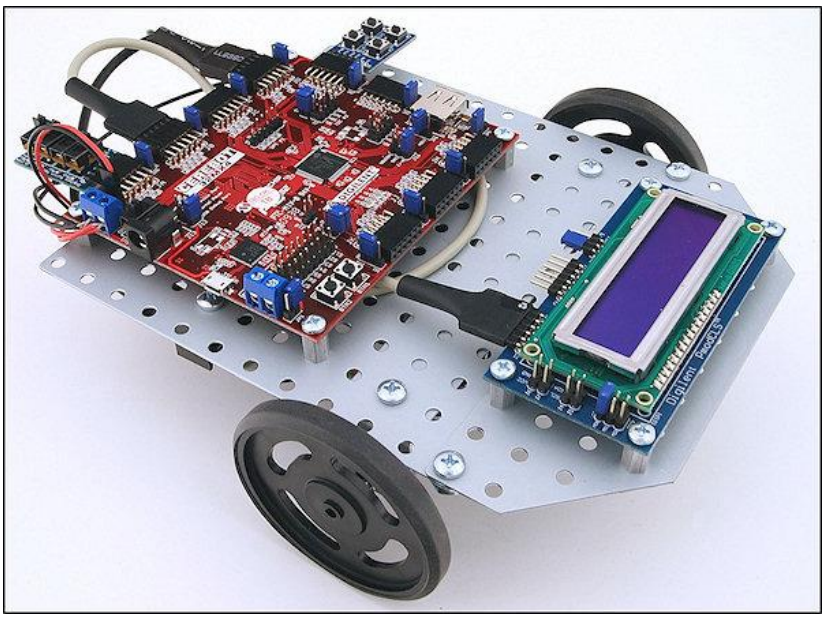
BÀI 5

TIMER/COUNTER CỦA PIC - LẬP TRÌNH C



- **GIỚI THIỆU**
- **KHẢO SÁT TIMER/COUNTER**
  - KHẢO SÁT TIMER/COUNTER T0
  - KHẢO SÁT TIMER/COUNTER T1
  - KHẢO SÁT TIMER T2
- **TẬP LỆNH C CHO CÁC TIMER**
  - Lệnh SETUP_TIMER_X()
  - Lệnh SET_TIMER_X()
  - Lệnh SETUP_COUNTERS()
  - Lệnh SETUP_WDT()
  - Lệnh RESTART_WDT()
  - Lệnh GET_TIMER_X()
- **CÁC CHƯƠNG TRÌNH ỨNG DỤNG TIMER /COUNTER**
  - ỨNG DỤNG ĐẾM XUNG NGOẠI HIỂN THỊ TRÊN LED 7 ĐOẠN TRỰC TIẾP
  - ỨNG DỤNG ĐẾM XUNG NGOẠI HIỂN THỊ TRÊN LED 7 ĐOẠN QUÉT
  - ỨNG DỤNG ĐẾM XUNG NỘI - ĐỊNH THỜI
- **CÁC TOÁN TỬ**
- **FILE DEVICE**

I. GIỚI THIỆU

Vi điều khiển PIC họ 16F877A có 3 timer T0, T1 và T2. T0 là timer/counter 8 bit, T1 là timer/counter 16 bit, cả 2 đều có bộ chia trước. T2 chỉ là timer 8 bit có bộ chia trước và chia sau phục vụ cho các ứng dụng đặc biệt.

Phần tiếp sẽ khảo sát chi tiết từng timer/counter.

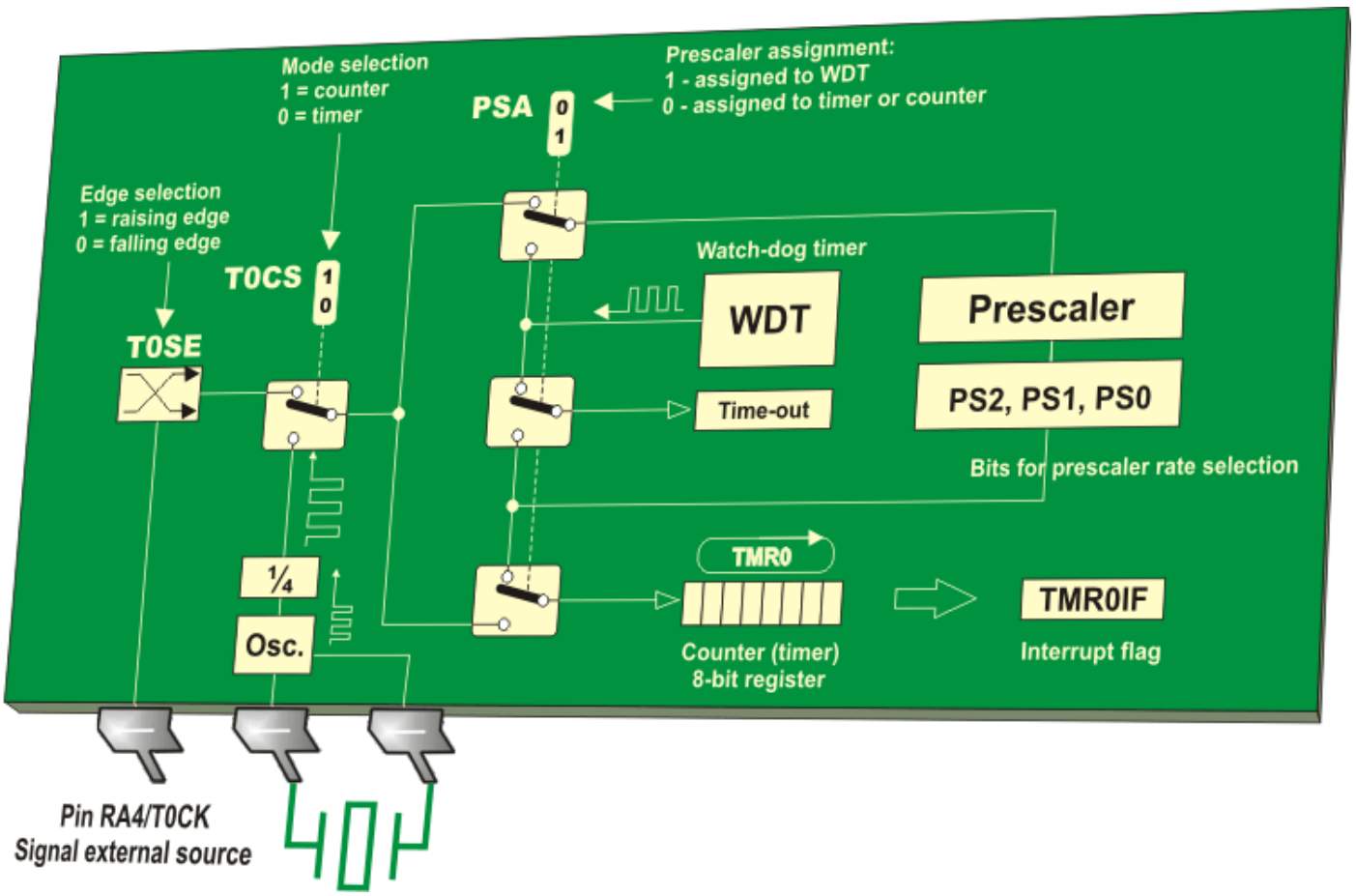
II. KHẢO SÁT CÁC TIMER/COUNTER

1. KHẢO SÁT TIMER0

Bộ timer0/counter0 có những đặc điểm sau:

- Là timer/counter 8 bit.
- Có thể đọc và ghi giá trị đếm của timer/counter.
- Có bộ chia trước 8 bit cho phép lập trình bằng phần mềm.
- Cho phép lựa chọn nguồn xung clock bên trong hoặc bên ngoài.
- Phát sinh ngắt khi bị tràn từ FFH đến 00H.
- Cho phép lựa chọn tác động cạnh lên hoặc cạnh xuống.

Sơ đồ khối của timer0 và bộ chia trước với WDT như hình 5-1:



Hình 5-1: Sơ đồ khối của timer0.

Để sử dụng timer0 thì phải khảo sát chức năng của thanh ghi điều khiển timer là OPTION_REG. Cấu hình thanh ghi và chức năng các bit như sau:

	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	Features
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

Legend

R/W	Readable/Writable bit
(1)	After reset, bit is set

Bit 7 $\overline{RBPU}$: bit điều khiển điện trở treo của portb.

Bit 6 **INTEDG**

Bit 5 **T0CS**: bit lựa chọn nguồn xung cho TMR0 - TMR0 Clock Source Select bit.

1= sẽ đếm xung ngoại đưa đến chân T0CKI.

0= sẽ đếm xung clock nội bên trong.

Bit 4 **T0SE**: bit lựa chọn cạnh tích cực T0SE - TMR0 Source Edge Select bit.

1= tích cực cạnh lên ở chân T0CKI.

0= tích cực cạnh xuống ở chân T0CKI.

Bit 3 **PSA**: bit gán bộ chia trước - prescaler assigment.

1= gán bộ chia cho WDT.

0= gán bộ chia Timer0.

Bit 2-0 **PS2:PS0**: các bit lựa chọn tỉ lệ bộ chia trước - prescaler rate select bits:

Bit lựa chọn	Tỉ lệ TMR0	Tỉ lệ WDT
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

Nếu bit T0CS (OPTION_REG<5>) bằng 1 thì chọn chế độ đếm xung ngoài Counter. Trong chế độ đếm xung ngoại thì bộ đếm sẽ tăng giá trị mỗi khi có xung tác động cạnh lên hoặc cạnh xuống ở chân RA4/T0CKI. Cạnh tác động của xung được chọn lựa bởi bit T0SE (OPTION_REG<4>): T0SE = 0 thì chọn cạnh lên, ngược lại thì chọn cạnh xuống.

Bộ chia trước không thể đọc/ghi và có mối quan hệ với Timer0 và Watchdog Timer.

a. Ngắt của Timer0:

Khi giá trị đếm trong thanh ghi TMR0 tràn từ FFh về 00h thì phát sinh ngắt, cờ báo ngắt TMR0IF (INTCON<2>) đổi trạng thái từ 0 lên 1. Ngắt có thể ngăn bằng cách xóa bit cho phép ngắt TMR0IE (INTCON<5>).

Trong chương trình con phục vụ ngắt Timer0 phải xóa cờ báo ngắt TMR0IF. Ngắt của TMR0 không thể kích CPU thoát khỏi chế độ ngủ vì bộ định thời sẽ ngừng khi CPU ở chế độ ngủ.

b. Timer0 với nguồn xung đếm từ bên ngoài:

Khi không sử dụng bộ chia trước thì ngõ vào xung clock bên ngoài giống như ngõ ra bộ chia trước. Việc đồng bộ hóa của T0CKI với các xung clock bên trong được thực hiện bằng cách lấy mẫu ngõ ra bộ chia

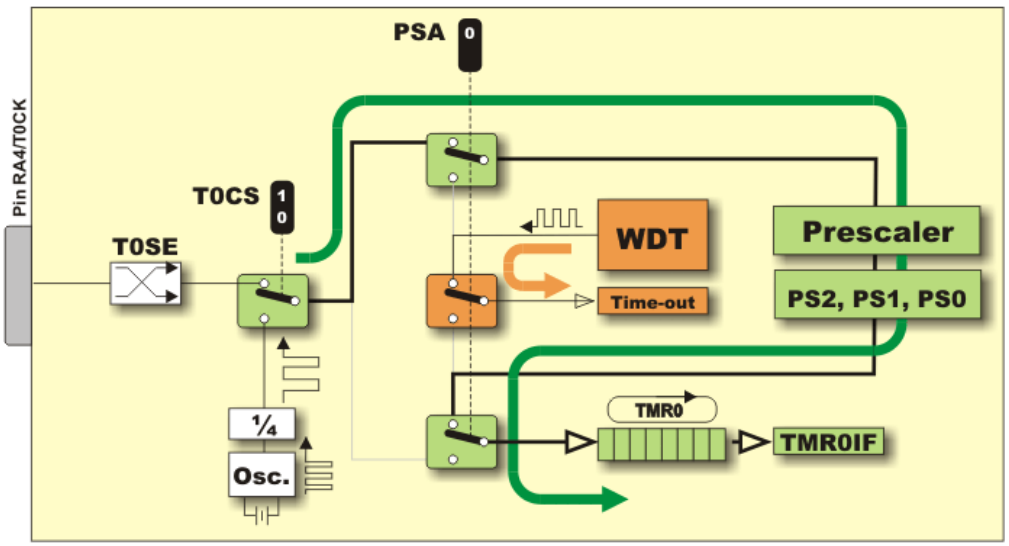
ở những chu kì Q2 và Q4 của xung clock bên trong. Do đó, nó rất cần thiết cho T0CKI ở trạng thái mức cao ít nhất 2 T_{OSC} và ở trạng thái mức thấp ít nhất 2 T_{OSC}.

c. Bộ chia trước:

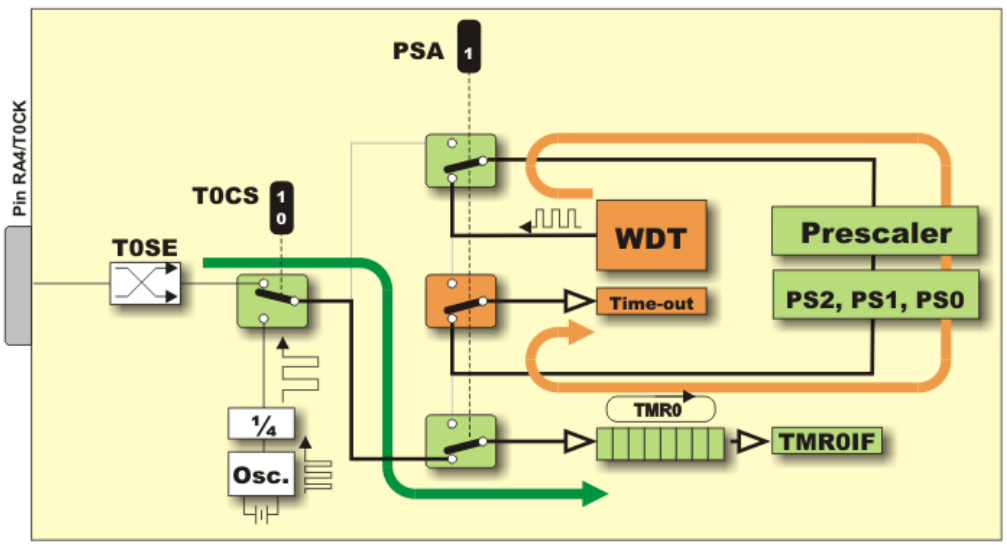
Bộ chia trước có thể gán cho Timer0 hoặc gán cho Watchdog Timer. Bộ chia trước thì không thể đọc hoặc ghi. Các bit PSA và PS2:PS0 (OPTION_REG<3:0>) quyết định đối tượng gán và tỉ lệ chia.

