

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ VI ĐIỀU KHIỂN PIC 16F877A.....	2
1. TỔNG QUAN VỀ HỌ VI ĐIỀU KHIỂN PIC.....	2
2. GIỚI THIỆU VỀ PIC16F8XX và PIC16F877A.....	4
CHƯƠNG 2: TỔ CHỨC BỘ NHỚ - CÁC THANH GHI CHÚC NĂNG.....	6
2.1. SƠ ĐỒ CHÂN VI ĐIỀU KHIỂN PIC16F877A.....	7
2.2. MỘT VÀI THÔNG SỐ VỀ VI ĐIỀU KHIỂN PIC16F877A.....	9
2.3. SƠ ĐỒ KHỐI VI ĐIỀU KHIỂN PIC16F877A.....	11
2.4. TỔ CHỨC BỘ NHỚ.....	12
2.4.1. BỘ NHỚ CHƯƠNG TRÌNH.....	12
2.4.2. BỘ NHỚ DỮ LIỆU.....	13
2.5. CÁC THANH GHI NĂNG ĐẶC BIỆT	14
2.6. STACK.....	16
CHƯƠNG 3: TẬP LỆNH - CẤU TRÚC CHƯƠNG TRÌNH.....	17
3.1. TẬP LỆNH.....	17
3.1.1. NHÓM LỆNH DI CHUYỂN	17
3.1.2. NHÓM LỆNH SỐ HỌC.....	18
3.1.3. NHÓM LỆNH LOGIC	19
3.1.4. NHÓM LỆNH RÊ NHÁNH.....	22
3.2. TẠO TRỄ BẰNG DÒNG LẶP.....	25
3.3. CẤU TRÚC CHƯƠNG TRÌNH.....	26
3.4. CÁC KHỐI GIAO TIẾP.....	31
3.4.1. GIAO TIẾP VỚI LED 7 ĐOẠN.....	31
3.4.2. GIAO TIẾP VỚI BÀN PHÍM HEX.....	35
3.4.3. GIAO TIẾP VỚI LED MA TRẬN.....	37
3.4.4. GIAO TIẾP VỚI LCD.....	40
CHƯƠNG 4: CÁC KHỐI CHÚC NĂNG.....	46
4.1. BỘ ĐỊNH THỜI.....	46
4.1.1. TIMER 0.....	46
4.1.2. TIMER1.....	49
4.1.3. TIMER2.....	52
4.2. ADC	53
4.3. PMW_ ĐIỀU CHẾ ĐỘ RỘNG XUNG.....	58
CHƯƠNG 5: CỔNG NỐI TIẾP.....	67
5.1. USART.....	67
5.2. CHẾ ĐỘ LÀM VIỆC.....	68
5.2.1. TRUYỀN DỮ LIỆU BẤT ĐỒNG BỘ.....	68
5.2.2. NHẬN DỮ LIỆU BẤT ĐỒNG BỘ.....	71
CHƯƠNG 6: NGẮT – INTERRUPT.....	80
6.1. KHÁI NIỆM.....	80
6.2. NGẮT RB0.....	82
6.3. NGẮT PORTB.....	84
6.4. NGẮT TIMER.....	85
6.5. NGẮT ADC.....	86
6.6. NGẮT PORT NỐI TIẾP.....	88
* PHỤ LỤC: GIỚI THIỆU LẬP TRÌNH CCS.....	94
* PHỤ LỤC: CÁC THANH GHI CHÚC NĂNG.....	105

CHƯƠNG 1

TỔNG QUAN VỀ VI ĐIỀU KHIỂN PIC

1.1. TỔNG QUAN VỀ HỌ VI ĐIỀU KHIỂN PIC

PIC là một họ vi điều khiển RISC được sản xuất bởi công ty Microchip Technology. Dòng PIC đầu tiên là PIC1650 được phát triển bởi Microelectronics Division thuộc General_Instrument. PIC bắt nguồn từ chữ viết tắt của “Programmable Intelligent Computer” (Máy tính khả trình thông minh) là một sản phẩm của hãng General Instruments đặt cho dòng sản phẩm đầu tiên của họ là PIC1650. Lúc này, PIC 1650 được dùng để giao tiếp với các thiết bị ngoại vi cho máy chủ 16 bit CP1600, vì vậy, người ta cũng gọi PIC với tên “Peripheral Interface Controller” (Bộ điều khiển giao tiếp ngoại vi). CP1600 là một CPU tốt, nhưng lại kém về các hoạt động xuất nhập, và vì vậy **PIC 8-bit được phát triển vào khoảng năm 1975** để hỗ trợ hoạt động xuất nhập cho CP1600. PIC sử dụng microcode đơn giản đặt trong ROM, và mặc dù, cụm từ RISC chưa được sử dụng thời bấy giờ, nhưng PIC thực sự là một vi điều khiển với kiến trúc RISC, chạy một lệnh một chu kỳ máy (4 chu kỳ của bộ dao động). Năm 1985 General Instruments bán bộ phận vi điện tử của họ, và chủ sở hữu mới hủy bỏ hầu hết các dự án – lúc đó quá lỗi thời. Tuy nhiên, PIC được bổ sung EPROM để tạo thành 1 bộ điều khiển vào ra khả trình. Ngày nay rất nhiều dòng PIC được xuất xưởng với hàng loạt các module ngoại vi tích hợp sẵn (như USART, PWM, ADC...), với bộ nhớ chương trình từ 512 Word đến 32K Word.

1.1.1 Một số đặc tính của Vi điều khiển PIC

Hiện nay có khá nhiều dòng PIC và có rất nhiều khác biệt về phần cứng, nhưng chúng ta có thể điểm qua một vài nét như sau :

- 8/16 bit CPU, xây dựng theo kiến trúc Harvard có sửa đổi
- Flash và ROM có thể tùy chọn từ 256 byte đến 256 Kbyte
- Các cổng Xuất/ Nhập (I/ O) (mức logic thường từ 0V đến 5.5V, ứng với logic 0 và logic 1)
- 8/16 bit Timer
- Các chuẩn giao tiếp nối tiếp đồng bộ/ khung đồng bộ USART
- Bộ chuyển đổi ADC Analog-to-digital converters, 10/12 bit
- Bộ so sánh điện áp (Voltage Comparator)
- Các module Capture/ Compare/ PWM
- LCD
- MSSP Peripheral dùng cho các giao tiếp I2C, SPI.
- Bộ nhớ nội EPROM – có thể ghi/ xóa lớn tới 1 triệu lần
- Module Điều khiển động cơ, đọc encoder
- Hỗ trợ giao tiếp USB

- Hỗ trợ giao tiếp CAN
- Hỗ trợ giao tiếp LIN
- Hỗ trợ giao tiếp IrDA
- Một số dòng có tích hợp bộ RF (PIC16f639, và RFPIC)
- KEELOQ mờ hoá và giải mờ
- DSP những tính năng xử lý tín hiệu số (dsPIC) Đặc điểm thực thi tốc độ cao của RISC CPU của họ vi điều khiển PIC16F87XA :
- Chỉ gồm 35 lệnh đơn.
- Tất cả các lệnh là 1 chu kỳ ngoại trừ chương trình con là 2 chu kỳ.
- Tốc độ hoạt động :
 - + DC- 20MHz ngõ vào xung clock.
 - + DC- 200ns chu kỳ lệnh.
- Độ rộng của bộ nhớ chương trình Flash là 8K x 14word, của bộ nhớ dữ liệu (RAM) là 368 x 8bytes, của bộ nhớ dữ liệu là EPROM là 256 x 8bytes.

1.1.2. Những đặc tính ngoại vi

- Timer0 : 8- bit định thời/ đếm với 8- bit prescaler
- Timer1 : 16- bit định thời/ đếm với prescaler, có thể được tăng lên trong suốt chế độ Sleep qua thạch anh/ xung clock bên ngoài.
- Timer2 : 8- bit định thời/đếm với 8- bit, prescaler và postscaler
- Hai module Capture, Compare, PWM
 - * Capture có độ rộng 16 bit, độ phân giải 12.5ns
 - * Compare có độ rộng 16 bit, độ phân giải 200ns
 - * Độ phân giải lớn nhất của PWM là 10bit.
- Có 13 ngõ I/O có thể điều khiển trực tiếp
- Dòng vào và dòng ra lớn :
 - * 25mA dòng vào cho mỗi chân
 - * 20mA dòng ra cho mỗi chân

1.1.3. Đặc điểm về tương tự

- 10 bit, với 8 kênh của bộ chuyển đổi tương tự sang số (A/D).
- Brown – out Reset (BOR).
- Module so sánh về tương tự.
 - * Hai bộ so sánh tương tự.
 - * Module điện áp chuẩn VREF có thể lập trình trên PIC.
- Có thể lập trình ngõ ra vào đến từ những ngõ vào của PIC và trên điện áp bên trong.
- Những ngõ ra của bộ so sánh có thể sử dụng cho bên ngoài.

1.1.4. Các đặc điểm đặc biệt :

- Có thể ghi/ xoá 100.000 lần với kiểu bộ nhớ chương trình Enhanced Flash.

- 1.000.000 ghi/ xoá với kiểu bộ nhớ EPROM.
- EPROM có thể lưu trữ dữ liệu hơn 40 năm.
- Có thể tự lập trình lại dưới sự điều khiển của phần mềm.
- Mạch lập trình nối tiếp qua 2 chân.
- Nguồn đơn 5V cấp cho mạch lập trình nối tiếp.
- Watchdog Timer (WDT) với bộ dao động RC tích hợp sẵn trên Chip cho hoạt động đáng tin cậy.
- Có thể lập trình mờ bảo vệ.
- Tiết kiệm năng lượng với chế độ Sleep.
- Có thể lựa chọn bộ dao động.
- Mạch gỡ sai (ICD : In- Circuit Debug) qua 2 chân

1.1.5. Công nghệ CMOS

- Năng lượng thấp, tốc độ cao Flash/ công nghệ EPROM
- Việc thiết kế hoàn toàn tĩnh
- Khoảng điện áp hoạt động từ 2V đến 5.5V
- Tiêu tốn năng lượng thấp.

1.2. GIỚI THIỆU VỀ PIC16F8XX và PIC16F877A

PIC16F8X là nhóm PIC trong họ PIC16XX của họ Vi điều khiển 8-bit, tiêu hao năng lượng thấp, đáp ứng nhanh, chế tạo theo công nghệ CMOS, chống tĩnh điện tuyệt đối. Nhóm bao gồm các thiết bị sau:

- PIC16F83
- PIC16CR83
- PIC16F84
- PIC16CR84

- Tất cả các PIC16/17 đều có cấu trúc RISC. PIC16CXX các đặc tính nổi bật, 8 mức ngăn xếp Stack, nhiều nguồn ngắt tích hợp bên trong lẫn ngoài. Có cấu trúc Harvard với các bus dữ liệu và bus thực thi chương trình riêng biệt nhau cho phép độ dài 1 lệnh là 14-bit và bus dữ liệu 8-bit cách biệt nhau. Tất cả các lệnh đều mất 1 chu kỳ lệnh ngoại trừ các lệnh rẽ nhánh chương trình mất 2 chu kỳ lệnh. Chỉ có 35 lệnh và 1 lượng lớn các thanh ghi cho phép đáp ứng cao trong ứng dụng.

- Họ PIC16F8X có nhiều tính năng đặc biệt làm giảm thiểu các thiết bị ngoại vi, vì vậy kinh tế cao, có hệ thống nổi bật đáng tin cậy và sự tiêu thụ năng lượng thấp. Ở đây có 4 sự lựa chọn bộ dao động và chỉ có 1 chân kết nối bộ dao động RC nên có giải pháp tiết kiệm cao. Chế độ SLEEP tiết kiệm nguồn và có thể được đánh thức bởi các nguồn reset. Và còn nhiều phần khác đó được giới thiệu bên trên sẽ được nói rõ ở các phần kế tiếp.

- PIC16F877A có 40/44 chân với sự phân chia cấu trúc như sau :

- + Có 5 port xuất/nhập

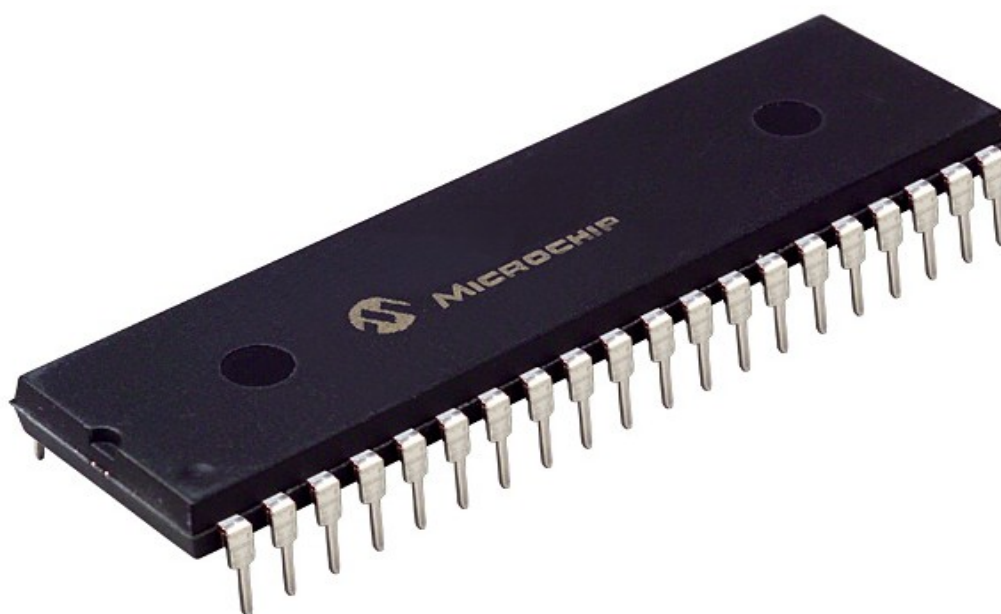
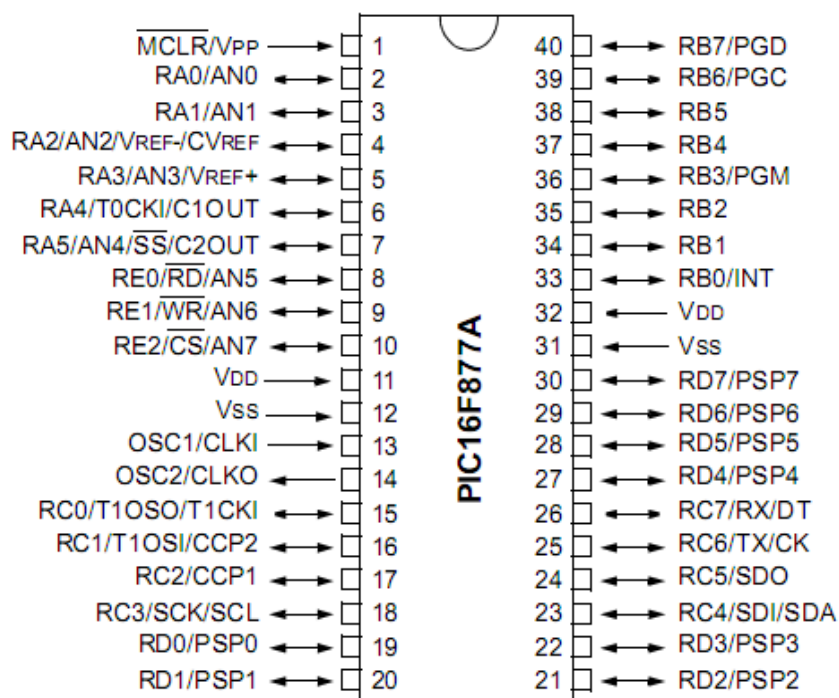
- + Có 8 kênh chuyển đổi A/D 10-bit
- + Có 2 bộ PWM
- + Có 3 bộ định thời: Timer0, timer1 và timer2
- + Có giao tiếp truyền nối tiếp: chuẩn RS 232, I2C...
- + Có giao tiếp LCD

CHƯƠNG 2

TỔ CHỨC BỘ NHỚ - CÁC THANH GHI CHỨC NĂNG

2.1 SƠ ĐỒ CHÂN VI ĐIỀU KHIỂN PIC16F877A

40-Pin PDIP



Hình 2.1: Sơ đồ chân và hình dạng của Píc 16F877A

❖ Chức năng các chân :

Chân	Tên	Chức năng
1	\overline{MCLR}/V_{PP}	- \overline{MCLR} : Hoạt động Reset ở mức thấp - V_{PP} : ngõ vào áp lập trình
2	RA0/AN0	- RA0 : xuất/nhập số - AN0 : ngõ vào tương tự
3	RA1/AN1	- RA1 : xuất/nhập số - AN1 : ngõ vào tương tự
4	RA2/AN2/ V_{REF-}/CV_{REF}	- RA2 : xuất/nhập số - AN2 : ngõ vào tương tự - V_{REF-} : ngõ vào điện áp chuẩn (thấp) của bộ A/D
5	RA3/AN3/ V_{REF+}	- RA3 : xuất/nhập số - AN3 : ngõ vào tương tự - V_{REF+} : ngõ vào điện áp chuẩn (cao) của bộ A/D
6	RA4/TOCKI/C1OUT	- RA4 : xuất/nhập số - TOCKI : ngõ vào xung clock bên ngoài cho timer0 - C1 OUT : Ngõ ra bộ so sánh 1
7	RA5/AN4/ \overline{SS} /C2OUT	- RA5 : xuất/nhập số - AN4 : ngõ vào tương tự 4 - SS : ngõ vào chọn lựa SPI phụ - C2 OUT : ngõ ra bộ so sánh 2
8	RE0/ \overline{RD} /AN5	- RE0 : xuất nhập số - RD : điều khiển việc đọc ở port nhánh song song - AN5 : ngõ vào tương tự
9	RE1/ \overline{WR} /AN6	- RE1 : xuất/nhập số - WR : điều khiển việc ghi ở port nhánh song song - AN6 : ngõ vào tương tự
10	RE2/ \overline{CS} /AN7	- RE2 : xuất/nhập số - CS : Chip lựa chọn sự điều khiển ở port nhánh song song - AN7 : ngõ vào tương tự
11	V_{DD}	Chân nguồn của PIC.
12	V_{SS}	Chân nối đất
13	OSC1/CLKI	Ngõ vào dao động thạch anh hoặc xung clock bên ngoài. - OSC1 : ngõ vào dao động thạch anh hoặc xung clock bên ngoài. Ngõ vào Schmit trigger khi được cấu tạo ở chế độ RC ; một cách khác của CMOS. - CLKI : ngõ vào nguồn xung bên ngoài. Luôn được kết hợp với chức năng OSC1.
14	OSC2/CLKO	Ngõ vào dao động thạch anh hoặc xung clock - OSC2 : Ngõ ra dao động thạch anh. Kết nối đến thạch anh hoặc bộ cộng hưởng. - CLKO : ở chế độ RC, ngõ ra của OSC2, bằng tần số

		của OSC1 và chỉ ra tốc độ của chu kỳ lệnh.
15	RC0/T1 OCO/T1CKI	- RC0 : xuất/nhập số - T1OCO : ngõ vào bộ dao động Timer 1 - T1CKI : ngõ vào xung clock bên ngoài Timer 1
16	RC1/T1OSI/CCP2	- RC1 : xuất/nhập số - T1OSI : ngõ vào bộ dao động Timer 1 - CCP2 : ngõ vào Capture 2, ngõ ra compare 2, ngõ ra PWM2
17	RC2/CCP1	- RC2 : xuất/nhập số - CCP1 : ngõ vào Capture 1, ngõ ra compare 1, ngõ ra PWM1
18	RC3/SCK/SCL	- RC3 : xuất/nhập số - SCK : ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ SPI - SCL : ngõ vào xung clock nối tiếp đồng bộ/ ngõ ra của chế độ I2C
19	RD0/PSP0	- RD0 : xuất/nhập số - PSP0 : dữ liệu port nhánh song song
20	RD1/PSP1	- RD1 : xuất/nhập số - PSP1 : dữ liệu port nhánh song song
21	RD2/PSP2	- RD2 : xuất/nhập số - PSP2 : dữ liệu port nhánh song song
22	RD3/PSP3	- RD3: xuất/nhập số - PSP3 : dữ liệu port nhánh song song
23	RC4/SDI/SDA	- RC4 : xuất/nhập số - SDI : dữ liệu vào SPI - SDA : xuất/nhập dữ liệu vào I2C
24	RC5/SDO	- RC5 : xuất/nhập số - SDO : dữ liệu ra SPI
25	RC6/TX/CK	- RC6 : xuất/nhập số - TX : truyền bất đồng bộ USART - CK : xung đồng bộ USART
26	RC7/RX/DT	- RC7 : xuất/nhập số - RX : nhận bất đồng bộ USART - DT : dữ liệu đồng bộ USART
27	RD4/PSP	- RD4: xuất/nhập số - PSP4 : dữ liệu port nhánh song song
28	RD5/PSP5	- RD5: xuất/nhập số - PSP5 : dữ liệu port nhánh song song
29	RD6/PSP6	- RD6: xuất/nhập số - PSP6 : dữ liệu port nhánh song song
30	RD7/PSP7	- RD7: xuất/nhập số - PSP7 : dữ liệu port nhánh song song
31	V _{SS}	Chân nối đất
32	V _{DD}	Chân nguồn của PIC.

33	RB0/INT	- RB0 : xuất/nhập số - INT : ngắt ngoài
34	RB1	xuất/nhập số
35	RB2	xuất/nhập số
36	RB3	- RB3 : xuất/nhập số - Chân cho phép lập trình điện áp thấp ICPS
37	RB4	- xuất/nhập số - Ngắt PortB
38	RB5	- xuất/nhập số - Ngắt PortB
39	RB6/PGC	- RB6 : xuất/nhập số - PGC : mạch vi sai và xung clock lập trình ICSP - Ngắt PortB
40	RB7/PGD	- RB7 : xuất/nhập số - PGD : mạch vi sai và dữ liệu lập trình ICSP - Ngắt PortB

2.2 MỘT VÀI THÔNG SỐ VỀ VI ĐIỀU KHIỂN PIC16F877A

Đây là vi điều khiển thuộc họ PIC16Fxxx với tập lệnh gồm 35 lệnh có độ dài 14 bit.

Mỗi lệnh đều được thực thi trong một chu kỳ xung clock. Tốc độ hoạt động tối đa cho phép là 20 MHz với một chu kỳ lệnh là 200ns. Bộ nhớ chương trình 8Kx14 bit, bộ nhớ dữ liệu 368x8 byte RAM và bộ nhớ dữ liệu EEPROM với dung lượng 256x8 byte. Số PORT I/O là 5 với 33pin I/O.

Các đặc tính ngoại vi bao gồm các khối chức năng sau:

Timer0: bộ đếm 8 bit với bộ chia tần số 8 bit.

Timer1: bộ đếm 16 bit với bộ chia tần số, có thể thực hiện chức năng đếm dựa vào xung clock ngoại vi ngay khi vi điều khiển hoạt động ở chế độ sleep.

Timer2: bộ đếm 8 bit với bộ chia tần số, bộ postcaler.

Hai bộ Capture/số sánh/điều chế độ rộng xung.

Các chuẩn giao tiếp nối tiếp SSP (Synchronous Serial Port), SPI và I2C.

Chuẩn giao tiếp nối tiếp USART với 9 bit địa chỉ.

Cổng giao tiếp song song PSP (Parallel Slave Port) với các chân điều khiển RD, WR, CS ở bên ngoài.

Các đặc tính Analog:

8 kênh chuyển đổi ADC 10 bit.

Hai bộ so sánh.

Bên cạnh đó là một vài đặc tính khác của vi điều khiển như:

Bộ nhớ flash với khả năng ghi xóa được 100.000 lần.

Bộ nhớ EEPROM với khả năng ghi xóa được 1.000.000 lần.

Dữ liệu bộ nhớ EEPROM, có 256 byte (có địa chỉ 00h÷FFh), có thể lưu trữ trên 40 năm.

Khả năng tự nạp chương trình với sự điều khiển của phần mềm.

Nạp được chương trình ngay trên mạch điện ICSP (In Circuit Serial Programming) thông qua 2 chân.

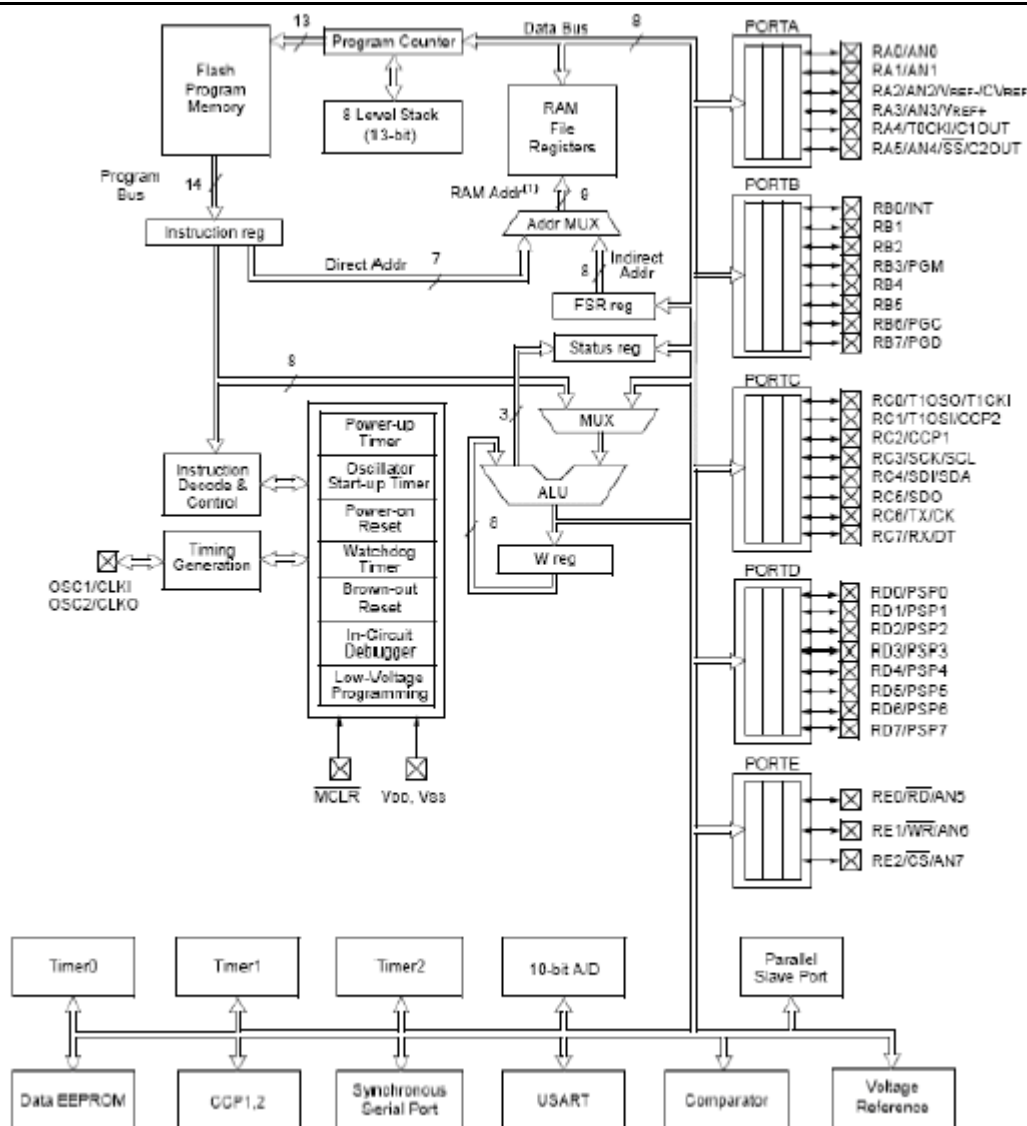
Watchdog Timer với bộ dao động trong.

Chức năng bảo mật mã chương trình.

Chế độ Sleep.

Có thể hoạt động với nhiều dạng Oscillator khác nhau.

2.3 SƠ ĐỒ KHỐI VI ĐIỀU KHIỂN PIC16F877A



Hình 2.2: Cấu trúc bên trong của Pic 16F877A

Như đã nói ở trên, vi điều khiển PIC có kiến trúc Harvard, trong đó CPU truy cập chương trình và dữ liệu được trên hai bus riêng biệt, nên làm tăng đáng kể băng thông so với kiến trúc Von Neumann trong đó CPU truy cập chương trình và dữ liệu trên cùng một bus.

Việc tách riêng bộ nhớ chương trình và bộ nhớ dữ liệu cho phép số bit của từ lệnh có thể khác với số bit của dữ liệu. Ở PIC 16F877A, từ lệnh dài 14 bit, từ dữ liệu 8 bit.

PIC 16F877A chứa một bộ ALU 8 bit và thanh ghi làm việc WR (working register).

ALU là đơn vị tính toán số học và logic, nó thực hiện các phép tính số và đại số Boole trên thanh ghi làm việc WR và các thanh ghi dữ liệu. ALU có thể thực hiện các phép cộng, trừ, dịch bit và các phép toán logic

2.4 TỔ CHỨC BỘ NHỚ

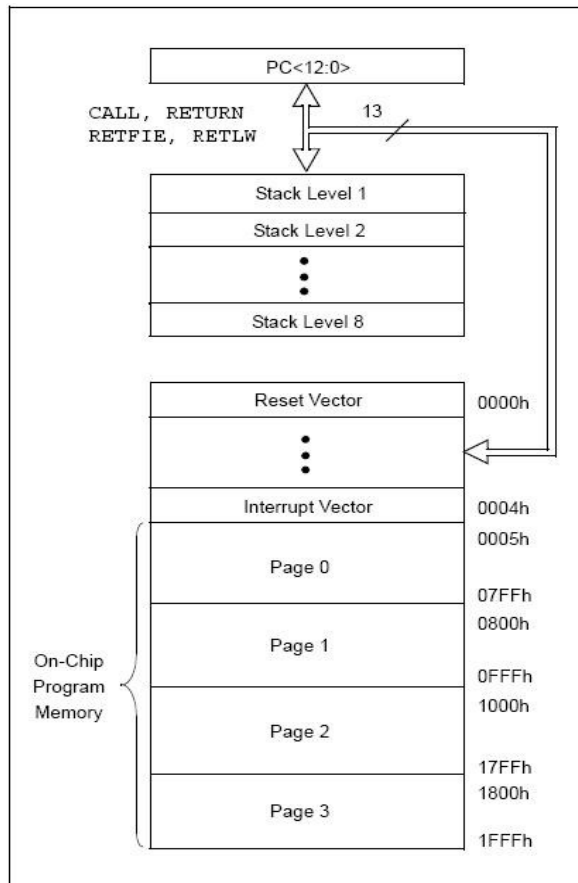
a. BỘ NHỚ CHƯƠNG TRÌNH

Bộ nhớ chương trình của vi điều khiển PIC16F877A là bộ nhớ flash, dung lượng bộ nhớ 8K word (1 word = 14 bit) và được phân thành nhiều trang (từ page0 đến page 3). Như vậy bộ nhớ chương trình có khả năng chứa được $8 \times 1024 = 8192$ lệnh (vì một lệnh sau khi mã hóa sẽ có dung lượng 1 word (14bit)).

Để mã hóa được địa chỉ của 8K word bộ nhớ chương trình, bộ đếm chương trình có dung lượng 13 bit (PC<12:0>).

Khi vi điều khiển được reset, bộ đếm chương trình sẽ chỉ đến địa chỉ 0000h (Resetvector). Khi có ngắt xảy ra, bộ đếm chương trình sẽ chỉ đến địa chỉ 0004h (Interruptvector).

Bộ nhớ chương trình không bao gồm bộ nhớ stack và không được địa chỉ hóa bởi bộ đếm chương trình. Bộ nhớ stack sẽ được đề cập cụ thể trong phần sau.



