ta cần phải cho phép ngắt toàn cục thông qua việt cho bit GIE =1(INTCON.7)

III) ĐIỀU CHẾ ĐỘ RỘNG XUNG(PWM)

Một trong những tính năng quan trọng của PIC được ứng dụng rất nhiều đó là điều chế độ rộng xung PWM(Pulse Width Modulation)



THANH GHI ĐIỀU KHIỂN CCP1CON/CCP2CON:

REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

Bit 3- 0: chọn chế độ làm việc

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

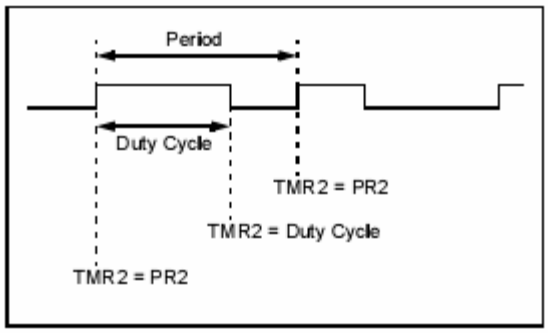
Bit 5 - 4:????

CCPxX:CCPxY: PWM Least Significant bits
Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL

Quá trình hoạt động của chức năng PWM như sau:

- Với PIC 16F877A chúng ta có 2 chân điều chế độ rộng xung là CCP1 và CCP2 ,sau chọn chức năng PWM bằng cách điều khiển 4 bít thấp của thanh ghi CCPxCON , chúng ta sẽ nạp giá trị cho thanh ghi PR2 và thanh ghi CCPRx . Khi Timer2 hoạt động giá trị của thanh TMR2 sẽ tăng cho đến khi bằng giá trị của thanh ghi PR2 lúc này chân CCPx tương ứng sẽ lên mức 1 đồng thời thanh ghi TMR2 sẽ bị xoá về giá trị ban đầu.Mức 1 tại chân CCPx sẽ được dữ cho đên khi giá trị thanh ghi TMR2 bằng giá tri thanh ghi CCPRx sau đó chân CCPx lại trở về 0 cho đến khi giá trị thanh ghi TMR2=PR2 cứ như vậy quá trình sẽ lặp lại như ban đầu..
- Như vậy chúng ta có thể rút ra như sau:
  Chu kì xung là khoảng thời gian để giá trị thanh ghi TMR2 tăng đến giá trị thanh ghi PR2
  Khoảng xung dương là khoảng thời gian để thanh ghi TMR2 tăng đến giá trị thanh ghi CCPRx
  Để hiểu rõ hơn chúng ta có thể xem trong sơ đồ sau:

FIGURE 8-4: PWM OUTPUT



CHƯƠNG 5:
TẬP LỆNH CCS C CHO PIC 16F877A

I) GIỚI THIỆU VỀ TRÌNH DỊCH CCS C:

- CCS là trình biên dịch dùng ngôn ngữ C lập trình cho VĐK . Đây là ngôn ngữ lập trình đầy sức mạnh , giúp bạn nhanh chóng trong việc viết chương trình hơn là Assembly
- CCS chứa rất nhiều hàm phục vụ cho mọi mục đích và có rất nhiều cách lập trình mà cho cùng 1 vấn đề với tốc độ thực thi và độ dài chương trình khác nhau .Sự tối ưu là do kĩ năng lập trình của mỗi người
- CCS cung cấp các công cụ tiện ích giám sát hoạt động chương trình như:

+ C/ASM list: cho phép mã ASM của file bạn biên dịch , giúp bạn quản lý và nắm rõ cách thức nó được sinh ra , là công cụ rất quan trọng giúp bạn có thể gỡ rối chương trình
+ SYMBOL: hiển thị bộ nhớ cấp phát cho từng biến , giúp bạn quản lý bộ nhớ các biến của của chương trình
+ CALLTREE: hiển thị phân bổ bộ nhớ

II) CHỈ THỊ TIỀN XỬ LÝ :

1) #INCLUDE :

- Cú pháp: #include<filename>

Filename: tên file cho thiết bị *.h, *.c . Chỉ định đường dẫn cho trình biên địch , luôn phải có để khai báo chương trình viết cho VĐK nào và phảI luôn đặt ở dòng đầu tiên

VD: #include<16F877A.H>

2) #BIT :

- Cú pháp: #BIT name = x.y

Name: tên biến

X: biến C(8,16,32…bit) hay hắng số địa chỉ thanh ghi

Y: vị trí của bit trong x

Tạo biến 1bit đặt ở byte x vị trí y tiện dùng kiểm tra hay gán giá trị cho thanh ghi

VD : #Bit TMR1IF = 0x0B.2;

3) #BYTE :

- Cú pháp: #BYTE name = x

Name: tên biến

X:địa chỉ

Gán tên biến name cho địa chỉ x , name thường dùng để gán cho các thanh ghi

VD : #Byte portb = 0x06;

4) #DEFINE :

- Cú pháp: #DEFINE name text

Name: tên biến

Text : chuỗi hay số

VD : #Define A 12345

5) #USE :

- Cú pháp: #USE delay(clock = speed)

Speed: tốc độ dao động của thạch anh
Có chỉ thị này chúng ta mới dùng được hàm delay_ms hoặc delay_us
VD: #USE delay(clock = 4000000);

6) #USE FAST_IO :

- Cú pháp: #USE FAST_IO(PORT)

Port : các cổng vào ra của PIC(từ A-G)
Dùng cái này chúng ta có thể điều chỉnh các port với chỉ 1 lệnh
VD: # USE FAST_IO(A);

III) CÁC HÀM DELAY :

1) DELAY_MS(TIME)
Time: giá trị thời gian cần tạo trễ
VD : delay_ms(1000); // trễ 1s

2) DELAY_US(TIME)
Time: giá trị thời gian cần tạo trễ
VD : delay_us(1000); // trễ 1ms
Hàm delay này không sử dụng bất cứ Timer nào cả mà chỉ là 1 nhóm lệnh vô nghĩa thực hiện trong khoảng thời gian bạn đã định sẵn
Trước khi sử dụng các hàm này cần phảI khai báo tiền định #use_delay(….)