Khi được gán cho Timer0 thì tất cả các lệnh ghi cho thanh ghi TMR0 (ví dụ CLRF 1, MOVWF 1, BSF 1, …) sẽ xoá bộ chia trước.

Khi được gán cho WDT thì lệnh CLRWDT sẽ xoá bộ chia trước cùng với Watchdog Timer.



Hình 5-2: Bộ chia trước được gán cho timer.

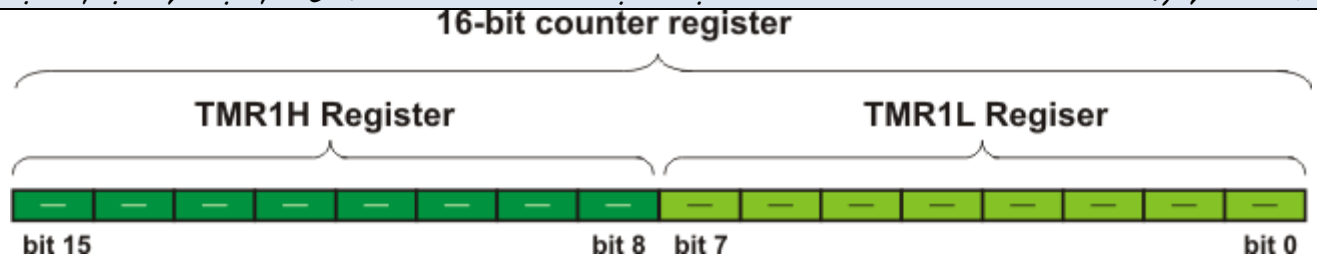


Hình 5-3: Bộ chia trước được gán cho WDT.

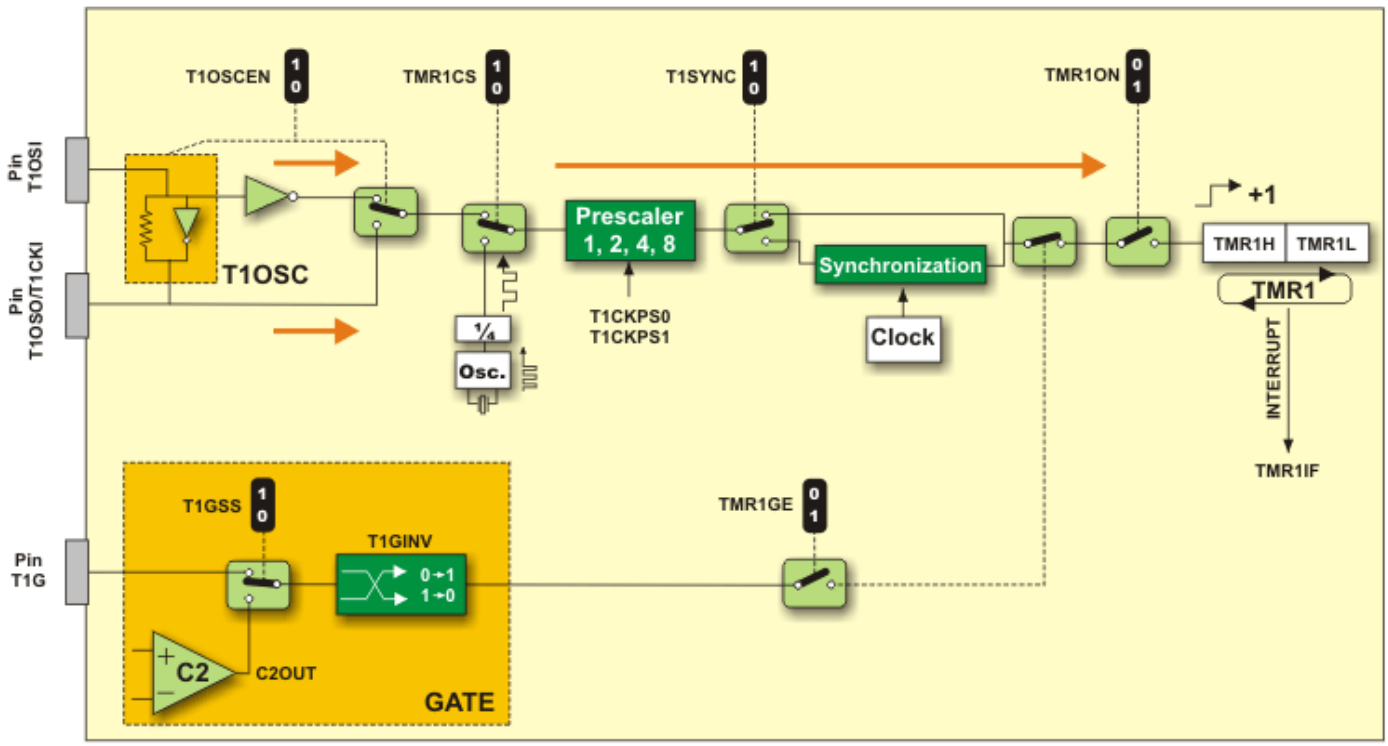
2. BỘ ĐỊNH THỜI TIMER1:

Là bộ định thời/đếm 16 bit gồm 2 thanh ghi 8 bit (TMR1H và TMR1L) – có thể đọc và ghi. Hai thanh ghi này tăng từ 0000h đến FFFFh và quay trở lại 0000h.

Khi bị tràn thì Timer1 sẽ phát sinh ngắt, cờ báo ngắt TMR1IF (PIR1<0>) lên mức 1. Timer1 có bit cho phép/cấm là TMRIE (PIE1<1>).



Cấu trúc của Timer1:



Hình 5-4: Cấu trúc timer T1.

Khảo sát thanh ghi điều khiển Timer1:

	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	Features
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

Legend

R/W	Readable/Writable bits
(0)	After reset, bit is cleared

Bit 7-6 Chưa sử dụng nếu đọc sẽ có giá trị ‘0’.

Bit 5-4 **T1CKPS1:T1CKPS0:** các bit lựa chọn bộ chia - Timer1 input Clock Prescale Select bits

11=1:8 giá trị chia.

10=1:4 giá trị chia.

01=1:2 giá trị chia.

00=1:1 giá trị chia.

Bit 3 **T1OSCEN:** bit ĐK cho phép bộ dao động Timer1 - Timer1 Oscillator Enable Control bit

1= bộ dao động được phép.

0= Tắt bộ dao động.

Bit 2 **T1SYNC**: bit ĐK đồng bộ ngõ vào xung clock bên ngoài của timer1

Khi **TMR1CS = 1**

1= không thể đồng bộ ngõ vào clock ở từ bên ngoài.

0= đồng bộ ngõ vào clock ở từ bên ngoài.

Khi **TMR1CS = 0**

Bit này bị bỏ qua. Timer1 dùng xung clock bên trong khi **TMR1CS = 0.**

Bit 1 **TMR1CS**: bit lựa chọn nguồn xung clock của timer1

1= Chọn nguồn xung clock từ bên ngoài ở chân RC0/T1OSO/T1CKI (cạnh lên).

0= Chọn xung nội bên trong ($F_{OSC}/4$).

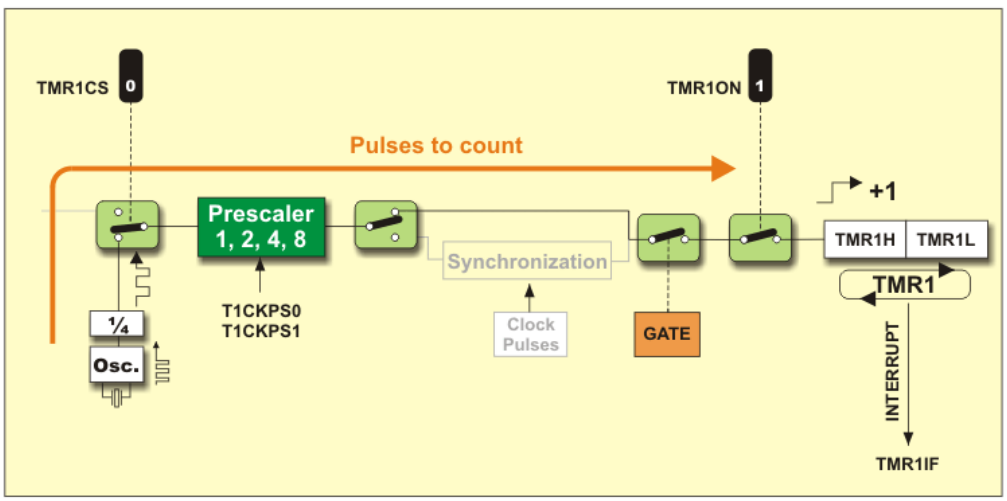
Bit 0 **TMR1ON**: bit điều khiển Timer14

1= Cho phép Timer1 đếm.

0= Timer1 ngừng đếm.

a. Timer1 ở chế độ định thời:

Nếu bit TMR1CS bằng 0 thì T1 hoạt động định thời đếm xung nội có tần số bằng $F_{OSC}/4$. Bit điều khiển đồng bộ $\overline{T1SYNC}$ không bị ảnh hưởng do xung clock bên trong luôn đồng bộ.



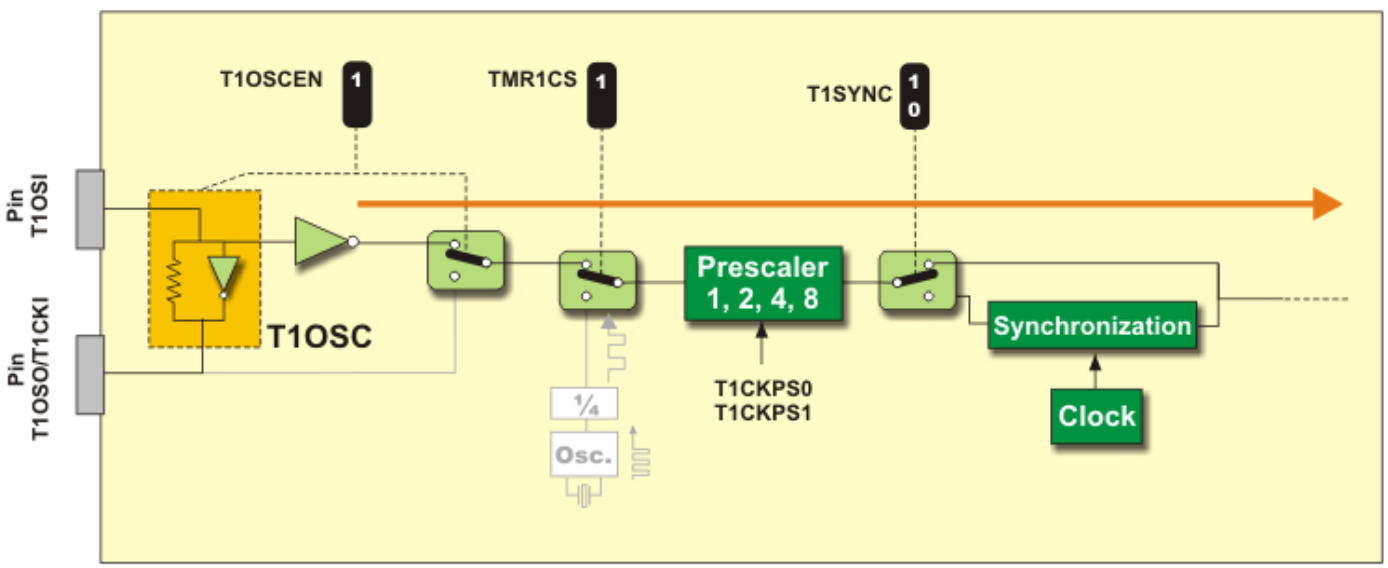
Hình 5-5. T1 hoạt động định thời.

b. Timer1 ở chế độ đếm xung ngoại:

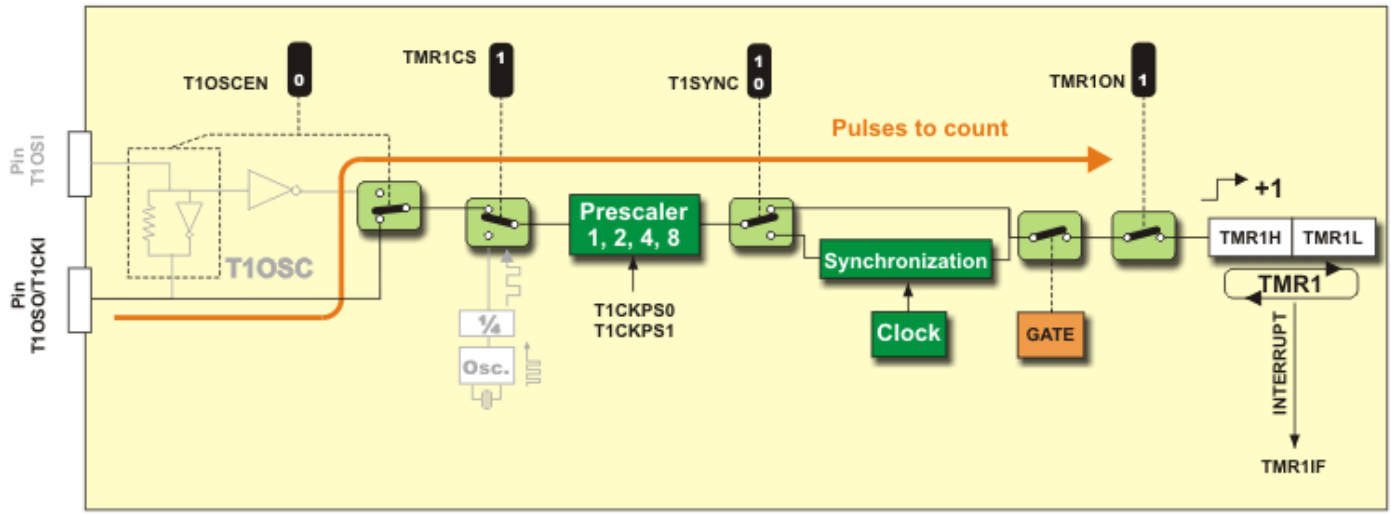
Nếu bit TMR1CS bằng 1 thì T1 hoạt động đếm xung ngoại. Xung ngoại có 2 nguồn xung phụ thuộc vào bit T1OSCEN:

Nếu bit T1OSCEN bằng 1 thì T1 đếm xung ngoại từ mạch dao động của T1 - xem hình 5-6.