Hình 2.3: Bộ nhớ chương trình của Pic

b. BỘ NHỚ DỮ LIỆU

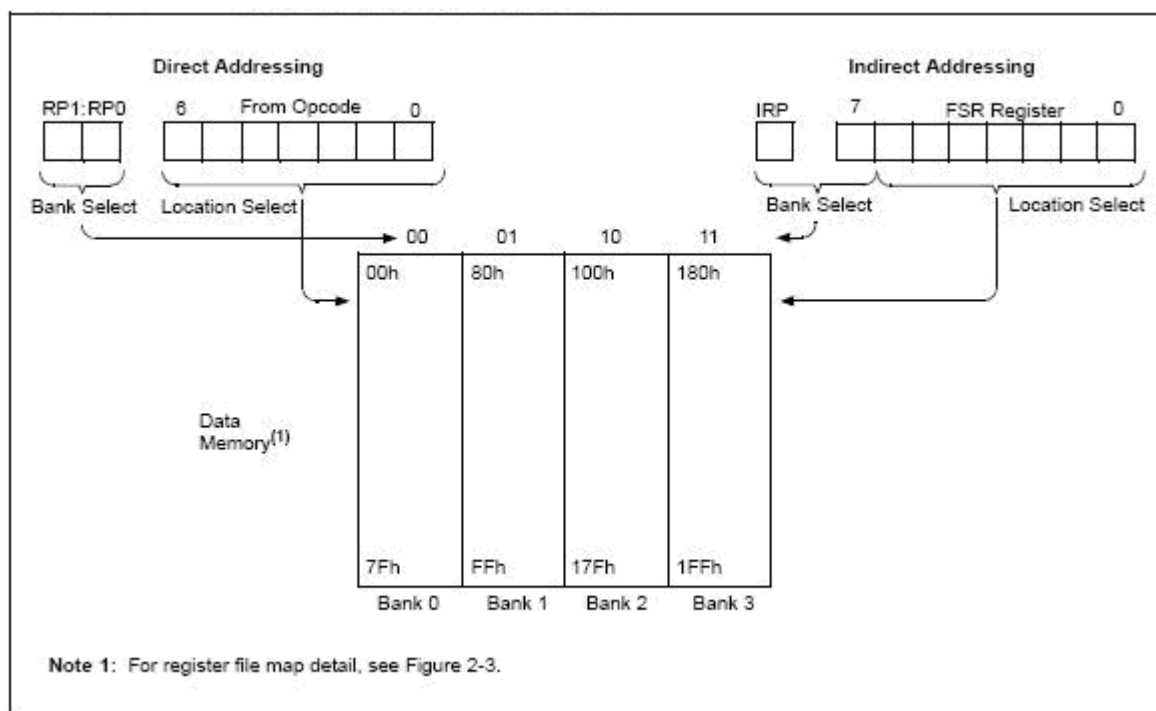
Bộ nhớ dữ liệu của PIC là bộ nhớ EEPROM được chia ra làm nhiều bank. Đối với PIC16F877A bộ nhớ dữ liệu được chia ra làm 4 bank. Mỗi bank có dung lượng 128 byte, bao gồm các thanh ghi có chức năng đặc biệt SFG (Special Function Register) nằm ở các vùng địa chỉ thấp và các thanh ghi mục đích chung GPR (General Purpose Register) nằm ở vùng địa chỉ còn lại trong bank. Các thanh ghi SFR thường xuyên được sử dụng (ví dụ như thanh ghi STATUS) sẽ được đặt ở tất cả các bank của bộ nhớ dữ liệu giúp thuận tiện trong

quá trình truy xuất và làm giảm bớt lệnh của chương trình. Sơ đồ cụ thể của bộ nhớ dữ liệu PIC16F877A như sau:

File Address		File Address		File Address		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	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADDD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		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
			F0h		170h		1F0h
		accesses 70h-7Fh		accesses 70h-7Fh		accesses 70h - 7Fh	
Bank 0	7Fh	Bank 1	FFh	Bank 2	17Fh	Bank 3	1FFh

Hình 2.4: Bộ nhớ bộ nhớ của Pic

2.5 CÁC THANH GHI ĐẶC BIỆT - THANH GHI FSR VÀ INDF



Hình 2.5: Sơ đồ thanh ghi FSR

Thanh ghi FSR chứa địa chỉ “con trỏ” chỉ đến, thanh ghi INDF chứa nội dung có địa chỉ nằm trong thanh ghi FSR.

Ví dụ: Thanh ghi 22H có giá trị là 10. Nếu FSR = 22H thì INDF = 10.

Tóm lại, Thanh ghi **INDF** không phải là một thanh ghi vật lí. Nó chứa giá trị của thanh ghi có địa chỉ nằm ở thanh ghi **FSR**.

-THANH GHI STATUS

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

Thanh ghi trạng thái chứa các trạng thái số học của bộ ALU, trạng thái Reset và các bit chọn Bank của bộ nhớ dữ liệu.

Bit 7 **IRP**: Bit lựa chọn bank thanh ghi (Sử dụng cho định địa chỉ gián tiếp).

1 = Bank 2, 3 (100h – 1FFh)

0 = Bank 0, 1 (00h – FFh)

Bit 6 – 5: **RP1 – RP0**: Bit lựa chọn bank thanh ghi (Dùng trong định địa chỉ trực tiếp).

11 = Bank 3 (180h – 1FFh)

10 = Bank 2 (100h – 17Fh)

01 = Bank 1 (80h – FFh)

00 = Bank 0 (00h – 7Fh)

Each bank is 128 bytes

Bit 4 \overline{TO} : Bit báo hiệu hoạt động của WDT.

1: Lệnh xóa WDT hoặc Sleep xảy ra.

0: WDT hoạt động.

Bit 3 $\overline{\text{PD}}$: Bit báo công suất thấp (Power down bit).

1: Sau khi nguồn tăng hoặc có lệnh xóa WDT.

0: Thực thi lệnh Sleep.

Bit 2 **Z**: bit Zero

1: Khi kết quả của một phép toán bằng 0.

0: Khi kết quả của một phép toán khác 0.

Bit 1 **DC**: Digit Carry

1: Có một số nhớ sinh ra bởi phép cộng hoặc phép trừ 4 bit thấp.

0: Không có số nhớ sinh ra.

Bit 0 **C**: cờ nhớ (Carry Flag)/ borrow

1: Có một số nhớ sinh ra bởi phép cộng hoặc phép trừ 4 bit cao.

0: Không có số nhớ sinh ra.

Ví dụ: Nếu $A - B < 0$ thì $C = 0$ ngược lại $C = 1$

- THANH GHI ĐIỀU KHIỂN NGẮT INTCON (Interrupt Control Register)

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 7 **GIE**: Bit cho phép ngắt toàn cục

1: Cho phép ngắt toàn cục

0: Không cho phép ngắt

Bit 6 **PEIE**: Bit cho phép ngắt khi ghi vào **EEPROM** hoàn tất.

1: Cho phép ngắt ghi vào **EEPROM** hoạt động

0: Không cho phép ngắt ghi vào **EEPROM** hoạt động

Bit 5 **TMR0IE**: Bit cho phép ngắt khi timer 0 tràn

1: Cho phép ngắt khi **timer 0** tràn

0: Không cho phép ngắt khi **timer 0** tràn

Bit 4 **INTE**: Bit cho phép ngắt ngoại vi trên chân **RB0/INT**

1: Cho phép ngắt ngoại vi

0: Không cho phép ngắt ngoại vi

Bit 3 **RBIE**: Cho phép ngắt khi trạng thái **PORTB** thay đổi

1: Cho phép

0: Không cho phép

Bit 2 **TMR0IF**: Cờ báo ngắt **Timer 0**

1: **Timer 0** tràn

0: **Timer 0** chưa tràn

Bit 1 **INTF**:Cờ báo ngắt ngoài **RB0/INT**

1: Có ngắt

0: Không xảy ra ngắt.

Bit 0 **RBIF**:Cờ báo ngắt khi có thay đổi trạng thái **PORTB**

1: Có thay đổi

0: Không có thay đổi xảy ra trên **PORTB**

*** Ngoài ra còn một số thanh ghi chức năng khác như:**

Thanh ghi PIE1 (địa chỉ 8Ch): chứa các bit điều khiển chi tiết các ngắt của các khối chức năng ngoại vi.

Thanh ghi PIR1 (địa chỉ 0Ch) chứa cờ ngắt của các khối chức năng ngoại vi, các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PIE1.

Thanh ghi PIE2 (8Dh): chứa các bit điều khiển các ngắt của các khối chức năng CCP2, SSP bus, ngắt của bộ so sánh và ngắt ghi vào bộ nhớ EEPROM.

Thanh ghi PIR2 (0Dh): chứa các cờ ngắt của các khối chức năng ngoại vi, các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PIE2.

Thanh ghi PCON (8Eh): chứa các cờ hiệu cho biết trạng thái các chế độ reset của vi điều khiển.

Để biết thêm chi tiết xem phần Phụ lục

2.6 STACK

Stack cho phép 8 lệnh gọi chương trình con và ngắt hoạt động. **Stack** chứa địa chỉ mà chương trình chính sẽ quay về thực hiện từ sau chương trình con hay ngắt. Đối với **PIC16F877A** Stack có độ sâu 8 lớp.

Stack không nằm trong bộ nhớ chương trình hay bộ nhớ dữ liệu mà là một vùng nhớ đặc biệt không cho phép đọc hay ghi. Khi lệnh CALL được thực hiện hay khi một ngắt xảy ra làm chương trình bị rẽ nhánh, giá trị của bộ đếm chương trình PC tự động được vi điều khiển cất vào trong stack. Khi một trong các lệnh RETURN, RETLW hat RETFIE được thực thi, giá trị PC sẽ tự động được lấy ra từ trong stack, vi điều khiển sẽ thực hiện tiếp chương trình theo đúng qui trình định trước.

Bộ nhớ Stack trong vi điều khiển PIC họ 16F87xA có khả năng chứa được 8 địa chỉ và hoạt động theo cơ chế xoay vòng. Nghĩa là giá trị cất vào bộ nhớ Stack lần thứ 9 sẽ ghi đè lên giá trị cất vào Stack lần đầu tiên và giá trị cất vào bộ nhớ Stack lần thứ 10 sẽ ghi đè lên giá trị cất vào Stack lần thứ 2.

Cần chú ý là không có cờ hiệu nào cho biết trạng thái stack, do đó ta không biết được khi nào stack tràn. Bên cạnh đó tập lệnh của vi điều khiển dòng PIC cũng không có lệnh POP hay PUSH, các thao tác với bộ nhớ stack sẽ hoàn toàn được điều khiển bởi CPU.

CHƯƠNG 3

TẬP LỆNH - CẤU TRÚC CHƯƠNG TRÌNH

3.1. TẬP LỆNH

3.1.1. NHÓM LỆNH DI CHUYỂN

1. Lệnh MOVLW

Cú pháp: MOVLW k ($0 \leq k \leq 255$)

Tác dụng: Dem giá trị k vào thanh ghi W

Ví dụ: Để gián cho thanh ghi W một giá trị cụ thể là 20H ta làm như sau:

MOVLW 20H

⇔ MOVLW B'0010 0000'

⇔ MOVLW D'32'

2. Lệnh MOVWF

Cú pháp: MOVWF f ($0 \leq f \leq 255$)

Tác dụng: Dem giá trị của thanh ghi W vào thanh ghi f

Để gián cho thanh ghi một giá trị cụ thể, đầu tiên đưa giá trị cần gián cho thanh ghi W, sau đó ta thực hiện lệnh **MOVWF** để di chuyển giá trị trong thanh ghi W sang thanh ghi cần gián.

Ví dụ:

MOVLW D'15'; W=15

MOVWF PORTB; PORTB=15

Tuy nhiên, còn có cách khác thông qua thanh ghi “con trỏ” FSR, **khi thanh ghi “con trỏ” FSR trở đến byte có địa chỉ nào thì nội dung của thanh ghi đó di chuyển vào thanh ghi INDF.**

Để hiểu một cách đơn giản ta hiểu thanh ghi FSR chứa địa chỉ còn thanh ghi INDF chứa nội dung.

Ví dụ:

MOVLW 30H

MOVWF FSR

MOVLW D'20'

MOVWF INDF

Ở lệnh đầu tiên W=30H, sau đó gián giá trị 30H vào thanh ghi FSR tức là “con trỏ” chỉ đến byte có địa chỉ 30H. **Khi đó giá trị của thanh ghi có địa chỉ 30H được chứa trong thanh ghi INDF.** Như vậy sau khi gián giá trị 20 vào thanh ghi INDF tức là gián giá trị đó vào thanh ghi có địa chỉ 30H.

Vậy sau khi thực hiện đoạn chương trình trên $(30H) = 20$, tức là byte có địa chỉ 30H có giá trị là 20.

Để cụ thể hơn chúng ta xét **ví dụ** sau:

MOVLW D'5'

MOVWF PORTB

Thông qua 2 lệnh trên PORTB = 5, nhưng ta có thể viết lại:

MOVLW 06H

MOVWF FSR

MOVLW D'5'

MOVWF INDF

Vậy sau khi thực hiện đoạn chương trình trên (06H) = 5, tức là byte có địa chỉ 06H (PORTB) có giá trị là 5.

3. Lệnh MOVF

Cú pháp: MOVF f, W f ($0 \leq f \leq 255$)

Tác dụng: Dem giá trị của thanh ghi f vào thanh ghi W

Để di chuyển giá trị ở thanh ghi COUNT1 sang thanh ghi COUNT2 thì ta bắt buộc qua thanh ghi “trung gian” W thông qua lệnh **MOVF**.

Ví dụ:

MOVF COUNT1, W

MOVWF COUNT2

Đầu tiên dem giá trị có được ở thanh ghi COUNT1 vào W, sau đó thông qua lệnh MOVWF dem giá trị có được ở thanh ghi W vào COUNT2.

3.1.2. NHÓM LỆNH SỐ HỌC

4. Lệnh ADDLW

Cú pháp: ADDLW k

Tác dụng: Cộng giá trị k vào thanh ghi W, kết quả được chứa trong thanh ghi W.

Bit trạng thái: C, DC, Z

Ví dụ:

MOVLW D'200'; W=200

ADDLW D'55'

MOVWF PORTB; PORTB = 255

5. Lệnh ADDWF

Cú pháp: ADDWF f, d

($d \in [0, 1]$).

Tác dụng: Cộng giá trị hai thanh ghi W và thanh ghi f. Kết quả được chứa trong thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: C, DC, Z

6. Lệnh SUBLW

Cú pháp: SUBLW k

Tác dụng: Lấy giá trị k trừ giá trị trong thanh ghi W. Kết quả được chứa trong thanh ghi W.

Bit trạng thái: C, DC, Z

Ví dụ:

MOVLW D'100'; W=100

SUBLW D'155'

MOVWF PORTB; PORTB=55

7. Lệnh SUBWF

Cú pháp: SUBWF f,d

($d \in [0,1]$)

Tác dụng: Lấy giá trị trong thanh ghi f đem trừ cho thanh ghi W. Kết quả được lưu trong thanh ghi W nếu $d=0$ hoặc thanh ghi f nếu $d=1$.

Bit trạng thái: C, DC, Z

8. Lệnh INCF

Cú pháp: INCF f,d

($d \in [0,1]$)

Tác dụng: Tăng giá trị thanh ghi f lên 1 đơn vị. Kết quả được đưa vào thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: Z

Ví dụ:

MOVLW D'10'

MOVWF COUNT; COUNT=10

INCF COUNT,1; COUNT=11

9. Lệnh DECF

Cú pháp: DECF f,d

($d \in [0,1]$).

Tác dụng: Giá trị thanh ghi f được giảm đi 1 đơn vị. Kết quả được đưa vào thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: Z

Ví dụ:

MOVLW D'10'

MOVWF COUNT; COUNT=10

DECF COUNT,1; COUNT=9

3.1.3. NHÓM LỆNH LOGIC

10. Lệnh BCF

Cú pháp: BCF f,b ($0 \leq b \leq 7$)

Tác dụng: Xóa bit b trong thanh ghi f về giá trị 0.

Bit trạng thái: không có.

Ví dụ:

BCF PORTB,2; RB2 =0

11. Lệnh BSF

Cú pháp: BSF f,b ($0 \leq b \leq 7$)

Tác dụng: Set bit b trong thanh ghi f.

Bit trạng thái: không có

Ví dụ:

BSF PORTB,2; RB2 =1

12. Lệnh CLRW

Cú pháp CLRW

Tác dụng: Xóa thanh ghi W và bit Z được set.

Bit trạng thái: Z

13. Lệnh CLRF

Cú pháp CLRF f

Tác dụng: Xóa thanh ghi f và bit Z được set.

Bit trạng thái: Z

14. Lệnh CLRWD

Cú pháp: CLRWD

Tác dụng: Reset Watchdog Timer, đồng thời prescaler cũng được reset, các bit và được set lên 1.

15. Lệnh ANDLW

Cú pháp: ANDLW k

Tác dụng: Thực hiện phép toán AND giữa thanh ghi và giá trị k, kết quả được chứa trong thanh ghi W.

Bit trạng thái: Z

Chú ý: And các bit tương ứng

Ví dụ:

MOVLW B'1111 0000'

ANDLW B'0011 1111'; W = B'0011 0000'

16. Lệnh ANDWF

Cú pháp: ANDWF f,d

($d \in [0,1]$).

Tác dụng: Thực hiện phép toán AND giữa các giá trị chứa trong hai thanh ghi W và f. Kết quả được đưa vào thanh ghi W nếu $d=0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: Z

17. Lệnh IORLW

Cú pháp: IORLW k

Tác dụng: Thực hiện phép toán OR giữa thanh ghi W và giá trị k. Kết quả được chứa trong thanh ghi W.

Bit trạng thái: Z

18. Lệnh IORWF

Cú pháp: IORWF f,d

($d \in [0,1]$)

Tác dụng: Thực hiện phép toán OR giữa hai thanh ghi W và f. Kết quả được đưa vào thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d=1$.

Bit trạng thái: Z

19. Lệnh XORLW

Cú pháp: XORLW k

Tác dụng: Thực hiện phép toán XOR giữa giá trị k và giá trị trong thanh ghi W. Kết quả được lưu trong thanh ghi W.

Bit trạng thái: Z

20. Lệnh XORWF

Cú pháp: XORWF f,d

Tác dụng: Thực hiện phép toán XOR giữa hai giá trị chứa trong thanh ghi W và thanh ghi f. Kết quả được lưu vào trong thanh ghi W nếu $d=0$ hoặc thanh ghi f nếu $d=1$.

Bit trạng thái: Z

21. Lệnh SWAPF

Cú pháp: SWAPF f,d

($d \in [0,1]$)

Tác dụng: Đảo 4 bit thấp với 4 bit cao trong thanh ghi f. Kết quả được chứa trong thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: không có

22. Lệnh RLF

Cú pháp: RLF f,d

($d \in [0,1]$)

Tác dụng: Dịch trái các bit trong thanh ghi f qua cờ carry. Kết quả được lưu trong thanh ghi W nếu $d=0$ hoặc thanh ghi f nếu $d=1$.

Bit trạng thái: C

23. Lệnh RRF

Cú pháp: RRF f,d

($d \in [0,1]$)

Tác dụng: Dịch phải các bit trong thanh ghi f qua cờ carry. Kết quả được lưu trong thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: C

24. Lệnh COMF

Cú pháp: COMF f,d

($d \in [0,1]$).

Tác dụng: Đảo các bit trong thanh ghi f. Kết quả được đưa vào thanh ghi W nếu $d=0$ hoặc thanh ghi f nếu $d=1$.

Bit trạng thái: Z

3.1.4.NHÓM LỆNH Rẽ NHÁNH

25. Lệnh BTFSS

Cú pháp: BTFSS f,b

($0 \leq b \leq 7$)

Tác dụng: Kiểm tra bit b trong thanh ghi f. Nếu bit b bằng 0, lệnh tiếp theo được thực thi. Nếu bit b bằng 1, lệnh tiếp theo được bỏ qua và thay vào đó là lệnh NOP.

Bit trạng thái: không có

Ví dụ:

BTFSS PORTB,1

LỆNH 1

LỆNH 2

“1” ở đây là vị trí bit được kiểm tra của portB. Nếu bit này ở mức cao thì sẽ bỏ qua lệnh 1 để thực thi lệnh 2. Ngược lại, mức thấp sẽ thực thi lệnh 1

26. Lệnh BTFSC

Cú pháp: BTFSC f,b

($0 \leq b \leq 7$)

Tác dụng: kiểm tra bit b trong thanh ghi f. Nếu bit b bằng 1, lệnh tiếp theo được thực thi. Nếu bit b bằng 0, lệnh tiếp theo được bỏ qua và thay vào đó là lệnh NOP.

Bit trạng thái: không có

27. Lệnh DECFSZ

Cú pháp: DECFSZ f,d

($d \in [0,1]$)

Tác dụng: giá trị thanh ghi f được giảm 1 đơn vị. Nếu kết quả sau khi giảm khác 0, lệnh tiếp theo được thực thi, nếu kết quả bằng 0, lệnh tiếp theo không được thực thi và thay vào đó là lệnh NOP. Kết quả được đưa vào thanh ghi W nếu $d = 0$ hoặc thanh ghi f nếu $d = 1$.

Bit trạng thái: không có

Ví dụ:

DECFSZ DEM,1

LỆNH 1

LỆNH 2

Sau khi giảm giá trị trong thanh ghi “DEM” xuống 1 đơn vị, nếu chưa bằng 0 thì thực thi “LỆNH 1”. Ngược lại, thực thi “LỆNH 2”

28. Lệnh INCFSZ

Cú pháp: INCFSZ f,d

($d \in [0,1]$)

Tác dụng: tăng giá trị thanh ghi f lên 1 đơn vị. Nếu kết quả khác 0, lệnh tiếp theo được thực thi, nếu kết quả bằng 0, lệnh tiếp theo được thay bằng lệnh NOP. Kết quả sẽ được đưa vào thanh ghi f nếu $d=1$ hoặc thanh ghi W nếu $d = 0$.

Bit trạng thái: không có.

29. Lệnh GOTO

Cú pháp: GOTO k ($0 \leq k \leq 2047$)

Tác dụng: nhảy tới một label được định nghĩa bởi tham số k và 2 bit PCLATH <4:3>.

Bit trạng thái: không có.

30. Lệnh CALL

Cú pháp: CALL k ($0 \leq k \leq 2047$)

Tác dụng: gọi một chương trình con. Trước hết địa chỉ quay trở về từ chương trình con ($PC+1$) được cất vào trong Stack, giá trị địa chỉ mới được đưa vào bộ đếm gồm 11 bit của biến k và 2 bit PCLATH<4:3>.

Bit trạng thái: không có

31. Lệnh RETURN

Cú pháp: RETURN

Tác dụng: quay trở về chương trình chính từ một chương trình con

Bit trạng thái: không có

Ngoại trừ lệnh trên còn có một số lệnh được trong chương trình nhô:

32 Lệnh #DEFINE

Cú pháp: #DEFINE <text1> <text2>

Tác dụng: thay thế một chuỗi kí tự này bằng một chuỗi kí tự khác, có nghĩa là mỗi khi chuỗi kí tự text1 xuất hiện trong chương trình, trình biên dịch sẽ tự động thay thế chuỗi kí tự đó bằng chuỗi kí tự <text2>.

33. Lệnh INCLUDE

Cú pháp: #INCLUDE <filename> hoặc #INCLUDE "filename"

Tác dụng: đính kèm một file khác vào chương trình, tương tự như việc ta copy file đó vào vị trí xuất hiện lệnh INCLUDE. Nếu dùng cú pháp <filename> thì file đính kèm là file hệ thống (système file), nếu dùng cú pháp "filename" thì file đính kèm là file của người sử dụng. Thông thường chương trình được đính kèm theo một "header file" chứa các thông tin định nghĩa các biến (thanh ghi W, thanh ghi F,...) và các địa chỉ của các thanh ghi chức năng đặc

biệt trong bộ nhớ dữ liệu. Nếu không có header file, chương trình sẽ khó đọc và khó hiểu hơn.

34 .Lệnh CONSTANT

Cú pháp: CONSTANT <name>=<value>

Tác dụng: Khai báo một hằng số, có nghĩa là khi phát hiện chuỗi kí tự "name" trong chương trình, trình biên dịch sẽ tự động thay bằng chuỗi kí tự bằng giá trị "value" đã được định nghĩa trước đó.

35. Lệnh VARIABLE

Cú pháp: VARIABLE <name>=<value>

Tác dụng: Tương tự như lệnh CONSTANT, chỉ có điểm khác biệt duy nhất là giá trị "value" khi dùng lệnh VARIABLE có thể thay đổi được trong quá trình thực thi chương trình còn lệnh CONSTANT thì không.

36. Lệnh SET

Cú pháp: <name variable> SET <value>

Tác dụng: Gán giá trị cho một tên biến. Tên của biến có thể thay đổi được trong quá trình thực thi chương trình.

37 Lệnh EQU

Cú pháp: <name constant> EQU <value>

Tác dụng: Gán giá trị cho tên của hằng số. Tên của hằng số không thay đổi trong quá trình thực thi chương trình.

38. Lệnh ORG

Cú pháp: ORG <value>

Tác dụng: Định nghĩa một địa chỉ chứa chương trình trong bộ nhớ chương trình của vi điều khiển.

39. Lệnh END

Cú pháp: END

Tác dụng: Đánh dấu kết thúc chương trình.

40. Lệnh __CONFIG

Tác dụng: Thiết lập các bit điều khiển các khối chức năng của vi điều khiển được chứa trong bộ nhớ chương trình (Configuration bit).

41. Lệnh PROCESSOR

Cú pháp: PROCESSOR <processor type>

Tác dụng: Định nghĩa vi điều khiển nào sử dụng chương trình.

3.2. TẠO TRỄ BẰNG VÒNG LẶP

Thực chất của chương trình DELAY là cho vi điều khiển làm một công việc vô nghĩa nào đó trong một khoảng thời gian định trước. Khoảng thời gian này được tính toán dựa trên quá trình thực thi lệnh, hay cụ thể hơn là dựa vào thời gian của một chu kì lệnh.

Có thể viết chương trình DELAY dựa trên đoạn chương trình sau:

```

DELAY
    MOVLW    D'5'
    MOVWL    DEM
LOOP
    DECFSZ   DEM
    GOTO     LOOP
    RETURN
    
```

Bây giờ ta tính toán xem đoạn chương trình trên tạo trễ bao lâu? (Hai lệnh đầu xem như bỏ qua, tính từ ngay nhãn “LOOP” cho đến lệnh “RETURN”)

$5 \rightarrow 4$; 3 chu kỳ máy
 $4 \rightarrow 3$; 3 chu kỳ máy
 $3 \rightarrow 2$; 3 chu kỳ máy
 $2 \rightarrow 1$; 3 chu kỳ máy
 $1 \rightarrow 0$; 4 chu kỳ máy

Đối với các lệnh trong Pic những lệnh thông thường khi thực thi tốn 1 chu kỳ máy, các lệnh “nhảy” tốn 2 chu kỳ máy. Riêng các lệnh: BTFSS, BTFSC, DECFSZ... Khi chưa “nhảy” cũng tốn 1 chu kỳ máy, khi thỏa điều kiện thì “nhảy” thì tốn 2 chu kỳ máy.

Do đó, ở vòng lặp đầu tiên lệnh DECFSZ tốn 1 chu kỳ máy, lệnh GOTO tốn 2 chu kỳ máy. Ở vòng lặp cuối, sau khi thực thi xong lệnh DECFSZ giá trị trong thanh ghi DEM giảm từ $1 \rightarrow 0$ thì nhảy qua khỏi lệnh GOTO tốn 2 chu kỳ máy nhưng gặp lệnh RETURN là lệnh “nhảy” tốn 2 chu kỳ máy. Do đó, ở vòng lặp cuối tốn 4 chu kỳ máy.

$$T_d = (3DEM + 1)T_i$$

$$\approx 3DEM.$$

Với: $T_i = 4 / f_{osc}$

DEM ≤ 255 : Giá trị cài vào để đếm

Td: Thời gian tạo trễ.

Ví dụ: Viết chương trình tạo trễ $500\mu s$, thạch anh 4Mhz

Tacó: $T_d = 500\mu s$, $T_i = 4 / f_{osc} = 1\mu s \Rightarrow DEM = 500 / 3 = 167$

```

DELAY
    MOVLW    D'167'
    MOVWL    DEM
LOOP
    DECFSZ   DEM
    
```

GOTO LOOP

RETURN

Nhận xét: Nếu dùng thạch anh 4Mhz thì Td đạt giá trị tối đa là 765 μ s. Vậy để tăng thời gian Td chúng ta dùng 2 vòng lặp lồng vào nhau:

DELAY

MOVLW D'255'

MOVWF DEM1

LOOP

DECFSZ DEM1

GOTO LOOP1

GOTO THOAT

LOOP1

MOVLW D'255'

MOVWF DEM2

LOOP2

DECFSZ DEM2

GOTO LOOP2

GOTO LOOP

THOAT

NOP

RETURN

Với đoạn chương trình trên ta tính được:

$$Td \approx 3 * DEM2 * DEM1.$$

$$= 3.255.255 = 172125 \mu s \approx 0.196S$$

3.3. CẤU TRÚC CHƯƠNG TRÌNH

; Không có sử dụng ngắt, nếu có sử dụng ngắt xem chương 6

PROCESSOR 16F877A ; Khai báo dùng VI ĐIỀU KHIỂN gì?

INCLUDE <P 16F877A.INC> ; Đính kèm file có sẵn trong thư viện.

ORG 0000H ; Địa chỉ Vector Reset

- CHỌN BANK ; Dựa vào thanh ghi Status chọn bank phù hợp

- CHỌN I/O ; Dựa vào mục đích thiết kế, chọn ngõ vào/ra
; phù hợp.

MAIN ; Bắt đầu viết chương trình

- ; Thực thi chương trình

-

```
GOTO $ ; Vòng lặp vô hạn
END ; Kết thúc chương trình
```

Ví dụ: Viết chương trình xuất ra chân RB7 mức cao.

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

```
BCF STATUS,6 ; Chọn bank0
BCF STATUS,5 ; Chọn bank0
CLRF PORTB ; Xóa PORTB
BSF STATUS,5 ; Chọn bank1
BCF TRISB,7 ; Khai báo RB7 là output
BCF STATUS,5 ; Trở lại bank0
BSF PORTB,7 ; Set RB7 mức cao
GOTO $ ; Tạo vòng lặp vô hạn
```

END

Để biết khai báo I/O như thế nào? chúng ta có thể nắm cách thức khai báo I/O cụ thể như sau:

- Thanh ghi TRISA chọn tính I/O cho PORTA
- Thanh ghi TRISB chọn tính I/O cho PORTB
- Thanh ghi TRISC chọn tính I/O cho PORTC
- Thanh ghi TRISD chọn tính I/O cho PORTD
- Thanh ghi TRISE chọn tính I/O cho PORTE

Cách chọn cũng khá đơn giản:

Muốn xác lập chức năng của một chân trong PORTA là input, ta "set" bit điều khiển tương ứng với chân đó trong thanh ghi TRISA và ngược lại, muốn xác lập chức năng của một chân trong PORTA là output, ta "clear" bit điều khiển tương ứng với chân đó trong thanh ghi TRISA. Thao tác này hoàn toàn tương tự đối với các PORT và các thanh ghi điều khiển tương ứng TRIS (đối với PORTA là TRISA, đối với PORTB là TRISB, đối với PORTC là TRISC, đối với PORTD là TRISD và đối với PORTE là TRISE).