IV) CÁC HÀM VÀO RA TRONG CCS C

1) Output_low(pin) – Output_high(pin)
Thiết lập mức 0v(low) hoặc 5v(high) cho các chân của PIC
VD : output_low(pin_D0) ;

2) Output_bit(pin,value)
Pin: tên chân của PIC
Value: giá trị 0 hay 1
VD: output_bit(pin_C0,1);

3) Output_X(value)
X: tên các port trên chíp
Value: giá trị 1 byte
VD: output_B(255);

4) Input_X( )
X: tên các port trên chip
Hàm này trả giá trị 8 bit là giá trị hiện hữu của port đó
VD: n = input_A( );

5) Set_tris_X(value)
X: tên chân (A – G)
Value: là giá trị 8 bít điều khiển vào ra cho các chân của chip
1: nhập dữ liệu 0: xuất dữ liệu
VD: set_tris_B(0); // tất cả các chân của portb là ngõ ra

V) HÀM SỬ DỤNG TRONG CÁC TIMER:

1)TIMER0:

- SETUP_TIMER_0(mode);

Mode: là một trong 2 constant (nếu dùng 2 thì chèn dấu " | " ở giữa) được định nghĩa trong file <16F877A.h>

+RTCC_INTERNAL: chọn xung dao động nội
+RTCC_EXT_H_TO_L: chọn kiểu tác động là cạch xuống của xung
+RTCC_EXT_L_TO_H: chọn kiểu tác động là cạch lên của xung
+RTCC_DIV_2 : Sử dụng bộ chia tần với tỉ lệ 1:2
+RTCC_DIV_4 : Sử dụng bộ chia tần với tỉ lệ 1:4
+RTCC_DIV_8 : Sử dụng bộ chia tần với tỉ lệ 1:8
+RTCC_DIV_16 : Sử dụng bộ chia tần với tỉ lệ 1:16
+RTCC_DIV_32 : Sử dụng bộ chia tần với tỉ lệ 1:32
+RTCC_DIV_64 : Sử dụng bộ chia tần với tỉ lệ 1:64
+RTCC_DIV_128 : Sử dụng bộ chia tần với tỉ lệ 1:128
+RTCC_DIV_256 : Sử dụng bộ chia tần với tỉ lệ 1:256

- SETUP_COUNTER_0(rtcc_state , ps_state)
Rtcc_state:
+RTCC_INTERNAL: chọn xung dao động nội
+RTCC_EXT_H_TO_L: chọn kiểu tác động là cạch xuống của xung
+RTCC_EXT_L_TO_H: chọn kiểu tác động là cạch lên của xung

Ps_state:
+RTCC_DIV_2 : Sử dụng bộ chia tần với tỉ lệ 1:2
+RTCC_DIV_4 : Sử dụng bộ chia tần với tỉ lệ 1:4
+RTCC_DIV_8 : Sử dụng bộ chia tần với tỉ lệ 1:8
+RTCC_DIV_16 : Sử dụng bộ chia tần với tỉ lệ 1:16
+RTCC_DIV_32 : Sử dụng bộ chia tần với tỉ lệ 1:32
+RTCC_DIV_64 : Sử dụng bộ chia tần với tỉ lệ 1:64
+RTCC_DIV_128 : Sử dụng bộ chia tần với tỉ lệ 1:128
+RTCC_DIV_256 : Sử dụng bộ chia tần với tỉ lệ 1:256

- SET_TIMER0(value) : xác định giá trị 8 bit ban đầu của Timer0(value=TMR0)
- GET_TIMER0( ) : trả lại giá trị 8 bit cho Timer0

2)TIMER1:

- SETUP_TIMER_1(mode);
Mode: có thể kết hợp với nhau bằng đấu " | "

+T1_DISABLED : tắt hoạt động của Timer1
+T1_INTERNAL : sử dụng giao động nội
+T1_EXTERNAL : chọn xung clock trên chân RC0
+T1_EXTERNAL_SYNC : chọn xung lock ngoài đồng bộ
+T1_DIV_BY_1 : Sử dụng bộ chia tần với tỉ lệ 1:1
+T1_DIV_BY_2 : Sử dụng bộ chia tần với tỉ lệ 1:2

+T1_DIV_BY_4 : Sử dụng bộ chia tần với tỉ lệ 1:4
+T1_DIV_BY_8 : Sử dụng bộ chia tần với tỉ lệ 1:8

- SET_TIMER0(value) : xác định giá trị 8 bit ban đầu của Timer1(value=TMR1)
- GET_TIMER0( ) : trả lại giá trị 8 bit cho Timer1

3) TIMER2:

- SETUP_TIMER_1(mode , period , postcale);

Mode: có thể kết hợp với nhau bằng đấu " | "
+T2_DISABLED : tắt hoạt động của Timer2
+T2_DIV_BY_1 : Sử dụng bộ chia tần với tỉ lệ 1:1
+T2_DIV_BY_4 : Sử dụng bộ chia tần với tỉ lệ 1:4
+T2_DIV_BY_16 : Sử dụng bộ chia tần với tỉ lệ 1:16

Period : số nguyên từ 0→255 xác định giá trị xung reset
Postcale : xác định số reset trước khi ngắt

- SET_TIMER2(value) : xác định giá trị 8 bit ban đầu của Timer2(value=TMR2)
- GET_TIMER2( ) : trả lại giá trị 8 bit cho Timer2

VI) CÁC HÀM PHỤC VỤ NGẮT

- ENABLE_INTERRUPTS(level)

Level:
+GLOBAL : cho phép ngắt toàn cục
+INT_TIMER0 : ngắt do tràn Timer0
+INT_TIMER1 : ngắt do tràn Timer1
+INT_TIMER2 : ngắt do tràn Timer2
+INT_RB : có thay đổi 1 trong các chân RB4 → RB7
+INT_EXT : ngắt ngoài trên chân RB0

Chú ý : sau khi khai báo trên thì để vào chương trình ngắt cần khai báo
#INT_.....
VD:

```
#INT_TIMER1
Void ngắt_Timer1( )
   {
      //chương trình ngắt viết ở đây
    }
```