Nếu bit T1OSCEN bằng 0 thì T1 đếm xung ngoại đưa đến ngõ vào T1CKI - xem hình 5-7.

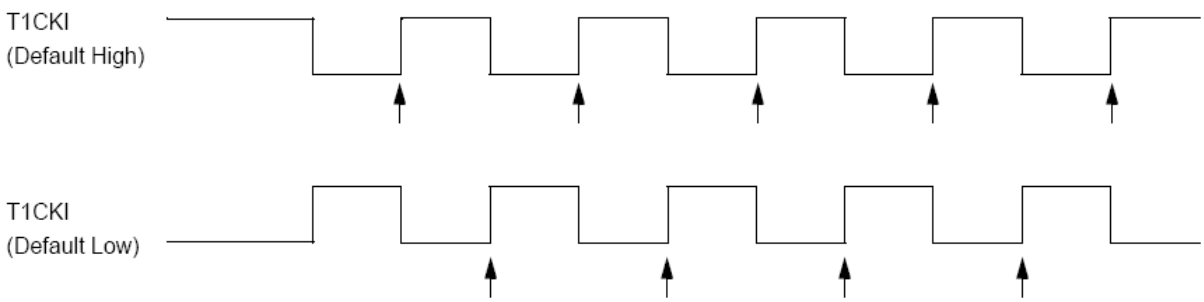


Hình 5-6. T1 hoạt động đếm xung ngoại từ mạch dao động T1.



Hình 5-7. T1 hoạt động đếm xung ngoại đưa đến ngõ vào T1CKI.

Timer1 tăng giá trị khi có xung cạnh lên. Counter chỉ tăng giá trị đếm sau khi nhận 1 xung cạnh xuống được minh hoạ hình 5-8.



Hình 5-8. Giản đồ thời gian xung đếm của Counter1.

Hình trên thì ngõ vào đang ở mức 1, counter sẽ đếm khi có xung cạnh lên.

Hình dưới thì ngõ vào đang ở mức thấp, xung cạnh lên thứ nhất thì mạch vẫn chưa đếm, xung xuống mức thấp và khi có cạnh lên thứ 2 thì counter bắt đầu đếm.

c. Hoạt động của Timer1 ở chế độ Counter đồng bộ:

Khi bit TMR1CS bằng 1 thì T1 hoạt động ở chế độ Counter:

- Nếu bit T1OSCEN bằng 1 thì đếm xung từ mạch dao động của T1.
- Nếu bit T1OSCEN bằng 1 thì đếm xung cạnh lên đưa đến ngõ vào RC0/T1OSO/T1CKI.

Nếu bit $\overline{T1SYNC}$ bằng 0 thì ngõ vào xung ngoại được đồng bộ với xung bên trong. Ở chế độ đồng bộ, nếu CPU ở chế độ ngủ thì Timer1 sẽ không đếm vì mạch đồng bộ ngừng hoạt động.

d. Hoạt động của Timer1 ở chế độ Counter bất đồng bộ:

Nếu bit $\overline{T1SYNC}$ bằng 1 thì xung ngõ vào từ bên ngoài không được đồng bộ. Bộ đếm tiếp tục tăng bất đồng bộ với xung bên trong. Bộ đếm vẫn đếm khi CPU ở trong chế độ ngủ và khi tràn sẽ phát sinh ngắt và đánh thức CPU. Ngắt T1 có thể ngăn được.

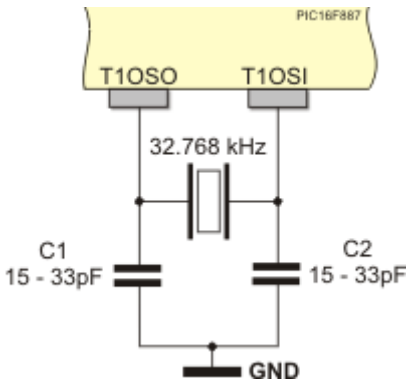
e. Đọc và ghi Timer1 trong chế độ đếm không đồng bộ:

Timer T1 cho phép đọc giá trị các thanh ghi TMR1H hoặc TMR1L khi timer đang đếm xung bất đồng bộ bên ngoài.

Khi ghi thì nên ngừng timer lại rồi mới ghi giá trị mong muốn vào các thanh ghi. Nếu ghi mà timer đang đếm vẫn được nhưng có thể tạo ra một giá trị đếm không dự đoán được hay không chính xác.

f. Bộ dao động của Timer1:

Mạch dao động được tích hợp bên trong và tụ thạch anh nối giữa 2 chân T1OSI và T1OSO để tạo dao động - xem hình 5-9. Bộ dao động được phép hoạt động khi bit T1OSCEN bằng 1.



Hình 5-9. Kết nối thạch anh tạo dao động.

Bộ dao động là dao động công suất thấp, tốc độ 200 kHz. Bộ dao động vẫn tiếp tục chạy khi CPU ở chế độ ngủ. Bộ dao động chỉ dùng với tụ thạch anh 32 kHz. Bảng sau trình bày cách lựa chọn tụ cho bộ dao động Timer1.

Osc Type	Freq.	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF

g. Reset Timer1 sử dụng ngõ ra CCP Trigger:

Nếu khối CCP1 và CCP2 được định cấu hình ở chế độ so sánh để tạo ra "xung kích" (CCP1M3:CCP1M0 = 1011), tín hiệu này sẽ reset Timer1.

Chú ý: "xung kích" từ khối CCP1 và CCP2 sẽ không làm bit cờ ngắt TMR1IF (PIR1<0>) bằng 1.

Timer1 phải định cấu hình ở chế độ định thời hoặc bộ đếm đồng bộ để tạo tiện ích cho cấu trúc này. Nếu Timer1 đang hoạt động ở chế độ đếm bất đồng bộ thì hoạt động Reset không thể thực hiện được.

h. Reset cặp thanh ghi TMR1H, TMR1L của Timer1:

Hoạt động reset lúc cấp nguồn POR (Power On Reset) hoặc bất kì reset nào khác không ảnh hưởng đến hai thanh ghi TMR1H và TMR1L, ngoại trừ khi xảy ra "xung kích" của CCP1 và CCP2 thì xóa hai thanh ghi về 00H.

Thanh ghi T1CON được reset về 00h

Reset lúc cấp nguồn POR hoặc khi Brown-out Reset sẽ xóa thanh ghi T1CON và timer T1 ở trạng thái ngừng (OFF) và hệ số chia trước là 1:1. Các reset khác thì thanh ghi không bị ảnh hưởng.

Các thanh ghi của Timer1 như bảng sau:

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

3. BỘ ĐỊNH THỜI TIMER2:

Timer2 là timer 8 bit có bộ chia trước (prescaler) và có bộ chia sau (postscaler).

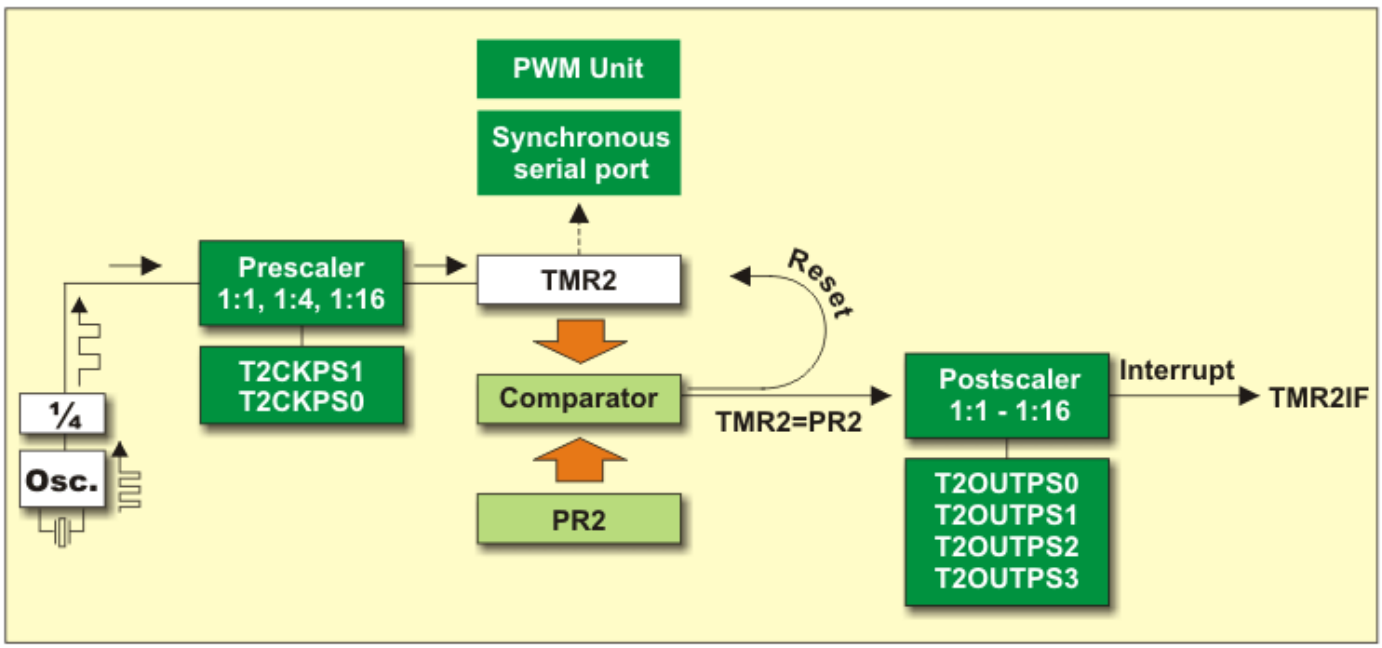
Timer2 được sử dụng điều chế xung khi khối CCP hoạt động ở chế độ PWM (pulse width modulation). Thanh ghi TMR2 có thể đọc/ghi và xoá khi bị reset.

Xung nội (F_{OSC}/4) qua bộ chia trước có hệ số chia: "1:1", "1:4" hoặc "1:16" được lựa chọn bằng các bit điều khiển T2CKPS1:T2CKPS0 (T2CON<1:0>).

PR2 là thanh ghi chu kỳ 8 bit dùng để so sánh. Timer2 tăng giá trị từ 00h cho đến khi bằng giá trị lưu trong thanh ghi PR2 thì reset về 00h rồi lặp lại.

PR2 là thanh ghi có thể đọc/ghi, khi hệ thống bị reset thì thanh ghi PR được khởi tạo giá trị FFH.

Ngõ ra của TMR2 đi qua bộ chia sau (postscaler) 4 bit trước khi phát sinh yêu cầu ngắt làm cờ TMR2IF (PIR1<1>) lên 1. Khi bit TMR2ON bằng 0 thì tắt Timer2 để giảm công suất tiêu thụ.



Hình 5-10. Sơ đồ khối của Timer2.

Thanh ghi điều khiển timer2:

T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Bit 7	Chưa sử dụng nếu đọc sẽ có giá trị '0'.
Bit 6-3	**TOUTPS3:TOUTPS0:** các bit lựa chọn ngõ ra bộ chia sau (Postscaler) của Timer2
	0000=1:1
	0001=1:2
	0010=1:3
	.
	.
	1111=1:16
Bit 2	**TMR2ON**: Bit điều khiển cho phép/cấm Timer2
	1= cho phép timer2 đếm.
	0= Timer2 ngừng đếm.
Bit 1-0	**T2CKPS1:T2CKPS0**: bit lựa chọn hệ số chia trước cho nguồn xung clock của timer2
	00= hệ số chia là 1.
	01= hệ số chia là 4.

1x= hệ số chia là 16.

a. Bộ chia trước và chia sau của Timer2:

Bộ đếm chia trước và bộ chia sau sẽ bị xóa khi xảy ra một trong các sự kiện sau:

- Thực hiện ghi dữ liệu vào thanh ghi TMR2.
- Thực hiện ghi dữ liệu vào thanh ghi T2CON.
- Bất kì Reset nào tác động.

TMR2 không bị xóa khi ghi dữ liệu vào thanh ghi T2CON.

b. Ngõ ra của TMR2:

Ngõ ra của TMR2 được nối tới khối SSP – khối này có thể tùy chọn để tạo ra xung nhịp.

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

III. TẬP LỆNH C CHO TIMER /COUNTER

Các lệnh của ngôn ngữ lập trình C liên quan đến TIMER/COUNTER bao gồm:

Lệnh SETUP_TIMER_X()
Lệnh SET_TIMER_X()
Lệnh SETUP_COUNTERS()
Lệnh SETUP_WDT()
Lệnh RESTART_WDT()
Lệnh GET_TIMER_X()

1. LỆNH SETUP_TIMER_0(MODE) - LỆNH ĐỊNH CẤU HÌNH CHO TIMER_0

Cú pháp: setup_timer_0(***mode***)

Thông số: ***mode*** có thể là 1 hoặc 2 hằng số định nghĩa trong file device.h. Các thông số gồm

RTCC_INTERNAL, RTCC_EXT_L_TO_H hoặc RTCC_EXT_H_TO_L

RTCC_DIV_2, RTCC_DIV_4, RTCC_DIV_8, RTCC_DIV_16, RTCC_DIV_32, RTCC_DIV_64, RTCC_DIV_128, RTCC_DIV_256.

Các hằng số từ nhiều nhóm khác nhau thì có thể or với nhau.

Chức năng: Định cấu hình cho TIMER0.

Có hiệu lực: cho tất cả các vi điều khiển PIC.

Ví dụ: **SETUP_TIMER_0 (RTCC_DIV_2|RTCC_EXT_L_TO_H);**

```
SETUP_TIMER_0(RTCC_EXT_L_TO_H | RTCC_DIV_2);// KHOI TAO COUNTER
SETUP_TIMER_0(RTCC_EXT_L_TO_H);                    // MAC NHIEN DIV_2

SET_TIMER0(0);                                      // GIA TRI BAT DAU
DEM=GET_TIMER0();

SETUP_COUNTERS(RTCC_EXT_L_TO_H,WDT_18MS);          // COUNTER
```

```
SET_TIMER0(0);                                  // GIA TRI BAT DAU
DEM=GET_TIMER0();
```

2. LỆNH SETUP_TIMER_1(MODE) - LỆNH ĐỊNH CẤU HÌNH CHO TIMER_1

Cú pháp: setup_timer_1(***mode***)

Thông số: ***mode*** có thể là 1 hoặc 2 hằng số định nghĩa trong file device.h. Các thông số gồm

T1_DISABLED, T1_INTERNAL, T1_EXTERNAL, T1_EXTERNAL_SYNC

TC_CLK_OUT

T1_DIV_BY_1, T1_DIV_BY_2, T1_DIV_BY_4, T1_DIV_BY_8.