Ví dụ:

Chúng ta muốn RA1 là output, RA0 là input

```
BCF TRISA,1
```

```
BSF TRISA,0
```

Tương tự, RB5 là input, RB7 là output

```
BSF TRISB,5
```

```
BCF TRISB,7
```

BÀI TẬP THAM KHẢO

Bài tập 1:

Viết chương trình tạo xung vuông tại chân RB7, có tần số $f = 50\text{hz}$ (thạch anh 4Mhz)

Ta có:

$$T = 1/f = 20.000\mu\text{S}$$

$$\Rightarrow T_d = 10.000\mu\text{S}$$

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

DEM2 EQU 20H

DEM1 EQU 21H

ORG 0000H

```

                                BCF      STATUS,6
                                BCF      STATUS,5
                                CLRF     PORTB
                                BSF      STATUS,5
                                BCF      TRISB,7
                                BCF      STATUS,5

MAIN
                                BSF      PORTB,7
                                CALL     DELAY
                                BCF      PORTB,7
                                CALL     DELAY
                                GOTO     MAIN

DELAY
                                MOVLW    D'33'
                                MOVWF    DEM1

LOOP
                                DECFSZ   DEM1
                                GOTO     LOOP1
                                GOTO     THOAT

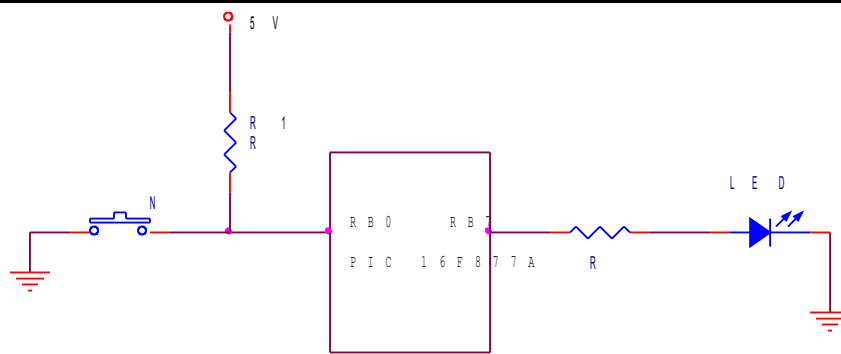
LOOP1
                                MOVLW    D'100'
                                MOVWF    DEM2

LOOP3
                                DECFSZ   DEM2
                                GOTO     LOOP3
                                GOTO     LOOP

THOAT
                                NOP
                                RETURN
                                END
    
```

Bài tập 2:

Viết chương trình điều khiển đèn: Ở trạng thái ban đầu đèn tắt, nhấn N buông ra đèn sang. Nếu đèn đang sang nhấn N buông ra đèn tắt và ngược lại.



Hình 3.1

```
PROCESSOR 16F877A
#include <P16F877A.INC>
ORG 0000H
```

```

                BCF      STATUS,6
                BCF      STATUS,5
                CLRF     PORTB
                BSF      STATUS,5
                BSF      TRISB,0
                BCF      TRISB,7
                BCF      STATUS,5

MAIN
                BTFSS    PORTB,0
                GOTO     LOOP1
                GOTO     MAIN

LOOP1
                BTFSC    PORTB,0
                GOTO     LOOP2
                GOTO     LOOP1

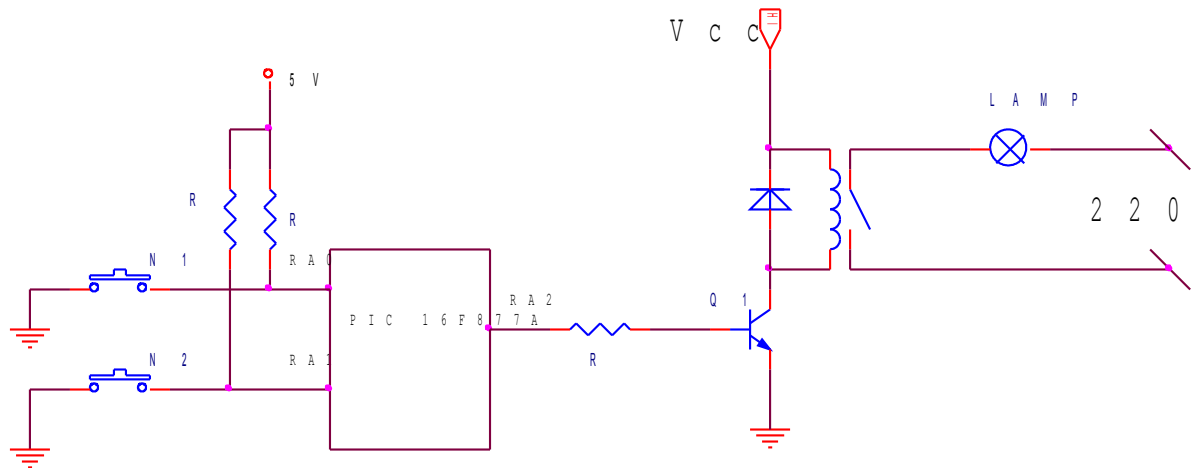
LOOP2
                BTFSS    PORTB,7
                GOTO     ON
                GOTO     OFF

OFF
                BCF      PORTB,7
                GOTO     MAIN

ON
                BSF      PORTB,7
                GOTO     MAIN
END
```

Bài tập 3:

Viết chương trình điều khiển đèn cầu thang: Nếu đèn đang tắt nhấn N1(hoặc N2), rồi buông ra đèn sáng và ngược lại.



Hình 3.2

```

PROCESSOR 16F877A
#include <P16F877A.INC>
ORG 0000H

        BCF      STATUS,6
        BCF      STATUS,5
        CLRF     PORTA
        BSF      STATUS,5
        BSF      TRISA,0
        BSF      TRISA,1
        BCF      TRISA,2
        BCF      STATUS,5

MAIN
        BTFSC    PORTA,0
        GOTO     LOOP1
        GOTO     KT_1

LOOP1
        BTFSS    PORTA,1
        GOTO     KT_2
        GOTO     MAIN

KT_1
        BTFSS    PORTA,0
        GOTO     KT_1
        GOTO     ON/OFF

KT_2
        BTFSS    PORTA,1
        GOTO     KT_2
        GOTO     ON/OFF

ON/OFF
        BTFSS    PORTA,2
        GOTO     ON
        GOTO     OFF
    
```

ON

BSF	PORTA,2
GOTO	MAIN

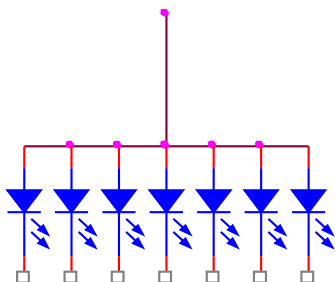
OFF

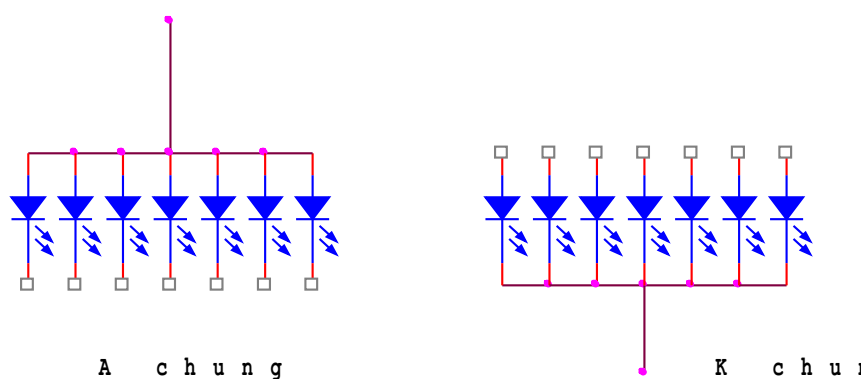
BCF	PORTA,2
GOTO	MAIN
END	

3.4. CÁC KHỐI GIAO TIẾP

3.4.1. GIAO TIẾP VỚI LED 7 ĐOẠN

Trong các thiết bị, để báo trạng thái hoạt động của thiết bị đó cho người sử dụng với thông số chỉ là các dãy số đơn thuần, thường người ta sử dụng "**led 7 đoạn**". Led 7 đoạn được sử dụng khi các dãy số không đòi hỏi quá phức tạp, chỉ cần hiện thị số là đủ, chẳng hạn led 7 đoạn được dùng để hiển thị nhiệt độ phòng, trong các đồng hồ treo tường bằng điện tử, hiển thị số lượng sản phẩm được kiểm tra sau một công đoạn nào đó...

Led 7 đoạn có cấu tạo bao gồm 7 led đơn có dạng thanh xếp theo hình  và có thêm một led đơn hình tròn nhỏ thể hiện dấu chấm tròn ở góc dưới, bên phải của led 7 đoạn. Tám led đơn trên led 7 đoạn có Anode(cực +) hoặc Cathode(cực -) được nối chung với nhau vào một điểm, được đưa chân ra ngoài để kết nối với mạch điện. 8 cực còn lại trên mỗi led đơn được đưa thành 8 chân riêng, cũng được đưa ra ngoài để kết nối với mạch điện. Nếu led 7 đoạn có Anode(cực +) chung, đầu chung này được nối với +Vcc, các chân còn lại dùng để điều khiển trạng thái sáng tắt của các led đơn, led chỉ sáng khi tín hiệu đặt vào các chân này ở mức 0. Nếu led 7 đoạn có Cathode(cực -) chung, đầu chung này được nối xuống Ground (hay Mass), các chân còn lại dùng để điều khiển trạng thái sáng tắt của các led đơn, led chỉ sáng khi tín hiệu đặt vào các chân này ở mức 1.



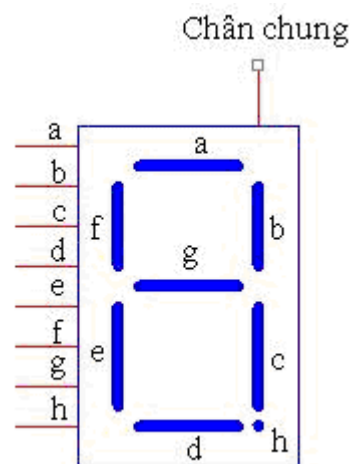
Hình 3.3

Vì led 7 đoạn chứa bên trong nó các led đơn, do đó khi kết nối cần đảm bảo dòng qua mỗi led đơn trong khoảng 10mA-20mA để bảo vệ led.

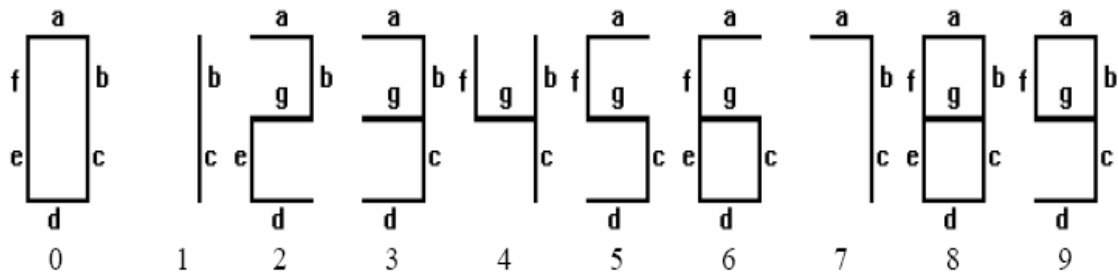
Nếu kết nối với nguồn 5V có thể hạn dòng bằng điện trở 330Ω trước các chân nhận tín hiệu điều khiển

Sơ đồ vị trí các led được trình bày như hình bên: Các điện trở 330Ω là các điện trở bên ngoài được kết nối để giới hạn dòng điện qua led nếu led 7 đoạn được nối với nguồn 5V.

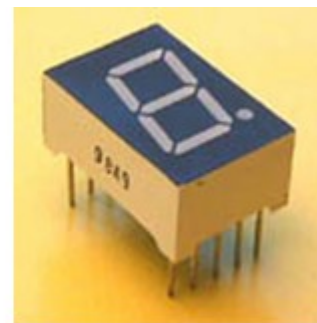
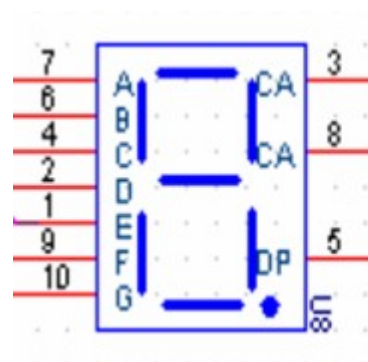
Chân nhận tín hiệu a điều khiển led a sáng tắt, ngõ vào b để điều khiển led b. Tương tự với các chân và các led còn lại



Hình 3.4: Ký hiệu Led 7 đoạn



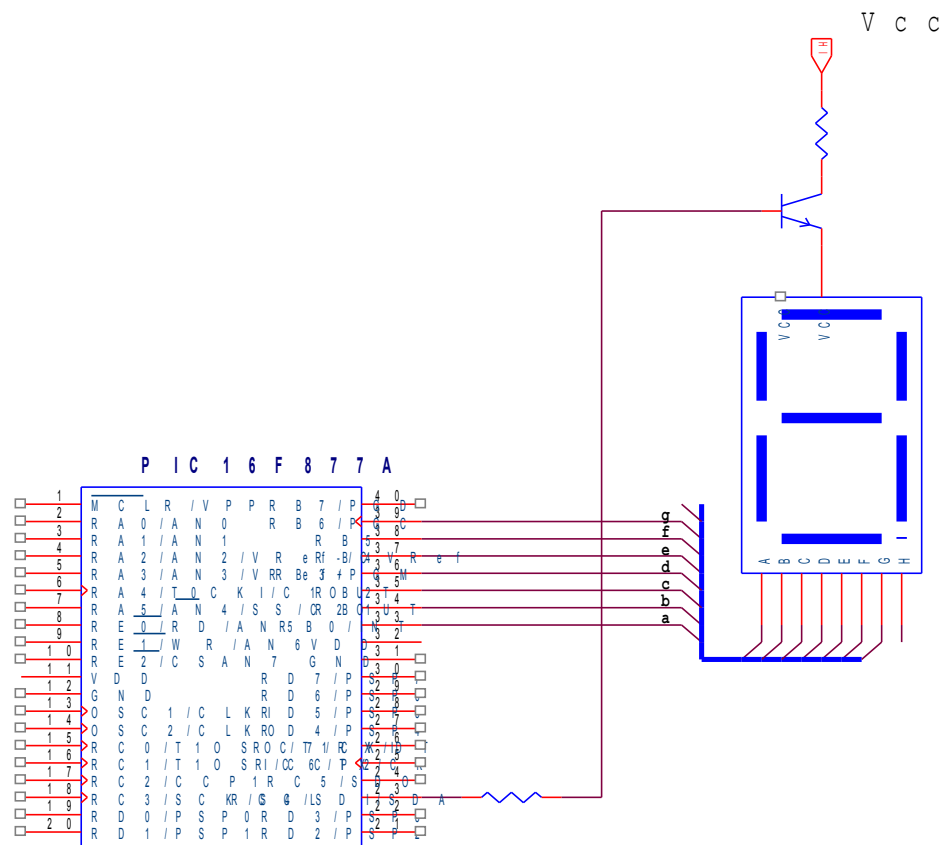
Hình 3.5: Dạng số hiển thị lên Led 7 đoạn



Hình 3.6: Hình dạng Led 7 đoạn

Bài tập 1:

Viết chương trình hiển thị số “3”



Hình 3.7

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

```
BCF    STATUS,6
BCF    STATUS,5
CLRF   PORTB
CLRF   PORTC
```

```
BSF    STATUS,5
CLRF   TRISB
BCF    TRISC,4
BCF    STATUS,5
```

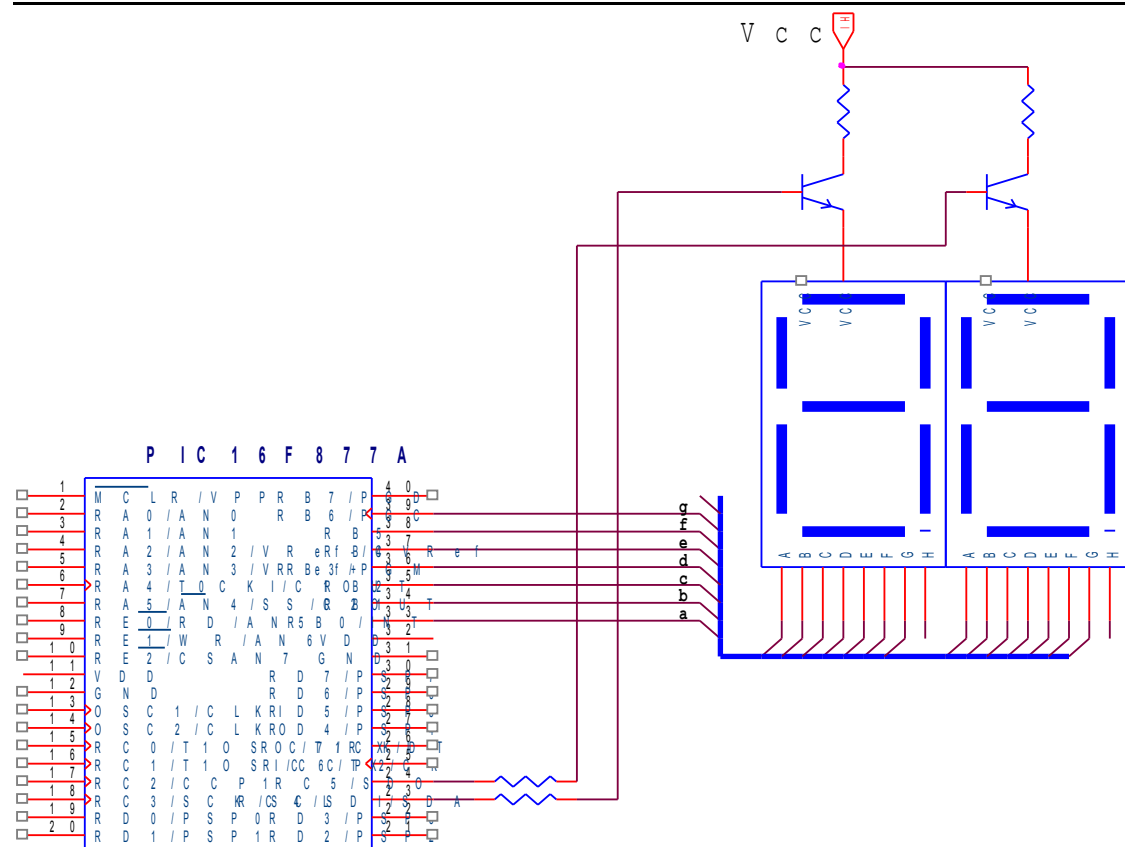
MAIN

```
BSF    PORTC,4
MOVLW  B'011 0000'
MOVWF  PORTB
GOTO   $
```

END

Bài tập 2:

Tuy nhiên, để hiển thị 2 số ví dụ như “37” chúng ta không nhất thiết phải dùng 2 port, mà có thể ghép song song 2 led 7 đoạn. Để hiển thị số “37”, tại một thời điểm ta cho một con Led sáng. Thời gian chớp tắt liên tục (tần số khoảng 40 Hz) làm cho mắt ta có cảm giác như 2 Led đang sáng liên tục. Phương pháp này gọi là “quét led”



Hình 3.8

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

```

    DEM1    EQU    20H
    DEM2    EQU    21H

```

ORG 0000H

```

    BCF      STATUS,6
    BCF      STATUS,5
    CLRF     PORTB
    CLRF     PORTC
    BSF      STATUS,5
    CLRF     TRISB
    BCF      TRISC,4
    BCF      TRISC,5
    BCF      STATUS,5

```

MAIN

```

    BSF      PORTC,4
    BCF      PORTC,5
    MOVLW    B'0110000'
    MOVWF    PORTB

```

```

CALL    DELAY_10ms
BCF     PORTC,4
BSF     PORTC,5
MOVLW   B'1111000'
MOVWF   PORTB
CALL    DELAY_10ms
GOTO    MAIN

DELAY_10ms
    MOVLW D'33'
    MOVWF DEM1
LOOP
    DECFSZ DEM1
    GOTO   LOOP1
    GOTO   THOAT

LOOP1
    MOVLW D'100'
    MOVWF DEM2

LOOP3
    DECFSZ DEM2
    GOTO   LOOP3
    GOTO   LOOP

THOAT
    NOP
    RETURN

```

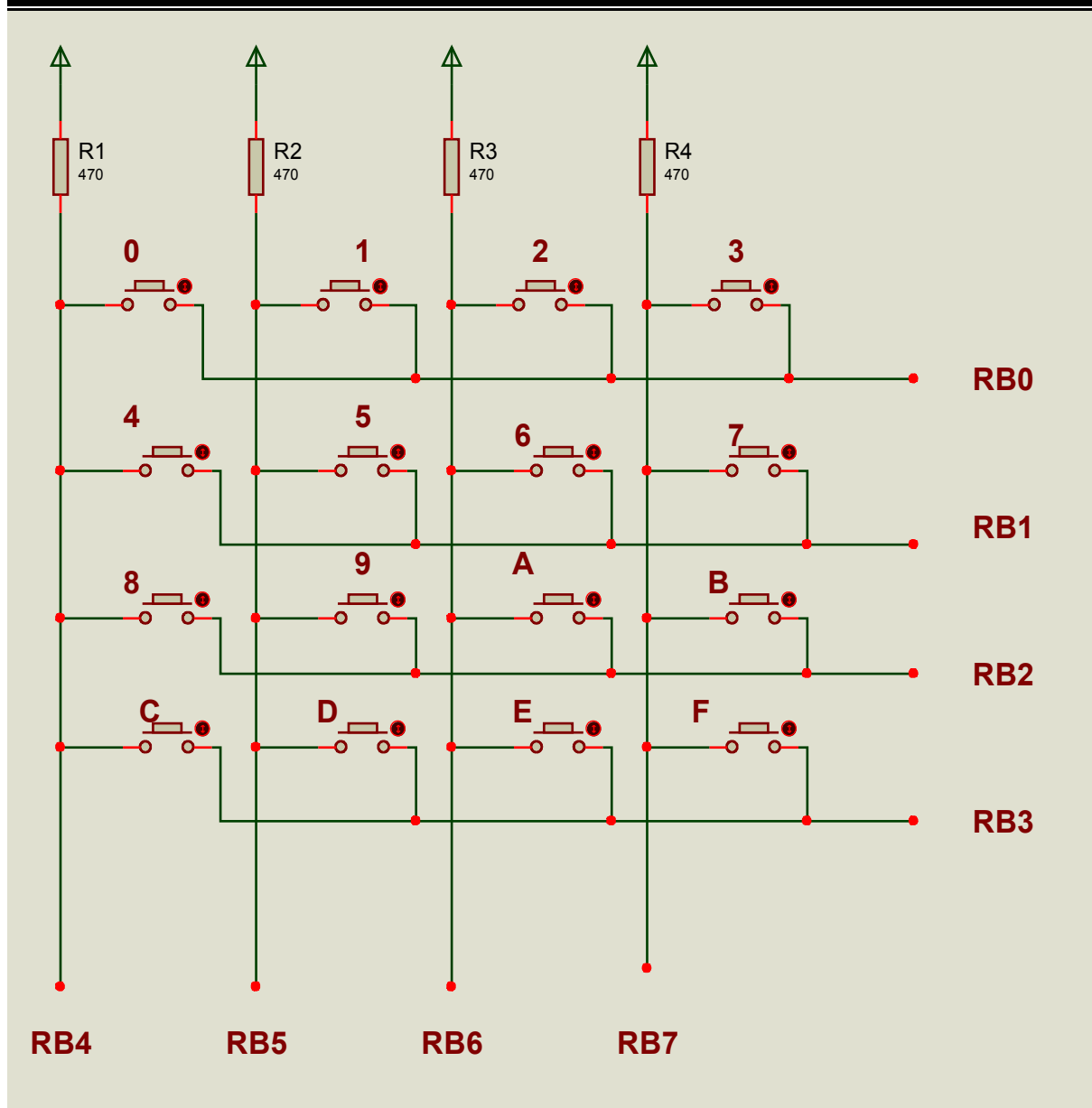
END

3.4.2. GIAO TIẾP VỚI BÀN PHÍM HEX

Khi giao tiếp với bàn phím Hex như hình 3.9 ta chọn RB0÷RB3 là output, RB4÷RB7 là input. Ban đầu ta cho RB3RB2RB1RB0=1110 sau đó chúng ta kiểm tra ngõ vào từ RB4÷RB7 để xác định tại vị trí nào mức thấp tương ứng với nút nhấn đó được tác động.

Nếu RB4=0 tức là vị trí số “0” được nhấn, tương tự RB5=0 thì số “1” được nhấn.

Tiếp theo ta cho RB3RB2RB1RB0=1101, khi đó kiểm tra nếu RB4 =0 thì nút số “4” thì được nhấn.....



Hình 3.9

DOC_BP

MOVLW	B'1110'
MOVWF	PORTB
BTFSS	PORTB,4
RETLW	D'0'
BTFSS	PORTB,5
RETLW	D'1'
BTFSS	PORTB,6
RETLW	D'2'
BTFSS	PORTB,7
RETLW	D'3'

```

MOVLW    B'1101'
MOVWF    PORTB
BTFSS    PORTB,4
RETLW    D'4'
BTFSS    PORTB,5
RETLW    D'5'
BTFSS    PORTB,6
RETLW    D'6'
BTFSS    PORTB,7
RETLW    D'7'

```

```

MOVLW    B'1011'
MOVWF    PORTB
BTFSS    PORTB,4
RETLW    D'8'
BTFSS    PORTB,5
RETLW    D'9'
BTFSS    PORTB,6
RETLW    0AH
BTFSS    PORTB,7
RETLW    0BH

```

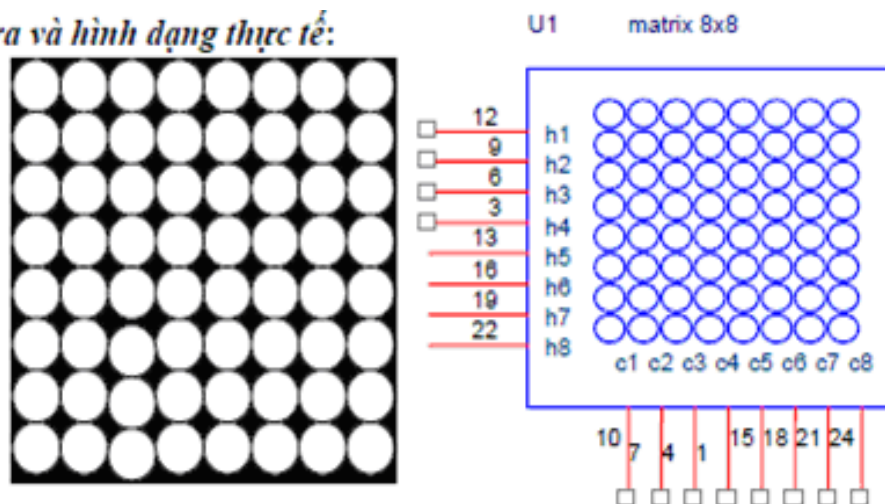
```

MOVLW    B'0111'
MOVWF    PORTB
BTFSS    PORTB,4
RETLW    0CH
BTFSS    PORTB,5
RETLW    0DH
BTFSS    PORTB,6
RETLW    0EH
BTFSS    PORTB,7
RETLW    0FH
RETURN

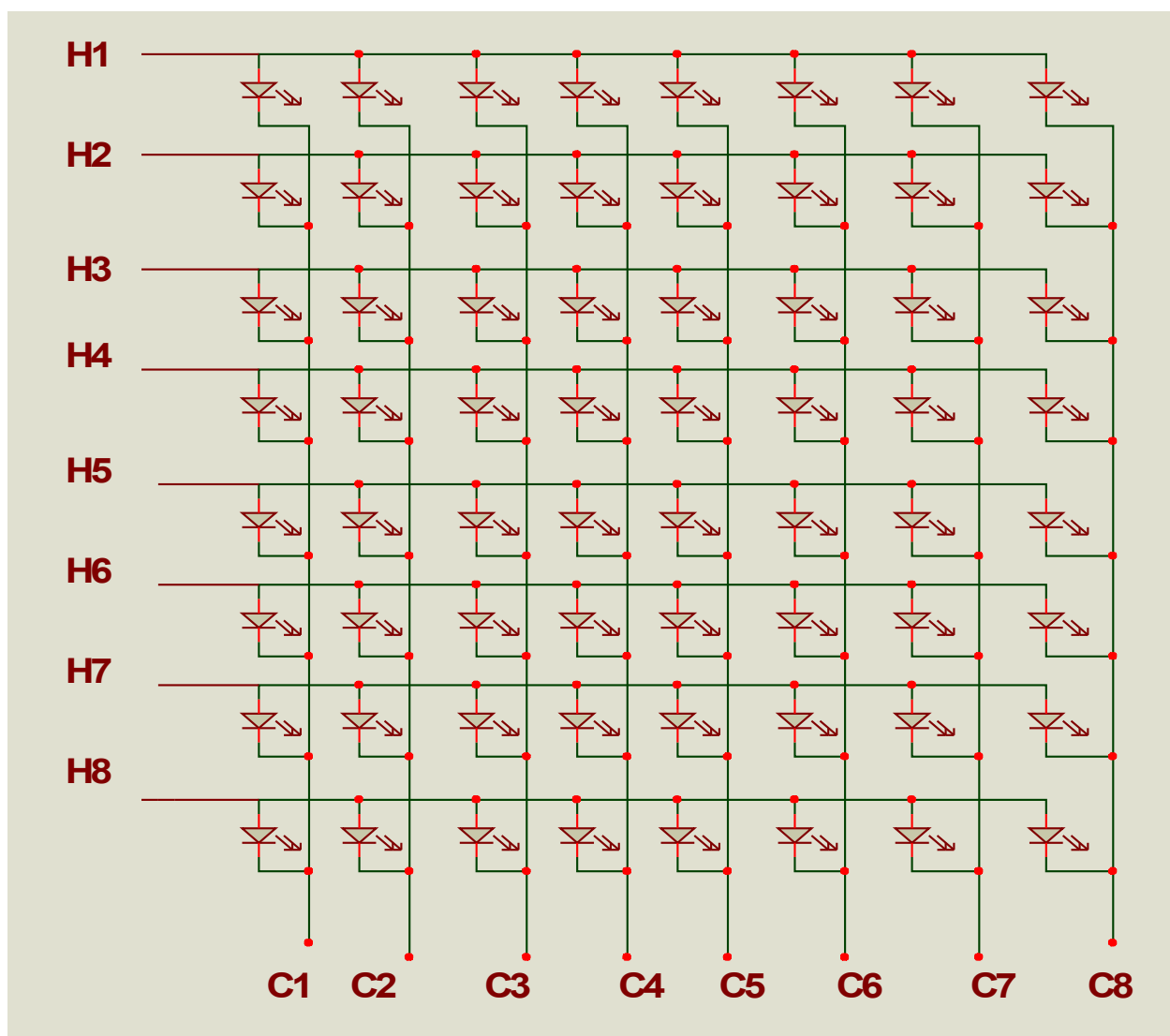
```

3.4.3. GIAO TIẾP VỚI LED MA TRẬN

- Sơ đồ chân ra và hình dạng thực tế:



Hình 3.10



Hình 3.11

Để Led matrận hiển thị 1 ký tự chúng ta dùng phương pháp quét, cũng như quét led 7 đoạn, ở đây chúng ta dùng cách quét cột. Tức là tại một thời điểm chỉ cho 1 cột cột sáng bằng cách tác động cho hàng và cột tích cực hợp lý.

Ví dụ: Điều khiển hàng là portb (RB0→H1, RB1→H2 RB7→H8), điều khiển cột là portd (RD0→C1, RD1→C2 RD7→C8).