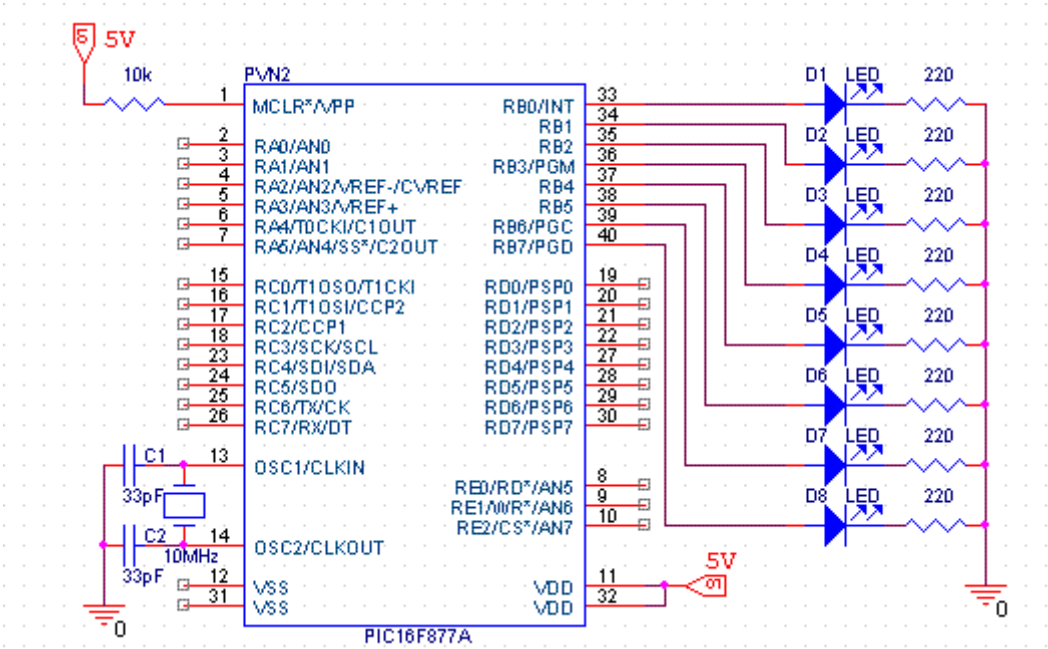
VII) CÁC HÀM ĐIỀU CHẾ ĐỘ RỘNG XUNG

- SETUP_CCPx(mode):
  Dùng trước hết để thiết lập chế độ hoạt động hay vô hiệu hoá tính năng CCP
  X: tên chân CCP trên chip (với PIC 16F877A đó là các chân RC1-CCP2 ; RC2-CCP1)
  Mode: CCP_PWM (bật chế độ PWM)

- SET_CCPx_DUTY(value)
  X: tên chân CCP trên chip
  Value: giá trị 8 hay 16 bit
  Nó ghi 10 bit giá trị vào thanh ghi CCPx , nếu value chỉ có 8 bit thì nó sẽ dịch thêm 2 bit nữa để đủ 10 bit nạp vào CCPx
  Tuỳ độ phân giả mà giá trị của value không phảI lúc nào cũng đạt tới giá trị 1023

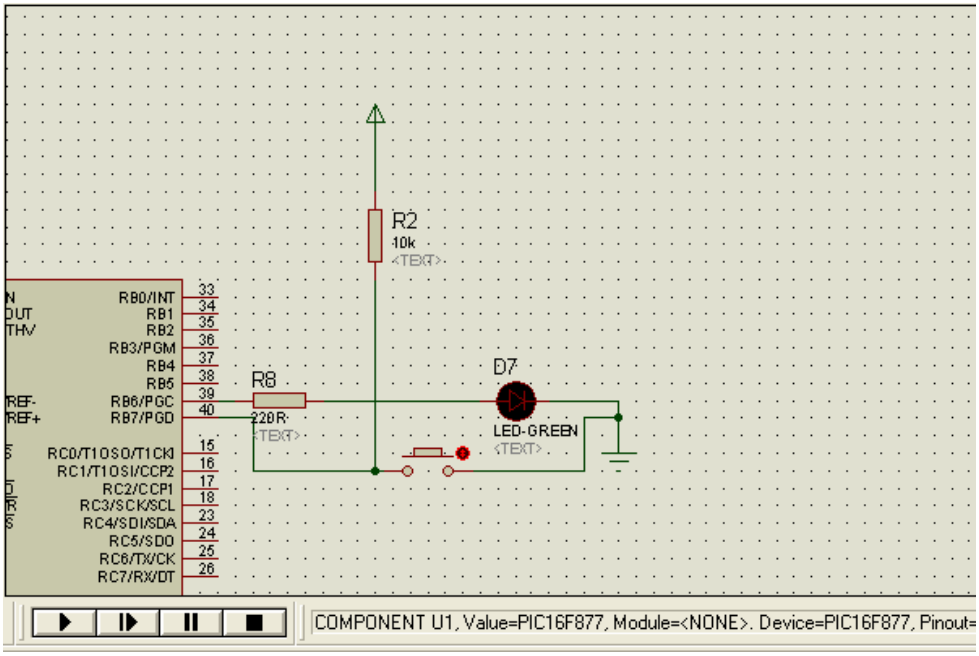
CHƯƠNG6
CÁC VÍ DỤ CƠ BẢN DÙNG PIC 16F877A

1) XUẤT 0-1 RA CÁC CHÂN:




```
#INCLUDE<16F877A.H>  //KHAI BAO DEVICE CHO PIC
#FUSES XT,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOLVP
    /*
      XT: SU DUNG THACH ANH TAN SO THAP (=4MHZ)
      NOWDT:KHONG SU DUNG CHE DO WATCH DOG TIMER
      PUT(POWER UP TIMER): CHON CHE DO LAM VIEC CHO PIC KHI DIEN AP DA ON DINH
      NOPROTECT : KHONG CHO PHEP DOC LAI MA CHUONG TRINH TRONG CHIP
      NOBROWNOUT : KHONG RESET LAI PIC KHI BI SUT AP
      NOLVP(LOW VOLTAGE PROGRAMMING) : ??
    */
#USE DELAY(CLOCK=4000000) //GIA TRI OSC MA BAN DUNG
#USE FAST_IO(B)

VOID MAIN()
{
  SET_TRIS_B(0);          //THIET LAP CAC CHAN CUA PORTA LA CAC CHAN XUAT DU LIEU
  WHILE(TRUE)  //TAO VONG LAP VO TAN
    {
      OUTPUT_HIGH(PIN_B0);  //XUAT GIA TRI 1 RA CHAN RB0
      DELAY_MS(500);       //TAO TRE 0,5S
      OUTPUT_LOW(PIN_B0);   //XUAT GIA TRI 0 RA CHAN RB0
      DELAY_MS(500);
      OUTPUT_B(255);       //DUA GIA TRI 1 RA CAC CHAN CUA PORTB
      DELAY_MS(500);
      OUTPUT_B(0);         //DUA GIA TRI 0 RA CAC CHAN CUA PORTB
      DELAY_MS(500);
    }
}
```

2)GIAO TIẾP VỚI PHÍM BẤM




```
#INCLUDE<16F877A.H>
#FUSES XT,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#USE FAST_IO(B)
#BYTE PORTB=0x06
#BIT BUTTON=PORTB.7
#BIT LED=PORTB.6

VOID MAIN()
{
   SET_TRIS_B(0b10111111); //LAP CHE CHO PIN_B6 LA XUAT  CON CAC CHAN KHAC LA NHAP
   WHILE(TRUE)
    {
      IF(BUTTON==0)    //KIEM TRA NUT BAM
        {
         LED=1;           //BAT DEN
         DELAY_MS(1000);
         LED=0;           //TAT DEN
        }
    }


}
```

3)DELAY DÙNG VÒNG LẶP

```
    #INCLUDE<16F877A.H>
#FUSES XT,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#BYTE INTCON=0X0B
#BYTE PIR1=0X0C
#BYTE T1CON=0X10
#BIT TMR1ON=T1CON.0 //KHOI DONG TIMER1
#BIT TMR1IF=PIR1.0  //CO TRAN TIMER1
#BIT INTF=INTCON.1  //CO NGAT


VOID DELAY(INT8 TIMER)
  {
   WHILE(TIMER--)
     {
      TMR1ON=1;
       WHILE(TMR1IF==0);
        TMR1ON=0;
        TMR1IF=0;
     }
  }




VOID MAIN()
   {
   WHILE(TRUE)
     {

       OUTPUT_HIGH(PIN_C4);
       DELAY(100);
       OUTPUT_LOW(PIN_C4);
       DELAY(100);
     }
   }


//TUONG TU VOI CAC TIMER0 VA TIMER2 CAC BAN CO THE XEM DATASHEET DE BIET THEM VE DIA
CHI
// CAC THANH GHI CHUC NANG CUA CHUNG
```