Các hằng số từ nhiều nhóm khác nhau thì có thể or với nhau.

Với tần số thạch anh là 20MHz và khởi tạo T1_DIV_BY_8 thì timer sẽ tăng giá trị sau mỗi khoảng thời gian 1.6µs. Timer sẽ tràn sau 104.8576 ms.

Chức năng: khởi tạo cho TIMER1.

Với tần số thạch anh là 20MHz và khởi tạo T1_DIV_BY_8 thì timer sẽ tăng giá trị sau mỗi khoảng thời gian 1.6µs. Timer sẽ tràn sau 104.8576 ms.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer 1.

Ví dụ:
```
SETUP_TIMER_1 (T1_DISABLED);
SETUP_TIMER_1 (T1_INTERNAL|T1_DIV_BY_4);
SETUP_TIMER_1 (T1_INTERNAL|T1_DIV_BY_8);
```

```
SETUP_TIMER_1(T1_EXTERNAL | T1_DIV_BY_1);       // KHOI TAO COUNTER
SET_TIMER1(0);                                  // GIA TRI BAT DAU
DEM=GET_TIMER1();
```

3. LỆNH SETUP_TIMER_2(MODE) - LỆNH ĐỊNH CẤU HÌNH CHO TIMER_2

Cú pháp: setup_timer_2(***mode, period, postscale***)

Thông số: ***mode*** có thể là 1 trong các thông số: T2_DISABLED, T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16

Period là số nguyên có giá trị từ 0 đến 255 dùng để xác định khi nào giá trị clock bị reset.

postscale là số nguyên có giá trị từ 1 đến 16 dùng để xác định timer tràn bao nhiêu lần trước khi phát sinh tín hiệu ngắt.

Chức năng: khởi tạo cho TIMER2.

Mode chỉ định kiểu bộ chia của timer từ tần số của mạch dao động.

Giá trị của timer có thể đọc hoặc ghi dùng lệnh GET_TIMER2() và SET_TIMER2().

TIMER2 là timer 8 bit.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer 2.

Ví dụ: SETUP_TIMER_2 (T1_ DIV_BY_4,0XC0,2);

Với lệnh khởi tạo này thì timer2 sẽ tăng giá trị sau mỗi khoảng thời gian 800ns, sẽ tràn sau mỗi khoảng thời gian 154.4µs và phát sinh yêuc cầu ngắt sau 308.8 µs.

4. LỆNH SET_TIMERx(value) - LỆNH THIẾT LẬP GIÁ TRỊ BẮT ĐẦU CHO TIMER

Cú pháp: set_timerX(***value***) ; x là 0, 1, 2

Thông số: ***value*** là hằng số nguyên 8 hoặc 16 bit dùng để thiết lập giá trị mới cho timer.

Chức năng: thiết lập giá trị bắt đầu cho TIMER.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer.

Ví dụ: SET_TIMER2 (0); //reset timer2

5. LỆNH GET_TIMERx() - LỆNH ĐỌC GIÁ TRỊ CỦA TIMER

Cú pháp: value = get_timerX() ; x là 0, 1, 2

Thông số: ***không có***.

Chức năng: đọc giá trị của TIMER/COUNTER.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer.

Ví dụ: DEM=GET_TIMER0(); //đọc giá trị của timer0

6. LỆNH SETUP_WDT() - LỆNH THIẾT LẬP CHO WDT

Cú pháp: setup_wdt (***mode***)

Thông số: For PCB/PCM parts: WDT_18MS, WDT_36MS, WDT_72MS, WDT_144MS,WDT_288MS, WDT_576MS, WDT_1152MS, WDT_2304MS
For PIC®18 parts: WDT_ON, WDT_OFF

Chức năng: Thiết lập cho bộ định thời watchdog.
Bộ định thời watchdog được sử dụng để reset phần cứng nếu phần mềm lập trình bị sa lầy ở những vòng lặp vô tận hoặc một sự kiện nào đó chờ chúng xảy ra nhưng chúng lại không bao giờ xảy ra, để chấm dứt việc chờ đợi này thì ta phải sử dụng bộ định thời watchdog.
Trước khi tiến hành vào vòng lặp thì ta phải cho phép bộ định thời watchdog đếm với khoảng thời gian lập trình trước và nếu sự kiện đó không xảy ra thì bộ định thời watchdog sẽ hết thời gian lập trình sẽ reset phần cứng thoát khỏi vòng lặp.

	PCB/PCM	PCH
Enable/Disable	#fuses	setup_wdt()
Timeout time	setup_wdt()	#fuses
Restart	restart_wdt()	restart_wdt()

Có hiệu lực: Cho tất cả các thiết bị

Yêu cầu: #fuses, hằng số được định nghĩa trong file devices .h

Ví dụ:
```
#fuses WDT1     // PIC18 example, See  restart_wdt for a PIC18 example
main() {        // WDT1 means 18ms*1 for old PIC18s and 4ms*1 for new PIC18s
                // setup_wdt(WDT_ON);
  while (TRUE) {
   restart_wdt();
   perform_activity();
  }
}
```

7. LỆNH RESTART_WDT() - LỆNH KHỞI ĐỘNG LẠI CHO WDT

Cú pháp: restart_wdt()

Thông số: Không có

Chức năng: Khởi tạo lại bộ định thời watchdog. Nếu bộ định thời watchdog được cho phép thì lệnh này sẽ khởi tạo lại giá trị định thời để ngăn chặn không cho bộ định thời reset CPU vì sự kiện chờ đợi đã xảy ra.

	PCB/PCM	PCH
Enable/Disable	#fuses	setup_wdt()
Timeout time	setup_wdt()	#fuses
restart	restart_wdt()	restart_wdt()

Availability: All devices

Requires: #fuses

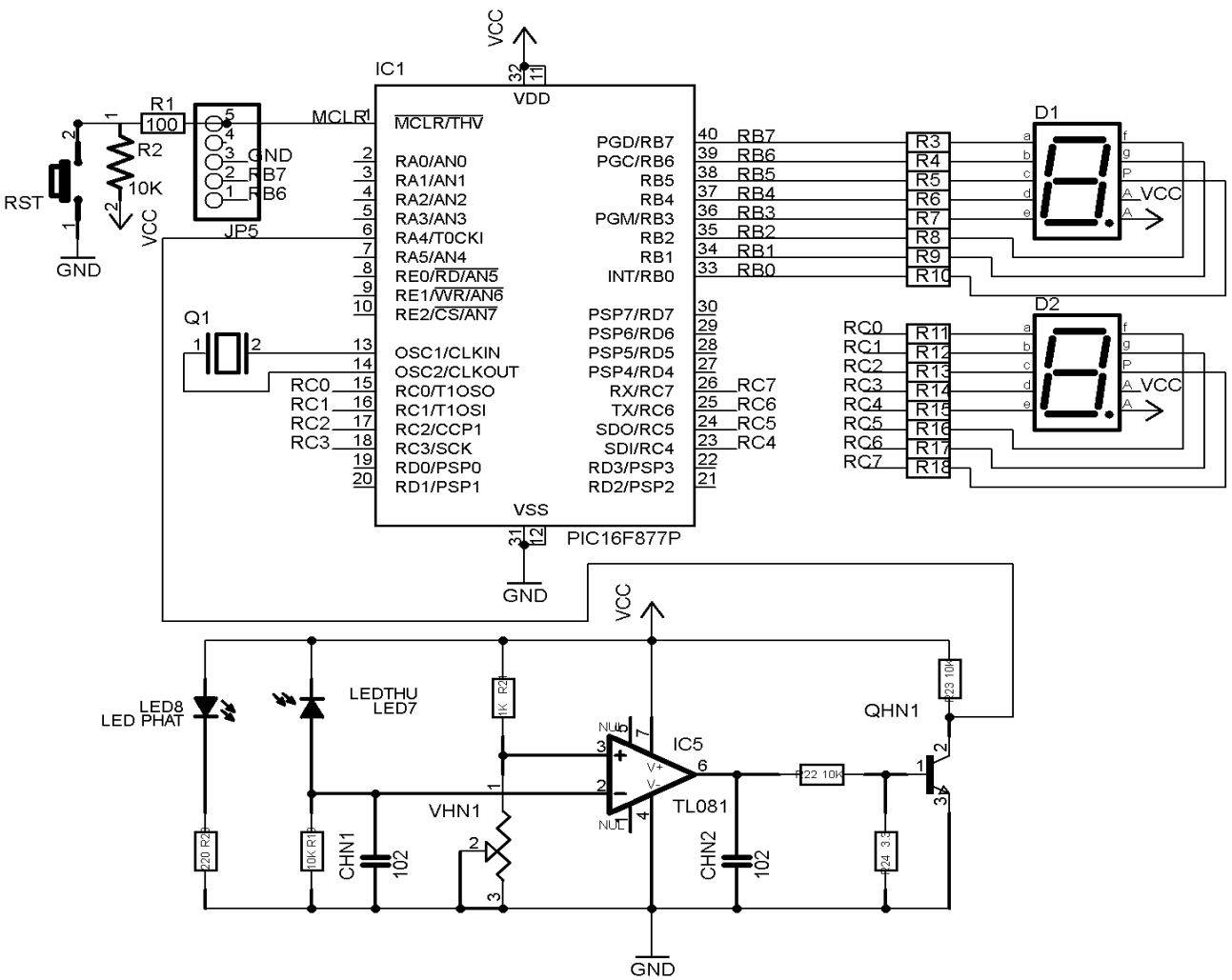
Examples:
```
#fuses WDT    // PCB/PCM example
              // See setup_wdt for a PIC18 example
main() {
  setup_wdt(WDT_2304MS);
  while (TRUE) {
   restart_wdt();
   perform_activity();
  }
}
```

IV. CÁC CHƯƠNG TRÌNH ỨNG DỤNG TIMER /COUNTER

1. ỨNG DỤNG ĐẾM XUNG NGOẠI HIỂN THỊ LED 7 ĐOẠN TRỰC TIẾP

BÀI 5-1 –ĐẾM XUNG NGOẠI DÙNG COUNTER0, GIÁ TRỊ TỪ 00 ĐẾN 99 HIỂN THỊ TRÊN 2 LED 7 ĐOẠN NỐI TRỰC TIẾP VỚI PORTB VÀ PORTC

SƠ ĐỒ MẠCH ĐIỆN



Hình 5-11. Mạch đếm hiển thị trên 2 led.

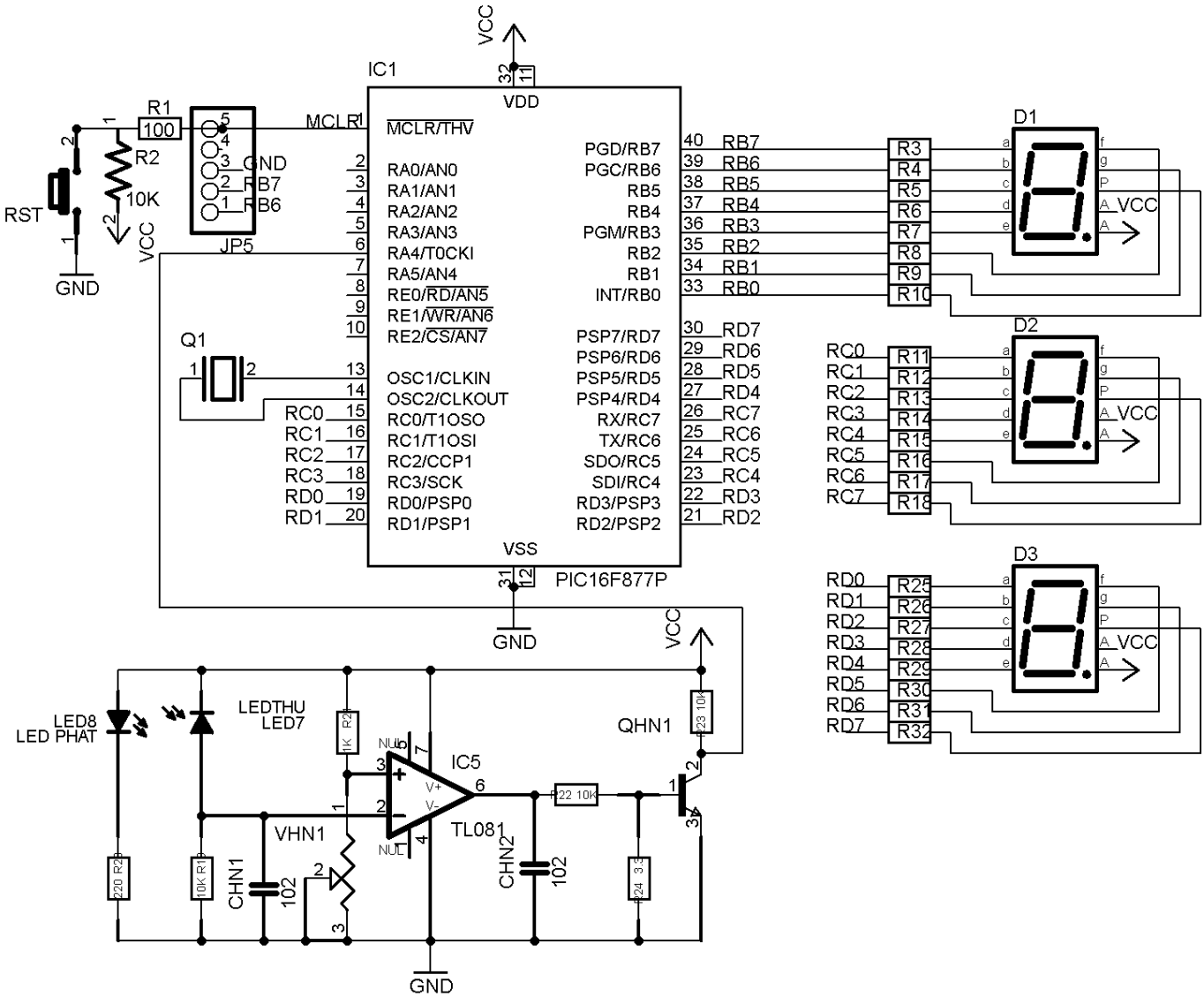
CHƯƠNG TRÌNH

```
#INCLUDE <16F877A.H>
```

```
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)

CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT             DONVI, CHUC, DEM;
UNSIGNED  CHAR  MDONVI,MCHUC;

VOID  DIV_DECODE (INT TAM)
{
        CHUC  = TAM/10;                         //LAY HANG CHUC
        DONVI= TAM %10;                         //LAY DONVI
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];
}
VOID DISPLAY ()
  {     OUTPUT_B(MDONVI);
        OUTPUT_C(MCHUC);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                       //PORTB OUT
        SET_TRIS_C(0x00);                       // PORTC OUT
        SETUP_TIMER_0 (RTCC_EXT_L_TO_H |  RTCC_DIV_1);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU

        WHILE(1)
        {       DEM=GET_TIMER0();
                DIV_DECODE(DEM);
                DISPLAY();
        }
}
```

BÀI 5-2. ĐẾM XUNG NGOẠI DÙNG COUNTER0, GIÁ TRỊ TỪ 000 ĐẾN 255 HIỂN THỊ TRÊN 3 LED 7 ĐOẠN NỐI TRỰC TIẾP VỚI 3 PORTB, PORTC VÀ PORTD.