Để hiển chữ “N” ta viết như sau:

MAIN

```
MOVLW    B'11111111'
MOVWF    PORTB
MOVLW    B'11111110'
MOVWF    PORTD
CALL     DELAY
```

```
MOVLW    B'11111111'
MOVWF    PORTB
MOVLW    B'11111101'
MOVWF    PORTD
CALL     DELAY
```

```
MOVLW    B'00000011'
MOVWF    PORTB
MOVLW    B'11111011'
MOVWF    PORTD
CALL     DELAY
```

```
MOVLW    B'00000110'
MOVWF    PORTB
MOVLW    B'11110111'
MOVWF    PORTD
CALL     DELAY
```

```
MOVLW    B'00001100'
MOVWF    PORTB
MOVLW    B'11101111'
MOVWF    PORTD
CALL     DELAY
```

```
MOVLW    B'00011000'
MOVWF    PORTB
MOVLW    B'11011111'
MOVWF    PORTD
CALL     DELAY
```

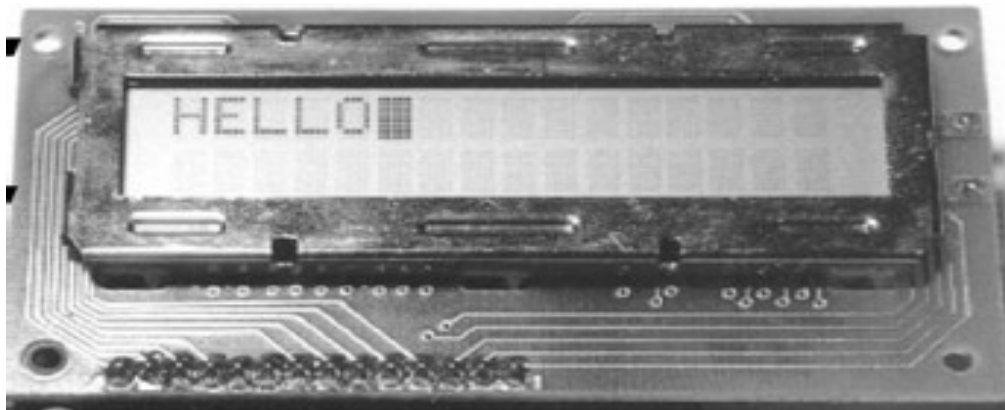
```
MOVLW    B'11111111'
MOVWF    PORTB
MOVLW    B'10111111'
MOVWF    PORTD
CALL     DELAY
```

```
MOVLW    B'11111111'
MOVWF    PORTB
MOVLW    B'01111111'
```

	MOVWF	PORTD
	CALL	DELAY
	GOTO	MAIN
; TAO TRE 2.5mS = 2500uS		
DELAY		
	MOVLW	D'8'
	MOVWF	DEM1
LOOP		
	DECFSZ	DEM1
	GOTO	LOOP1
	GOTO	THOAT
LOOP1		
	MOVLW	D'104'
	MOVWF	DEM2
LOOP3		
	DECFSZ	DEM2
	GOTO	LOOP3
	GOTO	LOOP
THOAT		
	NOP	
	RETURN	

END

3.4.4 GIAO TIẾP VỚI LCD



Hình 3.12

Bên trong LCD có 2 thanh ghi 8 bit quan trọng : Thanh ghi lệnh IR (Instructor Register) và thanh ghi dữ liệu DR (Data Register)

Thanh ghi IR : Để điều khiển LCD

Như vậy để điều khiển LCD chúng ta cần đưa mã lệnh điều khiển thích hợp vào thanh ghi IR thông qua bảng mã sau:

Bảng mã lệnh

Mã Số Hex	Lệnh Đến Thanh Ghi Của LCD
01	Xóa màn hình hiển thị
02	Trở về đầu dòng
04	Giảm con trỏ (dịch con trỏ sang trái)
06	Tăng con trỏ (dịch con trỏ sang phải)
05	Dịch hiển thị sang phải
07	Dịch hiển thị sang trái
08	Tắt con trỏ, tắt hiển thị
0A	Bật con trỏ, tắt hiển thị
0C	Tắt con trỏ, bật hiển thị
0E	Nhấp nháy con trỏ, bật hiển thị
0F	Tắt con trỏ, nhấp nháy con trỏ
10	Dịch vị trí con trỏ sang trái
14	Dịch vị trí con trỏ sang phải
18	Dịch toàn bộ hiển thị sang trái
1C	Dịch toàn bộ hiển thị sang phải
80	Đưa con trỏ về đầu dòng thứ nhất
C0	Đưa con trỏ về đầu dòng thứ hai
38	Cài LCD chạy chế độ 2 dòng và dùng ma trận 5 x 7

Ví dụ: Muốn xóa màn hình chúng ta cần đưa giá trị 01H vào IR

Thanh ghi DR : Thanh ghi DR dùng để chứa dữ liệu 8 bit để ghi vào vùng RAM DDRAM hoặc CGRAM (chế độ ghi) hoặc dùng để chứa dữ liệu từ 2 vùng RAM này gởi ra cho MCU (ở chế độ đọc).

DR cũng là thanh Ram chứa dữ liệu cần hiển thị lên LCD.

Ví dụ: Để hiển thị lên LCD chữ “A” chúng ta đưa giá trị 65 vào DR (mã ascii của A là 65)

Để đưa giá trị thích hợp vào IR hoặc DR thông qua 3 chân điều khiển: E, RS và RW

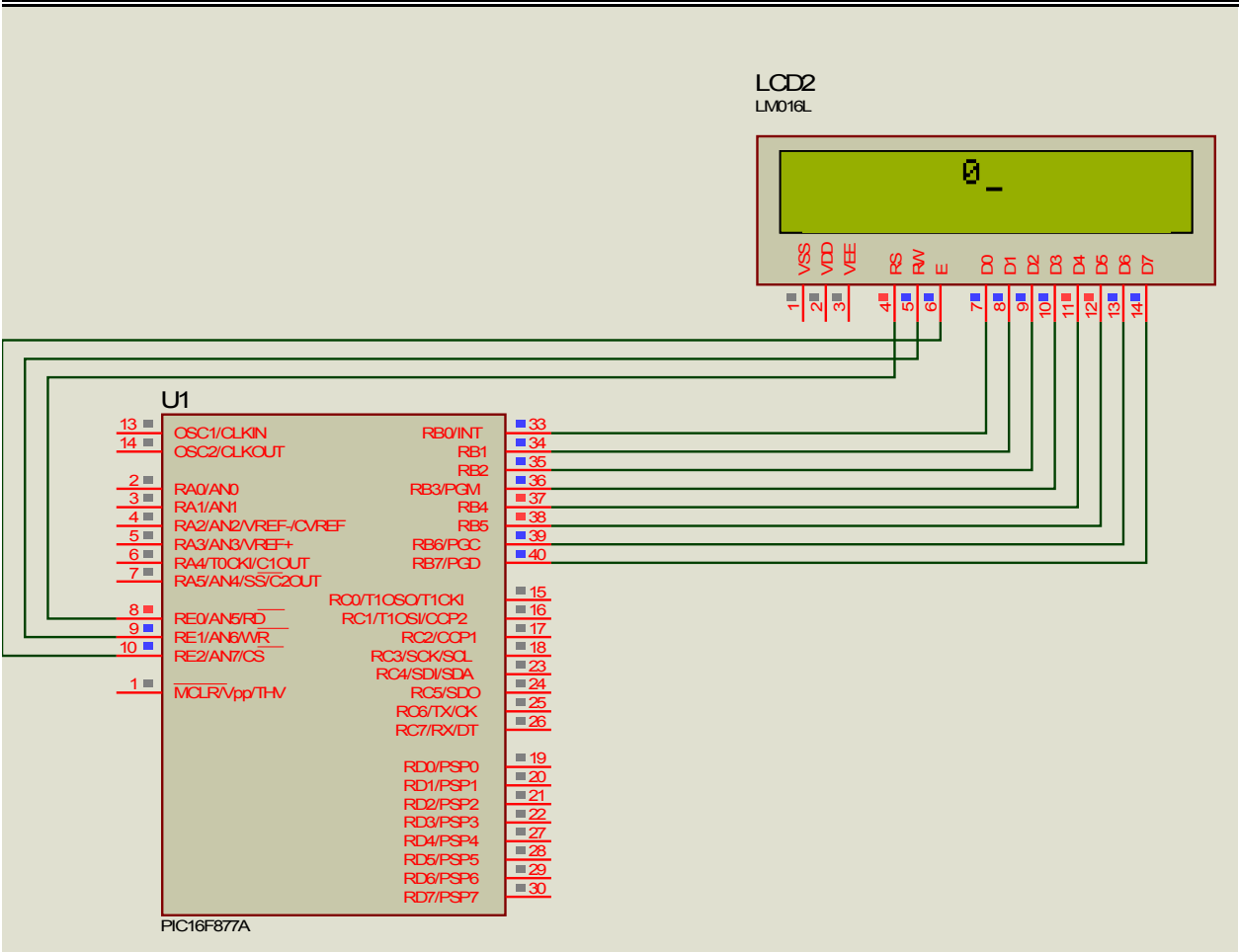
Bảng chức năng chân

Số Chân	Tên Gọi	Chức Năng
1	VSS	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển.
2	VDD	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với VCC=5V của mạch điều khiển.
3	VEE	Chân này dùng để điều chỉnh độ tương phản của LCD. Khi thiết kế nối chân này với chân điều chỉnh của biến trở khoảng 5K đến 10k.
4	RS	Chân chọn thanh ghi (Register select). Nối chân RS với logic “0” (GND) hoặc logic “1” (VCC) để chọn thanh ghi. + Logic “0”: Bus DB0-DB7 sẽ nối với thanh ghi lệnh

		IR của LCD (ở chế độ “ghi” - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ “đọc” - read) + Logic “1”: Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
5	R/W	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E (xung này có độ rộng hơn $\geq 4\mu s$). + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào (chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (high-to-low transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7 --> 14	D0 --> D7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này : + Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7. Chi tiết sử dụng 2 giao thức này được đề cập ở phần sau.
15	A	Chân dương (+) của đèn nền LCD.
16	K	Chân âm (-) của đèn nền LCD.

Ví dụ:

Cho sơ đồ như hình 3.13. Viết chương trình cho LCD hiển thị số “0” ở giữa hàng thứ nhất.



Hình 3.13

```

PROCESSOR      16F877A
#include <P16F877A.INC>
TAM            EQU    20H
DEM1           EQU    21H
DEM2           EQU    22H
DEM3           EQU    23H

ORG            0000H

BSF            STATUS,5
BCF            STATUS,6
CLRF          TRISB
CLRF          TRISE
BCF            STATUS,5
CLRF          PORTB
CLRF          PORTE

MAIN
MOVLW         01h ; Dưa 01h vào IR, xóa màn hình
MOVWF         TAM
CALL          KHOITAO
CALL          DELAY; Tao tre
    
```

	MOVLW	38h; Dưa 38h vào IR, khởi tạo LCD 2 hàng
	MOVWF	TAM
	CALL	KHOITAO
	CALL	DELAY
	MOVLW	0Eh ; ; Dưa 0eh vào IR, nhập nhảy con trỏ, bắt hiển thị
	MOVWF	TAM
	CALL	KHOITAO
	CALL	DELAY
	MOVLW	87h
	MOVWF	TAM
	CALL	KHOITAO
	CALL	DELAY
	MOVLW	D'48'
	MOVWF	TAM
	CALL	XUATLCD
	CALL	DELAY
	GOTO	\$
XUATLCD	MOVLW	B'101' ; E=1, RW=0, RS=1
	MOVWF	PORTE
	MOVF	TAM,0
	MOVWF	PORTB
	MOVLW	B'001' ; E=0 tạo cạnh xuống
	MOVWF	PORTE
	RETURN	
KHOITAO	MOVLW	B'100' ; ; E=1, RW=0, RS=0
	MOVWF	PORTE
	MOVF	TAM,0
	MOVWF	PORTB
	MOVLW	B'000'
	MOVWF	PORTE
	RETURN	
DELAY	MOVLW	D'20'
	MOVWF	DEM1
LOOP	DECFSZ	DEM1
	GOTO	LOOP1
	GOTO	THOAT
LOOP1	MOVLW	D'100'
	MOVWF	DEM2

LOOP3

DECFSZ

DEM2

GOTO

LOOP3

GOTO

LOOP

THOAT

NOP

RETURN

END

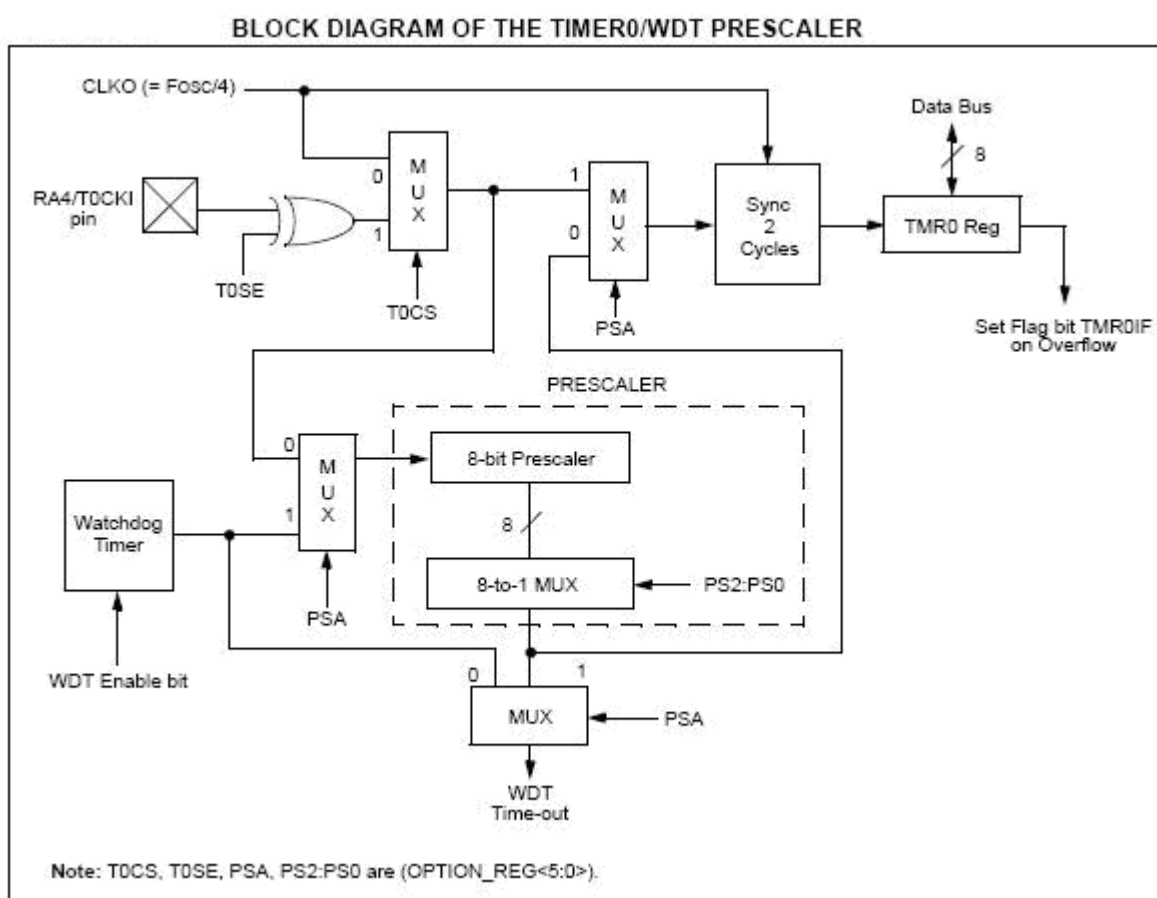
CHƯƠNG 4 CÁC KHỐI CHỨC NĂNG

4.1.BỘ ĐỊNH THỜI

4.1.1.TIMER 0

Đây là một trong ba bộ đếm hoặc bộ định thời của vi điều khiển PIC16F877A. Timer0 là bộ đếm 8 bit được kết nối với bộ chia tần số (prescaler) 8 bit. Cấu trúc của Timer0 cho phép ta lựa chọn xung clock tác động và cạnh tích cực của xung clock. Ngắt Timer0 sẽ xuất hiện khi Timer0 bị tràn. Bit TMR0IE (INTCON<5>) là bit điều khiển của Timer0. TMR0IE=1 cho phép ngắt Timer0 tác động, TMR0IF= 0 không cho phép ngắt Timer0 tác động.

Sơ đồ khối của Timer0 như sau:



Hình 4.7: Cấu trúc bên trong của bộ định thời Timer0

Muốn Timer0 hoạt động ở chế độ Timer ta clear bit TOSC (OPTION_REG<5>), khi đó giá trị thanh ghi TMR0 sẽ tăng theo từng chu kỳ xung đồng hồ (**tần số vào Timer0 bằng ¼tần số oscillator**). Khi giá trị thanh ghi TMR0 từ FFh trở về 00h, ngắt Timer0 sẽ xuất hiện. Thanh ghi TMR0 cho phép ghi và xóa được giúp ta ấn định thời điểm ngắt Timer0 xuất hiện một cách linh động.

Muốn Timer0 hoạt động ở chế độ counter ta set bit TOSC (OPTION_REG<5>). Khi

đó xung tác động lên bộ đếm được lấy từ chân RA4/TOCK1. Bit TOSE (OPTION_REG<4>) cho phép lựa chọn cạnh tác động vào bộ đếm. **Cạnh tác động sẽ là cạnh lên nếu TOSE=0 và cạnh tác động sẽ là cạnh xuống nếu TOSE=1.**

Khi thanh ghi TMR0 bị tràn, bit TMR0IF (INTCON<2>) sẽ được set. Đây chính là cờ ngắt của Timer0. Cờ ngắt này phải được xóa bằng chương trình trước khi bộ đếm bắt đầu thực hiện lại quá trình đếm. Ngắt Timer0 không thể "đánh thức" vi điều khiển từ chế độ sleep. **Bộ chia tần số (prescaler) được chia sẻ giữa Timer0 và WDT (Watchdog Timer).** Điều đó có nghĩa là nếu prescaler được sử dụng cho Timer0 thì WDT sẽ không có được hỗ trợ của prescaler và ngược lại. Prescaler được điều khiển bởi thanh ghi OPTION_REG. Bit PSA (OPTION_REG<3>) xác định đối tượng tác động của prescaler. Các bit PS2:PS0 (OPTION_REG<2:0>) xác định tỉ số chia tần số của prescaler. Xem lại thanh ghi OPTION_REG để xác định lại một cách chi tiết về các bit điều khiển trên. Các lệnh tác động lên giá trị thanh ghi TMR0 sẽ xóa chế độ hoạt động của prescaler. Khi đối tượng tác động là Timer0, tác động lên giá trị thanh ghi TMR0 sẽ xóa prescaler nhưng không làm thay đổi đối tượng tác động của prescaler. Khi đối tượng tác động là WDT, lệnh CLRWDT sẽ xóa prescaler, đồng thời prescaler sẽ ngưng tác vụ hỗ trợ cho WDT

Chú ý:

* **Thanh ghi liên quan đến Timer0:** TMR0 (địa chỉ 01h, 101h) : chứa giá trị đếm của Timer0. (chi tiết xem bảng phụ lục trang 96)

***Thanh ghi điều khiển Timer0:**

OPTION_REG (địa chỉ 81h, 181h): điều khiển prescaler.

Thanh ghi này cho phép đọc và ghi, cho phép điều khiển chức năng pull-up của các chân trong PORTB, xác lập các tham số về xung tác động, cạnh tác động của ngắt ngoại vi và bộ đếm Timer0. Thanh ghi tùy chọn chứa các bit điều khiển để cấu hình cho các chức năng như:

ngắt ngoại, Timer 0 chức năng kéo lên Vdd của các chân Port B, và thời gian chờ của WDT.

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
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Bit 7 **$\overline{\text{RBPU}}$** : Bit cho phép PORTB được kéo lên nguồn.

1: Không cho phép PORTB kéo lên nguồn.

0: Cho phép PORTB kéo lên nguồn.

Bit 6 **INTEDG**: Bit lựa chọn cạnh tác động ngắt (**INTERRUPT EDGE**)

1: Ngắt sẽ được tác động bởi cạnh lên của chân **RB0/INT**

0: Ngắt sẽ được tác động bởi cạnh xuống của chân **RB0/INT**

Bit 5 **T0CS**: Bit lựa chọn nguồn xung Clock cho **Timer 0**

1: Xung Clock cung cấp bởi nguồn ngoài qua chân **RA4/T0CKI**

0: Xung Clock cung cấp bởi nguồn dao động nội.

Bit 4 **T0SE**: Bit lựa chọn cạnh nào của xung clock (bên ngoài) tác động lên **timer 0**

1: Cạnh xuống

0: Cạnh lên

Bit 3 **PSA**: Bit quyết định tốc độ đếm **PS2:PS0** sẽ tác động lên Timer 0 hay **WDT**

1: Tốc độ đếm **PS2:PS0** sẽ tác động lên **WDT**

0: Tốc độ đếm **PS2:PS0** sẽ tác động lên **Timer 0**

Bit 2-0 **PS2:PS0**: Dùng để lựa chọn tốc độ đếm của timer hay **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

Chú ý1:

Các bước viết chương trình Delay dùng Timer0

+ Chọn chia tần **OPTION_REG <2÷0 >**

+ Đặt giá trị vào thanh ghi **TMR0**

+ Cho phép bộ định thời **Timer0** hoạt động (cho bit **OPTION_REG <5> =0**)

+ Kiểm tra đếm xong chưa? Kiểm tra cờ tràn.

Chú ý2:

Các bước viết chương trình đọc xung dùng Timer0

a. không có chia tần (có 1 xung thì giá trị trong TRM0 tăng 1 đơn vị)

BSF **OPTION_REG,4**; chọn tác động cạnh xuống

BSF **OPTION_REG,5**

BSF **OPTION_REG,3**

b. có chia tần (có nhiều xung thì giá trị trong TRM0 tăng 1 đơn vị, tùy thuộc vào giá trị chia tần. nếu chia tần 1:8 thì có 8 xung đọc về TM0 tăng 1 đơn vị)

BSF **OPTION_REG,4**; chọn tác động cạnh xuống

BCF **OPTION_REG,3**

+ thực hiện chia tần **OPTION_REG <2÷0 >**

BSF

OPTION_REG,5; cho phép đọc xung

Ví dụ1:

Viết chương trình con tạo trễ 40ms, thạch anh 4Mhz

DELAY

```
BSF      STATUS,5;  Chọn bank 1
BCF      STATUS,6;  Chọn bank1
BCF      OPTION_REG,3 ; Kết quả tác động lên TMR0
BCF      OPTION_REG,2;  Chọn chia tần 1:4
BCF      OPTION_REG,1
BSF      OPTION_REG,0
BCF      STATUS,5;  Trở lại bank0
MOVLW    D'50'
MOVWF    DEM;  Giá trị DEM=50
```

BATDAU

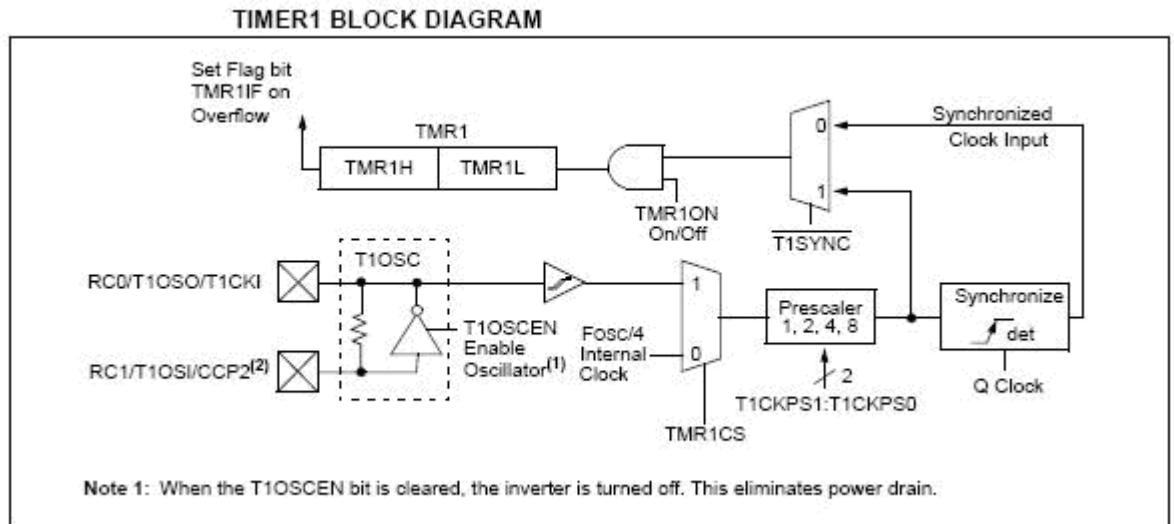
```
MOVLW    D'55'
MOVWF    TMR0;      TMR0= 55
BSF      STATUS,5
BCF      OPTION_REG,5; Cho phép bộ định thời hoạt động
BCF      STATUS,5
```

LOOP

```
BTFSS    INTCON,2;      Đếm xong chưa?
GOTO     LOOP
BCF      INTCON,2;      Xóa cờ tràn
DECFSZ   DEM,1;        Giảm giá trị đếm 1 đơn vị
GOTO     BATDAU
RETURN
```

4.1.2.TIMER1

Timer1 là bộ định thời 16 bit, giá trị của Timer1 sẽ được lưu trong hai thanh ghi (TMR1H:TMR1L). Cờ ngắt của Timer1 là bit TMR1IF (PIR1<0>). Bit điều khiển của Timer1 sẽ là TMR1IE (PIE<0>). Tương tự như Timer0, Timer1 cũng có hai chế độ hoạt động: chế độ định thời (timer) với xung kích là xung clock của oscillator (tần số của timer bằng 1/4 tần số của oscillator) và chế độ đếm (counter) với xung kích là xung phản ánh các sự kiện cần đếm lấy từ bên ngoài thông qua chân RC0/T1OSO/T1CKI (cạnh tác động là cạnh lên). Việc lựa chọn xung tác động (tương ứng với việc lựa chọn chế độ hoạt động là timer hay counter) được điều khiển bởi bit TMR1CS (T1CON<1>). Sau đây là sơ đồ khối của Timer1:



Hình 4.8: Cấu trúc bên trong của bộ định thời Timer1

Ngoài ra Timer1 còn có chức năng reset input bên trong được điều khiển bởi một trong hai khối CCP (Capture/Compare/PWM).

Khi bit T1OSCEN (T1CON<3>) được set, Timer1 sẽ lấy xung clock từ hai chân RC1/T1OSI/CCP2 và RC0/T1OSO/T1CKI làm xung đếm. Timer1 sẽ bắt đầu đếm sau cạnh xuống đầu tiên của xung ngõ vào. Khi đó PORTC sẽ bỏ qua sự tác động của hai bit TRISC<1:0> và PORTC<2:1> được gán giá trị 0. Khi clear bit T1OSCEN Timer1 sẽ lấy xung đếm từ oscillator hoặc từ chân RC0/T1OSO/T1CKI. Timer1 có hai chế độ đếm là đồng bộ (Synchronous) và bất đồng bộ (Asynchronous). Chế độ đếm được quyết định bởi bit điều khiển (T1CON<2>). Khi =1 xung đếm lấy từ bên ngoài sẽ không được đồng bộ hóa với xung clock bên trong, Timer1 sẽ tiếp tục quá trình đếm khi vi điều khiển đang ở chế độ sleep và ngắt do Timer1 tạo ra khi bị tràn có khả năng "đánh thức" vi điều khiển. Ở chế độ đếm bất đồng bộ, Timer1 không thể được sử dụng để làm nguồn xung clock cho khối CCP (Capture/Compare/Pulse width modulation). Khi =0 xung đếm vào Timer1 sẽ được đồng bộ hóa với xung clock bên trong. Ở chế độ này Timer1 sẽ không hoạt động khi vi điều khiển đang ở chế độ sleep.

Chú ý:

***Các thanh ghi liên quan đến Timer1 bao gồm:**

INTCON (địa chỉ 0Bh, 8Bh, 10Bh, 18Bh): cho phép ngắt hoạt động (GIE và PEIE).

PIR1 (địa chỉ 0Ch): chứa cờ ngắt Timer1 (TMR1IF).

PIE1 (địa chỉ 8Ch): cho phép ngắt Timer1 (TMR1IE).

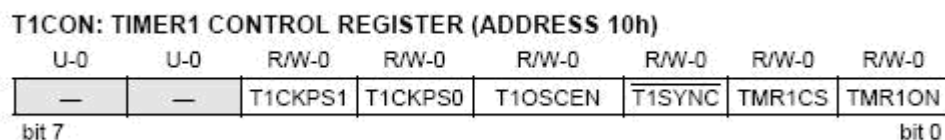
TMR1L (địa chỉ 0Eh): chứa giá trị 8 bit thấp của bộ đếm Timer1.

TMR1H (địa chỉ 0Fh): chứa giá trị 8 bit cao của bộ đếm Timer1.

Các thanh ghi trên chi tiết xem bảng phụ lục trang 96

***Thanh điều khiển Timer1:**

T1CON (địa chỉ 10h): xác lập các thông số cho Timer1.



Bit 7,6 Không sử dụng, đọc là 0.

Bit 5,4 **T1CKPS1 : T1CKPS0** : Các bit chọn tỉ lệ xung ngõ vào cho Timer1.

11 1 : 8 giá trị tỉ lệ

10 1 : 4 giá trị tỉ lệ

01 1 : 2 giá trị tỉ lệ

00 1 : 1 giá trị tỉ lệ

Bit 3 **T1OSCEN** : Bit cho phép bộ dao động Timer 1 Oscillator

1 : Cho phép dao động

0 : Không cho phép dao động

Bit 2 **T1SYNC** : Bit lựa chọn đồng bộ hóa xung clock ngoài của Timer 1

(**Chú ý:** Bit này chỉ có tác dụng khi bit **TMR1CS** = 1)

1: Không đồng bộ hóa xung clock ngoài

0: Đồng bộ hóa xung clock ngoài.

Bit 1 **TMR1CS** : Bit chọn nguồn xung clock cho **Timer 1**

1: Chọn xung clock ngoài qua chân **T1OSC/T1CKI** (tác động cạnh lên)

0: Chọn xung clock nội (Fosc/4)

Bit 0 **TMR1ON**: Bit cho phép ngoặc ngưng **Timer 1**

1: Cho phép

0: Không cho phép

Chi tiết về các thanh ghi khác sẽ được trình bày cụ thể trong phụ lục 2.

Chú ý:

Các bước viết chương trình Delay dùng Timer1:

+ Chọn chia tần thông qua thanh ghi **T1CON**

+ Đặt giá trị vào thanh ghi TMR1 (8 bit cao đưa vào TMR1H, 8 bit thấp đưa vào TMR1L)

+ Cho phép bộ định thời Timer0 hoạt động (set bit **T1CON<0>**)

+ Kiểm tra đếm xong chưa? Kiểm tra cờ tràn.