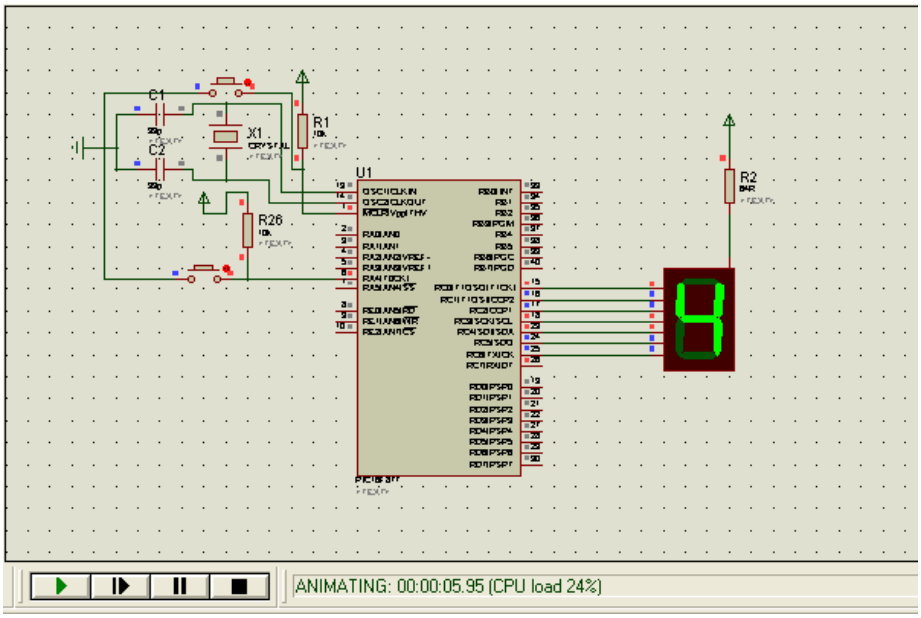
4)DELAY DÙNG TIMER KHÔNG DÙNG NGẮT

```
#INCLUDE<16F877A.H>
#FUSES XT,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#BYTE INTCON=0X0B
#BYTE PIR1=0X0C
#BYTE T1CON=0X10
#BYTE T2CON=0X12
#BIT TMR1ON=T1CON.0 //KHOI DONG TIMER1
#BIT TMR1IF=PIR1.0  //CO TRAN TIMER1
#BIT TMR0IF=INTCON.2 //CO TRAN TIMER0
#BIT TMR2ON=T2CON.2  //KHOI DONG TIMER2
#BIT TMR2IF=PIR1.1   //CO TRAN TIMER2


VOID MAIN()
{
 SET_TRIS_E(0);
 SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_2); //SU DUNG TAN SO DAO DONG NOI VA BO CHIA
TAN 1:2
 SET_TIMER0(0);                    //DAT GIA TRI THANH GHI DINH THOI TMR0=0
 SETUP_TIMER_1(T1_INTERNAL|T1_DIV_BY_4);  //SU DUNG TAN SO DAO DONG NOI VA BO CHIA TAN
1:4
 SET_TIMER1(0);                    //DAT GIA TRI THANH GHI DINH THOI TMR1=0
 SETUP_TIMER_2(T2_DIV_BY_16,255,1);      //SU DUNG TAN SO DAO DONG NOI VA BO CHIA TAN 1:16
 SET_TIMER2(0);                    //DAT GIA TRI THANH GHI DINH THOI TMR2=0
 TMR0IF=TMR1IF=TMR2IF=0;               //XOA CAC CO TRAN
 WHILE(TRUE)
   {
     IF(TMR0IF==1)                 //KIEM TRA CO TRAN TIMER0
       {
        OUTPUT_HIGH(PIN_E0);
        DELAY_MS(50);
        OUTPUT_LOW(PIN_E0);
        TMR0IF=0;
       }
        ELSE IF(TMR1IF==1)           //KIEM TRA CO TRAN TIMER1
          {
           OUTPUT_HIGH(PIN_E1);
           DELAY_MS(50);
           OUTPUT_LOW(PIN_E1);
           TMR1IF=0;
          }
           ELSE IF(TMR2IF==1)        //KIEM TRA CO TRAN TIMER2
             {
              OUTPUT_HIGH(PIN_E2);
              DELAY_MS(50);
              OUTPUT_LOW(PIN_E2);
              TMR2IF=0;
             }
   }


}
```

5)DELAY DÙNG TIMER CÓ DÙNG NGẮT

```
 #INCLUDE<16F877A.H>
#FUSES XT,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#BYTE PORTB=0X06
#BYTE PORTC=0X07
#BYTE PORTD=0X08

#INT_TIMER0             //CHUONG TRINH NGAT TIMER0
VOID NGAT_TIMER0()
  {
  PORTB=0B10101010;
  DELAY_MS(200);
  PORTB=0B01010101;
  }


#INT_TIMER1             //CHUONG TRINH NGAT TIMER1
VOID NGAT_TIMER1()
   {
    PORTC=0B11110000;
    DELAY_MS(200);
    PORTC=0B00001111;
   }

#INT_TIMER2             //CHUONG TRINH NGAT TIMER2
VOID NGAT_TIMER2()
   {
    PORTD=0B11001100;
    DELAY_MS(200);
    PORTD=0B00001111;
   }



VOID MAIN()
   {
    SET_TRIS_B(0);
    SET_TRIS_C(0);
    SET_TRIS_D(0);
    ENABLE_INTERRUPTS(GLOBAL);     //CHO PHEP TAT CA CAC NGAT HOAT DONG
    ENABLE_INTERRUPTS(INT_TIMER0); //CHO PHET NGAT BANG TIMER0
    ENABLE_INTERRUPTS(INT_TIMER1); //CHO PHET NGAT BANG TIMER1
    ENABLE_INTERRUPTS(INT_TIMER2); //CHO PHET NGAT BANG TIMER2
    SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_1);
    SET_TIMER0(0);
    SETUP_TIMER_1(T1_INTERNAL|T1_DIV_BY_4);
    SET_TIMER1(0);
    SETUP_TIMER_2(T2_DIV_BY_16,255,1);
    SET_TIMER2(0);
    WHILE(TRUE);    //KHONG LAM GI CA CHO NGAT
   }
```