SƠ ĐỒ MẠCH ĐIỆN



Hình 5-12. Mạch đếm hiển thị trên 3 led.

CHƯƠNG TRÌNH

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT                 DEM, TRAM, CHUC, DONVI;
UNSIGNED  CHAR      MTRAM, MCHUC, MDONVI;

VOID  DIV_DECODE (INT TAM)                  //TAM=123
{
        TRAM = TAM/100;                     //LAY HANG TRAM=1
        TAM = TAM %100;                     //LAY HANG VA DONVI =23
        CHUC  = TAM/10;                     //LAY HANG CHUC=2
        DONVI= TAM %10;                     //LAY DONVI=3
        MTRAM = MALED7DOAN[TRAM];
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];
}
VOID DISPLAY ()
```

```
  {       OUTPUT_B(MDONVI);
          OUTPUT_D(MCHUC);
          OUTPUT_C(MTRAM);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                       //PORTB OUT
        SET_TRIS_D(0x00);                       // PORTD OUT
        SET_TRIS_C(0x00);                       // PORTC OUT

        SETUP_TIMER_0 (RTCC_EXT_L_TO_H |  RTCC_DIV_1);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU

        WHILE(1)
        {       DEM=GET_TIMER0();
                DIV_DECODE(DEM);
                DISPLAY();
        }
}
```

BÀI 5-3. ĐẾM XUNG NGOẠI DÙNG COUNTER0, GIÁ TRỊ TỪ 00 ĐẾN 99 HIỂN THỊ TRÊN 2 LED 7 ĐOẠN NỐI TRỰC TIẾP VỚI 2 PORTB, PORTC.

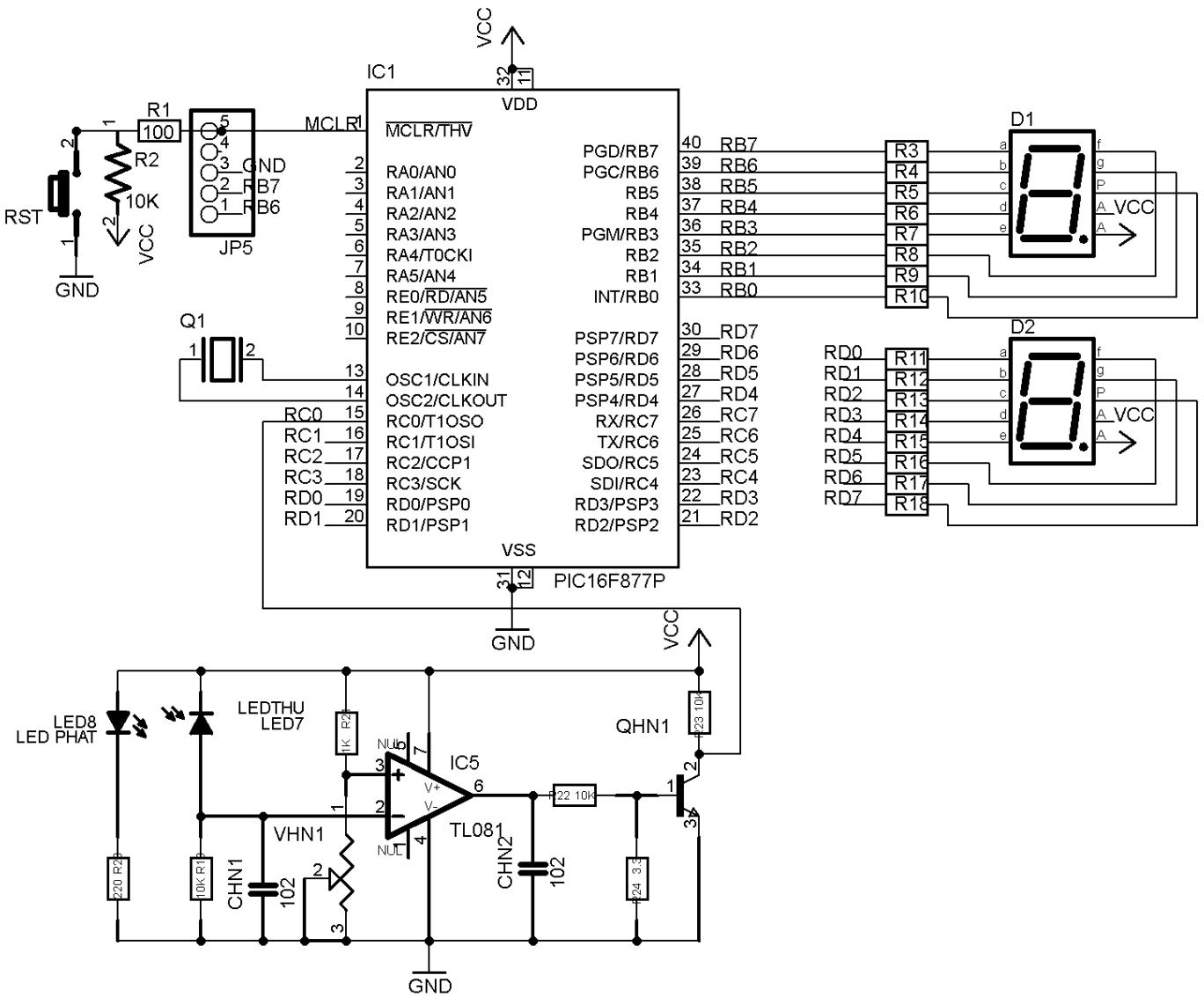
SƠ ĐỒ MẠCH ĐIỆN – GIỐNG BÀI 5-1.
CHƯƠNG TRÌNH – CŨNG GIỐNG BÀI 5-1

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT                     DONVI, CHUC, DEM;
UNSIGNED  CHAR  MDONVI,MCHUC;

VOID  DIV_DECODE (INT TAM)
{
        CHUC  = TAM/10;                 //LAY HANG CHUC
        DONVI= TAM %10;                 //LAY DONVI
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];
}
VOID DISPLAY ()
  {     OUTPUT_B(MDONVI);
        OUTPUT_C(MCHUC);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                       //PORTB OUT
        SET_TRIS_C(0x00);                       // PORTC OUT
        SETUP_TIMER_0 (RTCC_EXT_L_TO_H |  RTCC_DIV_1);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU

        WHILE(1)
        {       DEM=GET_TIMER0();
                DIV_DECODE(DEM);
                IF (DEM==100) SET_TIMER0(1);
```

```
                DISPLAY();
            }
    }
```

BÀI 5-4. ĐẾM XUNG NGOẠI DÙNG COUNTER1, GIÁ TRỊ TỪ 00 ĐẾN 99 HIỂN THỊ TRÊN 2 LED 7 ĐOẠN NỐI TRỰC TIẾP VỚI 2 PORTB, PORTD. CHÚ Ý KHI BẰNG 100 THÌ QUAY VỀ 1

SÔ ÑOÀ MAÏCH ÑIEÄN



Hình 5-13. Mạch đếm hiển thị trên 2 led.

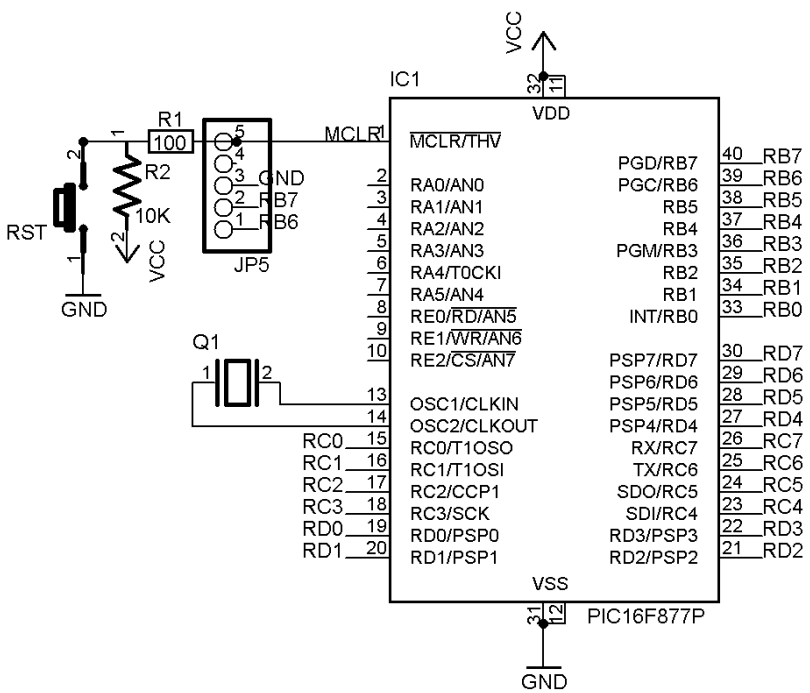
CHƯƠNG TRÌNH

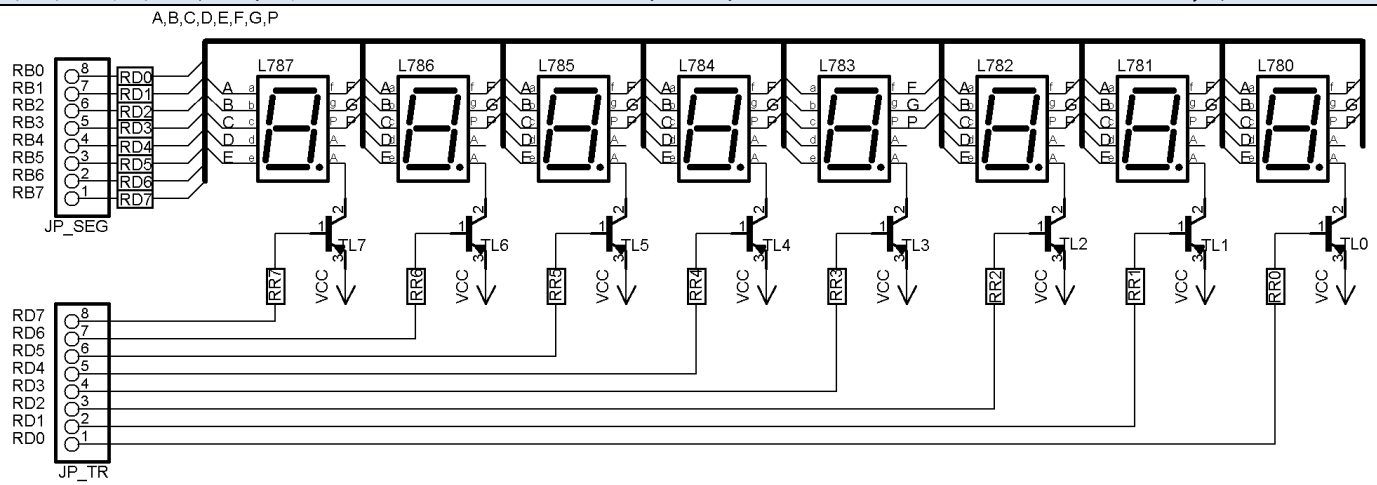
```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT                 DONVI, CHUC, DEM;
UNSIGNED  CHAR      MDONVI, MCHUC;

VOID  DIV_DECODE (INT TAM)
```

```
{
        CHUC  = TAM/10;                         //LAY HANG CHUC
        DONVI= TAM %10;                         //LAY DONVI
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];
}
VOID DISPLAY ()
  {     OUTPUT_B(MDONVI);
        OUTPUT_D(MCHUC);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                       //PORTB OUT
        SET_TRIS_D(0x00);                       // PORTD OUT
        SETUP_TIMER_1 (T1_EXTERNAL | T1_DIV_BY_1);      // KHOITAOTIMER
        SET_TIMER1(0);                          // GIA TRI BAT DAU

        WHILE(1)
        {       DEM=GET_TIMER1();
                DIV_DECODE(DEM);
                IF (DEM==100) SET_TIMER1(1);
                DISPLAY();
        }
}
```

2. ỨNG DỤNG ĐẾM XUNG NGOẠI HIỂN THỊ LED 7 ĐOẠN QUÉT

SƠ ĐỒ MẠCH GIAO TIẾP PIC VỚI 8 LED 7 ĐOẠN DÙNG PORTB VÀ PORTB





Hình 5-14. Vi điều khiển PIC giao tiếp với 8 led 7 đoạn.