Ví dụ:

Viết chương trình con tạo trễ 1s, thạch anh 4Mhz

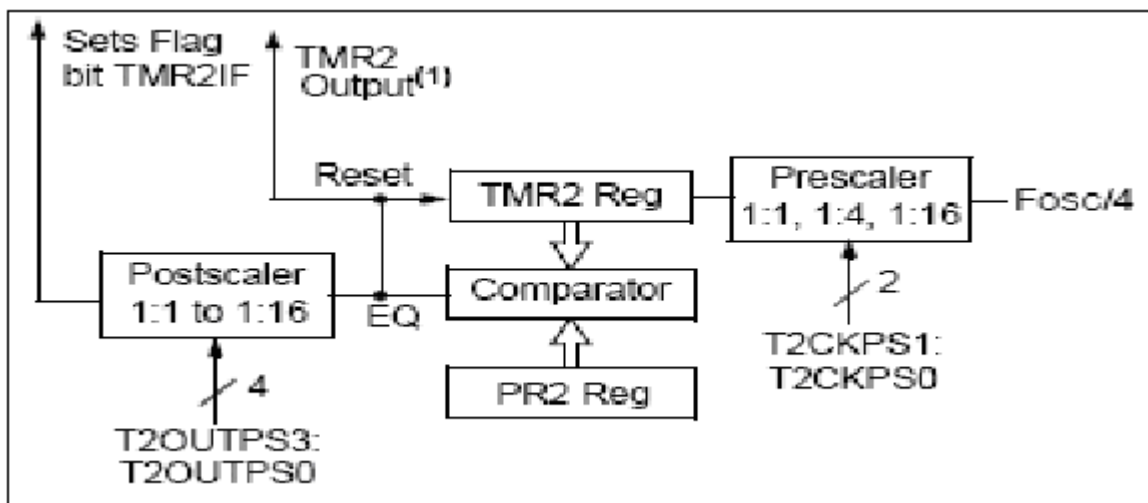
DELAY

```
BCF          STATUS,5
MOVLW       b'00000000';    Chọn chia tần 1:1
MOVWF       T1CON
```

	MOVLW	d'20';	DEM=20
	MOVWF	DEM	
BATDAU			
	MOVLW	3CH	
	MOVWF	TMR1H;	TMR1H=B'0011 1100'
	MOVLW	AFH	
	MOVWF	TMR1L;	TMR1L=B'1010 1111'
			=> TMR1=15535
	BSF	T1CON,0;	Cho phép bộ định thời hoạt động
LOOP			
	BTFSS	PIR1,0	
	GOTO	LOOP	;Đếm xong chưa?
	BCF	PIR1,0	
	DECFSZ	DEM,1;	Giảm giá trị đếm 1 đơn vị
	GOTO	BATDAU	
	RETURN		

4.1.3.TIMER2

Timer2 là bộ định thời 8 bit và được hỗ trợ bởi hai bộ chia tần số prescaler và postscaler. Thanh ghi chứa giá trị đếm của Timer2 là TMR2. Bit cho phép ngắt Timer2 tác động là TMR2ON (T2CON<2>). Cờ ngắt của Timer2 là bit TMR2IF (PIR1<1>). Xung ngõ vào (tần số bằng $\frac{1}{4}$ tần số oscillator) được đưa qua bộ chia tần số prescaler 4 bit (với các tỉ số chia tần số là 1:1, 1:4 hoặc 1:16 và được điều khiển bởi các bit T2CKPS1:T2CKPS0 (T2CON<1:0>)).



Hình 4.9: Cấu trúc bên trong của bộ định thời Timer2

Timer2 còn được hỗ trợ bởi thanh ghi PR2. Giá trị đếm trong thanh ghi TMR2 sẽ tăng

từ 00h đến giá trị chứa trong thanh ghi PR2, sau đó được reset về 00h. Khi reset thanh ghi PR2 được nhận giá trị mặc định FFh.

Ngõ ra của Timer2 được đưa qua bộ chia tần số postscaler với các mức chia từ 1:1 đến 1:16. Postscaler được điều khiển bởi 4 bit T2OUTPS3:T2OUTPS0. Ngõ ra của postscaler đóng vai trò quyết định trong việc điều khiển cờ ngắt.

Ngoài ra ngõ ra của Timer2 còn được kết nối với khối SSP, do đó Timer2 còn đóng vai trò tạo ra xung clock đồng bộ cho khối giao tiếp SSP.

Chú ý:

***Các thanh ghi liên quan đến Timer2 bao gồm:**

INTCON (địa chỉ 0Bh, 8Bh, 10Bh, 18Bh): cho phép toàn bộ các ngắt (GIE và PEIE).

PIR1 (địa chỉ 0Ch): chứa cờ ngắt Timer2 (TMR2IF).

PIE1 (địa chỉ 8Ch): chứa bit điều khiển Timer2 (TMR2IE).

TMR2 (địa chỉ 11h): chứa giá trị đếm của Timer2.

Các thanh ghi trên chi tiết xem bảng phụ lục trang 96

***Thanh điều khiển Timer2:**

T2CON (địa chỉ 12h): xác lập các thông số cho 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: không sử dụng

Bit 6:3 **TOUTPS3:TOUTPS0**: Bit chọn tỉ lệ ngõ ra của Timer 2

0000: 1:1 Tỷ lệ ngõ ra

0001: 1:2 Tỷ lệ ngõ ra

.

1111: 1:16 Tỷ lệ ngõ ra

Bit 2 **TMR2ON**: Bit cho phép hoạt động của Timer 2

1: Cho phép

0: Không cho phép.

Bit 1:0 **T2CKPS1:T2CKPS0**: Bit chọn tỉ lệ ngõ vào của Timer 2

00 : Prescaler 1

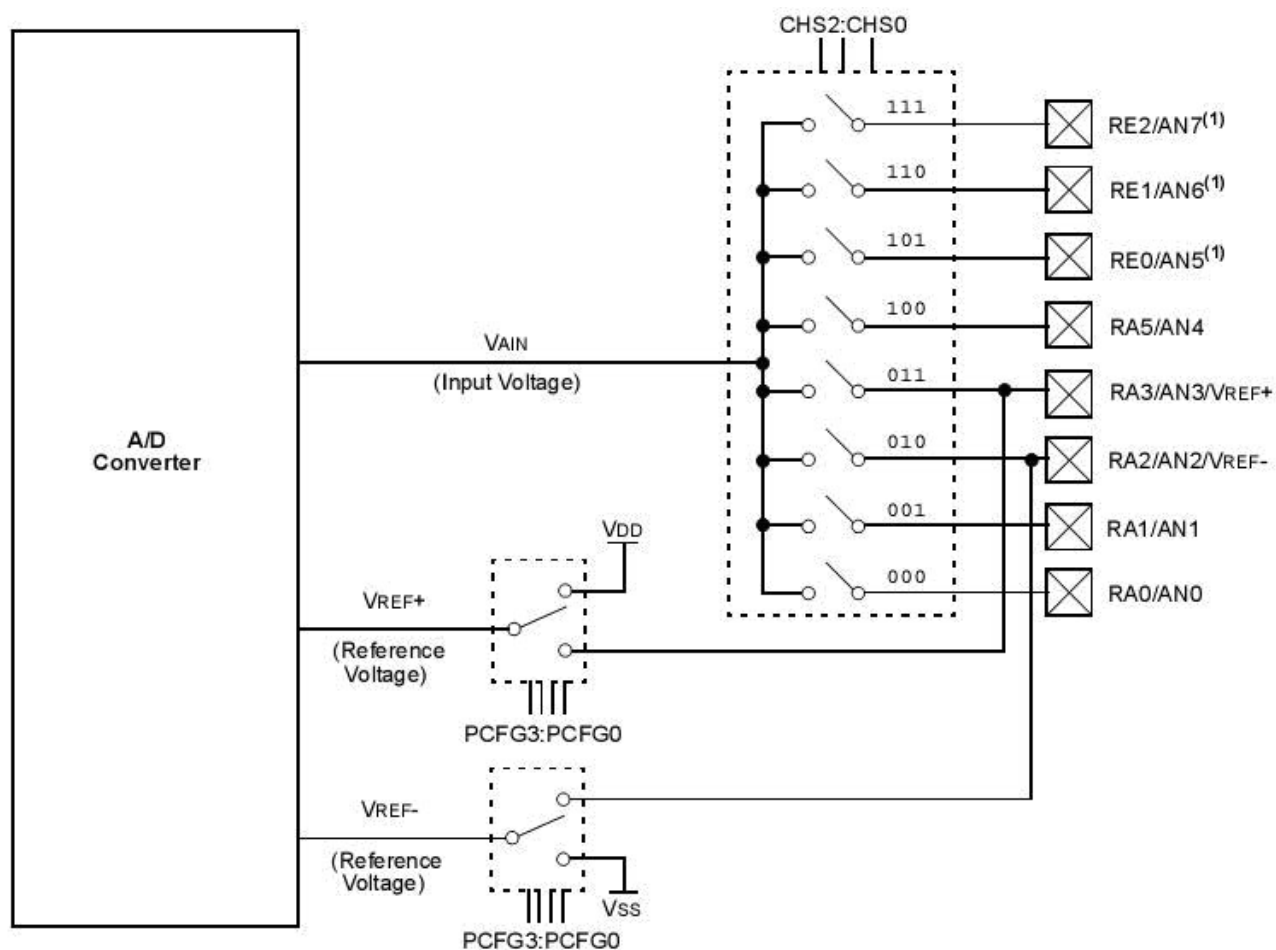
01 : Prescaler 4

1x : Prescaler 16

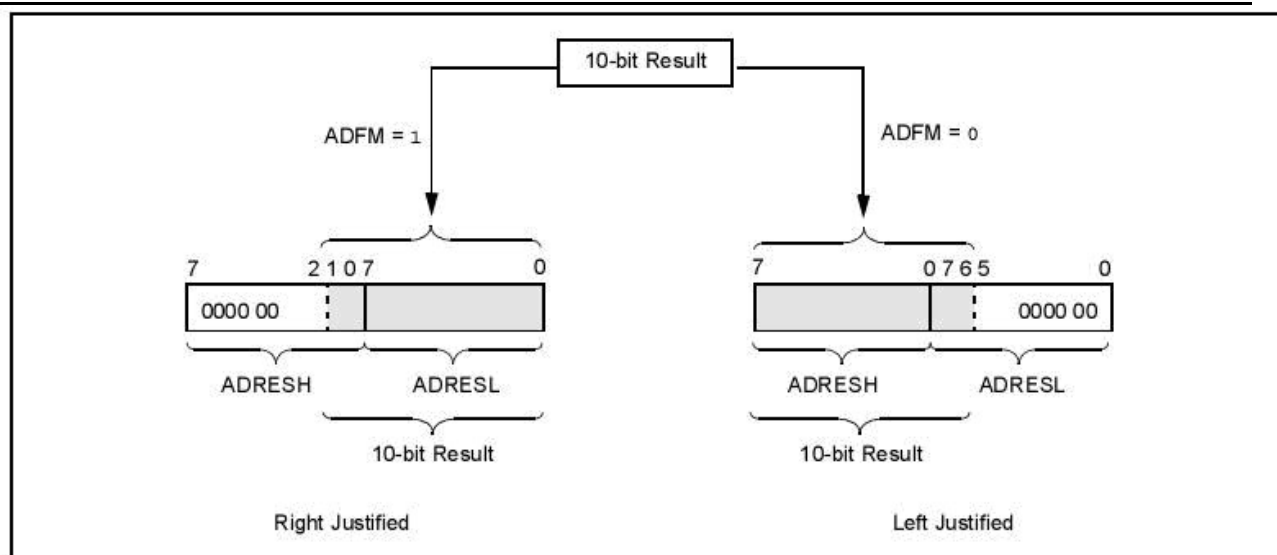
4.2. ADC

ADC (Analog to Digital Converter) là bộ chuyển đổi tín hiệu giữa hai dạng tương tự và số. PIC16F877A có 8 ngõ vào analog (RA4:RA0 và RE2:RE0). Hiệu điện thế chuẩn VREF có thể được lựa chọn là VDD, VSS hay hiệu điện thế chuẩn được xác lập trên hai chân RA2

và RA3. Kết quả chuyển đổi từ tín hiệu tương tự sang tín hiệu số là 10 bit số tương ứng và được lưu trong hai thanh ghi ADRESH:ADRESL. Khi không sử dụng bộ chuyển đổi ADC, các thanh ghi này có thể được sử dụng như các thanh ghi thông thường khác. Khi quá trình chuyển đổi hoàn tất, kết quả sẽ được lưu vào hai thanh ghi ADRESH:ADRESL, bit ADCON0<2>) được xóa về 0 và cờ ngắt ADIF được set.



Hình 4.10: Sơ đồ khối bộ chuyển đổi ADC:



Hình 4.11: cách lưu kết quả chuyển đổi AD:

Quy trình chuyển đổi từ tương tự sang số bao gồm các bước sau:

Bước 1: Chọn số ngõ vào, điện áp chuẩn V_{cc} .

Bước 2: Chọn ngõ vào cụ thể.

Bước 3: Chọn tần số chuyển đổi

Bước 4: Chọn nơi chứa kết quả chuyển đổi.

Bước 5: Bật bộ chuyển đổi, cho phép bộ chuyển đổi hoạt động

Bước 6 :Kiểm tra chuyển đổi xong chưa? Đọc kết quả về. Nếu muốn tiếp tục thì trở lại bước 5.

Chú ý:

*** Các thanh ghi liên quan đến bộ chuyển đổi ADC bao gồm:**

INTCON (địa chỉ 0Bh, 8Bh, 10Bh, 18Bh): cho phép các ngắt (các bit GIE, PEIE).

PIR1 (địa chỉ 0Ch): chứa cờ ngắt AD (bit ADIF).

PIE1 (địa chỉ 8Ch): chứa bit điều khiển AD (ADIE).

ADRESH (địa chỉ 1Eh) và ADRESL (địa chỉ 9Eh): thanh ghi chứa kết quả.

PORTA (địa chỉ 05h) và TRISA (địa chỉ 85h): liên quan đến I/O

PORTE (địa chỉ 09h) và TRISE (địa chỉ 89h): liên quan đến I/O

Các thanh ghi trên chi tiết xem bảng phụ lục trang 96

*** Thanh ghi điều khiển ADC:**

ADCON0 (địa chỉ 1Fh) và ADCON1 (địa chỉ 9Fh): xác lập các thông số cho bộ chuyển đổi AD.

- Thanh ghi điều khiển ADCON0:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

Bit 7:6 ADCS1:ADCS0: Các bit lựa chọn tần số chuyển đổi A/D

00 =FOSC/2

01 =FOSC/4

10 =FOSC/32

11 =FRC (xung clock được lấy từ dao động nội RC)

Bit 5:3 CHS2:CHS0: Các bit lựa chọn kênh Analog

000: Kênh 0, (AN0)

001: Kênh 1, (AN1)

010: Kênh 2, (AN2)

011: Kênh 3, (AN3)

100: Kênh 4, (AN4)

101: Kênh 5, (AN5)

110: Kênh 6, (AN6)

111: Kênh 7, (AN7)

Bit 2 GO/ DONE: Bit báo trạng thái chuyển đổi A/D

Khi bit ADON = 1

1: Quá trình A/D đang thực hiện (Khi chúng ta set bit này lên thì quá trình chuyển đổi sẽ xảy ra, khi quá trình kết thúc nó sẽ tự động được xóa bằng phần mềm).

0: Quá trình A/D không xảy ra hoặc đã hoàn tất.

Bit 1 Không sử dụng, giá trị là 0

Bit 0 ADON : Bit cho phép module A/D hoạt động.

1: Nguồn được cung cấp cho A/D

0: Ngưng cung cấp nguồn cho A/D

- Thanh ghi điều khiển ADCON1:

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Bit 7 ADFM: Bit lựa chọn định dạng kết quả A/D

1: Canh phải, 6 bit cao nhất của thanh ghi ADRESH có giá trị 0

0: Canh trái, 6 bit thấp nhất của thanh ghi ADRESL có giá trị 0

Bit 6 ADCS2: Bit lựa chọn clock chuyển đổi A/D

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

Bit 5,4 không sử dụng

Bit 3:0 **PCFG3:PCFG0**: Các bit điều khiển cấu hình các chân ADC

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	Vss	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	Vss	7/1
0010	D	D	D	A	A	A	A	A	VDD	Vss	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	Vss	4/1
0100	D	D	D	D	A	D	A	A	VDD	Vss	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	Vss	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	Vss	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	Vss	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	Vss	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

Chi tiết về các thanh ghi khác sẽ được trình bày cụ thể ở phụ lục trang 94.

Ví dụ:

Viết chương trình con đọc ADC từ ngõ RA1, Kết quả đọc về (8 bit được lưu trong ADRESH) $U_c = 5V$, tốc độ chuyển đổi 1Mhz, thạch anh 4Mhz.

DOC_ADC

BSF STATUS,5

BCF STATUS,6

;.....**Chọn số ngõ vào**.....

BCF ADCON1,3

BSF ADCON1,2

BCF ADCON1,1

BCF ADCON1,0

```

BCF STATUS,5
;.....Chọn ngõ vào.....
BCF ADCON0,5
BCF ADCON0,4
BSF ADCON0,3
;.....Chọn tần số lấy mẫu.....
BCF ADCON0,7
BCF ADCON0,6
BSF STATUS,5
BSF ADCON1,6
;.....Chọn nơi lưu kết quả.....
BCF ADCON1,7
BCF STATUS,5
; .....Cho phép bộ chuyển đổi ADC hoạt động.....
BSF ADCON0,0
BSF ADCON0,2
;.....Chuyển đổi xong chưa?.....
LOOP

BCF STATUS, 5
BTFSC ADCON0 ,2
GOTO LOOP

```

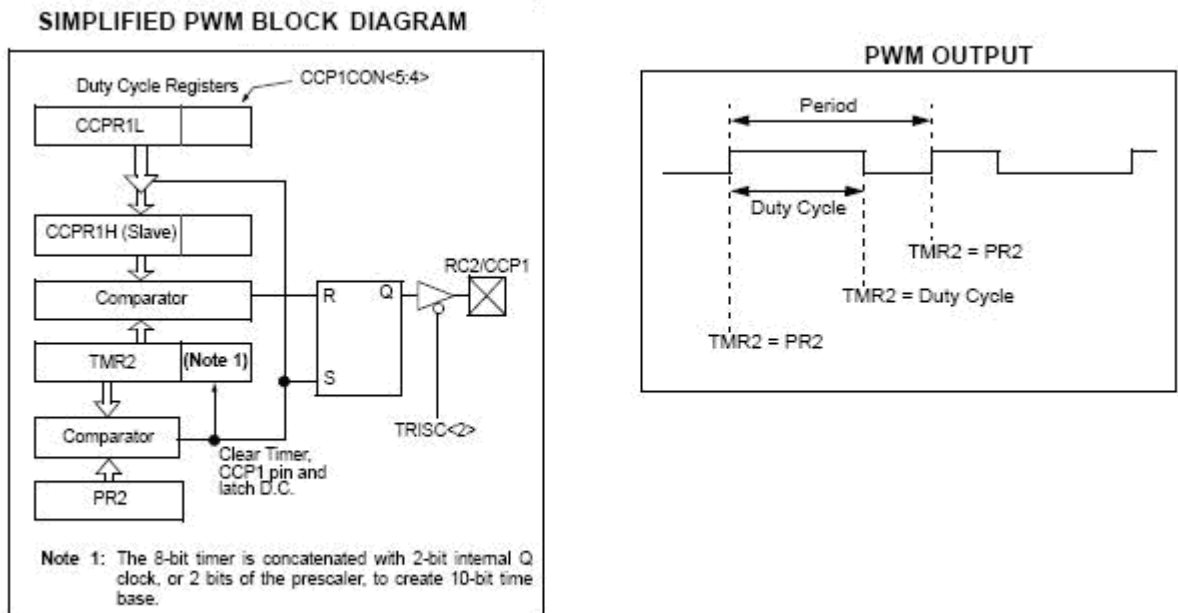
RETURN

4.3. PWM_ ĐIỀU CHẾ ĐỘ RỘNG XUNG

Khi hoạt động ở chế độ PWM (Pulse Width Modulation _ khối điều chế độ rộng xung), tín hiệu sau khi điều chế sẽ được đưa ra các pin của khối CCP (cần ấn định các pin này là output).

Các bước cài đặt bộ PWM:

1. Thiết lập thời gian của 1 chu kì của xung điều chế cho PWM (period) bằng cách đưa giá trị thích hợp vào thanh ghi PR2.
2. Thiết lập độ rộng xung cần điều chế (duty cycle) bằng cách đưa giá trị vào thanh ghi CCPRxL và các bit CCP1CON<5:4>..
3. Thiết lập giá trị bộ chia tần số prescaler của Timer2 và cho phép Timer2 hoạt động bằng cách đưa giá trị thích hợp vào thanh ghi T2CON.
4. Cho phép CCP hoạt động ở chế độ PWM



Hình 4.12: Sơ đồ của bộ PWM

Trong đó giá trị 1 chu kì (period) của xung điều chế được tính bằng công thức:

PWM period = [(PR2)+1]*4*T_{osc}*(giá trị bộ chia tần số của TMR2).

Bộ chia tần số prescaler của Timer2 chỉ có thể nhận các giá trị 1,4 hoặc 16 (xem lại Timer2 để biết thêm chi tiết). Khi giá trị thanh ghi PR2 bằng với giá trị thanh ghi TMR2 thì quá trình sau xảy ra:

Thanh ghi TMR2 tự động được xóa. Pin của khối CCP được set.

Giá trị thanh ghi CCPR1L (chứa giá trị ấn định độ rộng xung điều chế duty cycle) được đưa vào thanh ghi CCPRxH.

Độ rộng của xung điều chế (duty cycle) được tính theo công thức:

PWM duty cycle = (CCPRxL:CCPxCON<5:4>)*T_{osc}*(giá trị bộ chia tần số TMR2)

Như vậy 2 bit CCPxCON<5:4> sẽ chứa 2 bit LSB. Thanh ghi CCPRxL chứa byte cao của giá trị quyết định độ rộng xung. Thanh ghi CCPRxH đóng vai trò là buffer cho khối PWM. Khi giá trị trong thanh ghi CCPRxH bằng với giá trị trong thanh ghi TMR2 và hai bit CCPxCON<5:4> bằng với giá trị 2 bit của bộ chia tần số prescaler, pin của khối CCP lại được đưa về mức thấp, như vậy ta có được hình ảnh của xung điều chế tại ngõ ra của khối PWM như hình 4.12

Một số điểm cần chú ý khi sử dụng khối PWM:

Timer2 có hai bộ chia tần số prescaler và postscaler. **Tuy nhiên bộ postscaler không được sử dụng trong quá trình điều chế độ rộng xung của khối PWM.**

Nếu thời gian duty cycle dài hơn thời gian chu kì xung period thì xung ngõ ra tiếp tục được giữ ở mức cao sau khi giá trị PR2 bằng với giá trị TMR2.

Chú ý:

***Các thanh ghi liên quan:**

Thanh ghi PIR2: ãòa chæ 0Dh

TMR2 (địa chỉ 11h): chứa giá trị đếm của Timer2.

T2CON (địa chỉ 12h): xác lập các thông số cho Timer2

Các thanh ghi trên chi tiết xem bảng phụ lục trang 96

* Thanh ghi điều khiển bộ PWM

Thanh ghi CCP1CON và thanh ghi CCP2CON: ãòa chæ 17h (CCP1CON) và 1Dh (CCP2CON)

-	-	CCPXX	CCPXY	CCPxMP3	CCPxMP2	CCPxMP1	CCPxMP0
7							0

Bit 7,6 Không có tác dụng và mặc định mang giá trị 0.

Bit 5,4 CCPxX:CCPxY: PWM least Significant bits (các bit này không có tác dụng ở chế độ Capture và Compare). Ở chế độ PWM, đây là 2 bit MSB chứa giá trị tính độ rộng xung (duty cycle) của khối PWM (8 bit còn lại được chứa trong thanh ghi CCPxL).

Bit 3-0 CCPxM3:CCPxM0 CCPx Mode Select bit

Các bit dùng để xác lập các chế độ hoạt động của khối CCPx

0000 không cho phép CCPx (hoặc dùng để reset CCPx)

0100 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh xuống tại pin dùng cho khối CCPx.

0101 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh lên tại pin dùng cho khối CCPx.

0110 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh lên thứ 4 tại pin dùng cho khối CCPx.

0111 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh lên thứ 16 tại pin dùng cho khối CCPx.

1000 CCPx hoạt động ở chế độ Compare, ngõ ra được đưa lên mức cao và bit CCPxIF được set khi các giá trị cần so sánh bằng nhau.

1001 CCPx hoạt động ở chế độ Compare, ngõ ra được xuống mức thấp và bit CCPxIF được set khi các giá trị cần so sánh bằng nhau.

1010 CCPx hoạt động ở chế độ Compare, khi các giá trị cần so sánh bằng nhau, ngắt xảy ra, bit CCPxIF được set và trạng thái pin output không bị ảnh hưởng.

1011 CCPx hoạt động ở chế độ Compare, khi các giá trị cần so sánh bằng nhau, xung trigger đặc biệt (Trigger Special Event) sẽ được tạo ra, khi đó cờ ngắt CCPxIF được set, các pin output không thay đổi trạng thái, CCP1 reset Timer1, CCP2 reset Timer1 và khởi động khối ADC.

11xx CCPx hoạt động ở chế độ PWM.

Ví dụ1:

Viết chương trình xuất ra xung vuông tại chân RC2 có tần số 1Khz (thạch anh 4Mhz)

PWM period = [(PR2)+1]*4*T_{osc}*(giá trị bộ chia tần số của TMR2).

$$= 1000\mu S$$

Ta có: T_{osc} = 0.25 μS, chọn giá trị bộ chia tần số của TMR2 =1: 4

$$\Rightarrow PR2 = 250$$

PWM duty cycle = (CCPR1L:CCP1CON<5:4>)*T_{osc}*(giá trị bộ chia tần số TMR2)

$$= 500\mu S$$

$$\Rightarrow (CCPR1L:CCP1CON<5:4>) = 500 = B'0111110100'$$

$$\Rightarrow (CCP1CON<5:4>) = B'00'$$

$$\Rightarrow CCP1L = B'01111101'$$

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

```

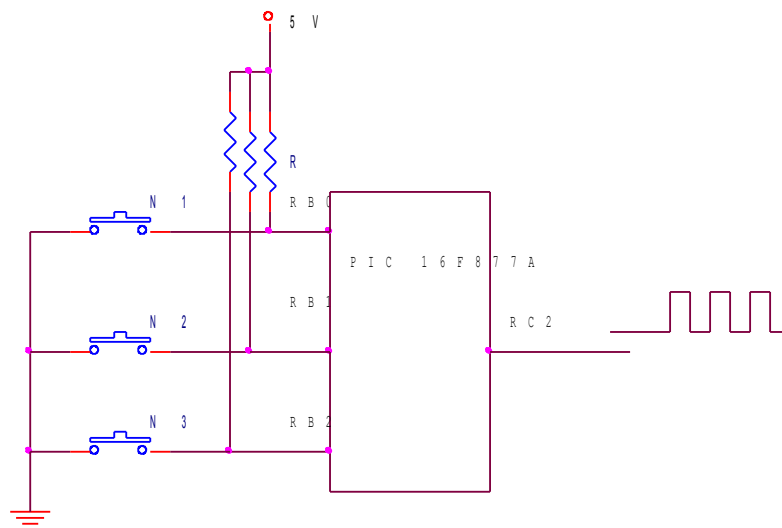
                BCF        STATUS,6
                BCF        STATUS,5
                CLRF       PORTC
;.....Bước 1.....
                BSF        STATUS,5
                BCF        TRISC,2
;.....Bước 2.....
                MOVLW      D'250'
                MOVWF      PR2
;.....Bước 3.....
                MOVLW      B'01111101'
                MOVWF      CCPR1L
                BCF        CCP1CON,5
                BCF        CCP1CON,4
;.....Bước4.....
                BCF        T2CON,1
                BSF        T2CON,0
                BSF        T2CON,2;
;.....Bước5.....
                BSF        CCP1CON,3
                BSF        CCP1CON,2
    
```

GOTO \$

END

BÀI TẬP THAM KHẢO CHƯƠNG 4**Bài 1:**

Viết chương trình điều khiển xung vuông tại chân RC2 như sau: Nếu nhấn nút N1 thì xuất ra chân RC2 tần số 1Khz, nhấn N2 thì xuất ra chân RC2 tần số 2Khz, nhấn N3 thì tần số xuất ra là 3Khz. (thạch anh 4Mhz).

**Hình 4.13**

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

DEM EQU 20H

DEM1 EQU 21H

ORG 0000H

BCF STATUS,6

BCF STATUS,5

CLRF PORTC

CLRF PORTB

BSF STATUS,5; BANK1

BCF TRISC,2

BSF TRISB,0

BSF TRISB,1

BSF TRISB,2

MAIN

BCF STATUS,5; BANK0

	BTFSS	PORTB,0	
	GOTO	KT_0	
	BTFSS	PORTB,1	
	GOTO	KT_1	
	BTFSS	PORTB,2	
	GOTO	KT_2	
	GOTO	MAIN	
KT_0			
	BTFSS	PORTB,0	
	GOTO	KT_0	
	MOVLW	D'250'	
	MOVWF	DEM	
	MOVLW	D'127'	
	MOVWF	DEM1	
	GOTO	TAO_XUNG	
KT_1			
	BTFSS	PORTB,1	
	GOTO	KT_1	
	MOVLW	D'125'	
	MOVWF	DEM	
	MOVLW	D'64'	
	MOVWF	DEM1	
	GOTO	TAO_XUNG	
KT_2			
	BTFSS	PORTB,2	
	GOTO	KT_2	
	MOVLW	D'63'	
	MOVWF	DEM	
	MOVLW	D'32'	
	MOVWF	DEM1	
	GOTO	TAO_XUNG	
TAO_XUNG			
	;..... BUOC2		
	BCF	STATUS,5;	BANK1
	MOVF	DEM,0	
	BSF	STATUS,5	
	MOVWF	PR2	

;.....**BUOC3**.....

BCF STATUS,5; BANK0

MOVF DEM1,0

MOVWF CCPR1L

BCF CCP1CON,5

BCF CCP1CON,4

;.....**BUOC4**.....

BCF T2CON,1

BSF T2CON,0

BSF T2CON,2; CHO TIMER2 HOAT DONG

;.....**BUOC5**.....

BSF CCP1CON,3

BSF CCP1CON,2

GOTO MAIN

END

Bài 2:

Viết chương trình đếm xung (từ chân RA4),sau mỗi giây rồi xuất ra portB

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

DEM EQU 22H

ORG 0000H

BCF STATUS,6

BCF STATUS,5

CLRF PORTB

BSF STATUS,5

CLRF TRISB

BCF STATUS,5

CLRF TMR0

BSF STATUS,5

MAIN

BSF OPTION_REG,4 ; **tac dong canh xuong**

BSF OPTION_REG,5

BSF OPTION_REG,3

LOOP

CALL DELAY_1S

BCF STATUS,5

MOVF TMR0,W

	MOVWF	PORTB
	CLRF	TMR0
	GOTO	LOOP
DELAY_1S		
	BCF	STATUS,5
	MOVLW	B'00100000'
	MOVWF	T1CON
	MOVLW	D'20'
	MOVWF	DEM
BATDAU		
	MOVLW	3CFH
	MOVWF	TMR1H
	MOVLW	0AFH
	MOVWF	TMR1L
	BSF	T1CON,0
LOOP1		
	BTFSS	PIR1,0
	GOTO	LOOP1
	BCF	PIR1,0
	DECFSZ	DEM,1
	GOTO	BATDAU
	RETURN	
	END	

Bài 3:

Viết chương trình đọc ADC từ ngõ RA0, Kết quả đọc về (8 bit cao được lưu trong ADRESH, $U_c = 4.5V$, tốc độ chuyển đổi 1Mhz, thạch anh 4Mhz) xuất ra portB.

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

	BCF	STATUS,6
	BCF	STATUS,5
	CLRF	PORTB
	BSF	STATUS,5
	CLRF	TRISB
MAIN		
	CALL	DOC_ADC
	MOVF	ADRESH,0

```

MOVWF    PORTB
GOTO     MAIN

DOC_ADC

BSF       STATUS,5
BCF       STATUS,6
;.....Chọn số ngõ vào.....
BCF       ADCON1,3
BCF       ADCON1,2
BCF       ADCON1,1
BSF       ADCON1,0
BCF       STATUS,5
;.....Chọn ngõ vào.....
BCF       ADCON0,5
BCF       ADCON0,4
BCF       ADCON0,3
;.....Chọn tần số lấy mẫu.....
BCF       ADCON0,7
BCF       ADCON0,6
BSF       STATUS,5
BSF       ADCON1,6
;.....Chọn nơi lưu kết quả.....
BCF       ADCON1,7
BCF       STATUS,5
; .....Cho phép bộ chuyển đổi ADC hoạt động.....
BSF       ADCON0,0
BSF       ADCON0,2
;.....Chuyển đổi xong chưa?.....
LOOP
BCF       STATUS, 5
BTFSC    ADCON0 ,2
GOTO     LOOP
RETURN
END

```

CHƯƠNG 5

CÁC CÔNG NỐI TIẾP

5.1. USART

USART (Universal Synchronous Asynchronous Receiver Transmitter) là một trong hai chuẩn giao tiếp nối tiếp. USART còn có thể gọi là giao diện giao tiếp nối tiếp nối tiếp SCI (Serial Communication Interface). Còn thể sử dụng giao diện này cho các giao tiếp với các thiết bị ngoại vi, với các vi xử lý khác nhau hay với máy tính. Các dạng của giao diện USART ngoại vi bao gồm:

- * Bất đồng bộ (Asynchronous).
- * Đồng bộ Master mode.
- * Đồng bộ Slave mode.