6) DÙNG CHẾ ĐỘ COUNTER

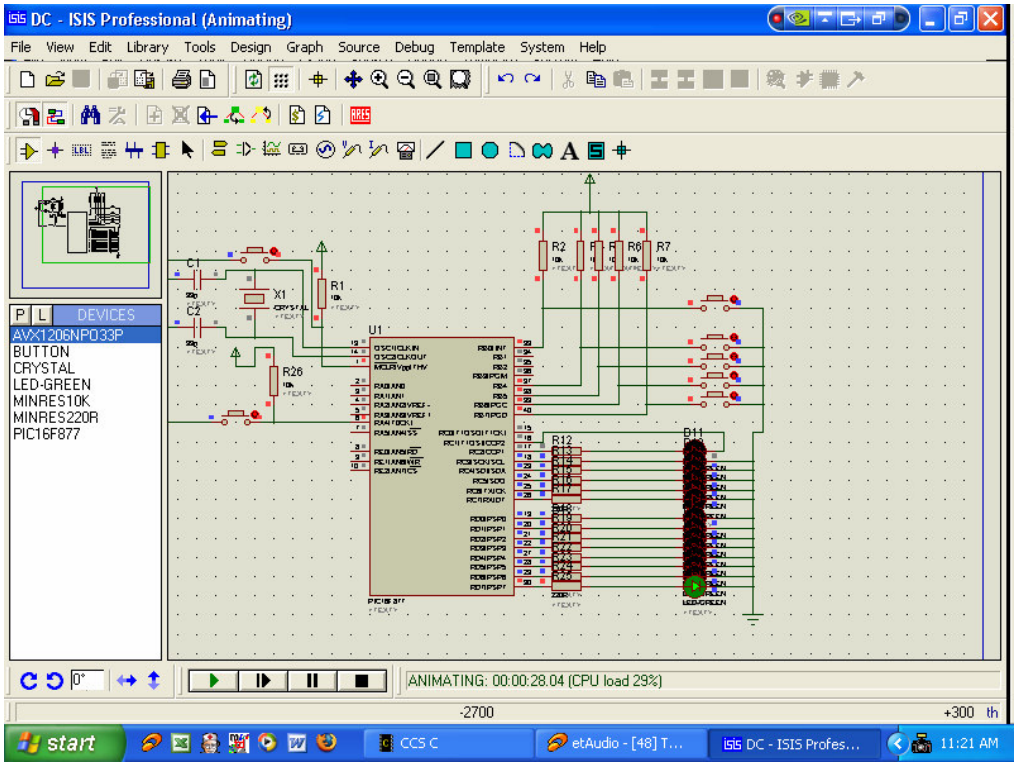



```
 #INCLUDE<16F877A.H>
#FUSES XT,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#USE FAST_IO(C)
#BIT TMR0IF=0X0B.2
#BYTE PORTC=0X07
CONST CHAR FONT[]={0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X98};
INT8 I;


VOID MAIN()
{
  SET_TRIS_C(0);
  PORTC=255;
  SETUP_COUNTERS(RTCC_EXT_H_TO_L,RTCC_DIV_1);
  SET_TIMER0(255);
  WHILE(TRUE)
    {
   IF(TMR0IF==1)
     {
      IF(I<=9)
        {
         PORTC=FONT[I++];
        }
          ELSE
            {
             I=0;
             PORTC=FONT[I];
            }

       TMR0IF=0;
       SET_TIMER0(255);
      }

    }
}
```

7) DÙNG CHẾ ĐỘ PWM

```
#INCLUDE<16F877A.H>
#FUSES NOWDT,PUT,XT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#USE FAST_IO(B)
#USE FAST_IO(C)
#BYTE PORTB=0X06
#BIT RB1=PORTB.1
#BIT RB2=PORTB.2
 INT16 U;

VOID MAIN()
 {
  SET_TRIS_B(255);
  SET_TRIS_C(0);
  U=200;
  SETUP_TIMER_2(T2_DIV_BY_16,255,1);  //THIET LAP CAU HINH CHO TIMER0
  SET_TIMER2(0);
  SETUP_CCP1(CCP_PWM);              //CHON CHE DO PWM
  SET_PWM1_DUTY(U);                 //XAC DINH SO XUNG DE CHAN PWM1 O MUC 1

  WHILE(TRUE)
   {
    IF(RB1==0) //TANG DO SANG LED
      {
       U=U+10;
        IF(U>255)
           {
              U=255;
              SET_PWM1_DUTY(U);//TANG TOC PHAI
           }
             ELSE
                 {

                   SET_PWM1_DUTY(U);
                 }
      }
     ELSE IF(RB2==0)
       {
          U=U-10;
           IF(U>145) SET_PWM1_DUTY(U);// GIAM TOC PHAI
                ELSE
                   {
                     U=145;
                     SET_PWM1_DUTY(U);
                   }

       }


   }

}
```