BÀI 5-5: ĐIỀU KHIỂN 8 LED 7 ĐOẠN QUÉT HIỂN THỊ 8 SỐ TỪ 0 ĐẾN 7 TRÊN 8 LED DÙNG PORTB ĐIỀU KHIỂN CÁC ĐOẠN A,B,C,D,E,F,G,DP. PORTD ĐIỀU KHIỂN CÁC TRANSISTOR ĐỂ TẮT MỞ LED

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
VOID MAIN()
{       SET_TRIS_B(0x00);          SET_TRIS_D(0x00);
        WHILE(1)
                {
                        OUTPUT_B(0xC0);  OUTPUT_D(0B11111110);        DELAY_MS(1);
                        OUTPUT_B(0XF9);  OUTPUT_D(0B11111101);        DELAY_MS(1);
                        OUTPUT_B(0XA4);  OUTPUT_D(0B11111011);        DELAY_MS(1);
                        OUTPUT_B(0XB0);  OUTPUT_D(0B11110111);        DELAY_MS(1);
                        OUTPUT_B(0x99);  OUTPUT_D(0B11101111);        DELAY_MS(1);
                        OUTPUT_B(0X92);  OUTPUT_D(0B11011111);        DELAY_MS(1);
                        OUTPUT_B(0X82);  OUTPUT_D(0B10111111);        DELAY_MS(1);
                        OUTPUT_B(0XF8);  OUTPUT_D(0B01111111);        DELAY_MS(1);
                }
}
```

BÀI 5-6: CHƯƠNG TRÌNH ĐẾM TỪ 00 ĐẾN 99 HIỂN THỊ LED 7 ĐOẠN QUÉT DÙNG PORTB ĐIỀU KHIỂN CÁC ĐOẠN A,B,C,D,E,F,G,DP. PORTD ĐIỀU KHIỂN CÁC TRANSISTOR ĐỂ TẮT MỞ LED

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
INT                 DEM, DONVI, CHUC, I;
UNSIGNED  CHAR      MDONVI, MCHUC;
VOID DIV_BCD(INT TAM)          //CH TR CON TACH CHUC VA DON VI – GIAI MA
{
        CHUC        =       TAM /10;      // CHUC BANG  TAM CHIA CHO 10
        DONVI       =       TAM%10;
        MCHUC       =       MALED7DOAN[CHUC];
        MDONVI      =       MALED7DOAN[DONVI];
```

```
}
 VOID DELAY_HIENTHI()
{       FOR (I=0; I<250;I++)
        {       OUTPUT_B(MDONVI);       OUTPUT_D(0xFE);         DELAY_MS(1);
                OUTPUT_B(MCHUC);        OUTPUT_D(0xFD);         DELAY_MS(1);
        }
}
VOID MAIN()
{       SET_TRIS_B(0x00);           SET_TRIS_D(0x00);
        WHILE(1)
        {       FOR (DEM=0; DEM<100; DEM++)
                        {       DIV_BCD(DEM);
                                DELAY_HIENTHI();
                        }
        }
}
```

BÀI 5-7: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM TỪ 00 ĐẾN 99 HIỂN THỊ LED QUÉT VÀ HIỂN THỊ GIÁ TRỊ ĐẾM DẠNG NHỊ PHÂN
DÙNG PORTB ĐIỀU KHIỂN CÁC ĐOẠN A,B,C,D,E,F,G,DP.
PORTD ĐIỀU KHIỂN CÁC TRANS

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT                     DEM, DONVI, CHUC, I;
UNSIGNED  CHAR          MDONVI, MCHUC;

VOID DIV_BCD(INT TAM)           //CH TR CON TACH CHUC VA DON VI – GIAI MA
{
        CHUC        =       TAM /10;      // CHUC BANG  TAM CHIA CHO 10
        DONVI       =       TAM%10;
        MCHUC       =       MALED7DOAN[CHUC];
        MDONVI      =       MALED7DOAN[DONVI];
}
 VOID HIENTHI()
{
        FOR (I=0; I<250;I++)
        {
                OUTPUT_B(MDONVI);       OUTPUT_D(0xFE);         DELAY_MS(1);
                OUTPUT_B(MCHUC);        OUTPUT_D(0xFD);         DELAY_MS(1);
        }
}

VOID MAIN()
{       SET_TRIS_B(0x00);
        SET_TRIS_D(0x00);
        SET_TRIS_C(0x00);

        WHILE(1)
        {
                FOR (DEM=0;DEM<100;DEM++)
                        {
```

```
                    OUTPUT_C(DEM);
                    DIV_BCD(DEM);
                    HIENTHI();
                }
            }
        }
```

BÀI 5-8: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM TỪ 0000 ĐẾN 9999 HIỂN THỊ LED QUÉT DÙNG PB ĐIỀU KHIỂN CÁC ĐOẠN A,B,C,D,E,F,G,DP. PD ĐIỀU KHIỂN CÁC TRANS

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};
INT16               DEM;
INT                 DONVI, CHUC, TRAM, NGAN, I;
UNSIGNED  CHAR      MDONVI, MCHUC, MTRAM, MNGAN;

VOID DIV_BCD (INT16 TAM)                    //VI DU TAM = 3567
{
        NGAN    =   TAM/1000;       //NGAN =3
        TAM     =   TAM%1000;       //TAM = 567
        TRAM    =   TAM/100;        //TRAM = 5
        TAM     =   TAM%100;        //TAM = 67
        CHUC    =   TAM /10;        //CHUC = 6
        DONVI   =   TAM%10;         //DONVI = 7

        MNGAN   =   MALED7DOAN[NGAN];
        MTRAM   =   MALED7DOAN[TRAM];
        MCHUC   =   MALED7DOAN[CHUC];
        MDONVI  =   MALED7DOAN[DONVI];
}
 VOID HIENTHI()
{
        FOR (I=0; I<2;I++)
        {
            OUTPUT_B(MDONVI);   OUTPUT_D(0B11111110);   DELAY_MS(1);
            OUTPUT_B(MCHUC);    OUTPUT_D(0B11111101);   DELAY_MS(1);
            OUTPUT_B(MTRAM);    OUTPUT_D(0B11111011);   DELAY_MS(1);
            OUTPUT_B(MNGAN);    OUTPUT_D(0B11110111);   DELAY_MS(1);
        }
}

VOID MAIN()
{       SET_TRIS_B(0x00);
        SET_TRIS_D(0x00);
        WHILE(1)
        {
            FOR (DEM=0;DEM<10000;DEM++)
                {
                    DIV_BCD(DEM);
                    HIENTHI();
                }
        }
}
```

BÀI 5-9: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM TỪ 0000 ĐẾN 9999 HIỂN THỊ LED QUÉT DÙNG PB ĐIỀU KHIỂN CÁC ĐOẠN A,B,C,D,E,F,G,DP. PD ĐIỀU KHIỂN CÁC TRANS – XÓA SỐ 0 VÔ NGHĨA

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};
INT16                DEM;
INT                  DONVI, CHUC, TRAM, NGAN, I;
UNSIGNED  CHAR       MDONVI, MCHUC, MTRAM, MNGAN;

VOID DIV_BCD (INT16 TAM)                    //VI DU TAM = 3567
{
        NGAN    =    TAM/1000;              //NGAN =3
        TAM     =    TAM%1000;              //TAM = 567
        TRAM    =    TAM/100;               //TRAM = 5
        TAM     =    TAM%100;               //TAM = 67
        CHUC    =    TAM /10;               //CHUC = 6
        DONVI   =    TAM%10;                //DONVI = 7

        MNGAN   =    MALED7DOAN[NGAN];
        MTRAM   =    MALED7DOAN[TRAM];
        MCHUC   =    MALED7DOAN[CHUC];
        MDONVI  =    MALED7DOAN[DONVI];
        IF      (MNGAN == 0XC0)
                {
                        MNGAN=0XFF;
                        IF (MTRAM ==0XC0)
                                {MTRAM=0XFF;
                                        IF (MCHUC == 0XC0)
                                                {MCHUC =0XFF;
                                                }
                                }
                }
}
 VOID HIENTHI()
{
        FOR (I=0; I<2;I++)
        {
                OUTPUT_B(MDONVI);    OUTPUT_D(0B11111110);        DELAY_MS(1);
                OUTPUT_B(MCHUC);     OUTPUT_D(0B11111101);        DELAY_MS(1);
                OUTPUT_B(MTRAM);     OUTPUT_D(0B11111011);        DELAY_MS(1);
                OUTPUT_B(MNGAN);     OUTPUT_D(0B11110111);        DELAY_MS(1);
        }
}

VOID MAIN()
{       SET_TRIS_B(0x00);
        SET_TRIS_D(0x00);
        WHILE(1)
        {
                FOR (DEM=0;DEM<2000;DEM++)
                        {
                                DIV_BCD(DEM);
```

```
                    HIENTHI();
                }
            }
        }
```

BÀI 5-10: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM TỪ 000 ĐẾN 200 – LED QUÉT

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)

CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT             DEM,DONVI,CHUC,I,TRAM;      // BIEN DEM
UNSIGNED  CHAR  MDONVI,MCHUC,MTRAM;         // BIEN CHUA MA 7 DOAN

VOID  DIV_DECODE (INT TAM)
{
        TRAM = TAM / 100;                   //LAY HANG TRAM
        TAM   = TAM %100;                   /LAY CHUC VA DON VI
        CHUC  = TAM/10;                     //LAY HANG CHUC
        DONVI= TAM %10;                     //LAY DONVI
        MDONVI= MALED7DOAN[DONVI];          //LAY MA 7 D
        MCHUC = MALED7DOAN[CHUC];           //LAY MA 7 D
        MTRAM = MALED7DOAN[TRAM];           //LAY MA 7 D

}
VOID DISPLAY ()
  {
            OUTPUT_B(MDONVI);   OUTPUT_D(0B11111110);   DELAY_MS(1);
            OUTPUT_B(MCHUC);    OUTPUT_D(0B11111101);   DELAY_MS(1);
            OUTPUT_B(MTRAM);    OUTPUT_D(0B11111011);   DELAY_MS(1);

  }
VOID MAIN()
{       SET_TRIS_B(0x00);                   //PORTB OUT
        SET_TRIS_D(0x00);                   // PORTD OUT
        WHILE(1)
        {       FOR (DEM=0;DEM<201;DEM++)
                {       DIV_DECODE(DEM);
                        FOR (I=0;I<50;I++)
                                {       DISPLAY();
                                }
                }
        }
}
```

BÀI 5-11: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM TỪ 000 ĐẾN 300 VÀ XÓA SỐ 0 VÔ NGHĨA

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)

CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT16           DEM;                        // BIEN DEM 16 BIT
INT             DONVI,CHUC,I,TRAM;          // BIEN 8 BIT
```

```
UNSIGNED  CHAR     MDONVI,MCHUC,MTRAM;        // BIEN CHUA MA 7 DOAN

VOID  DIV_DECODE (INT16 TAM)
{       TRAM = TAM / 100;                     //LAY HANG TRAM
        TAM   = TAM %100;                     //LAY CHUC VA DON VI
        CHUC  = TAM/10;                       //LAY HANG CHUC
        DONVI= TAM %10;                       //LAY DONVI

        MTRAM = MALED7DOAN[TRAM];
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];

        IF (TRAM == 0)                        // XOA SO 0 VO NGHIA
            {   MTRAM=0XFF;
                IF (CHUC == 0 )
                    { MCHUC=0XFF;
                    }
            }
}
VOID DISPLAY ()
  {         OUTPUT_B(MDONVI);   OUTPUT_D(0B11111110);   DELAY_MS(1);
            OUTPUT_B(MCHUC);    OUTPUT_D(0B11111101);   DELAY_MS(1);
            OUTPUT_B(MTRAM);    OUTPUT_D(0B11111011);   DELAY_MS(1);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                     //PORTB OUT
        SET_TRIS_D(0x00);                     // PORTD OUT
        WHILE(1)
        {   FOR (DEM=0;DEM<301;DEM++)
                {   DIV_DECODE(DEM);
                    FOR (I=0;I<20;I++)
                        {   DISPLAY();
                        }
                }
        }
}
```

BÀI 5-12: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM PHÚT GIÂY

```
 #INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};
INT             PHUT,GIAY,DVPHUT,CHPHUT,DVGIAY,CHGIAY,I;
UNSIGNED  CHAR     MCHPHUT, MDVPHUT,MCHGIAY, MDVGIAY;

VOID DIV_BCD_PHUT (INT TAMP)
{
        CHPHUT      =       TAMP/10;
        DVPHUT      =       TAMP%10;
        MCHPHUT     =       MALED7DOAN[CHPHUT];
        MDVPHUT     =       MALED7DOAN[DVPHUT];
}
VOID DIV_BCD_GIAY (INT TAMG)
```

```
{
        CHGIAY      =       TAMG/10;
        DVGIAY      =       TAMG%10;
        MCHGIAY     =       MALED7DOAN[CHGIAY];
        MDVGIAY     =       MALED7DOAN[DVGIAY];
}
 VOID HIENTHI()
{
        FOR (I=0; I<2;I++)
        {
                OUTPUT_B(MDVGIAY);   OUTPUT_D(0B11111110);        DELAY_MS(1);
                OUTPUT_B(MCHGIAY);   OUTPUT_D(0B11111101);        DELAY_MS(1);

                OUTPUT_B(MDVPHUT);   OUTPUT_D(0B11110111);        DELAY_MS(1);
                OUTPUT_B(MCHPHUT);   OUTPUT_D(0B11101111);        DELAY_MS(1);
        }
}
VOID MAIN()
{       SET_TRIS_B(0x00);           SET_TRIS_D(0x00);
        WHILE(1)
        {
                FOR (PHUT=0;PHUT<60;PHUT++)
                        {
                        FOR (GIAY=0; GIAY <60; GIAY ++)
                                {
                                        DIV_BCD_PHUT(PHUT);
                                        DIV_BCD_GIAY(GIAY);
                                        HIENTHI();
                                }
                        }
        }
}
```

BÀI 5-13: ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT ĐẾM GIỜ PHÚT GIÂY

```
 #INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};
INT             PHUT,GIAY ,DVPHUT,CHPHUT,DVGIAY,CHGIAY,I;
UNSIGNED  CHAR    MCHPHUT, MDVPHUT,MCHGIAY, MDVGIAY;

INT             GIO ,DVGIO,CHGIO;
UNSIGNED  CHAR    MCHGIO, MDVGIO;
VOID DIV_BCD_GIO (INT TAMGIO)
{
        CHGIO       =       TAMGIO/10;
        DVGIO       =       TAMGIO%10;
        MCHGIO      =       MALED7DOAN[CHGIO];
        MDVGIO      =       MALED7DOAN[DVGIO];
}

VOID DIV_BCD_PHUT (INT TAMP)
{
        CHPHUT      =       TAMP/10;
        DVPHUT      =       TAMP%10;
```

```
        MCHPHUT   =   MALED7DOAN[CHPHUT];
        MDVPHUT   =   MALED7DOAN[DVPHUT];
}
VOID DIV_BCD_GIAY (INT TAMG)
{
        CHGIAY    =   TAMG/10;
        DVGIAY    =   TAMG%10;
        MCHGIAY   =   MALED7DOAN[CHGIAY];
        MDVGIAY   =   MALED7DOAN[DVGIAY];
}
 VOID HIENTHI()
{
        FOR (I=0; I<2;I++)
        {
                OUTPUT_B(MDVGIAY);   OUTPUT_D(0B11111110);   DELAY_MS(1);
                OUTPUT_B(MCHGIAY);   OUTPUT_D(0B11111101);   DELAY_MS(1);

                OUTPUT_B(MDVPHUT);   OUTPUT_D(0B11110111);   DELAY_MS(1);
                OUTPUT_B(MCHPHUT);   OUTPUT_D(0B11101111);   DELAY_MS(1);

                OUTPUT_B(MDVGIO);    OUTPUT_D(0B10111111);   DELAY_MS(1);
                OUTPUT_B(MCHGIO);    OUTPUT_D(0B01111111);   DELAY_MS(1);