Hai pin dùng cho giao diện này là RC6/TX/CK và RC7/RX/DT, trong đó RC6/TX/CK dùng để truyền xung clock (baud rate) và RC7/RX/DT dùng để truyền data. Trong trường hợp này ta phải set bit TRISC<7:6> và SPEN (RCSTA<7>) để cho phép giao diện USART.

PIC16F877A có tích hợp sẵn bộ tạo tốc độ baud BRG (Baud Rate Generator) 8 bit dùng cho giao diện USART. BRG thực chất là một bộ đếm có thể có thể sử dụng cho cả hai dạng đồng bộ và bất đồng bộ và có thể nhiều khi cần phải thanh ghi PSBRG. Ở đây bất đồng bộ, BRG còn có thể nhiều khi cần phải bit BRGH (TXSTA<2>). Ở đây đồng bộ ta cần phải bit BRGH để có thể đi qua. Tốc độ baud do BRG tạo ra có thể tính theo công thức sau:

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = $F_{osc}/(64 (X + 1))$	Baud Rate = $F_{osc}/(16 (X + 1))$
1	(Synchronous) Baud Rate = $F_{osc}/(4 (X + 1))$	N/A

Trong đó X là giá trị của thanh ghi PSBRG (X là số nguyên và $0 < X < 255$).

Các thanh ghi liên quan đến BRG bao gồm:

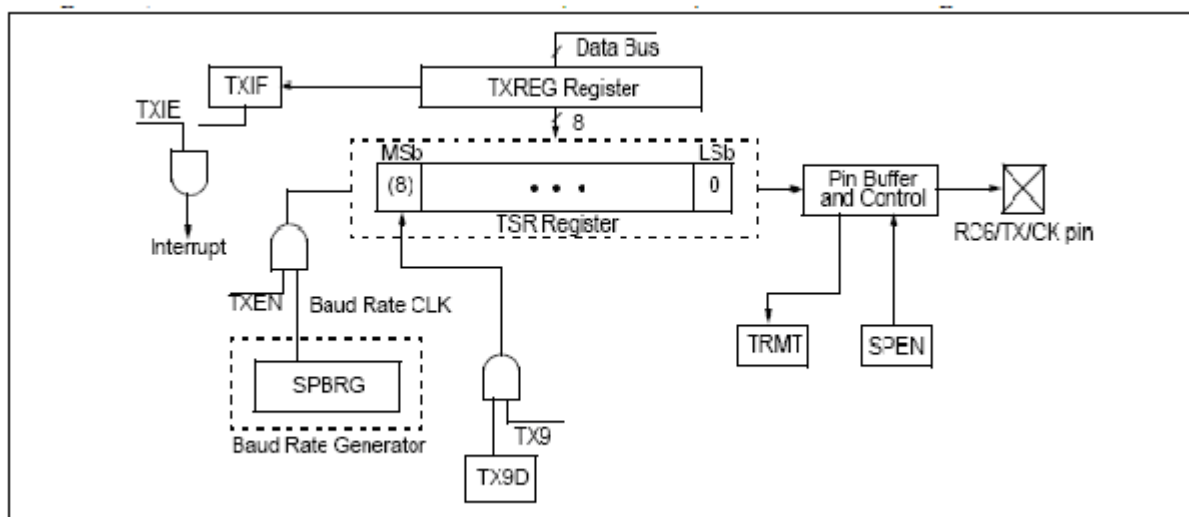
TXSTA (địa chỉ 98h): chọn chế độ đồng bộ hay bất đồng bộ (bit SYNC) và chọn mức tốc độ baud (bit BRGH).

RCSTA (địa chỉ 18h): cho phép hoạt động của giao tiếp (bit SPEN).

5.2. CHẾ ĐỘ LÀM VIỆC

5.2.1. TRUYỀN DỮ LIỆU BẤT ĐỒNG BỘ

Thành phần quan trọng nhất của khối truyền dữ liệu là thanh ghi dịch dữ liệu TSR (Transmit Shift Register). Thanh ghi TSR sẽ lấy dữ liệu từ thanh ghi đệm dùng cho quá trình truyền dữ liệu TXREG. Dữ liệu cần truyền phải được đưa trước vào thanh ghi TXREG. Ngay sau khi bit Stop của dữ liệu cần truyền trước đó được truyền xong, dữ liệu từ thanh ghi TXREG sẽ được đưa vào thanh ghi TSR, thanh ghi TXREG bị rỗng, ngắt xảy ra và cờ hiệu TXIF (PIR1<4>) được set. Ngắt này được điều khiển bởi bit TXIE (PIE1<4>). Cờ hiệu TXIF vẫn được set bất chấp trạng thái của bit TXIE hay tác động của chương trình (không thể xóa TXIF bằng chương trình) mà chỉ reset về 0 khi có dữ liệu mới được đưa vào thanh ghi TXREG.



Hình 5.1: Sơ đồ của bộ truyền nối tiếp

Trong khi cờ hiệu TXIF đóng vai trò chỉ thị trạng thái thanh ghi TXREG thì cờ hiệu TRMT (TXSTA<1>) có nhiệm vụ thể hiện trạng thái thanh ghi TSR. Khi thanh ghi TSR rỗng, bit TRMT sẽ được set. Bit này chỉ đọc và không có ngắt nào được gắn với trạng thái của nó. Một điểm cần chú ý nữa là thanh ghi TSR không có trong bộ nhớ dữ liệu và chỉ được điều khiển bởi CPU.

Khởi truyền dữ liệu được cho phép hoạt động khi bit TXEN (TXSTA<5>) được set. Quá trình truyền dữ liệu chỉ thực sự bắt đầu khi đã có dữ liệu trong thanh ghi TXREG và xung truyền baud được tạo ra. Khi khởi truyền dữ liệu được khởi động lần đầu tiên, thanh ghi TSR rỗng. Tại thời điểm đó, dữ liệu đưa vào thanh ghi TXREG ngay lập tức được load vào thanh ghi TSR và thanh ghi TXREG bị rỗng. Lúc này ta có thể hình thành một chuỗi dữ liệu liên tục cho quá trình truyền dữ liệu. Trong quá trình truyền dữ liệu nếu bit TXEN

bị reset về 0, quá trình truyền kết thúc, khối truyền dữ liệu được reset và pin RC6/TX/CK chuyển đến trạng thái high-impedance.

Trong trường hợp dữ liệu cần truyền là 9 bit, bit TX9 (TXSTA<6>) được set và bit dữ liệu thứ 9 sẽ được lưu trong bit TX9D (TXSTA<0>). Nên ghi bit dữ liệu thứ 9 vào trước, vì khi ghi 8 bit dữ liệu vào thanh ghi TXREG trước có thể xảy ra trường hợp nội dung thanh ghi TXREG sẽ được load vào thanh ghi TSG trước, như vậy dữ liệu truyền đi sẽ bị sai khác so với yêu cầu.

Tóm lại, để truyền dữ liệu theo giao diện USART bất đồng bộ, ta cần thực hiện tuần tự các bước sau:

1. Tạo xung truyền baud bằng cách đưa các giá trị cần thiết vào thanh ghi SPBRG và bit điều khiển mức tốc độ baud BRGH.
2. Cho phép cổng giao diện nối tiếp nối tiếp bất đồng bộ bằng cách clear bit SYNC và set bit PSEN.
3. Đưa 8 bit dữ liệu cần truyền vào thanh ghi TXREG.
4. Set bit TXIE nếu cần sử dụng ngắt truyền.
5. Set bit TX9 nếu định dạng dữ liệu cần truyền là 9 bit.
6. Set bit TXEN để cho phép truyền dữ liệu (lúc này bit TXIF cũng sẽ được set).
7. Nếu định dạng dữ liệu là 9 bit, đưa bit dữ liệu thứ 9 vào bit TX9D.
8. Nếu sử dụng ngắt truyền, cần kiểm tra lại các bit GIE và PEIE (thanh ghi INTCON)

Chú ý:

*** Các thanh ghi liên quan:**

Thanh ghi INTCON (địa chỉ 0Bh, 8Bh, 10Bh, 18Bh): cho phép tắt cả các ngắt.

Thanh ghi PIE1 (địa chỉ 8Ch): chứa bit cho phép ngắt truyền TXIE.

Thanh ghi RCSTA (địa chỉ 18h): chứa bit cho phép cổng truyền dữ

Thanh ghi TXREG (địa chỉ 19h): thanh ghi chứa dữ liệu cần truyền.

Thanh ghi SPBRG (địa chỉ 99h): quyết định tốc độ baud.

Chi tiết xem bảng phụ lục trang 96

*** Thanh ghi điều khiển bộ truyền nối tiếp:**

Thanh ghi PIR1 (địa chỉ 0Ch): chứa cờ hiệu TXIF

Thanh ghi TXSTA (địa chỉ 98h): Thanh ghi chứa các bit trạng thái và điều khiển việc truyền dữ liệu thông qua chuẩn giao tiếp USART.

CSRC	TX-9	TXEN	SYNC	-	BRGH	TRMT	TX9D
Bit 7							Bit 0

Bit 7 CSRC Clock Source Select bit

Ở chế độ bất đồng bộ: không cần quan tâm. Ở chế độ đồng bộ:

CSRC = 1 Master mode (xung clock được lấy từ bộ tạo xung BRG).

CSRC = 0 Slave mode (xung clock được nhận từ bên ngoài).

Bit 6 TX-9 9-bit Transmit Enable bit

TX-9 = 1 truyền dữ liệu 9 bit.

TX-9 = 0 truyền dữ liệu 8 bit.

Bit 5 TXEN Transmit Enable bit

TXEN = 1 cho phép truyền.

TXEN = 0 không cho phép truyền.

Bit 4 SYNC USART Mode Select bit

SYNC = 1 dạng đồng bộ

SYNC = 0 dạng bất đồng bộ.

Bit 3 Không cần quan tâm và mặc định mang giá trị 0.

Bit 2 BRGH High Baud Rate Select bit, Bit này chỉ có tác dụng ở chế độ bất đồng bộ.

BRGH = 1 tốc độ cao.

BRGL = 0 tốc độ thấp.

Bit 1 TRMT Transmit Shift Register Status bit

TRMT = 1 thanh ghi TSR không có dữ liệu.

TRMT = 0 thanh ghi TSR có chứa dữ liệu.

Bit 0 TX9D

Bit này chứa bit dữ liệu thứ 9 khi dữ liệu truyền nhận là 9 bit.

Ví dụ:

Viết đoạn chương trình nhận dữ liệu từ PORTB sau đó xuất ra Port nối tiếp, chế độ 8 bit,

Baud rate = 9600, Fosc = 4Mhz.

TRUYEN_NOI_TIEP

;**.....Chọn baud rate.....**

BSF TXSTA, BRGH

MOVLW D'25'

MOVWF SPBRG

;**.....Cho phép cổng nối tiếp hoạt động.....**

BCF TXSTA, SYNC

BSF RCSTA, SPEN

;**.....Xuất giá trị cần truyền vào thanh ghi TXREG....**

LOOP

MOVF PORTB, W

MOVWF TXREG

;**.....Cho phép truyền.....**

BSF TXSTA, TXEN

;.....Truyền xong chưa?.....

LOOP1

BTFSS PIR1,TXIF

GOTO LOOP1

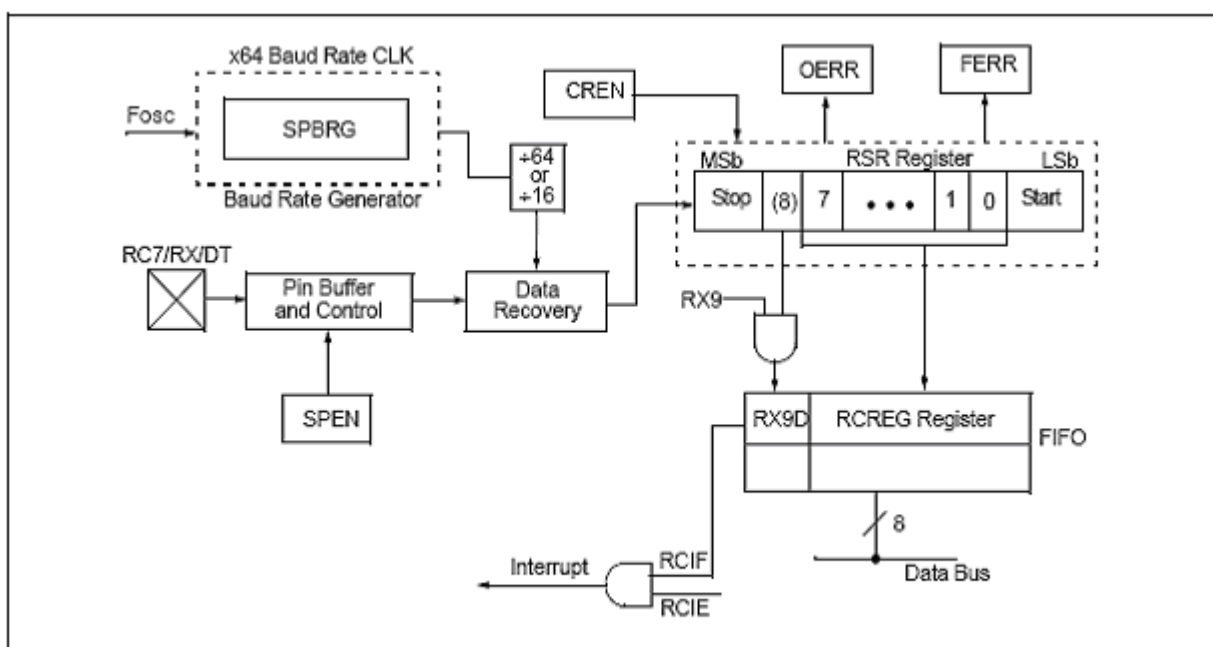
NOP

GOTO LOOP

RETURN

5.2.2. NHẬN DỮ LIỆU BẤT ĐỒNG BỘ

Dữ liệu được đưa vào từ chân RC7/RX/DT sẽ kích hoạt khối phục hồi dữ liệu. Khối phục hồi dữ liệu thực chất là một bộ dịch dữ liệu tốc độ cao và có tần số hoạt động gấp 16 lần hoặc 64 lần tần số baud. Trong khi đó tốc độ dịch của thanh ghi nhận dữ liệu sẽ bằng với tần số baud hoặc tần số của oscillator.



Hình 5.2: Sơ đồ của bộ truyền nối tiếp

Bit điều khiển cho phép khởi nhận dữ liệu là bit RCEN (RCSTA<4>). Thành phần quan trọng nhất của khối nhận dữ liệu là thanh ghi nhận dữ liệu RSR (Receive Shift Register). Sau khi nhận diện bit Stop của dữ liệu truyền tới, dữ liệu nhận được trong thanh ghi RSR sẽ được đưa vào thanh ghi RCGER, sau đó cờ hiệu RCIF (PIR1<5>) sẽ được set và ngắt nhận được kích hoạt. Ngắt này được điều khiển bởi bit RCIE (PIE1<5>). Bit cờ hiệu RCIF là bit chỉ đọc và không thể được tác động bởi chương trình. RCIF chỉ reset về 0 khi dữ liệu nhận vào ở thanh ghi RCREG đã được đọc và khi đó thanh ghi RCREG rỗng. Thanh ghi RCREG là thanh ghi có bộ đệm kép (double-buffered register) và hoạt động theo cơ chế FIFO (First In First Out) cho phép nhận 2 byte và byte thứ 3 tiếp tục được đưa vào thanh ghi RSR. Nếu sau khi nhận được bit Stop của byte dữ liệu thứ 3 mà thanh ghi

RCREG vẫn còn đầy, cờ hiệu báo tràn dữ liệu (Overrun Error bit) OERR(RCSTA<1>) sẽ được set, dữ liệu trong thanh ghi RSR sẽ bị mất đi và quá trình đưa dữ liệu từ thanh ghi RSR vào thanh ghi RCREG sẽ bị gián đoạn. Trong trường hợp này cần lấy hết dữ liệu ở thanh ghi RSREG vào trước khi tiếp tục nhận byte dữ liệu tiếp theo. Bit OERR phải được xóa bằng phần mềm và thực hiện bằng cách clear bit RCEN rồi set lại. Bit FERR(RCSTA<2>) sẽ được set khi phát hiện bit Stop của dữ liệu được nhận vào. Bit dữ liệu thứ 9 sẽ được đưa vào bit RX9D(RCSTA<0>). Khi đọc dữ liệu từ thanh ghi RCREG, hai bit FERR và RX9D sẽ nhận các giá trị mới. Do đó cần đọc dữ liệu từ thanh ghi RCSTA trước khi đọc dữ liệu từ thanh ghi RCREG để tránh bị mất dữ liệu.

Tóm lại, khi sử dụng giao diện nhận dữ liệu USART bất đồng bộ cần tiến hành tuần tự các bước sau:

1. Thiết lập tốc độ baud (đưa giá trị thích hợp vào thanh ghi SPBRG và bit BRGH).
2. Cho phép cổng giao tiếp USART bất đồng bộ (clear bit SYNC và set bit SPEN).
3. Nếu cần sử dụng ngắt nhận dữ liệu, set bit RCIE.
4. Nếu dữ liệu truyền nhận có định dạng là 9 bit, set bit RX9.
5. Cho phép nhận dữ liệu bằng cách set bit CREN.
6. Sau khi dữ liệu được nhận, bit RCIF sẽ được set và ngắt được kích hoạt (nếu bit RCIE được set).
7. Đọc giá trị thanh ghi RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra xem quá trình nhận dữ liệu có bị lỗi không.
8. Đọc 8 bit dữ liệu từ thanh ghi RCREG.
9. Nếu quá trình truyền nhận có lỗi xảy ra, xóa lỗi bằng cách xóa bit CREN.
10. Nếu sử dụng ngắt nhận cần set bit GIE và PEIE (thanh ghi INTCON).

Chú ý:

*** Các thanh ghi liên quan:**

Thanh ghi INTCON (địa chỉ 0Bh, 8Bh, 10Bh, 18Bh): chứa các bit cho phép ngắt

Thanh ghi PIE1 (địa chỉ 8Ch): chứa bit cho phép ngắt RCIE.

Thanh ghi RCREG (địa chỉ 1Ah): chứa dữ liệu nhận được.

Thanh ghi TXSTA (địa chỉ 98h): chứa các bit điều khiển SYNC và BRGH.

Thanh ghi SPBRG (địa chỉ 99h): điều khiển tốc độ baud.

Chi tiết các thanh ghi xem bảng phụ lục trang 96

*** Thanh ghi điều khiển bộ nhận nối tiếp:**

Thanh ghi PIR1 (địa chỉ 0Ch): chứa cờ hiệu RCIE.

Thanh ghi RCSTA: (địa chỉ 18h) Thanh ghi chứa các bit trạng thái và các bit điều khiển quá trình nhận dữ liệu qua chuẩn giao tiếp USART.

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
Bit 7				Bit 0			

Bit 7 SPEN Serial Port Enable bit

SPEN = 1 Cho phép cổng giao tiếp USART (pin RC7/RX/DT và RC6/TX/CK).

SPEN = 0 không cho phép cổng giao tiếp USART.

Bit 6 RX9 9-bit Receive Enable bit

RX9 = 1 nhận 9 bit dữ liệu.

RX9 = 0 nhận 8 bit dữ liệu.

Bit 5 SREN Single Receive Enable bit

Ở chế độ USART bất đồng bộ: bit này không cần quan tâm.

Ở chế độ USART Master đồng bộ:

SREN = 1 cho phép chức năng nhận 1 byte dữ liệu (8 bit hoặc 9 bit).

SREN = 0 không cho phép chức năng nhận 1 byte dữ liệu.

Bit 4 CREN Continuous Receive Enable bit

Ở chế độ bất đồng bộ:

CREN = 1 cho phép nhận 1 chuỗi dữ liệu liên tục.

CREN = 0 không cho phép nhận 1 chuỗi dữ liệu liên tục.

Ở chế độ đồng bộ:

CREN = 1 cho phép nhận dữ liệu cho tới khi xóa bit CREN.

CREN = 0 không cho phép nhận chuỗi dữ liệu.

Bit 3 ADDEN Address Detect Enable bit

Ở chế độ USART bất đồng bộ 9 bit

ADDEN = 1 cho phép xác nhận địa chỉ, khi bit RSR<8> được set thì ngắt được cho phép thực thi và giá trị trong buffer được nhận vào.

ADDEN = 0 không cho phép xác nhận địa chỉ, các byte dữ liệu được nhận vào và bit thứ 9 có thể được sử dụng như là bit parity.

Bit 2 FERR Framing Error bit

FERR = 1 xuất hiện lỗi "Framing" trong quá trình truyền nhận dữ liệu.

FERR = 0 không xuất hiện lỗi "Framing" trong quá trình truyền nhận dữ liệu.

Bit 1 OERR Overrun Error bit,

OERR = 1 xuất hiện lỗi "Overrun"

OERR = 0 không xuất hiện lỗi "Overrun"

Bit 0 RX9D

Bit này chứa bit dữ liệu thứ 9 của dữ liệu truyền nhận

Ví dụ:

Viết đoạn chương trình đọc dữ liệu từ Port nối tiếp, sau đó xuất giá trị đọc về ra PORTD

Baud rate = 9600, Fosc = 4Mhz.

```

;..... Chọn baud rate .....
BSF          TXSTA, BRGH
MOVLW        D'25'
MOVWF        SPBRG
;..... Cho phép cổng nối tiếp hoạt động .....
BCF          TXSTA, SYNC
BSF          RCSTA, SPEN
;.....Cho phép nhận dữ liệu.....

BAT_DAU

BSF          RCSTA,CREN
;.....Nhận xong chưa?.....

LOOP

BTFSS        PIR1, RCIF
GOTO         LOOP
;.....Có lỗi không?.....
BTFSS        RCSTA, OERR
GOTO         DOC_VE
BCF          RCSTA, CREN
GOTO         BAT_DAU
;.....Đọc kết quả về.....

DOC_VE

MOVF         RCREG,W
MOVWF        PORTD
GOTO         LOOP
RETURN

```

BÀI TẬP THAM KHẢO CHƯƠNG 5

Bài 1:

Viết chương trình đọc dữ liệu từ Port nối tiếp (Baud rate =9600, chế độ 8 bit, thạch anh = 4Mhz), kiểm tra kết quả đọc về. Nếu là số lẻ thì xuất ra PortB, ngược lại không xuất.

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

TEMP EQU 20H

ORG 0000H

```

BCF          STATUS,6
BCF          STATUS,5
CLRF         PORTB
CLRF         PORTC

```

```

BSF      STATUS,5
CLRF     TRISB
BCF      TRISC,6
BCF      TRIC,7

MAIN
CALL     DOC_NOI_TIEP
BTFSS    TEMP,0
GOTO     LOOP1
MOVF     TEMP,0
MOVWF    PORTB
GOTO     MAIN

LOOP1
CLRF     RCREG
GOTO     MAIN

DOC_NOI_TIEP
;..... Chọn baud rate .....
BSF      TXSTA, BRGH
MOVLW    D'25'
MOVWF    SPBRG
;..... Cho phép cổng nối tiếp hoạt động .....
BCF      TXSTA, SYNC
BSF      RCSTA, SPEN
;.....Cho phép nhận dữ liệu.....

BAT_DAU
BSF      RCSTA,CREN
;.....Nhận xong chưa?.....

LOOP
BTFSS    PIR1, RCIF
GOTO     LOOP
;.....Có lỗi không?.....
BTFSS    RCSTA, OERR
GOTO     DOC_VE
BCF      RCSTA, CREN
GOTO     BAT_DAU
;.....Đọc kết quả về.....

DOC_VE
MOVF     RCREG,W

```

```
MOVWF    TEMP
RETURN
END
```

Bài 2:

Viết đoạn chương trình xuất 30byte trong Ram nội, byte đầu tiên có địa chỉ 22H ra Port nối tiếp (Baud rate =9600, chế độ 8 bit, thạch anh = 4Mhz)

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

DEM EQU 20H

TEMP EQU 21H

ORG 0000H

```
BCF      STATUS,6
BCF      STATUS,5
CLRF     PORTB
CLRF     PORTC
BSF      STATUS,5
BCF      TRISC,6
BCF      TRIC,7
BCF      STATUS,5
```

MAIN

```
MOVLW    D'30'
MOVWF    DEM
MOVLW    22H
MOWF     FSR
```

BAT_DAU

```
MOVF     INDF,0
MOVWF    TEMP
CALL     TRUYEN_NOI_TIEP
DECFSZ   DEM
GOTO     TIEP
GOTO     THOAT
```

TIEP

```
INCF     FSR,1
GOTO     MAIN
```

THOAT

```
NOP
GOTO     $
```

TRUYEN_NOI_TIEP

;.....Chọn baud rate.....

```

BSF          TXSTA, BRGH
MOVLW       D'25'
MOVWF       SPBRG
;.....Cho phép cổng nối tiếp hoạt động.....
BCF          TXSTA, SYNC
BSF          RCSTA, SPEN
;...Xuất giá trị cần truyền vào thanh ghi RCREG...
MOVF        TEMP,0
MOVWF       TXREG
;.....Cho phép truyền.....
BSF          TXSTA, TXEN
;.....Truyền xong chưa?.....
LOOP1      BTFSS      PIR1, TXIF
           GOTO       LOOP1
           NOP
           RETURN
           END

```

Bài 3:

Viết chương trình xuất kiểm tra trong Ram nội (bank0) nếu ký tự chữ hoa (A, B,C...Z) thì xuất ra Port nối tiếp (Baud rate =19200, chế độ 8 bit, thạch anh = 4Mhz).Ngược lại, không xuất. Biết chữ A có mã ASCH là 65.

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

TEMP EQU A1H

DEM EQU A0H

ORG 0000H

```

           BCF        STATUS,6
           BCF        STATUS,5
           CLRF       PORTB
           CLRF       PORTC
           BSF        STATUS,5
           CLRF       TRISB
           BSF        TRISC,6
           BSF        TRIC,7

MAIN      MOVLW       D'97'
           MOVWF      DEM
           MOVLW      20H

```

```

                                MOVWF    FSR

LOOP    DECFSZ    DEM
        GOTO     BAT_DAU
        GOTO     $

BAT_DAU
        MOVF     INDF,0
        MOVWF    TEMP
        MOVLW    D'65'
        SUBWF    TEMP,0
        BTFSS    STATUS,0
        GOTO     TIEP
        INCF     FSR,1
        GOTO     LOOP

TIEP
        MOVLW    D'92'
        SUBWF    TEMP,0
        BTFSS    STATUS,0
        GOTO     BO_QUA
        GOTO     XUAT

BO_QUA
        INCF     FSR,1
        GOTO     LOOP

XUAT
        BCF      TXSTA, BRGH
        MOVLW    D'12 '
        MOVWF    SPBRG
        ;.....Cho phép cổng nối tiếp hoạt động.....
        BCF      TXSTA, SYNC
        BSF      RCSTA, SPEN
        ;.....Xuất giá trị cần truyền vào thanh ghi RCREG...
        MOVF     TEMP,0
        MOVWF    TXREG
        ;.....Cho phép truyền.....
        BSF      TXSTA, TXEN
        ;.....Truyền xong chưa?.....

LOOP1   BTFSS    PIR1,TXIF

```

	GOTO	LOOP1
	INCF	FSR,1
	GOTO	LOOP
END		

CHƯƠNG 6

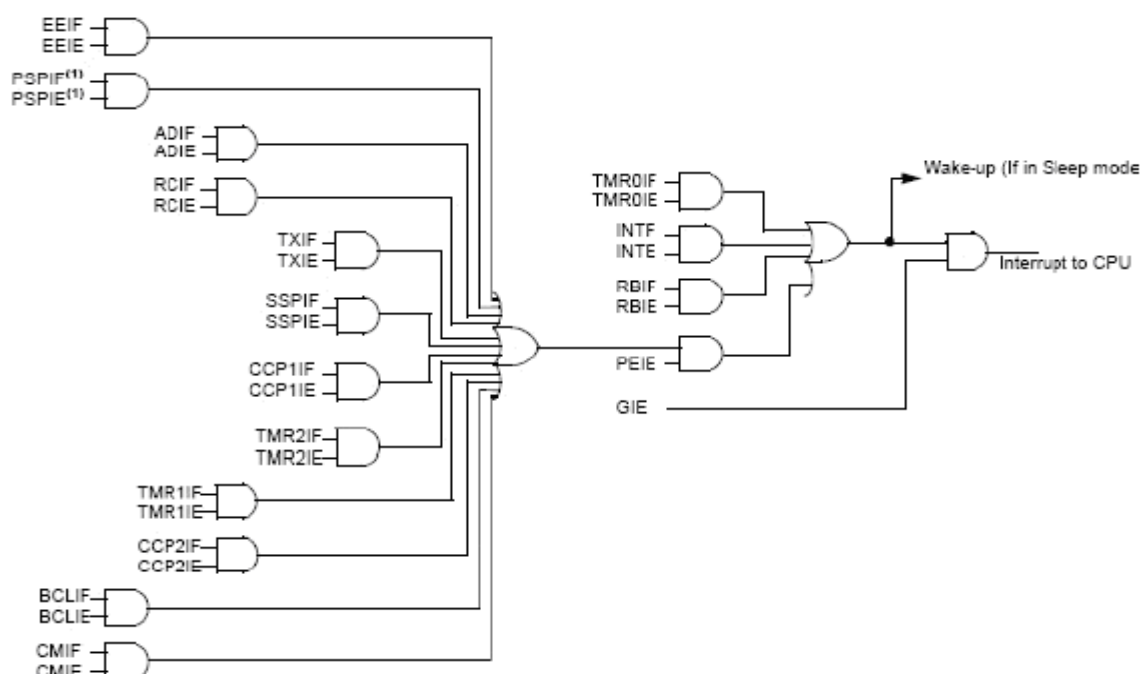
NGẮT - INTERRUPT

6.1 KHÁI NIỆM

6.1.1. GIỚI THIỆU

Đầu tiên, Ngắt (interrupt) là cái gì?, nó thật sự có ý nghĩa giống như tên gọi của nó, **một Interrupt là một tác vụ xử lý hay là một tín hiệu xử lý mà nó có thể bắt con Pic dừng lại những gì đang làm để làm một công việc khác.** Một ví dụ dễ hiểu, hãy lấy sinh hoạt hàng ngày của bạn, giả sử bạn đang ngồi ở nhà, rồi bạn đang tán gẫu với ai đó, thành linh chuông điện thoại reo, bạn ngưng cuộc nói chuyện lại, nhấc điện thoại lên và nói chuyện với người gọi đến. Khi bạn kết thúc cuộc nói chuyện bằng điện thoại bạn lại quay trở về và tiếp tục tán gẫu với người đã nói chuyện với bạn trước khi điện thoại reo. Bây giờ bạn hãy tưởng tượng, chương trình chính là quá trình tán gẫu của bạn với người bạn ngồi ở nhà, điện thoại reo tạo ra một Interrupt và thủ tục (routine) Interrupts là cuộc nói chuyện với người ở đầu dây bên kia, khi kết thúc cuộc nói chuyện bằng điện thoại bạn quay về “chương trình chính” để tiếp tục tán gẫu.