8) DÙNG NGẮT NGOÀI

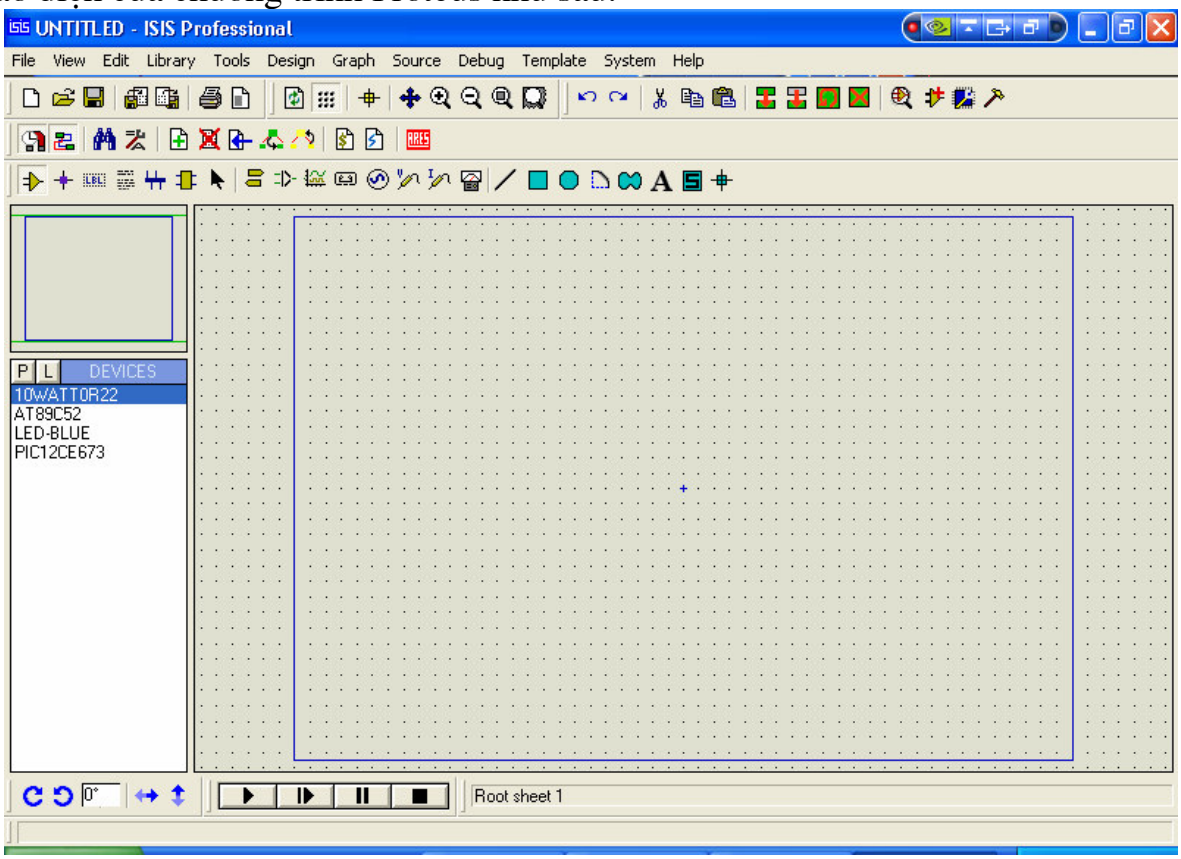



```
#INCLUDE<16F877A.H>
#FUSES NOWDT,PUT,XT,NOPROTECT,NOBROWNOUT,NOLVP
#USE DELAY(CLOCK=4000000)
#USE FAST_IO(B)
#USE FAST_IO(D)
#BYTE PORTB=0X06
#BYTE PORTD=0X08
#BIT RB0=PORTB.0
#BIT RB4=PORTB.4
#BIT RB5=PORTB.5
#BIT RB6=PORTB.6
#BIT RB7=PORTB.7



#INT_EXT                  //CHUONG TRINH NGAT TREN CHAN RB0
VOID NGATNGOAI_RB0()
 {
  OUTPUT_HIGH(PIN_D0);
  DELAY_MS(1000);
  OUTPUT_LOW(PIN_D0);
 }



#INT_RB
VOID NGATNGOAI_RB()        //CHUONG TRINH NGAT NGOAI TREN CHAN RB4-->RB7
  {
   IF(RB4==0)                      //KIEM TRA CHAN RB4
     {
      OUTPUT_HIGH(PIN_D0);
```

```
      DELAY_MS(1000);
      OUTPUT_LOW(PIN_D0);
     }
     ELSE IF(RB5==0)                //KIEM TRA CHAN RB5
       {
         OUTPUT_HIGH(PIN_D1);
         DELAY_MS(1000);
         OUTPUT_LOW(PIN_D1);

       }

         ELSE IF(RB6==0)            //KIEM TRA CHAN RB6
          {
            OUTPUT_HIGH(PIN_D2);
            DELAY_MS(1000);
            OUTPUT_LOW(PIN_D2);
          }
           ELSE IF(RB7==0)        //KIEM TRA CHAN RB7
             {
               OUTPUT_HIGH(PIN_D3);
               DELAY_MS(1000);
               OUTPUT_LOW(PIN_D3);
             }
 }

VOID MAIN()
 {
   SET_TRIS_B(255);
   SET_TRIS_D(0);
   PORTD=0;
   ENABLE_INTERRUPTS(GLOBAL);     //CHO PHEP TAT CA CAC NGAT HOAT DONG
   ENABLE_INTERRUPTS(INT_RB);     //CHO PHEP NGAT NGOAI TU CAC CHAN RB4-->RB7
   ENABLE_INTERRUPTS(INT_EXT);    //CHO PHEP NGAT NGOAI TREN CHAN RB0
   EXT_INT_EDGE(H_TO_L);          //KIEU TAC DONG LA KIEU SUONG AM
   WHILE(TRUE)
    {

     OUTPUT_HIGH(PIN_D7);
     DELAY_MS(1000);
     OUTPUT_LOW(PIN_D7);
     DELAY_MS(1000);
    }
 }
```

CHƯƠNG 7
CÁCH SỬ DỤNG CHƯƠNG TRÌNH MÔ PHỎNG PROTEUS

Giao diện của chương trình Proteus như sau:

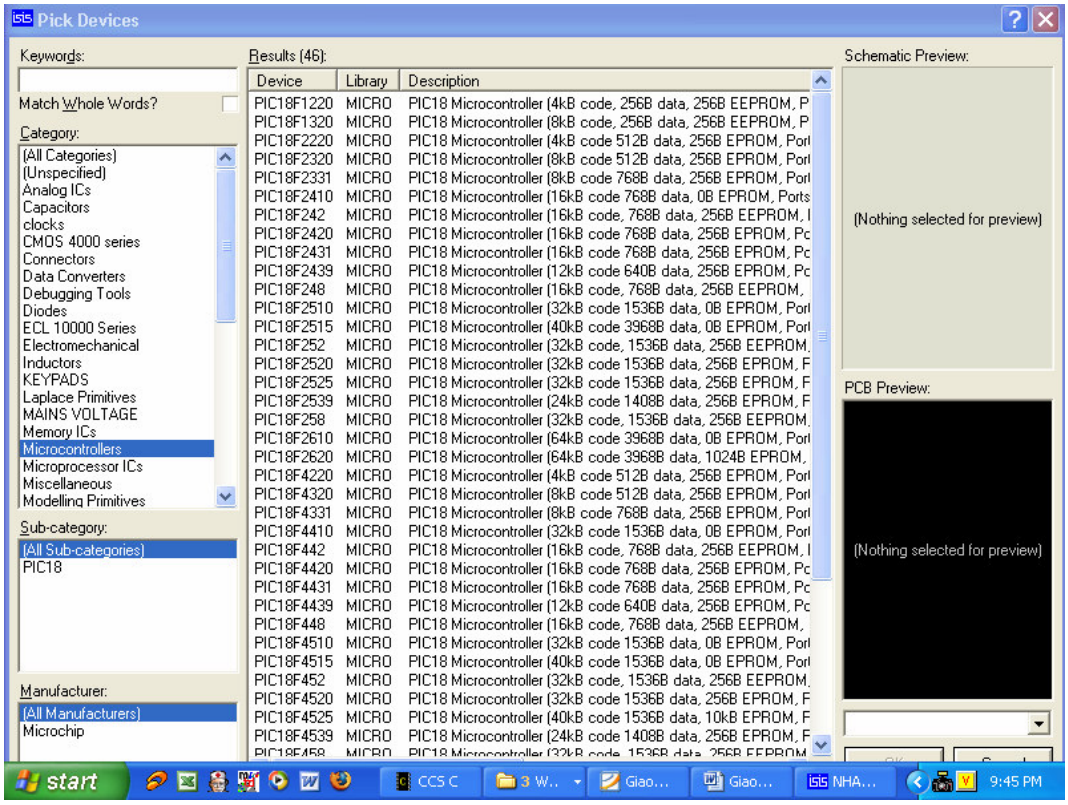


Để tìm các linh kiện cần dùng bạn ấn phím P hay click chuột vào nút P ở bên trái màn hình