        }
}
VOID MAIN()
{       SET_TRIS_B(0x00);       SET_TRIS_D(0x00);
        WHILE(1)
        {
        FOR(GIO=0;GIO<24;GIO++)
        {
                FOR (PHUT=0;PHUT<60;PHUT++)
                        {
                        FOR (GIAY=0; GIAY <60; GIAY ++)
                                {       DIV_BCD_GIO(GIO);
                                        DIV_BCD_PHUT(PHUT);
                                        DIV_BCD_GIAY(GIAY);
                                        HIENTHI();
                                }
                        }
        }
        }
}
```

BÀI 5-14: TIMER0 ĐẾM XUNG NGOẠI – COUNTER0 TỪ 000 ĐẾN 255 HIỂN THỊ TRÊN 3 LED QUET

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,
0x90};
INT                 DEM, TRAM, CHUC, DONVI;
UNSIGNED  CHAR  MTRAM, MCHUC, MDONVI;
```

```
VOID  DIV_DECODE (INT TAM)
      {       TRAM = TAM/100;                 TAM = TAM %100;
              CHUC = TAM/10;                          //LAY HANG CHUC
              DONVI= TAM %10;                         //LAY DONVI

              MTRAM = MALED7DOAN[TRAM];
              MCHUC = MALED7DOAN[CHUC];
              MDONVI= MALED7DOAN[DONVI];

              IF (TRAM == 0)                          // XOA SO 0 VO NGHIA
                    {       MTRAM=0XFF;
                            IF (CHUC == 0)
                                  { MCHUC=0XFF;     }
                     }
      }
      VOID DISPLAY ()
            {
                  OUTPUT_B(MDONVI);     OUTPUT_LOW(PIN_D0);
                  DELAY_MS(1);          OUTPUT_HIGH(PIN_D0);

                  OUTPUT_B(MCHUC);      OUTPUT_LOW(PIN_D1);
                  DELAY_MS(1);          OUTPUT_HIGH(PIN_D1);

                  OUTPUT_B(MTRAM);      OUTPUT_LOW(PIN_D2);
                  DELAY_MS(1);          OUTPUT_HIGH(PIN_D2);
            }
      VOID MAIN()
      {           SET_TRIS_B(0x00);     SET_TRIS_D(0x00);
                  OUTPUT_D(0XFF);
                  SETUP_TIMER_0(RTCC_EXT_H_TO_L | RTCC_DIV_1);// KHOITAOTIMER
                  SET_TIMER0(0);                                  // GIA TRI BAT DAU
                  WHILE(1)
                  {       DEM=GET_TIMER0();
                          DIV_DECODE(DEM);
                          DISPLAY();
                  }
      }
```

BÀI 5-15: TIMER0 ĐẾM SỐ VỊNG QUAY – 4 XUNG LÀ 1 VÒNG – COUNTER0 TỪ 000 ĐẾN 255 HIỂN THỊ TRÊN 3 LED QUET

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};
INT                 DEM, TRAM, CHUC, DONVI;
UNSIGNED  CHAR      MTRAM, MCHUC, MDONVI;

VOID  DIV_DECODE (INT TAM)
{
      TRAM = TAM/100;
      TAM = TAM %100;
      CHUC  = TAM/10;                         //LAY HANG CHUC
      DONVI= TAM %10;                         //LAY DONVI
```

```
        MTRAM = MALED7DOAN[TRAM];
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];

        IF (TRAM == 0)                          // XOA SO 0 VO NGHIA
            {   MTRAM=0XFF;
                IF (CHUC == 0 )
                    { MCHUC=0XFF;
                    }
            }
}
VOID DISPLAY ()
  {
        OUTPUT_B(MDONVI);           OUTPUT_LOW(PIN_D0);
        DELAY_MS(1);                OUTPUT_HIGH(PIN_D0);

        OUTPUT_B(MCHUC);            OUTPUT_LOW(PIN_D1);
        DELAY_MS(1);                OUTPUT_HIGH(PIN_D1);

        OUTPUT_B(MTRAM);            OUTPUT_LOW(PIN_D2);
        DELAY_MS(1);                OUTPUT_HIGH(PIN_D2);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                   //PORTB OUT
        SET_TRIS_D(0x00);                   // PORTD OUT
        OUTPUT_D(0XFF);
        SETUP_TIMER_0(RTCC_EXT_H_TO_L | RTCC_DIV_4);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU
        WHILE(1)
        {   DEM=GET_TIMER0();
            DIV_DECODE(DEM);
            DISPLAY();
        }
}
```

BÀI 5-16: TIMER0 ĐẾM 0 ĐẾN 100 HIỂN THỊ TRÊN 3 LED QUET

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};
INT             DEM, TRAM, CHUC, DONVI;
UNSIGNED  CHAR  MTRAM, MCHUC, MDONVI;

VOID  DIV_DECODE (INT TAM)
{
        TRAM = TAM/100;
        TAM = TAM %100;
        CHUC  = TAM/10;                 //LAY HANG CHUC
        DONVI= TAM %10;                 //LAY DONVI

        MTRAM = MALED7DOAN[TRAM];
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];
```

```
        IF (TRAM == 0)                          // XOA SO 0 VO NGHIA
            {   MTRAM=0XFF;
                IF (CHUC == 0 )
                    { MCHUC=0XFF;
                    }
            }
}
VOID DISPLAY ()
  {
        OUTPUT_B(MDONVI);           OUTPUT_LOW(PIN_D0);
        DELAY_MS(1);                OUTPUT_HIGH(PIN_D0);

        OUTPUT_B(MCHUC);            OUTPUT_LOW(PIN_D1);
        DELAY_MS(1);                OUTPUT_HIGH(PIN_D1);

        OUTPUT_B(MTRAM);            OUTPUT_LOW(PIN_D2);
        DELAY_MS(1);                OUTPUT_HIGH(PIN_D2);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);                   //PORTB OUT
        SET_TRIS_D(0x00);                   // PORTD OUT
        OUTPUT_D(0XFF);
        SETUP_TIMER_0(RTCC_EXT_H_TO_L | RTCC_DIV_1);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU
        WHILE(1)
        {
            DEM=GET_TIMER0();

            IF (DEM==101)
                {SET_TIMER0(1);
                }
            DIV_DECODE(DEM);
            DISPLAY();
        }
}
```

THAY DOI

```
VOID MAIN()
{       SET_TRIS_B(0x00);                   //PORTB OUT
        SET_TRIS_D(0x00);                   // PORTD OUT

        SETUP_COUNTERS(RTCC_EXT_L_TO_H,WDT_18MS);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU

        WHILE(1)
        {   DEM=GET_TIMER0();
            DIV_DECODE(DEM);
            DISPLAY();
        }
}
```

BÀI 5-17: TIMER0 ĐẾM XUNG NGOẠI – COUNTER0 TỪ 000 ĐẾN 255 VÀ XÓA SỐ 0 VÔ NGHĨA

```
#INCLUDE <16F877A.H>
```

```
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)

CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT             DONVI,CHUC, TRAM, I,DEM;
UNSIGNED  CHAR  MDONVI,MCHUC,MTRAM;

VOID  DIV_DECODE (INT TAM)
{       TRAM = TAM / 100;                   //LAY HANG TRAM
        TAM   = TAM %100;                   //LAY CHUC VA DON VI
        CHUC  = TAM/10;                     //LAY HANG CHUC
        DONVI= TAM %10;                     //LAY DONVI

        MTRAM = MALED7DOAN[TRAM];
        MCHUC = MALED7DOAN[CHUC];
        MDONVI= MALED7DOAN[DONVI];

        IF (TRAM == 0)
                {       MTRAM=0XFF;
                        IF (CHUC == 0 )
                                { MCHUC=0XFF;
                                }
                }
}
VOID DISPLAY ()
  {     OUTPUT_B(MDONVI);         OUTPUT_LOW(PIN_D0);
        DELAY_MS(1);              OUTPUT_HIGH(PIN_D0);

        OUTPUT_B(MCHUC);          OUTPUT_LOW(PIN_D1);
        DELAY_MS(1);              OUTPUT_HIGH(PIN_D1);

        OUTPUT_B(MTRAM);          OUTPUT_LOW(PIN_D2);
        DELAY_MS(1);              OUTPUT_HIGH(PIN_D2);
  }
VOID MAIN()
{       SET_TRIS_B(0x00);               //PORTB OUT
        SET_TRIS_D(0x00);               // PORTD OUT
        SET_TRIS_C(0x00);
        SETUP_COUNTERS(RTCC_EXT_L_TO_H,WDT_18MS);// KHOITAOTIMER
        SET_TIMER0(0);                                  // GIA TRI BAT DAU

        WHILE(1)
        {       DEM=GET_TIMER0();
                DIV_DECODE(DEM);
                FOR (I=0;I<20;I++)
                        {       DISPLAY();
                        }
        }
}
```

BÀI 5-18: TIMER1 ĐẾM XUNG NGOẠI – COUNTER TỪ 00000 ĐẾN 65535 VÀ XÓA SỐ 0 VÔ NGHĨA

```
#INCLUDE <16F877A.H>
```

```
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
CONST UNSIGNED CHAR
MALED7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80, 0x90};

INT16    DEM;
INT                DONVI,CHUC, TRAM, NGAN,CNGAN,I;
UNSIGNED  CHAR     MDONVI,MCHUC,MTRAM,MNGAN,MCNGAN;

VOID  DIV_DECODE (INT16 TAM)
{  CNGAN = TAM / 10000;          //LAY CHUC NGAN
   TAM      = TAM % 10000;
   NGAN   = TAM / 1000;          //LAY NGAN
   TAM     = TAM % 1000;
   TRAM   = TAM / 100;           //LAY HANG TRAM
  TAM   = TAM %100;
  CHUC  = TAM/10;                //LAY HANG CHUC
  DONVI= TAM %10;                //LAY DONVI

MCNGAN = MALED7DOAN[CNGAN];
MNGAN = MALED7DOAN[NGAN];
MTRAM = MALED7DOAN[TRAM];
MCHUC = MALED7DOAN[CHUC];
MDONVI= MALED7DOAN[DONVI];

IF (CNGAN == 0)
   {MCNGAN=0XFF;
          IF (NGAN == 0)
            { MNGAN=0XFF;
              IF (TRAM == 0)
                {    MTRAM=0XFF;
                   IF (CHUC == 0 )
                     { MCHUC=0XFF;
                     }
                }
            }
   }
}
VOID DISPLAY ()
  {        OUTPUT_B(MDONVI);      OUTPUT_LOW(PIN_D0);
           DELAY_MS(1);           OUTPUT_HIGH(PIN_D0);

           OUTPUT_B(MCHUC);       OUTPUT_LOW(PIN_D1);
           DELAY_MS(1);           OUTPUT_HIGH(PIN_D1);

           OUTPUT_B(MTRAM);       OUTPUT_LOW(PIN_D2);
           DELAY_MS(1);           OUTPUT_HIGH(PIN_D2);

           OUTPUT_B(MNGAN);       OUTPUT_LOW(PIN_D3);
           DELAY_MS(1);           OUTPUT_HIGH(PIN_D3);

           OUTPUT_B(MCNGAN);      OUTPUT_LOW(PIN_D4);
           DELAY_MS(1);           OUTPUT_HIGH(PIN_D4);
  }
VOID MAIN()
{          SET_TRIS_B(0x00);                    //PORTB OUT
```

```
        SET_TRIS_D(0x00);                       //PORTD OUT
        SETUP_TIMER_1(T1_EXTERNAL | T1_DIV_BY_1);     // KHOITAOTIMER
        SET_TIMER1(0);                                // GIA TRI BAT DAU

        WHILE(1)
        {       DEM=GET_TIMER1();
                DIV_DECODE(DEM);
                DISPLAY();

        }
}
```

3. ỨNG DỤNG ĐẾM XUNG NỘI - ĐỊNH THỜI

BÀI 5-19 – ĐIỀU KHIỂN LED CHÓP TẮT THEO THỜI GIAN DÙNG TIMER1 ĐẾM XUNG NỘI

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=16M)
INT16   DEM;

VOID MAIN()
{    SET_TRIS_A(0x00);          //PORTB OUT
     SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);   // KHOITAOTIMER
     SET_TIMER1(0);             // GIA TRI BAT DAU

     WHILE(1)
     {   DEM=GET_TIMER1();
       IF (DEM <35000)          OUTPUT_A(0X00);
       ELSE                     OUTPUT_A(0XFF);
     }
}
```

BÀI 5-20: ĐIỀU KHIỂN LED CHÓP TẮT THEO THỜI GIAN DÙNG TIMER1 ĐẾM XUNG NỘI DÙNG PHƯƠNG PHÁP KIỂM TRA CỜ TRÀN TIMER1

```
#INCLUDE <16F877A.H>
#include <DEF_16F877A.h>

#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=16M)
INT16   DEM;

VOID MAIN()
{    SET_TRIS_A(0x00);          //PORTB OUT
     DEM=0X0;
     SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);   // KHOITAOTIMER
     SET_TIMER1(0);             // GIA TRI BAT DAU
```

```
    WHILE(1)
    {   IF (tmr1if==1)
        {
          OUTPUT_A(DEM); DEM=0XFF-DEM;
         tmr1if=0;
         }
    }
}
```

BÀI 5-21: ĐIỀU KHIỂN LED CHỚP TẮT THEO THỜI GIAN DÙNG TIMER1 ĐẾM XUNG NỘI SỬ DỤNG NGẮT TIMER1

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
INT16   DEM;

#int_timer1
void interrupt_timer1()
{
        OUTPUT_TOGGLE(PIN_A0);
}

VOID MAIN()
{       SET_TRIS_A(0x00);
        SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);   // KHOITAOTIMER
        SET_TIMER1(0);              // GIA TRI BAT DAU
        enable_interrupts(GLOBAL);
        enable_interrupts(int_timer1);
    WHILE(1)
    {
    }
}
```

BÀI 5-22 – ĐIỀU KHIỂN LED CHỚP TẮT THEO THỜI GIAN DÙNG TIMER1 ĐẾM XUNG NỘI DÙNG NGẮT ĐỊNH THỜI 100MS

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=20000000)
INT16   DEM;

#int_timer1
void interrupt_timer1()
{
        OUTPUT_TOGGLE(PIN_A0); SET_TIMER1(15536);
}
VOID MAIN()
{   SET_TRIS_A(0x00);
    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);   // KHOITAOTIMER
```

```
        SET_TIMER1(15536);              // GIA TRI BAT DAU
        enable_interrupts(GLOBAL);
        enable_interrupts(int_timer1);
      WHILE(1)
      {
      }
   }
```

BÀI 5-23: ĐIỀU KHIỂN LED CHỚP TẮT THEO THỜI GIAN DÙNG TIMER1 ĐẾM XUNG NỘI DÙNG NGẮT ĐỊNH THỜI 1S

```
#INCLUDE <16F877A.H>
#FUSES NOWDT,PUT,HS,NOPROTECT,NOLVP
#USE DELAY(CLOCK=16M)
INT   DEM;