Ví dụ này giải thích chính xác một Interrupts tạo ra một tiến trình xử lý như thế nào. Một chương trình chính đang chạy, thực hiện một vài chức năng nào đó trên mạch điện, nhưng khi Interrupt xảy ra chương trình chính sẽ tạm ngưng và ngay lúc đó một thủ tục khác được thực hiện, khi thủ tục này kết thúc con Pic sẽ lại quay về chương trình chính. Con Pic có 15 nguồn ngắt, khi ngắt được xảy ra cần: **khai báo ngắt (Set các bit điều khiển IE tương ứng) và có cờ ngắt tác động (IF),** để biết ngắt như thế nào chúng ta cần xem sơ đồ sau:



Hình 6.1: Giải đồ ngắt

Chú thích:

- Các Bit điều khiển ngắt

+ Bit GIE: INTCON<7> Cho phép ngắt toàn cục

- + Bít PEIE: INTCON<6> Cho phép ngắt ngoại vi
- + Bít RBIE: INTCON<3> Cho phép ngắt PortB
- + Bít INTE: INTCON<4> Cho phép ngắt RB0
- + Bít TMR0IE: INTCON<3> Cho phép ngắt Timer0
- + Bít FEIE: PIE2<4> Cho phép ngắt EFROM
- + Bít PSPIE: PIE1<7> Cho phép ngắt truyền song song
- + Bít ADIE: PIE1<6> Cho phép ngắt chuyển đổi ADC
- + Bít RCIE: PIE1<5> Cho phép ngắt nhận nối tiếp
- + Bít TXIE: PIE1<4> Cho phép ngắt truyền nối tiếp
- + Bít SSPIE: PIE1<3> Cho phép ngắt truyền nhận nối tiếp đang bận
- + Bít CCP1IE: PIE1<2> Cho phép ngắt bộ CCP1
- + Bít TMR1IE: PIE1<0> Cho phép ngắt Timer1
- + Bít TMR2IE: PIE1<1> Cho phép ngắt Timer2
- + Bít CCP2IE: PIE2<0> Cho phép ngắt bộ CCP2
- + Bít BCLIE: PIE2<3> Cho phép ngắt truyền nhận nối xảy ra
- + Bít CMIE: PIE2<6> Cho phép ngắt bộ so sánh

• **Các Bít cờ ngắt**

- + Bít RBIF: INTCON<0> Cờ ngắt PortB
- + Bít INTF: INTCON<1> Cờ ngắt RB0
- + Bít TMR0IF: INTCON<2> Cờ ngắt Timer0
- + Bít FEIF: PIR2<4> Cờ ngắt EFROM
- + Bít PSPIF: PIR1<7> Cờ ngắt truyền song song
- + Bít ADIF: PIR1<6> Cờ ngắt chuyển đổi ADC
- + Bít RCIF: PIR1<5> Cờ ngắt nhận nối tiếp
- + Bít TXIF: PIR1<4> Cờ ngắt truyền nối tiếp
- + Bít SSPIF: PIR1<3> Cờ ngắt truyền nhận nối tiếp đang bận
- + Bít CCP1IF: PIR1<2> Cờ ngắt bộ CCP1
- + Bít TMR1IF: PIR1<0> Cờ ngắt Timer1
- + Bít TMR2IF: PIR1<1> Cờ ngắt Timer2
- + Bít CCP2IF: PIR2<0> Cờ ngắt bộ CCP2
- + Bít BCLIF: PIR2<3> Cờ ngắt truyền nhận nối xảy ra
- + Bít CMIF: PIR2<6> Cờ ngắt bộ so sánh

Giả sử trong chương trình chính chúng ta có sử dụng Timer0. khi chương trình đang thực thi, nếu bộ định thời Timer0 đếm xong sẽ báo cho chúng ta biết thông qua cờ tràn TR0IF. Vậy nếu chúng ta dùng ngắt Timer0 thì chúng ta cần SET bit TMR0IE và SET bit GIE. Tương tự nếu chúng ta cần dùng ngắt ngoại vi RB thì cần SET bit RBIE và

SET bit GIE. Tuy nhiên, chúng ta cần ngắt Timer1 thì ngoài việc cần SET bit TMR1IE và SET bit GIE và cần SET thêm bit PEIE

=> **Dựa vào sơ đồ trên chúng ta cần “Set” bit nào khi khai báo ngắt.**

6.1.2. CẤU TRÚC CHƯƠNG TRÌNH CÓ DỪNG NGẮT

ORG 0000H ; Địa chỉ RESET

GOTO MAIN ; Nhảy vào chương trình chính

;.....**INTERRUPT ROUTINE**.....

ORG 04H ; Địa chỉ vector ngắt

- Lưu các giá trị tạm thời vào Ram nội (Nếu các giá trị này thay đổi khi thực thi chương trình ngắt)
- Thực thi chương trình ngắt
- Thoát ngắt
 - Trả các giá trị từ Ram nội vào các thanh ghi đã lưu
 - Xóa cờ ngắt

RETFIE ; kết thúc chương trình ngắt

;..... **MAIN PROGRAM**.....

MAIN

- Khai báo ngắt (Chúng ta khai báo là ngắt gì? Ngắt ngoại vi, ngắt timer hay ngắt ADC....)
- Thực thi chương trình chính
- Vòng lặp vô hạn

END

Ở đoạn chương trình trên ta thấy, khi bắt đầu ngay điểm nhập RESET gặp lệnh Goto MAIN khi đó chương trình thực thi chương trình chính. Trong suốt quá trình thực thi nếu có ngắt xảy ra (giống như bạn đang tán gẫu với ai đó,thình lình chương điện thoại reo) thì chương trình lập tức “tạm ngừng” trở về địa chỉ vector ngắt ORG 04H để thực thi chương trình ngắt (bạn ngưng cuộc nói chuyện lại, nhặt điện thoại lên và nói chuyện với người gọi đến). Khi kết thúc chương trình ngắt gặp lệnh RETFIE thì chương trình trở lại nơi nó đã “ra đi” (giống như bạn nghe điện thoại xong trở lại với tán gẫu tiếp câu chuyện còn dang dở).

6.2 NGẮT RB0

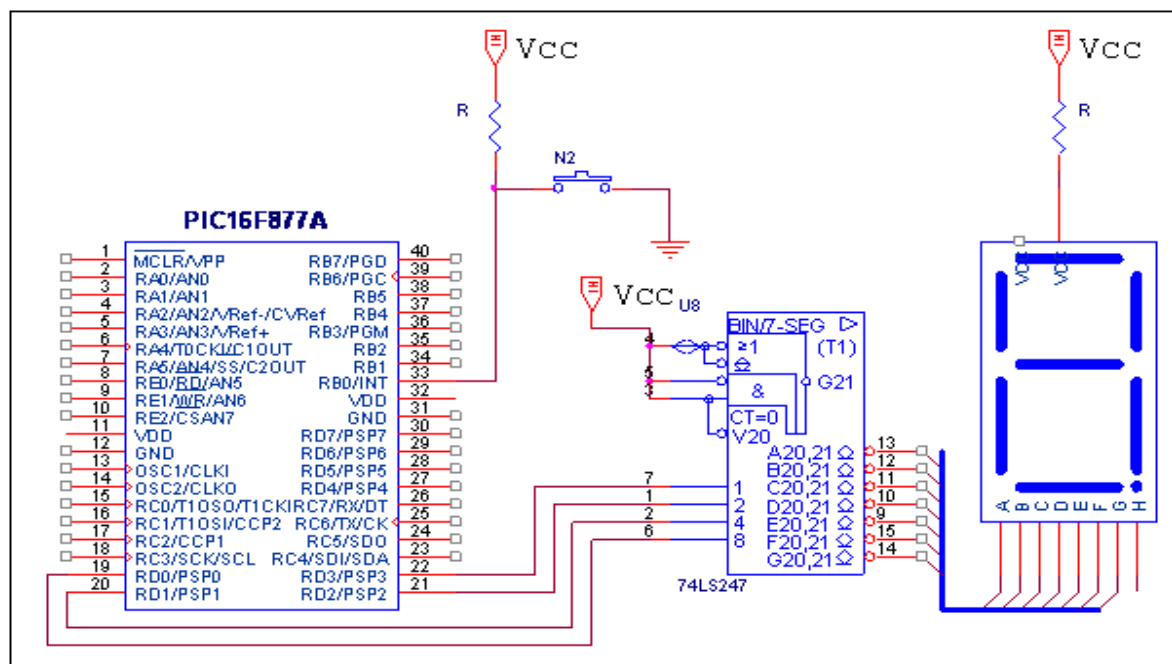
Ngắt này dựa trên sự thay đổi trạng thái của pin RB0/INT. Cạnh tác động gây ra ngắt có thể là cạnh lên hay cạnh xuống và được điều khiển bởi bit INTEDG (thanh ghi OPTION_REG <6>). Khi có cạnh tác động thích hợp xuất hiện tại pin RB0/INT, cờ ngắt INTF được set bất chấp trạng thái các bit điều khiển GIE và PEIE.

Thanh ghi OPTION chính là thanh ghi thiết lập chế độ cho Interrupt tích cực ở cạnh lên hay cạnh xuống của tín hiệu vào, Bit 6 của thanh ghi OPTION được gọi là INTEDG, nếu Set Bit6 sẽ thiết lập interrupt tích cực ở cạnh lên của tín hiệu vào (trạng thái

default), nếu Clear Bit6 sẽ thiết lập interrupt tích cực ở cạnh xuống của tín hiệu vào. Mặc nhiên sau khi bật nguồn con Pic sẽ thiết lập chế độ Interrupt cạnh lên, có nghĩa là interrupt xảy ra khi tín hiệu vào thay đổi từ thấp lên cao (cạnh lên).

Ví dụ:

Cho sơ đồ như hình vẽ, viết chương trình cho thỏa: Mỗi lần nhấn nút N, buông ra giá trị trên Led 7 đoạn tăng 1 đơn vị



Hình 6.2

```

PROCESSOR 16F877A
#include <P16F877A.INC>
DEM EQU 20H
ORG 0000H

        GOTO    MAIN

ORG 04H

        MOVWF   TEMP
        INCF    DEM,1
        MOVLW   D'10'
        XORWF   DEM,0
        BTFSC   STATUS,2
        CLRF    DEM
        MOVF    TEMP,W
        BCF     INTCON,1
        RETFIE

MAIN
    
```

```

                BSF      INTCON,7
                BSF      INTCON,4

                BCF      STATUS,6
                BSF      STATUS,5
                CLRF     TRISD
                BCF      STATUS,5
                CLRF     DEM

LOOP
                MOVF     DEM,W
                MOVWF    PORTD
                GOTO     LOOP
                END
    
```

6.3. NGẮT PORTB

Tương tự như ngắt RB0, nhưng dựa vào sự biến đổi trạng thái của các chân từ RB4÷RB7. Tức là khi RB4÷RB7 có sự biến đổi trạng thái thì cờ ngắt RBIF tích cực mức cao. Vậy để sử dụng ngắt PortB trong chương trình chính chúng ta cần Set bit RBIE (INTCON<4>) và Set bit GIE (INTCON<7>)

Ví dụ:

Viết chương trình cho mạch chống trộm, 4 ngõ vào (RB4÷RB7), 1 ngõ ra (RA0). Một trong 4 ngõ vào có tác động ngõ ra tích cực mức cao.

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

GOTO MAIN

ORG 04H

GOTO NGAT

MAIN

BSF INTCON,7

BSF INTCON,3

BCF STATUS,6

BSF STATUS,5

BCF TRISA,0

GOTO \$

NGAT

BCF STATUS,6

BCF STATUS,5

```

BSF      PORTA,0
BCF      INTCON,0
RETFIE
END

```

6.4. NGẮT TIMER

Như chúng ta đã biết, giá trị trong bộ Timer sẽ tăng theo mỗi xung nhịp tác động, đối với Timer0 và Timer2 thì khi thanh ghi TMR0 hoặc TMR2 đạt giá trị FFH (255) thì khi đó cờ tràn TMR0IF hoặc TMR2IF sẽ tích cực mức cao. Riêng đối với Timer1 để cờ tràn TMR1IF đạt giá trị tích cực mức cao thì giá trị trong thanh TMR1 (16 Bit gồm 2 thanh ghi TMR1H và TMR1L) phải đạt giá trị là FFFFH (65535)

Để sử dụng ngắt TIMER chúng ta biết rõ mình sử dụng Timer nào mà Set Bit TMR_xIE tương ứng khi khai báo, đồng thời Set Bit PEIE và Bit GIE (riêng đối Timer0 thì không cần Set Bit PEIE)

Ví dụ:

Viết chương trình tạo xung vuông tại chân RD0, tần số $f=10\text{hz}$, dùng ngắt

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

GOTO MAIN

ORG 04H

```

BCF      STATUS,6
BCF      STATUS,5
MOVLW    B'00000001'
XORWF    PORTD,1
BCF      PIR1,0
MOVLW    3CH
MOVWF    TMR1H
MOVLW    0AFH
MOVWF    TMR1L

```

RETFIE

MAIN

```

BSF      INTCON,7
BSF      INTCON,6
BCF      STATUS,6
BSF      STATUS,5
BSF      PIE1,0

BCF      STATUS,6

```

```

        BCF      STATUS,5
        CLRF     PORTD
        BSF      STATUS,5
        BCF      TRISD,0
        BCF      STATUS,5

LOOP
        MOVLW    B'00000000'
        MOVWF    T1CON
        MOVLW    3CH
        MOVWF    TMR1H
        MOVLW    0AFH
        MOVWF    TMR1L
        BSF      T1CON,0
        GOTO     $
        END
    
```

6.5. NGẮT ADC

Đối với bộ chuyển đổi ADC sau khi đã thiết lập các chế độ (Chọn ngõ vào, điện áp chuẩn, tần số chuyển đổi...) xong, sau đó cho bộ chuyển đổi ADC bắt đầu hoạt động thông qua bit ADON (ADCON0 <2>). Một thời gian sau, khoảng vài trăm μ S thì quá trình chuyển đổi hoàn tất, khi đó cờ ngắt ADIF được Set lên mức cao. Nếu chúng ta Set các Bit ADIE, PEIE và Bit GIE thì khi đó ngắt sẽ xảy ra.

Ví dụ:

Làm lại bài tập 3 chương 4 (trang 67), dùng ngắt

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

ORG 0000H

GOTO MAIN

ORG 04H

GOTO NGAT_ADC

MAIN

BSF INTCON,7

BSF INTCON,6

BSF PIE1,6

BSF STATUS,5

BCF STATUS,6

;.....**Chọn số ngõ vào**.....

BCF ADCON1,3

BCF ADCON1,2

```

BCF      ADCON1,1
BSF      ADCON1,0
;.....Chọn ngõ vào.....
BCF      STATUS,5
BCF      ADCON0,5
BCF      ADCON0,4
BCF      ADCON0,3
;.....Chọn tần số lấy mẫu.....
BCF      ADCON0,7
BCF      ADCON0,6
BSF      STATUS,5
BSF      ADCON1,6
;.....Chọn nơi lưu kết quả.....
BCF      ADCON1,7
BCF      STATUS,5
; .....Cho phép bộ chuyển đổi ADC hoạt động.....
STAR     BSF      ADCON0,2
          BSF      ADCON0,0
;.....Chuyển đổi xong chưa?.....
          BCF      STATUS, 5
LOOP
          BTFSC    PIR1,6
          GOTO     LOOP
          NOP
          GOTO     STAR
NGAT_ADC
          MOVF     ADRESH,0
          MOVWF    PORTB
          BCF      PIR1,6
          RETFIE
          END

```

6.6. NGẮT PORT NỐI TIẾP

Tương tự như ngắt ADC, đối với truyền nhận nối tiếp, khi truyền (nhận) xong 1 byte thì cờ ngắt Set lên mức cao. Để ngắt xảy ra ta Set các Bit điều khiển tương ứng, đối với truyền nối tiếp chúng ta cần Set 3 Bit điều khiển trong khi khai báo: GIE, PEIE và TXIE. Còn đối với nhận nối tiếp thì các Bit cần Set: GIE, PEIE và RCIE

Ví dụ:

Làm lại bài tập 2 chương 5 (trang 78), dùng ngắt

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

DEM EQU 20H

TEMP EQU 21H

ORG 0000H

GOTO MAIN

ORG 04H

GOTO NGAT_TRUYEN

MAIN

BSF INTCON,7

BSF INTCON,6

BSF PIE1,4

BCF STATUS,6

BCF STATUS,5

CLRF PORTC

BSF STATUS,5

BCF TRISC,6

BCF TRIC,7

BCF STATUS,5

MOVLW D'30'

MOVWF DEM

MOVLW 22H

MOWF FSR

TRUYEN_NOI_TIEP

;.....Chọn baud rate.....

BSF TXSTA, BRGH

MOVLW D'25'

MOVWF SPBRGN

;..... Cho phép cổng nối tiếp hoạt động.....

BCF TXSTA, SYNC

BSF RCSTA, SPEN

;.....Xuất giá trị cần truyền vào thanh ghi RCREG...

MOVF TEMP,0

MOVWF TXREG

;.....Cho phép truyền.....

BSF TXSTA, TXEN

;Truyền xong chưa? Nếu truyền xong thì vào ngắt, rồi trở lại, truyền tiếp.

LOOP1

```

        BTFSS      PIR1,TXIF
        GOTO       LOOP1
        NOP
        GOTO       TRUYEN_NOI_TIEP
    
```

NGAT_TRUYEN

```

        DECFSZ     DEM
        GOTO       TIEP
        GOTO       $
    
```

TIEP

```

        INCF       FSR,1
        MOVF       INDF,0
        MOVWF      TEMP
        BCF        PIR1,4
        RETFIE
        END
    
```

BÀI TẬP THAM KHẢO CHƯƠNG 6

Bài 1:

Nếu RB0 tác động cạnh lên thì xuất ra chân RC2 một chuỗi xung có tần số $f=1\text{Khz}$ trong 3 giây.

PROCESSOR 16F877A

#INCLUDE <P16F877A.INC>

DEM EQU 22H

ORG 0000H

```

        GOTO       MAIN
    
```

ORG 04H

```

        GOTO       NGAT
    
```

MAIN

```

        BCF        STATUS,6
        BCF        STATUS,5
        BSF        INTCON,7
        BSF        INTCON,4
        GOTO       $
    
```

NGAT

;Tạo xung dùng khối PWM, có tần số 1Khz

;.....Buoc 1.....

```

        BSF        STATUS,5
        BCF        TRISC,2
    
```

```

;.....Buoc 2.....
MOVLW    D'250'
MOVWF     PR2
;.....Buoc 3.....
BCF       STATUS,5
MOVLW     D'127'
MOVWF     CCPR1L
BCF       CCP1CON,5
BCF       CCP1CON,4
;.....Buoc4.....
BCF       T2CON,1
BSF       T2CON,0
BSF       T2CON,2; CHO TIMER2 HOAT DONG
;.....Buoc5.....
BSF       CCP1CON,3
BSF       CCP1CON,2
;.....TAO TRE.....
MOVLW     D'100'
MOVWF     DEM
MOVLW     B'00000000'
MOVWF     T1CON

STAR

MOVLW     3CH
MOVWF     TMR1H
MOVLW     0AFH
MOVWF     TMR1L
BSF       T1CON,0

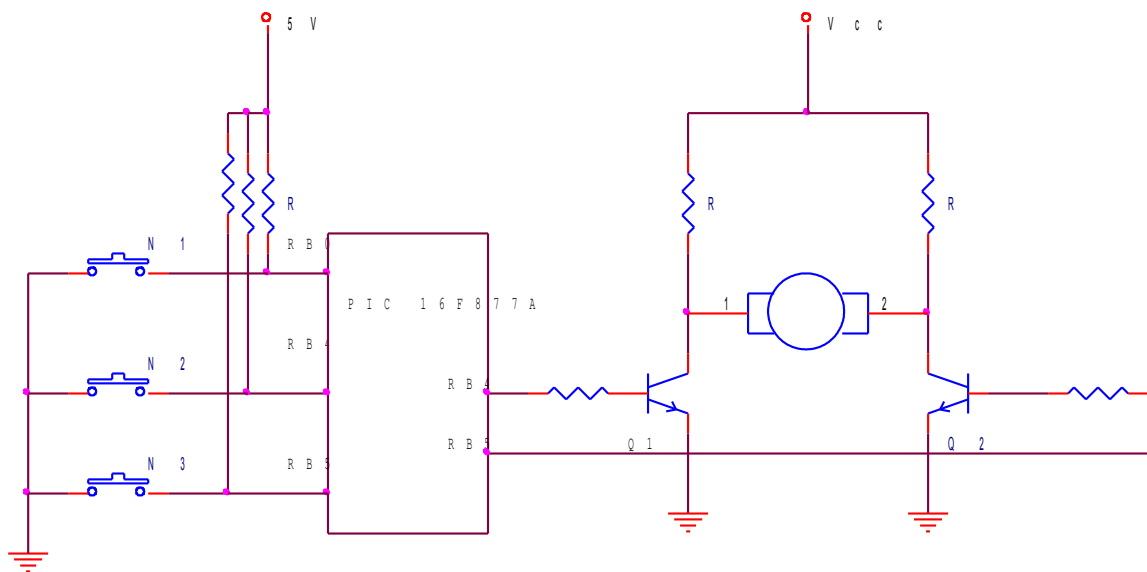
TEMPS

BTFSS     PIR1,0
GOTO      TEMPS
BCF       PIR1,0
DECFSZ    DEM
GOTO      STAR
;.....TẮT XUNG.....
BCF       STATUS,5
CLRF      CCPR1L
BCF       CCP1CON,5

```

END

Viết chương trình điều khiển động cơ: (hình 6.3)



Nút N1 (nút ON/OFF): Khi nhấn N1 nếu động cơ đang hoạt động thì ngừng và ngược lại
Nút N2 (thuận): Nhấn nút N2 động cơ quay thuận, cùng chiều kim đồng hồ.
Nút N3 (ngược): Nhấn nút N3 động cơ quay ngược, ngược chiều kim đồng hồ.
Khi động cơ ngừng (không hoạt động, nút N2, N3 không có tác dụng

BCF STATUS,6

	BSF	STATUS,5
	BCF	TRISD,0
	BCF	TRISD,7
	BCF	STATUS,5
	CLRF	PORTD
	CLRF	DEM
STAR		
	BTFSS	DEM,0
	GOTO	STOP
	GOTO	ON
ON		
	BSF	DEM,1
	GOTO	STAR
STOP		
	BCF	PORTD,0
	BCF	PORTD,7
	BCF	DEM,1
	GOTO	STAR
NGAT		
	BTFSC	INTCON,0
	GOTO	NGAT_2
	MOVLW	B'00000001'
	XORWF	DEM,1
	BCF	INTCON,1
	RETFIE	
NGAT_2		
	BTFSS	DEM,1
	GOTO	X
	BTFSS	PORTB,5
	GOTO	THUAN
	GOTO	NGHICH
THUAN		
	BSF	PORTD,7
	BCF	PORTD,0
	GOTO	X
NGHICH		
	BCF	PORTD,7
	BSF	PORTD,0
X	BTFSS	PORTB,4
	GOTO	X

Y	BTFSS	PORTB,5
	GOTO	Y
	BCF	INTCON,0
	RETFIE	
	END	

PHỤ LỤC: GIỚI THIỆU LẬP TRÌNH CCS

Chương I: Tập Lệnh Trong CCS

I.1. Các Phép Toán Trong CCS

I.1.1. Cách Khai Báo Biến, Hằng, Mảng

I.1.1.1. Các loại biến sau mỗi lần khởi tạo :

int1 số 1 bit = true hay false (0 hay 1)

int8 số nguyên 1 byte (8 bit)

int16 số nguyên 16 bit

int32 số nguyên 32 bit

char ký tự 8 bit

float số thực 32 bit

short maec ñòngh nhö kieåu int1

byte maec ñòngh nhö kieåu int8

int maec ñòngh nhö kieåu int8

long maec ñòngh nhö kieåu int16

Theâm signed hoaëc unsigned phía tröôùc ñeå chæ ñoù laø soá coù daáu hay khoâng daáu .Khai baùo nhö treân maec ñòngh laø khoâng daáu . 4 khai baùo cuoái khoâng neân duøng vì deã nhaàm laãn . Thay vaøo ñoù neân duøng 4 khai baùo ñeàu .

VD :

Signed int8 a ; // soá a laø 8 bit daáu (bit 7 laø bit daáu).

Signed int16 b , c , d ;

Signed int32 , . . .

Phaïm vi bieán :

Int8 : 0 , 255 signed int8 : -128 , 127

Int16 : 0 , $2^{15}-1$ signed int16 : -2^{15} , $2^{15}-1$

Int32 : 0 , $2^{32}-1$ signed int32 : -2^{31} , $2^{31}-1$

Khai baùo haøng : VD :

Int8 const a=231 ;

I.1.1.1.Khai baùo 1 maøung haøng soá :

VD : Int8 const a[5] = { 3,5,6,8,6 } ; //5 phaàn töû , chæ soá maøung baét ñeàu töø 0 : a[0]=3

Moät maøung haøng soá coù kích thoøùc **toái ña tuyø thuoäc loaïi VÑK:**

NeáuVÑK laø **PIC 14** (VD :16F877) : baïn chæ ñöôïc khai baùo 1 maøung haøng soá coù kích thoøùc toái ña laø256 byte .Caùc khai baùo sau laø hôïp leä.

Int8 const a[5]={ . . . }; // söû duïng 5 byte , daáu . . . ñeå baïn ñieàn soá vaøo

Int8 const a[256]={ . . . }; // 256 phaàn töû x 1 byte = 256 byte

Int16 const a[12] = { . . . }; // 12 x 2= 24 byte

Int16 const a[128] = { . . . }; // 128 x 2= 256 byte

I.1.2. Các phép toán số học

- + Cộng
- ++ Tăng 1 đơn vị
- Trừ
- Giảm 1 đơn vị
- * Nhân
- / Chia
- % Chia lấy dư
- = Bằng, thực hiện gián

I.1.3. Các phép toán Logic

- && Phép toán AND
- || Phép toán OR
- >> Dịch phải
- << Dịch trái
- ! Đảo bit

~ Lấy bù
& AND từng bit
| Or từng bit

I.1.4. Các phép toán so sánh

== So sánh bằng
> Lớn hơn.
>= Lớn hơn hoặc bằng.
< Nhỏ hơn
<= Nhỏ hơn hoặc bằng
!= Khác

I.2. Các Kiểu Điều Khiển Trong CCS

I.2.1. Kiểu: If – Else

Cú pháp:

```
If ( điều kiện)
{
    Những lệnh thỏa điều kiện;
}
Else
{
    Những lệnh thỏa điều kiện;
}
```

I.2.2. Kiểu: While

Cú pháp:

```
While ( điều kiện)
{
    Những lệnh thỏa điều kiện;
}
```

Chú ý: Trong vòng lặp While điều kiện luôn được kiểm tra

I.2.3. Kiểu: Do – While

Cú pháp:

```
Do
{
    Những lệnh ;
}
While ( điều kiện)
```

I.2.4. Kiểu: For

Cú pháp:

```
For (biểu thức khởi tạo; biểu thức điều kiện; biểu thức tác động)
{
    lệnh;
}
```

I.2.5. Kiểu: witch...case

```
switch (biểu thức)
{
    case giá trị 1:
    {
        lệnh 1;
        break;
    }
    case giá trị 2:
```

```

    {      lệnh 2;
          break;
    }
    ....
default:
{      lệnh ;
      break ;
}
}
}

```

I.3. Cấu Trúc Chương Trình

Ví Dụ: Viết chương trình tại RB0 mức cao

```

#include <16f877a.h>
#fuses nowdt,noprotect,nolvp,xt,put
#use delay(clock=4000000)
#use fast_io(b)
#byte portb = 0x6
#bit b0 = portb.0
void main()
{
    set_tris_b(0b0);
    while (true)
    {
        b0=1;
    }
}

```

Chương II: Sử Dụng Các Khối Chức Năng.

II.1. Sử Dụng Hàm Delay

III.1.1. Delay_cycles (count)

Count : hàng số từ 0 – 255 , là số chu kỳ lệnh .1 chu kỳ lệnh bằng 4 chu kỳ máy .
hàm không trả về trò . Hàm dùng delay 1 số chu kỳ lệnh cho trước .

VD : delay_cycles (25) ; // với OSC = 20 Mhz , hàm này delay 5 us

III.1.2.Delay_us (time)

Time : là biến số thì = 0 – 255 , time là hằng số thì = 0 -65535

Haøm không traû veà trò .Haøm naøy cho pheùp delay khoaùng thôøi gian daøi hôn theo ñôn vò us .Quan saùt trong C / asm list baïn seõ thaáy vòuì time daøi ngaén khaùc nhau , CSS sinh maõ khaùc nhau .

III.1.3.Delay_ms (time)

Time = 0-255 neáu là biến số hay = 0-65535 neáu là hằng số .

Haøm không traû veà trò .

Haøm naøy cho pheùp delay daøi hôn nũa .

VD :

```
Int a = 215;
```

```
Delay_us ( a ) ; // delay 215 us
```

```
Delay_us ( 4356 ) ; // delay 4356 us
```

```
Delay_ms ( 2500 ) ; // delay 2 . 5 s
```

II.2. Sử Dụng Khối ADC.

Bảng khai báo sử dụng số ngõ vào Analog:

- ALL_ANALOGS : dùng tất cả chân sau làm analog : A0 A1 A2 A3 A5 E0 E1 E2

(Vref=Vdd)

- NO_ANALOG : không dùng analog , các chân nào sẽ là chân I/O .

- AN0_AN1_AN2_AN4_AN5_AN6_AN7_VSS_VREF : A0 A1 A2 A5 E0 E1 E2 VRefh=A3

- AN0_AN1_AN2_AN3_AN4 : A0 A1 A2 A3 A5

- AN0_AN1_AN3 : A0 A1 A3 , Vref = Vdd

- AN0_AN1_VSS_VREF : A0 A1 VRefh = A3

- AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF : A0 A1 A5 E0 E1 E2

VRefh=A3 ,

VRefl=A2 .

- AN0_AN1_AN2_AN3_AN4_AN5 : A0 A1 A2 A3 A5 E0

- AN0_AN1_AN2_AN4_AN5_VSS_VREF : A0 A1 A2 A5 E0 VRefh=A3

- AN0_AN1_AN4_AN5_VREF_VREF : A0 A1 A5 E0 VRefh=A3 VRefl=A2

- AN0_AN1_AN4_VREF_VREF : A0 A1 A5 VRefh=A3 VRefl=A2

- AN0_AN1_VREF_VREF : A0 A1 VRefh=A3 VRefl=A2

- AN0 : A0

- AN0_VREF_VREF : A0 VRefh=A3 VRefl=A2

* Chương trình sử dụng đọc ADC:

```
Void Doc_ADC()
```

```
{
```

```
    setup_ADC(ADC_clock_internal); // div_by_2
```

```
    setup_ADC_ports(AN0);
```

```
    set_ADC_channel(0);
```

```
    delay_us(800);
```

```

    }
II.3. Sử Dụng PWM.
    Void Xuat_xung()
    {

```

```

        setup_ccp1(CCP_PWM);
        set_pwm1_duty(150); //  $T_H=150*4$ 
        setup_timer_2(t2_div_by_4,200,1); //  $T=200*4$ 
    }

```

II.4. Truyền Nối Tiếp

II.4.1. Chuẩn RS232

a. GETC(), GETCH(), GETCHAR():

Hàm này được dùng để đợi nhận 1 ký tự từ pin RS232 RCV. Nếu không muốn đợi ký tự gọi về.