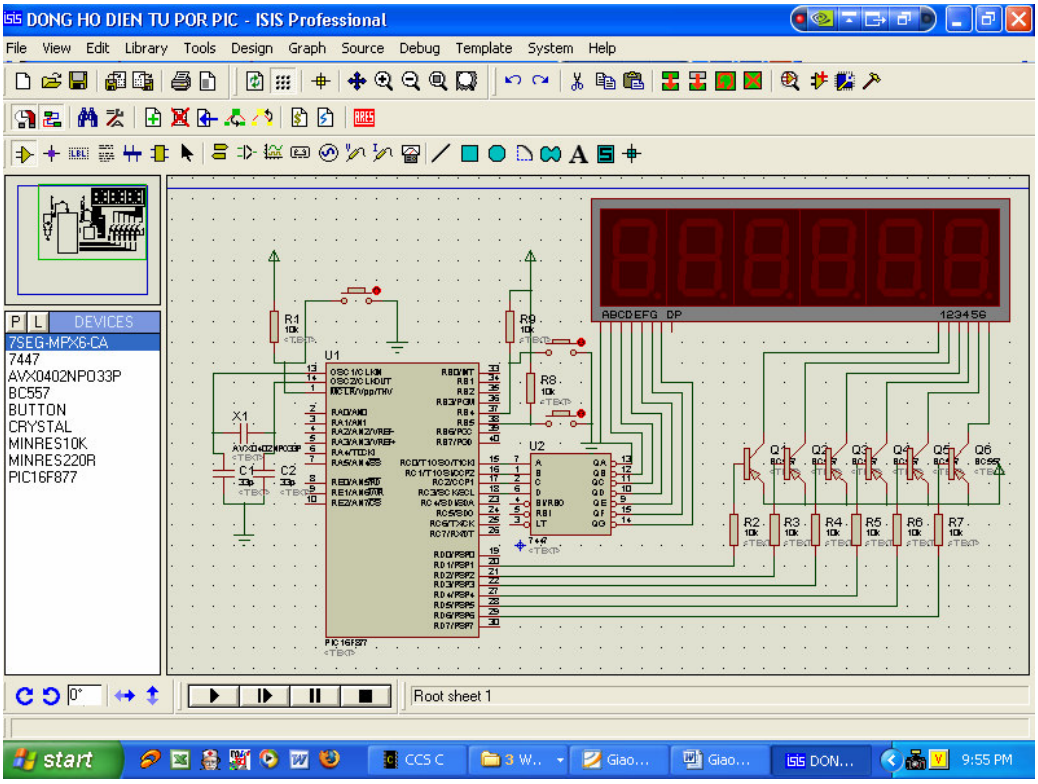
Một cửa sổ sẽ hiện ra:

-Trong đó có các thư mục giúp bạn dễ dàng trong việc tìm các linh kiện cần dùng

-Vi dụ như diodes (các loại diot),microcontrollers(các loại vi điều khiển)

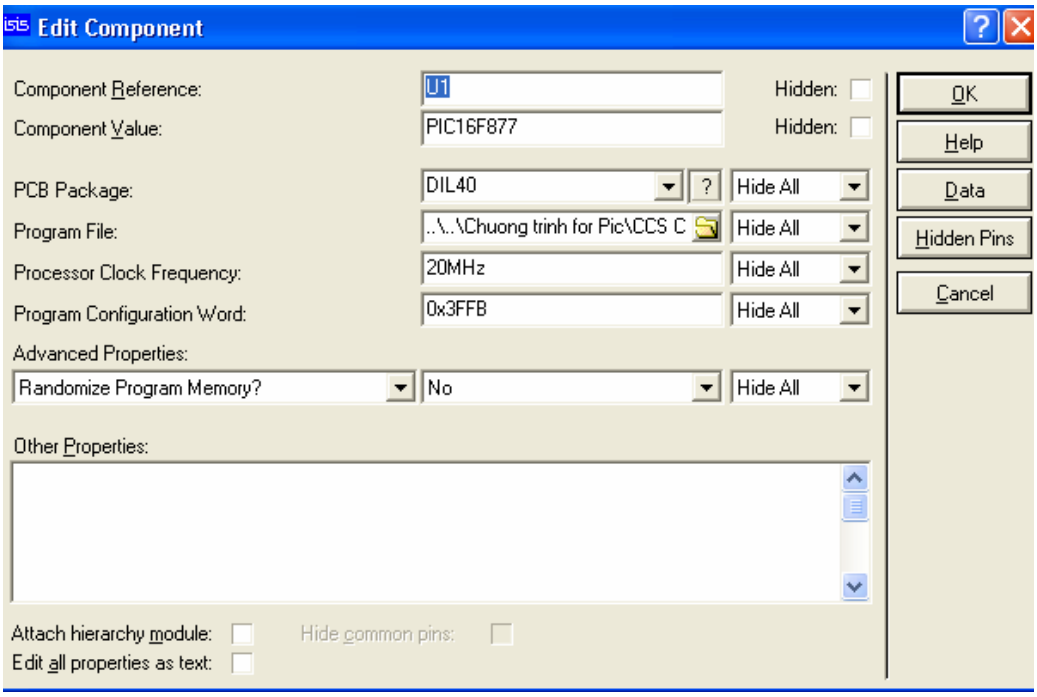


Sau khi sắp xếp các linh kiện ta có sơ đồ hoàn chỉnh như sau:



Để chạy mô phỏng mạch điện ta làm như sau:

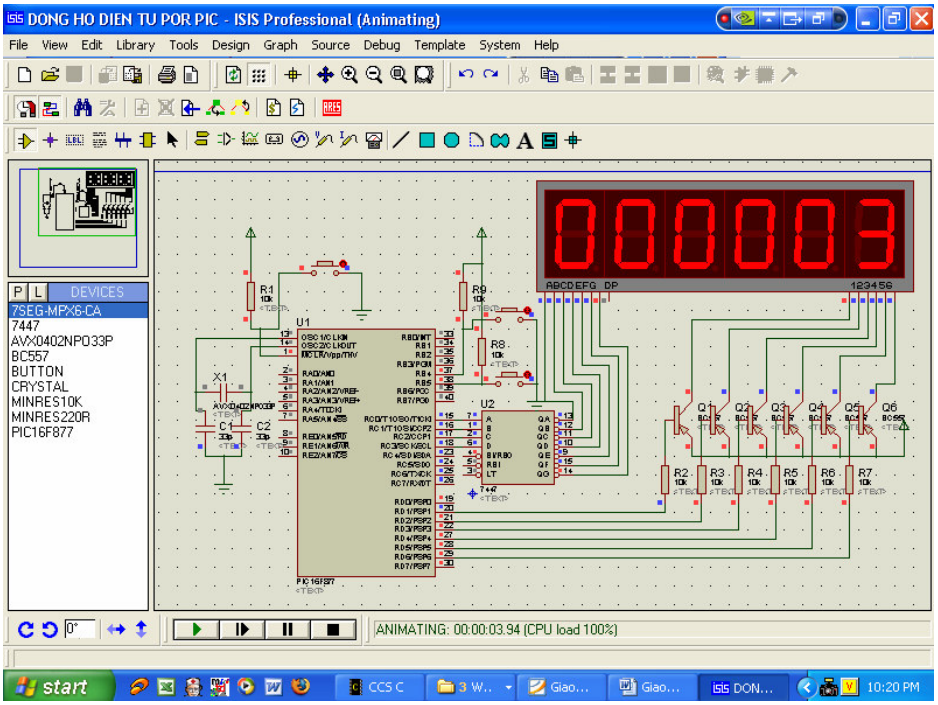
- Nháy chuột phải vào con vi điều khiển ta cần khi thấy nó đổi mầu đỏ thì ta nháy chuột trái và một cửa sổ sẽ hiện ra như sau:



Tại ô program file chúng ta chỉ đường dẫn đến file HEX cần phải nạp cho chip
Tại ô processor clock frequency chúng ta chọn tần số giao động của thạch anh
Sau đó ấn ok để hoàn tất

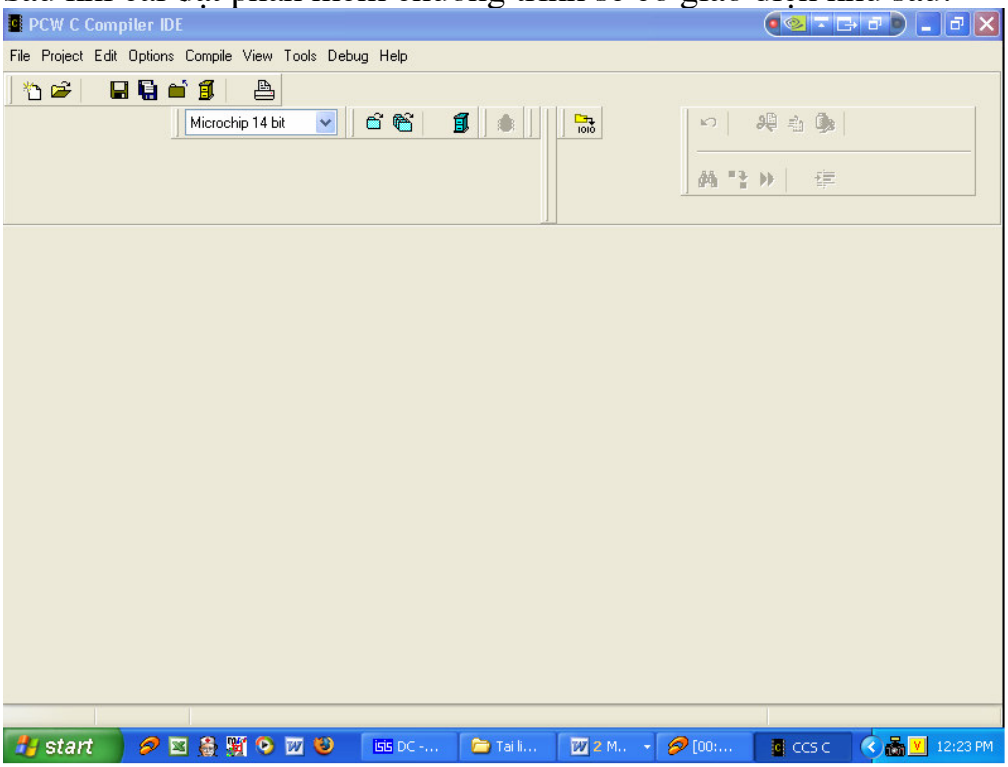
- để chạy mô phỏng chúng ta sử dụng các nút: 

Quá trình chạy mô phỏng được minh hoạ ở hình dưới:

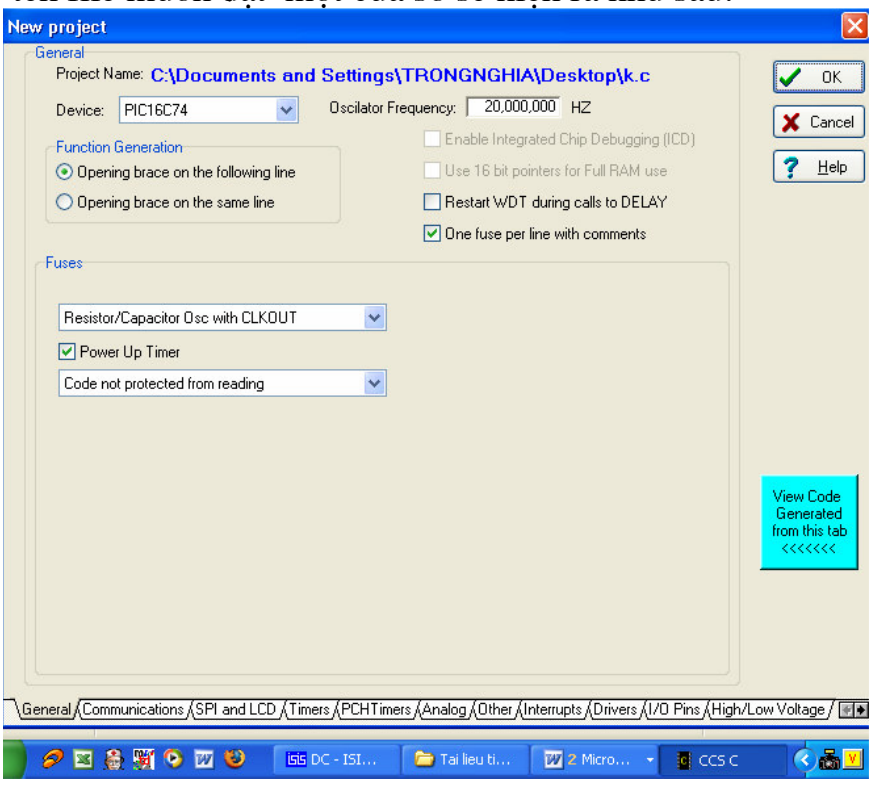


CHƯƠNG 8
HƯỚNG DẪN SỬ DỤNG TRÌNH BIÊN DỊCH CCS C

Sau khi cài đặt phần mềm chương trình sẽ có giao diện như sau:



Để tạo một project mới bạn hãy chọn Project/ New /Pic Winzard . Sau khi đánh tên file muốn đặt một cửa sổ sẽ hiện ra như sau:



Tại mục Device bạn chọn tên Pic đang dùng
Tại mục oscilator Frequency bạn chọn tần số giao động của thạch anh
Sau đó ấn ok để hoàn tất
Sau khi hoàn tất việc lập trình bạn ấn F9 để chương trình dịch sang file HEX
Với file Hex được tạo ra bạn có thể nạp được cho vi điều khiển