#int_timer1
void interrupt_timer1()
{  DEM++;
          SET_TIMER1(15536);
  IF (DEM==10)
    {  DEM=0;
      OUTPUT_TOGGLE(PIN_A0);}
}

VOID MAIN()
{       SET_TRIS_A(0x00);
       SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);   // KHOITAOTIMER
       SET_TIMER1(15536);              // GIA TRI BAT DAU
       enable_interrupts(GLOBAL);
       enable_interrupts(int_timer1);
       DEM=0;
    WHILE(1)
    {
    }
}
```

V. CÁC TOÁN TỬ

+	Addition Operator
+=	Addition assignment operator, x+=y, is the same as x=x+y
&=	Bitwise and assignment operator, x&=y, is the same as x=x&y
&	Address operator
&	Bitwise and operator
^=	Bitwise exclusive or assignment operator, x^=y, is the same as x=x^y
^	Bitwise exclusive or operator
l=	Bitwise inclusive or assignment

	operator, xl=y, is the same as x=xly
l	Bitwise inclusive or operator
?:	Conditional Expression operator
- -	Decrement
/=	Division assignment operator, x/=y, is the same as x=x/y
/	Division operator
==	Equality
>	Greater than operator
>=	Greater than or equal to operator
++	Increment
*	Indirection operator
!=	Inequality
<<=	Left shift assignment operator, x<<=y, is the same as x=x<<y
<	Less than operator
<<	Left Shift operator
<=	Less than or equal to operator
&&	Logical AND operator
!	Logical negation operator
ll	Logical OR operator
%=	Modules assignment operator x%=y, is the same as x=x%y
%	Modules operator
*=	Multiplication assignment operator, x*=y, is the same as x=x*y
*	Multiplication operator
~	One's complement operator
>>=	Right shift assignment, x>>=y, is the same as x=x>>y
>>	Right shift operator
->	Structure Pointer operation
-=	Subtraction assignment operator
-	Subtraction operator
sizeof	Determines size in bytes of operand

VI. FIL DEVICE

```
//////// Standard Header file for the PIC16F877A device ////////////////
#device PIC16F877A
#nolist
//////// Program memory: 8192x14  Data RAM: 367  Stack: 8
//////// I/O: 33   Analog Pins: 8
//////// Data EEPROM: 256
//////// C Scratch area: 77   ID Location: 2000
//////// Fuses: LP,XT,HS,RC,NOWDT,WDT,NOPUT,PUT,PROTECT,DEBUG,NODEBUG
//////// Fuses: NOPROTECT,NOBROWNOUT,BROWNOUT,LVP,NOLVP,CPD,NOCPD,WRT_50%
//////// Fuses: NOWRT,WRT_5%,WRT_25%
////////
////////////////////////////////////////////////////////////////// I/O
// Discrete I/O Functions: SET_TRIS_x(), OUTPUT_x(), INPUT_x(),
```

```
//                 PORT_x_PULLUPS(), INPUT(),
//                 OUTPUT_LOW(), OUTPUT_HIGH(),
//                 OUTPUT_FLOAT(), OUTPUT_BIT()
// Constants used to identify pins in the above are:

#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_A5  45

#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55

#define PIN_C0  56
#define PIN_C1  57
#define PIN_C2  58
#define PIN_C3  59
#define PIN_C4  60
#define PIN_C5  61
#define PIN_C6  62
#define PIN_C7  63

#define PIN_D0  64
#define PIN_D1  65
#define PIN_D2  66
#define PIN_D3  67
#define PIN_D4  68
#define PIN_D5  69
#define PIN_D6  70
#define PIN_D7  71

#define PIN_E0  72
#define PIN_E1  73
#define PIN_E2  74

//////////////////////////////////////////////////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define fgetc getch
#define getchar getch
#define putc putchar
#define fputc putchar
#define fgets gets
```

```
#define fputs puts

//////////////////////////////////////////////////////////////////// Control
// Control Functions:  RESET_CPU(), SLEEP(), RESTART_CAUSE()
// Constants returned from RESTART_CAUSE() are:
#define WDT_FROM_SLEEP  3
#define WDT_TIMEOUT     11
#define MCLR_FROM_SLEEP 19
#define MCLR_FROM_RUN   27
#define NORMAL_POWER_UP 24
#define BROWNOUT_RESTART 26


//////////////////////////////////////////////////////////////////// Timer 0
// Timer 0 (AKA RTCC)Functions: SETUP_COUNTERS() or SETUP_TIMER_0(),
//                         SET_TIMER0() or SET_RTCC(),
//                         GET_TIMER0() or GET_RTCC()
// Constants used for SETUP_TIMER_0() are:
#define RTCC_INTERNAL   0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48

#define RTCC_DIV_1      8
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7


#define RTCC_8_BIT      0

// Constants used for SETUP_COUNTERS() are the above
// constants for the 1st param and the following for
// the 2nd param:

//////////////////////////////////////////////////////////////////// WDT
// Watch Dog Timer Functions: SETUP_WDT() or SETUP_COUNTERS() (see above)
//                         RESTART_WDT()
//
#define WDT_18MS        8
#define WDT_36MS        9
#define WDT_72MS       10
#define WDT_144MS      11
#define WDT_288MS      12
#define WDT_576MS      13
#define WDT_1152MS     14
#define WDT_2304MS     15

//////////////////////////////////////////////////////////////////// Timer 1
// Timer 1 Functions: SETUP_TIMER_1, GET_TIMER1, SET_TIMER1
// Constants used for SETUP_TIMER_1() are:
//      (or (via |) together constants from each group)
```

```
#define T1_DISABLED        0
#define T1_INTERNAL        0x85
#define T1_EXTERNAL        0x87
#define T1_EXTERNAL_SYNC   0x83

#define T1_CLK_OUT         8

#define T1_DIV_BY_1        0
#define T1_DIV_BY_2        0x10
#define T1_DIV_BY_4        0x20
#define T1_DIV_BY_8        0x30

//////////////////////////////////////////////////////////////////// Timer 2
// Timer 2 Functions: SETUP_TIMER_2, GET_TIMER2, SET_TIMER2
// Constants used for SETUP_TIMER_2() are:
#define T2_DISABLED        0
#define T2_DIV_BY_1        4
#define T2_DIV_BY_4        5
#define T2_DIV_BY_16       6

//////////////////////////////////////////////////////////////////// CCP
// CCP Functions: SETUP_CCPx, SET_PWMx_DUTY
// CCP Variables: CCP_x, CCP_x_LOW, CCP_x_HIGH
// Constants used for SETUP_CCPx() are:
#define CCP_OFF                    0
#define CCP_CAPTURE_FE             4
#define CCP_CAPTURE_RE             5
#define CCP_CAPTURE_DIV_4          6
#define CCP_CAPTURE_DIV_16         7
#define CCP_COMPARE_SET_ON_MATCH   8
#define CCP_COMPARE_CLR_ON_MATCH   9
#define CCP_COMPARE_INT            0xA
#define CCP_COMPARE_RESET_TIMER    0xB
#define CCP_PWM                    0xC
#define CCP_PWM_PLUS_1             0x1c
#define CCP_PWM_PLUS_2             0x2c
#define CCP_PWM_PLUS_3             0x3c
long CCP_1;
#byte   CCP_1    =             0x15
#byte   CCP_1_LOW=             0x15
#byte   CCP_1_HIGH=            0x16
long CCP_2;
#byte   CCP_2    =             0x1B
#byte   CCP_2_LOW=             0x1B
#byte   CCP_2_HIGH=            0x1C
//////////////////////////////////////////////////////////////////// PSP
// PSP Functions: SETUP_PSP, PSP_INPUT_FULL(), PSP_OUTPUT_FULL(),
//                PSP_OVERFLOW(), INPUT_D(), OUTPUT_D()
// PSP Variables: PSP_DATA
// Constants used in SETUP_PSP() are:
#define PSP_ENABLED                0x10
#define PSP_DISABLED               0

#byte   PSP_DATA=              8

//////////////////////////////////////////////////////////////////// SPI
```

```
// SPI Functions: SETUP_SPI, SPI_WRITE, SPI_READ, SPI_DATA_IN
// Constants used in SETUP_SPI() are:
#define SPI_MASTER      0x20
#define SPI_SLAVE       0x24
#define SPI_L_TO_H      0
#define SPI_H_TO_L      0x10
#define SPI_CLK_DIV_4   0
#define SPI_CLK_DIV_16  1
#define SPI_CLK_DIV_64  2
#define SPI_CLK_T2      3
#define SPI_SS_DISABLED 1

#define SPI_SAMPLE_AT_END 0x8000
#define SPI_XMIT_L_TO_H 0x4000

//////////////////////////////////////////////////////////////////// UART
// Constants used in setup_uart() are:
// FALSE - Turn UART off
// TRUE  - Turn UART on
#define UART_ADDRESS          2
#define UART_DATA             4
//////////////////////////////////////////////////////////////////// COMP
// Comparator Variables: C1OUT, C2OUT
// Constants used in setup_comparator() are:
#define A0_A3_A1_A3 0xfff04
#define A0_A3_A1_A2_OUT_ON_A4_A5 0xfcf03
#define A0_A3_A1_A3_OUT_ON_A4_A5 0xbcf05
#define NC_NC_NC_NC 0x0ff07
#define A0_A3_A1_A2 0xfff02
#define A0_A3_NC_NC_OUT_ON_A4 0x9ef01
#define A0_VR_A1_VR 0x3ff06
#define A3_VR_A2_VR 0xcff0e

#bit C1OUT = 0x9c.6
#bit C2OUT = 0x9c.7

//////////////////////////////////////////////////////////////////// VREF
// Constants used in setup_vref() are:
//
#define VREF_LOW 0xa0
#define VREF_HIGH 0x80
// Or (with |) the above with a number 0-15
#define VREF_A2  0x40



//////////////////////////////////////////////////////////////////// ADC
// ADC Functions: SETUP_ADC(), SETUP_ADC_PORTS() (aka SETUP_PORT_A),
//                SET_ADC_CHANNEL(), READ_ADC()
// Constants used for SETUP_ADC() are:
#define ADC_OFF               0            // ADC Off
#define ADC_CLOCK_DIV_2   0x10000
#define ADC_CLOCK_DIV_4    0x4000
#define ADC_CLOCK_DIV_8    0x0040
#define ADC_CLOCK_DIV_16   0x4040
#define ADC_CLOCK_DIV_32   0x0080
```

```
#define ADC_CLOCK_DIV_64   0x4080
#define ADC_CLOCK_INTERNAL 0x00c0          // Internal 2-6us

// Constants used in SETUP_ADC_PORTS() are:
#define NO_ANALOGS                    7    // None
#define ALL_ANALOG                    0    // A0 A1 A2 A3 A5 E0 E1 E2
#define AN0_AN1_AN2_AN4_AN5_AN6_AN7_VSS_VREF 1    // A0 A1 A2 A5 E0 E1 E2 VRefh=A3
#define AN0_AN1_AN2_AN3_AN4           2    // A0 A1 A2 A3 A5
#define AN0_AN1_AN2_AN4_VSS_VREF      3    // A0 A1 A2 A5 VRefh=A3
#define AN0_AN1_AN3                   4    // A0 A1 A3
#define AN0_AN1_VSS_VREF              5    // A0 A1 VRefh=A3
#define AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF 0x08    // A0 A1 A5 E0 E1 E2 VRefh=A3
VRefl=A2
#define AN0_AN1_AN2_AN3_AN4_AN5       0x09    // A0 A1 A2 A3 A5 E0
#define AN0_AN1_AN2_AN4_AN5_VSS_VREF  0x0A    // A0 A1 A2 A5 E0 VRefh=A3
#define AN0_AN1_AN4_AN5_VREF_VREF     0x0B    // A0 A1 A5 E0 VRefh=A3 VRefl=A2
#define AN0_AN1_AN4_VREF_VREF         0x0C    // A0 A1 A5 VRefh=A3 VRefl=A2
#define AN0_AN1_VREF_VREF             0x0D    // A0 A1 VRefh=A3 VRefl=A2
#define AN0                           0x0E    // A0
#define AN0_VREF_VREF                 0x0F    // A0 VRefh=A3 VRefl=A2
#define ANALOG_RA3_REF        0x1       //!old only provided for compatibility
#define A_ANALOG              0x2       //!old only provided for compatibility
#define A_ANALOG_RA3_REF      0x3       //!old only provided for compatibility
#define RA0_RA1_RA3_ANALOG    0x4       //!old only provided for compatibility
#define RA0_RA1_ANALOG_RA3_REF 0x5      //!old only provided for compatibility
#define ANALOG_RA3_RA2_REF            0x8   //!old only provided for compatibility
#define ANALOG_NOT_RE1_RE2            0x9   //!old only provided for compatibility
#define ANALOG_NOT_RE1_RE2_REF_RA3    0xA   //!old only provided for compatibility
#define ANALOG_NOT_RE1_RE2_REF_RA3_RA2  0xB   //!old only provided for compatibility
#define A_ANALOG_RA3_RA2_REF          0xC   //!old only provided for compatibility
#define RA0_RA1_ANALOG_RA3_RA2_REF    0xD   //!old only provided for compatibility
#define RA0_ANALOG                    0xE   //!old only provided for compatibility
#define RA0_ANALOG_RA3_RA2_REF        0xF   //!old only provided for compatibility


// Constants used in READ_ADC() are:
#define ADC_START_AND_READ    7   // This is the default if nothing is specified
#define ADC_START_ONLY        1
#define ADC_READ_ONLY         6



////////////////////////////////////////////////////////////////////// INT
// Interrupt Functions: ENABLE_INTERRUPTS(), DISABLE_INTERRUPTS(),
//                      EXT_INT_EDGE()
//
// Constants used in EXT_INT_EDGE() are:
#define L_TO_H            0x40
#define H_TO_L              0
// Constants used in ENABLE/DISABLE_INTERRUPTS() are:
#define GLOBAL              0x0BC0
#define INT_RTCC            0x0B20
#define INT_RB             0xFF0B08
#define INT_EXT             0x0B10
#define INT_AD             0x8C40
#define INT_TBE             0x8C10
```

```
#define INT_RDA                 0x8C20
#define INT_TIMER1              0x8C01
#define INT_TIMER2              0x8C02
#define INT_CCP1                0x8C04
#define INT_CCP2                0x8D01
#define INT_SSP                 0x8C08
#define INT_PSP                 0x8C80
#define INT_BUSCOL              0x8D08
#define INT_EEPROM              0x8D10
#define INT_TIMER0              0x0B20
#define INT_COMP                0x8D40
```