+ Cú pháp:: ch = getc()

ch = getch()

ch = getchar()

+ Trị trả về: ký tự 8 bit

+ Yêu cầu: #use rs232

Ví dụ: #include <16f877.h>

#use delay(clock=20000000)

#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)

char answer;

```
void main()
```

```
{
```

```
printf("Continue (Y,N)?");
```

```
answer=getch();
```

```
}
```

```
while(answer!='Y' && answer!='N');
```

b. GETS(),

Hàm này được dùng để đọc các ký tự (dùng GETC()) trong chuỗi cho đến khi gặp lệnh RETURN

+ Cú pháp:: gets(char *string)

+ Tham số: string là con trỏ (pointer) chỉ đến dãy ký tự

+ Yêu cầu: #use rs232

Ví dụ: #include <16f877.h>

#include <string.h>

#use delay(clock=20000000)

#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)

char string[30];

```
void main()
```

```
{
```

```
printf("Input string: ");
```

```
gets(string);
```

```
printf("\n\r");
```

```
printf(string);
```

```
}
```

c. PUTC(), PUTCHAR():

Hàm này được dùng để gửi một ký tự thông qua pin RS232 XMIT. Phải dùng #USE RS232 trước khi thực hiện lệnh này để xác định tốc độ (baud rate) và pin truyền.

```
+ Cú pháp: putc(cdata)
putc(cdata)
+ Tham số: cdata là ký tự 8 bit
+ Trị trả về: không
+ Yêu cầu: #use rs232
Ví dụ: #include <16f877.h>
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)
int i;
char string[10];
void main()
{
strcpy(string,"Hello !"); //copy "Hello !" to string
for(i=0; i<10; i++) putc(string[i]); //put each charater of string onto screen
}
```

d. PUTS():

Hàm này được dùng để gửi mỗi ký tự trong chuỗi đến pin RS232 dùng PUTC(). Sau khi chuỗi được gửi đi thì RETURN (13) và LINE-FEED (10) được gửi đi. Lệnh printf() thường dùng hơn lệnh puts().

```
+ Cú pháp: puts(string)
+ Tham số: string là chuỗi hằng (constant string) hay dãy ký tự (character array)
+ Trị trả về: không
+ Yêu cầu: #use rs232
Ví dụ: Dùng PUTS()
#include <16f877.h>
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)
void main()
{
puts(" ----- ");
puts(" | Hello | ");
puts(" ----- ");
}
Dùng PRINTF()
#include <16f877.h>
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)
void main()
{
printf(" ----- \n\r");
printf(" | Hello | \n\r");
printf(" -----");
}
```

e. KBHIT():

Hàm này được dùng để báo đã nhận được bit start.

```
+ Cú pháp:: value = kbhit()
```

- + Tham số: không
- + Trị trả về: 0 (hay FALSE) nếu getc() cần phải đợi để nhận 1 ký tự từ bàn Phím 1 (hay TRUE) nếu đã có 1 ký tự sẵn sàng để nhận bằng getc().
- + Yêu cầu: #use rs232

f. PRINTF():

Hàm này được dùng để xuất một chuỗi theo chuẩn RS232 hoặc theo một hàm xác định. Dữ liệu được định dạng phù hợp với đối số của chuỗi.

Các định dạng dữ liệu như sau:

- C Kiểu ký tự
- S Chuỗi hoặc ký tự
- U Số nguyên không dấu
- x Hex int (xuất chữ thường)
- X Hex int (xuất chữ hoa)
- D số nguyên có dấu
- e Số thực định dạng kiểu số mũ
- f Kiểu dấu chấm động
- Lx Hex long int (chữ thường)
- LX Hex long int (chữ hoa)
- Iu số thập phân không dấu
- Id Số thập phân có dấu.
- % Dấu %

+ Cú pháp: printf(string)

printf(cstring, values...)

printf(fname, cstring, values...)

+ Tham số: String là một chuỗi hằng Hoặc một mảng ký tự không xác định.

Values là danh sách các biến phân cách nhau bởi dấu ‘,’ , fname is là tên hàm dùng để xuất dữ liệu (mặc nhiên là putc()).

+ Trị trả về: không

+ Yêu cầu: #use rs232

g. SET_UART_SPEED():

Hàm này được dùng để đặt tốc độ truyền dữ liệu thông qua cổng RS232.

+ Cú pháp:: set_uart_speed(baud)

+ Tham số: baud là hằng số tốc độ truyền (bit/giây) từ 100 đến 115200.

+ Trị trả về: không

+ Yêu cầu: #use rs232

Ví dụ: // Set baud rate based on setting of pins B0 and B1

switch(input_b() & 3)

{

case 0 : set_uart_speed(2400); break;

case 1 : set_uart_speed(4800); break;

case 2 : set_uart_speed(9600); break;

case 3 : set_uart_speed(19200); break;

}

II.4.2. Chuẩn I2C

a. #USE I2C():

Thư viện I2C gồm các hàm dùng cho I2C bus. #USE I2C dùng với các lệnh

I2C_START, I2C_STOP, I2C_READ, I2C_WRITE and I2C_POLL. Các hàm phần mềm được tạo ra trừ khi dùng lệnh FORCE_HW.

+ Cú pháp: #use i2c(mode, SDA=pin, SCL=pin[options])

- + Tham số: mode: master/slave - đặt master/slave mode
- SCL=pin – chỉ định pin SCL (pin là bit address)
- SDA=pin – chỉ định pin SDA options như sau
- ADDRESS=nn : chỉ định địa chỉ slave mode
- FAST : sử dụng fast I2C specification
- SLOW : sử dụng slow I2C specification
- RESTART_WDT : khởi động lại WDT trong khi chờ đọc I2C_READ
- FORCE_HW : sử dụng chức năng I2C phần cứng (hardware I2C functions)

b. I2C_START():

Hàm này được dùng để Khởi động start bit (bit khởi động) ở I2C master mode. Sau khi khởi động start bit, xung clock ở mức thấp chờ đến khi lệnh I2C_WRITE() được thực hiện. Chú ý I2C protocol phụ thuộc vào thiết bị slave.

- + Cú pháp: i2c_start()
- + Tham số: không
- + Trị trả về: không
- + Yêu cầu: #use i2c
- Ví dụ: i2c_start();
- i2c_write(0xa0); //Device address
- i2c_write(address); //Data to device
- i2c_start(); //Restart
- i2c_write(0xa1); //to change data direction
- data=i2c_read(0); //Now read from slave
- i2c_stop();

c. I2C_STOP():

Hàm này được sử dụng để tắt sử dụng I2C ở master mode.

- + Cú pháp: i2c_stop()
- + Tham số: không
- + Trị trả về: không
- + Yêu cầu: #use i2c
- Ví dụ: i2c_start(); //Start condition
- i2c_write(0xa0); //Device address
- i2c_write(5); //Device command
- i2c_write(12); //Device data
- i2c_stop(); //Stop condition

d. I2C_POLL():

Hàm này được dùng để hỏi vòng I2C, hàm này chỉ được dùng khi SSP được dùng. Hàm này trả về giá trị TRUE nếu nhận được giá trị ở bộ đệm. Khi hàm này lên TRUE, nếu dùng hàm I2C_READ thì ta được giá trị đọc về.

- + Cú pháp: i2c_poll()
- + Tham số: không
- + Trị trả về: 1 (TRUE) hay 0 (FALSE)
- + Yêu cầu: #use i2c
- Ví dụ: i2c_start(); //Start condition
- i2c_write(0xc1); //Device address/Read
- count=0;
- while(count!=4)
- {
- while(!i2c_poll());
- buffer[count++]=i2c_read(); //Read Next

```
}
i2c_stop(); // Stop condition
```

e. I2C_READ(), I2CREAD(ACK):

Hàm này được dùng để Đọc một byte qua cổng I2C Ở thiết bị master: lệnh này tạo xung clock và ở thiết bị claver, lệnh này chờ đọc xung clock. There is no timeout for the slave, use I2C_POLL to prevent a lockup. Use ESTART_WDT in the #USE I2C to strobe the watch-dog timer in the slave mode while waiting.

```
Cú pháp: i2c_stop()
i2c_stop(ack)
Tham số: tùy chọn, mặc định là 1
ack = 0: không kiểm tra trạng thái thu gởi tín hiệu (ack: acknowlegde)
ack = 1: kiểm tra trạng thái thu gởi tín hiệu
Trị trả về: 8 bit int
Yêu cầu: #use i2c
Ví dụ: i2c_start();
i2c_write(0xa1);
data1 = i2c_read();
data2 = i2c_read();
i2c_stop();
```

f. I2C_WRITE():

Hàm này được dùng để Gửi từng byte thông qua giao diện I2C. Ở chế độ chủ sẽ phát ra xung Clock với dữ liệu và ở chế độ Slave sẽ chờ xung Clock từ con chủ truyền về. Không tự động đếm ngoài là điều kiện của lệnh này. Lệnh này sẽ trả về bit ACK. Phát LSB trước khi truyền khi đã xác định hướng truyền của dữ liệu truyền (0 cho master sang slave). Chú ý chuẩn giao tiếp I2C

```
phụ thuộc vào thiết bị slave.
+ Cú pháp: i2c_write(data)
+ Tham số: data: 8 bit int
+ Trị trả về: Lệnh này trả về bit ACK
ack = 0: không kiểm tra trạng thái thu gởi tín hiệu (ack: acknowlegde)
ack = 1: kiểm tra trạng thái thu gởi tín hiệu
+ Yêu cầu: #use i2c
Ví dụ: long cmd;

...
i2c_start(); //Start condition
i2c_write(0xa0); //Device address
i2c_write(cmd); //Low byte of command
i2c_write(cmd>>8); //High byte of command
i2c_stop(); //Stop condition
```

Chương III: Sử Dụng Ngắt

III.1. Cấu Trúc Chương Trình Có Sử Dụng Ngắt.

III.1.1. Khai Báo Ngắt.

```
Enable_interrupts(int_EXT); // Cho phép ngắt ngoài
Enable_interrupts(global); // Cho phép ngắt toàn cục
#INT_AD : chuyển đổi A /D lẫn lộn hoàn tất , thông thường thì
không nên dùng
#INT_CCP1 : coù Capture hay compare trên CCP1
#INT_CCP2 : coù Capture hay compare trên CCP2
```

```
#INT_COMP : kiểm tra bằng nhau trên Comparator
#INT_EXT : ngắt ngoại
#INT_I2C : coù hoaït ñoäng I 2C
#INT_LCD : coù hoaït ñoäng LCD
#INT_PSP : coù data vaøo coäng Parallel slave
#INT_RB : baát kyø thay ñoãi naøo treân chaân B4 ñeán B7
#INT_RDA : data nhaän töø RS 232 saün saøng
#INT_RTCC : traøn Timer 0
#INT_SSP : coù hoaït ñoäng SPI hay I 2C
#INT_TBE : boä ñeäm chuyeån RS 232 troáng
#INT_TIMER0 : moät teân khaùc cuûa #INT_RTCC
#INT_TIMER1 : traøn Timer 1
#INT_TIMER2 : traøn Timer 2
```

III.1.2 Cấu Trúc Chương Trình

```
#include <16F877a.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#INT_x
void ngat();
void main()
{
    enable_interrupts(int_x); // Khai báo ngắt gì?
    enable_interrupts(global); // Cho phép ngắt toàn cục
    Chương trình chính;
}
void ngat()
{
    Xử lý ngắt;
}
```

III.2. Sử Dụng Ngắt

III.2.1. Ngắt RB0

```
#include <16F877a.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#use fast_io(b)
#byte portb=0x6
#bit b7=portb.7
#INT_EXT
void ngat()
{
    b7=!b7;
}
void main()
{
    enable_interrupts(int_EXT);
    enable_interrupts(global);
    set_tris_b(0b00000001);
    portb=0;
```

```
While (1)
    {} }
```

III.2.2. Ngắt Timer

```
#include <16F877a.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#use fast_io(b)
#byte portb=0x6
#bit b0=portb.0
int16 time;
#INT_TIMER1
void Ngat_timer1();
void main()
{
    enable_interrupts(int_timer1); //Ngat Timer1
    enable_interrupts(global); // Ngat toan cuc
    setup_timer_1(t1_INTERNAL|t1_div_by_4); // Chia tan
    set_timer1(55535); //Cai gia tri cho Timer1
    set_tris_b(0b0);
    While (1)
        {}
}
#INT_TIMER1
void Ngat_timer1()
{
    set_timer1(55535); //Cai gia tri cho Timer1
    b0=!b0;
}
```

PHỤ LỤC: CÁC THANH GHI CHỨC NĂNG

1. Thanh ghi TMRO: ñòa chæ 01h, 101h.

Thanh ghi 8 bit chõuà giaù trò cuûa boã ñònh thôøi Timer0.

2. Thanh ghi PCL: ñòa chæ 02h, 82h, 102h, 182h.

Thanh ghi chõuà 8 bit thaáp cuûa boã ñeám chõông trình (PC)

3. Thanh ghi STATUS: ñòa chæ 03h, 83h, 103h, 183h

IPR	RP1	RP0	TO	PD	Z	DC	C
Bit 7							Bit 0

Thanh ghi trạng thái chứa các trạng thái số học của bộ ALU, trạng thái Reset và các bit chọn Bank của bộ nhớ dữ liệu.

Bit 7 **IRP**: Bit lựa chọn bank thanh ghi (Sử dụng cho định địa chỉ gián tiếp).

1 = Bank 2, 3 (100h – 1FFh)

0 = Bank 0, 1 (00h – FFh)

Bit 6 – 5 **RP1 – RP0**: Bit lựa chọn bank thanh ghi (Dùng trong định địa chỉ trực tiếp).

11 = Bank 3 (180h – 1FFh)

10 = Bank 2 (100h – 17Fh)

01 = Bank 1 (80h – FFh)

00 = Bank 0 (00h – 7Fh)

Each bank is 128 bytes

Bit 4 **TO**: Bit báo hiệu hoạt động của WDT.

1: Lệnh xóa WDT hoặc Sleep xảy ra.

0: WDT hoạt động.

Bit 3 **PD**: Bit báo công suất thấp (Power down bit).

1: Sau khi nguồn tăng hoặc có lệnh xóa WDT.

0: Thực thi lệnh Sleep.

Bit 2 **Z**: bit Zero

1: Khi kết quả của một phép toán bằng 0.

0: Khi kết quả của một phép toán khác 0.

Bit 1 **DC**: Digit Carry

1: Có một số nhớ sinh ra bởi phép cộng hoặc phép trừ 4 bit thấp.

0: Không có số nhớ sinh ra.

Bit 0 **C**: cờ nhớ (Carry Flag)

1: Có một số nhớ sinh ra bởi phép cộng hoặc phép trừ 4 bit cao.

0: Không có số nhớ sinh ra.

4. Thanh ghi FSR: ñòa chæ 04h.

Thanh ghi chöùa con troö ñòa chæ giàuñ tieáp cuöa boä nhöu döõ lieäu.

5. Thanh ghi PORTA: ñòa chæ 05h.

Thanh ghi chöùa giàu trò nhaän vaøo hay xuaát ra PORTA.

6. Thanh ghi PORTB: ñòa chæ 06h, 106h.

Thanh ghi chöùa giàu trò nhaän vaøo hay xuaát ra PORTB.

7. Thanh ghi PORTC: ñòa chæ 07h.

Thanh ghi chöùa giàu trò nhaän vaøo hay xuaát ra PORTC

8. Thanh ghi PORTD: ñòa chæ 08h.

Thanh ghi chöùa giàu trò nhaän vaøo hay xuaát ra PORTD.

9. Thanh ghi PORTE: ñòa chæ 09h.

Thanh ghi ch÷uà giàu trò nhään vaoø hay xuaát ra PORTE.

10. Thanh ghi PCLATCH: ñòa chæ 0Ah, 8Ah, 10Ah, 18Ah.

Thanh ghi ñoùng vai troø laø buffer ñeäm trong quaù trình ghi giàu trò lên 5 bit cao cuøa boä

ñeäm ch÷ông trình PC.

11. Thanh ghi INTCON: ñòa chæ 0Bh, 8Bh, 10Bh, 18Bh.

Thanh ghi INTCON (0Bh, 8Bh, 10Bh, 18Bh): thanh ghi cho phép ñọc và ghi, chứa các bit ñiều khiển và các bit cò hiệu khi timer0 bị tràn, ngắt ngoại vi RB0/INT và ngắt interrupt-on-change tại các chân của PORTB.

GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
Bit 7							Bit 0

Bit 7: GIE Global Interrupt Enable bit

GIE = 1 cho phép tất cả các ngắt.

GIE = 0 không cho phép tất cả các ngắt.

Bit 6: PEIE Pheripheral Interrupt Enable bit

PEIE = 1 cho phép tất cả các ngắt ngoại vi

PEIE = 0 không cho phép tất cả các ngắt ngoại vi

Bit 5: TMR0IE Timer0 Overflow Interrupt Enable bit

TMR0IE = 1 cho phép ngắt Timer0

TMR0IE = 0 không cho phép ngắt Timer0

Bit 4: INTE RB0/INT External Interrupt Enable bit

INTIE = 1 cho phép ngắt ngoại vi RB0/INT

INTIE = 0 không cho phép ngắt ngoại vi RB0/INT

Bit3: RBIE RB Port change Interrupt Enable bit

RBIE = 1 cho phép ngắt RB Port change

RBIE = 0 không cho phép ngắt RB Port change

Bit 2: TMR0IF Timer0 Interrupt Flag bit

TMR0IF = 1 thanh ghi TMR0 bị tràn (phải xóa bằng chương trình).

TMR0IF = 0 thanh ghi TMR0 chưa bị tràn.

Bit 1: INTF BR0/INT External Interrupt Flag bit

INTF = 1 ngắt RB0/INT xảy ra (phải xóa cò hiệu bằng chương trình).

INTF = 0 ngắt RB0/INT chưa xảy ra.

Bit 0: RBIF RB Port Change Interrupt Flag bit

RBIF = 1 ít nhất có một chân RB7:RB4 có sự thay ñổi trạng thái. Bit này phải ñược xóa bằng chương trình sau khi ñã kiểm tra lại các giá trị của các chân tại PORTB.

RBIF = 0 không có sự thay đổi trạng thái các chân RB7:RB4

12. Thanh ghi PIR1: ñòa chæ 0Ch

Thanh ghi chõua cõø ngaét cuõa caùc khoái ngoaïi vi.

PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
Bit 7							Bit 0

Bit 7: PSPIF Parallel Slave Port Read/Write Interrupt Flag bit

PSPIF = 1 vừa hoàn tất thao tác đọc hoặc ghi PSP (phải xóa bằng chương trình).

PSPIF = 0 không có thao tác đọc ghi PSP nào diễn ra.

Bit 6: ADIF ADC Interrupt Flag bit

ADIF = 1 hoàn tất chuyển đổi ADC.

ADIF = 0 chưa hoàn tất chuyển đổi ADC.

Bit 5: RCIF USART Receive Interrupt Flag bit

RCIF = 1 buffer nhận qua chuẩn giao tiếp USART đã đầy.

RCIF = 0 buffer nhận qua chuẩn giao tiếp USART rỗng.

Bit 4: TXIF USART Transmit Interrupt Flag bit

TXIF = 1 buffer truyền qua chuẩn giao tiếp USART rỗng.

TXIF = 0 buffer truyền qua chuẩn giao tiếp USART đầy.

Bit 3: SSPIF Synchronous Serial Port (SSP) Interrupt Flag bit

SSPIF = 1 ngắt truyền nhận SSP xảy ra.

SSPIF = 0 ngắt truyền nhận SSP chưa xảy ra.

Bit 2: CCP1IF CCP1 Interrupt Flag bit

Khi CCP1 ở chế độ Capture

CCP1IF=1 đã cập nhật giá trị trong thanh ghi TMR1.

CCP1IF=0 chưa cập nhật giá trị trong thanh ghi TMR1.

Khi CCP1 ở chế độ Compare

CCP1IF=1 giá trị cần so sánh bằng với giá trị chứa trong TMR1

CCP1IF=0 giá trị cần so sánh không bằng với giá trị trong TMR1.

Bit 1: TMR2IF TMR2 to PR2 Match Interrupt Flag bit

TRM2IF = 1 giá trị chứa trong thanh ghi TMR2 bằng với giá trị chứa trong thanh ghi PR2.

TRM2IF = 0 giá trị chứa trong thanh ghi TMR2 chưa bằng với giá trị chứa trong thanh ghi PR2.

Bit 0: TMR1IF TMR1 Overflow Interrupt Flag bit

TMR1IF = 1 thanh ghi TMR1 bị tràn (phải xóa bằng chương trình).

TMR1IF = 0 thanh ghi TMR1 chưa bị tràn

13. Thanh ghi PIR2: ñòa chæ 0Dh

-	CMIF	-	EEIF	BCLIF	-	-	CCP2IF
Bit 7							Bit 0

Bit 7, 5, 2, 1: không quan tâm và mặc định mang giá trị 0.

Bit 6: CMIF Comparator Interrupt Flag bit

CMIF = 1 tín hiệu ngõ vào bộ so sánh thay đổi.

CMIF = 0 tín hiệu ngõ vào bộ so sánh không thay đổi.

Bit 4: EEIF EEPROM Write Operation Interrupt Flag bit

EEIF = 1 quá trình ghi dữ liệu lên EEPROM hoàn tất.

EEIF = 0 quá trình ghi dữ liệu lên EEPROM chưa hoàn tất hoặc chưa bắt đầu.

Bit 3: BCLIF Bus Collision Interrupt Flag bit

BCLIF = 1 Bus truyền nhận đang bận khi (đang có dữ liệu truyền đi trong bus) khi SSP hạt động ở chế độ I2C Master mode.

BCLIF = 0 Bus truyền nhận chưa bị tràn (không có dữ liệu truyền đi trong bus).

Bit 0: CCP2IF CCP2 Interrupt Flag bit

Ở chế độ Capture

CCP2IF = 1 đã cập nhật giá trị trong thanh ghi TMR1.

CCP2IF = 0 chưa cập nhật giá trị trong thanh ghi TMR1.

Ở chế độ Compare

CCP2IF = 1 giá trị cần so sánh bằng với giá trị chứa trong TMR1.

CCP2IF = 0 giá trị cần so sánh chưa bằng với giá trị chứa trong TMR1

14. Thanh ghi TMR1L: nãa chæ 0Eh

Thanh ghi chõu 8 bit thaáp cuûa boã ñòngh thôøi TMR1.

15. Thanh ghi TMR1H: nãa chæ 0Fh

Thanh ghi chõu 8 bit cao cuûa boã ñòngh thôøi TMR2.

16. Thanh ghi T1CON: nãa chæ 10h

Thanh ghi ñieàu khiễån Timer1.

-	-	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
Bit7							Bit 0

Bit 7,6 Không sử dụng, đọc là 0.

Bit 5,4 **T1CKPS1 : T1CKPS0** : Các bit chọn tỉ lệ xung ngõ vào cho Timer1.

11 1 : 8 giá trị tỉ lệ

10 1 : 4 giá trị tỉ lệ

01 1 : 2 giá trị tỉ lệ

00 1 : 1 giá trị tỉ lệ

Bit 3 **T1OSCEN** : Bit cho phép bộ dao động Timer 1 Oscillator

1 : Cho phép dao động

0 : Không cho phép dao động

Bit 2 **T1SYNC** : Bit lựa chọn đồng bộ hóa xung clock ngoài của Timer 1

(**Chú ý:** Bit này chỉ có tác dụng khi **bit TMR1CS = 1**)

1: Không đồng bộ hóa xung clock ngoài

0: Đồng bộ hóa xung clock ngoài.

Bit 1 **TMR1CS** : Bit chọn nguồn xung clock cho **Timer 1**

1: Chọn xung clock ngoài qua chân **T1OSC/T1CKI** (tác động cạnh lên)

0: Chọn xung clock nội (Fosc/4)

Bit 0 **TMR1ON**: Bit cho phép ngoặt ngưng **Timer 1**

1: Cho phép

0: Không cho phép

17. Thanh ghi TMR2: ñòa chæ 11h

Thanh ghi chõuà giầu trò boã ñéám Timer2.

18. Thanh ghi T2CON: ñòa chæ 12h

Thanh ghi ñieàu khiểån Timer2.

-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
Bit 7							Bit0

Bit 7: không sử dụng

Bit 6:3 **TOUTPS3:TOUTPS0**: Bit chọn tỉ lệ ngõ ra của Timer 2

0000: 1:1 Tỷ lệ ngõ ra

0001: 1:2 Tỷ lệ ngõ ra

.

1111: 1:16 Tỷ lệ ngõ ra

Bit 2 **TMR2ON**: Bit cho phép hoạt động của Timer 2

1: Cho phép

0: Không cho phép.

Bit 1:0 **T2CKPS1:T2CKPS0**: Bit chọn tỉ lệ ngõ vào của Timer 2

00 : Prescaler 1

01 : Prescaler 4

1x : Prescaler 16

19. Thanh ghi SSPBUF: ñòa chæ 13h

Thanh ghi ñeãm dõõ lieäu 8 bit cho chuaån giao tieáp MSSP.

20. Thanh ghi SSPCON: ñòa chæ 14h

Thanh ghi ñieàu khiểån chuaån giao tieáp MSSP.

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
Bit 7							Bit 0

Khi MSSP ở chế độ SPI:

Bit 7 WCOL Write Collision Detect bit

WCOL = 1 dữ liệu mới được đưa vào thanh ghi SSPBUF trong khi chưa truyền xong dữ liệu trước đó.

WCOL = 0 không có hiện tượng trên xảy ra.

Bit 6 SSPOV Receive Overflow Indicator bit (bit này chỉ có tác dụng ở chế độ SPI Slave mode).

SSPOV = 1 dữ liệu trong bufer đệm (thanh ghi SSPBUF) bị tràn (dữ liệu cũ chưa được đọc thì có dữ liệu mới ghi đè lên).

SSPOV = 0 không có hiện tượng trên xảy ra.

Bit 5 SSPEN Synchronous Serial Port Enable bit

SSPEN = 1 cho phép cổng giao tiếp MSSP (các pin SCK, SDO, SDI và).

SSPEN = 0 không cho phép cổng giao tiếp MSSP.

Bit 4 CKP Clock Polarity Select bit

CKP = 1 trạng thái chờ của xung clock là mức logic cao.

CKP = 0 trạng thái chờ của xung clock là mức logic thấp.

Bit 3-0 SSPM3:SSPM0 Synchronous Serial Mode Select bit

Các bit này đóng vai trò lựa chọn các chế độ hoạt động của MSSP.

0101 Slave mode, xung clock lấy từ pin SCK, không cho phép pin điều khiển (là pin I/O bình thường).

0100 SPI Slave mode, xung clock lấy từ pin SCK, cho phép pin điều khiển .

0011 SPI Master mode, xung clock bằng (ngõ ra TMR2)/2.

0010 SPI Master mode, xung clock bằng (FOSC/64).

0001 SPI Master mode, xung clock bằng (FOSC/16).

0000 SPI Master mode, xung clock bằng (FOSC/4).

Các trạng thái không được liệt kê hoặc không có tác dụng điều khiển hoặc chỉ

có tác dụng đối với chế độ I2C mode.

Khi MSSP ở chế độ I2C

Bit 7 WCOL Write Collision Detect bit

Khi truyền dữ liệu ở chế độ I2C Master mode:

WCOL = 1 đưa dữ liệu truyền đi vào thanh ghi SSPBUF trong khi chế độ truyền dữ liệu của I2C chưa sẵn sàng.

WCOL = 0 không xảy ra hiện tượng trên.

khi truyền dữ liệu ở chế độ I2C Slave mode:

WCOL = 1 dữ liệu mới được đưa vào thanh ghi SSPBUF trong khi dữ liệu cũ chưa được truyền đi.

WCOL = 0 không có hiện tượng trên xảy ra.

Ở chế độ nhận dữ liệu (Master hoặc Slave):

Bit này không có tác dụng chỉ thị các trạng thái.

Bit 6 SSPOV Receive Overflow Indicator Flag bit. Khi nhận dữ liệu:

SSPOV = 1 dữ liệu mới được nhận vào thanh ghi SSPBUF trong khi dữ liệu cũ chưa được đọc.

SSPOV = 0 không có hiện tượng trên xảy ra.

Khi truyền dữ liệu:

Bit này không có tác dụng chỉ thị các trạng thái.

Bit 5 SSPEN Synchronous Serial Port Enable bit

SSPEN = 1 cho phép cổng giao tiếp MSSP (các pin SDA và SCL).

SSPEN = 0 không cho phép cổng giao tiếp MSSP.

Cần chú ý là các pin SDA và SCL phải được điều khiển trạng thái bằng các bit tương ứng trong thanh ghi TRISC trước đó).

Bit 4 CKP SCK Release Control bit

Ở chế độ Slave mode:

CKP = 1 cho xung clock tác động.

CKP = 0 giữ xung clock ở mức logic thấp (để bảo đảm thời gian thiết lập dữ liệu).

Bit 3,0 SSPM3:SSPM0

Các bit này đóng vai trò lựa chọn các chế độ hoạt động của MSSP.

1111 I2C Slave mode 10 bit địa chỉ và cho phép ngắt khi phát hiện bit Start và bit Stop.

1110 I2C Slave mode 7 bit địa chỉ và cho phép ngắt khi phát hiện bit Start và bit Stop.

1011 I2C Firmwave Controlled Master mode (không cho phép chế độ Slave).

1000 I2C Master mode, xung clock = $FOSC/(4*(SSPAD+1))$.

0111 I2C Slave mode 10 bit địa chỉ.

Các trạng thái không được liệt kê hoặc không có tác dụng điều khiển hoặc chỉ có tác dụng đối với chế độ SPI mode.

21. Thanh ghi CCP1L: ñòa chæ 15h

Thanh ghi chõu 8 bit thaáp cuuá khoái CCP1.

22. Thanh ghi CCP1H: ñòa chæ 16h

Thanh ghi chõu 8 bit cao cuuá khoái CCP1.

23. Thanh ghi CCP1CON vaø thanh ghi CCP2CON: ñòa chæ 17h

(CCP1CON) vaø 1Dh

(CCP2CON)

Thanh ghi ñieàu khiễãn khoái CCP1.

-	-	CCPXX	CCPXY	CCPxMP3	CCPxMP2	CCPxMP1	CCPxMP0
---	---	-------	-------	---------	---------	---------	---------

Bit 7,6 Không có tác dụng và mặc định mang giá trị 0.

Bit 5,4 CCPxX:CCPxY: PWM least Significant bits (các bit này không có tác dụng ở chế độ Capture và Compare). Ở chế độ PWM, đây là 2 bit MSB chứa giá trị tính độ rộng xung (duty cycle) của khối PWM (8 bit còn lại được chứa trong thanh ghi CCPRxL).

Bit 3-0 CCPxM3:CCPxM0 CCPx Mode Select bit

Các bit dùng để xác lập các chế độ hoạt động của khối CCPx

0000 không cho phép CCPx (hoặc dùng để reset CCPx)

0100 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh xuống tại pin dùng cho khối CCPx.

0101 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh lên tại pin dùng cho khối CCPx.

0110 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh lên thứ 4 tại pin dùng cho khối CCPx.

0111 CCPx hoạt động ở chế độ Capture, "hiện tượng" được thiết lập là mỗi cạnh lên thứ 16 tại pin dùng cho khối CCPx.

1000 CCPx hoạt động ở chế độ Compare, ngõ ra được đưa lên mức cao và bit CCPxIF được set khi các giá trị cần so sánh bằng nhau.

1001 CCPx hoạt động ở chế độ Compare, ngõ ra được xuống mức thấp và bit CCPxIF được set khi các giá trị cần so sánh bằng nhau.

1010 CCPx hoạt động ở chế độ Compare, khi các giá trị cần so sánh bằng nhau, ngắt xảy ra, bit CCPxIF được set và trạng thái pin output không bị ảnh hưởng.

1011 CCPx hoạt động ở chế độ Compare, khi các giá trị cần so sánh bằng nhau, xung trigger đặc biệt (Trigger Special Event) sẽ được tạo ra, khi đó cờ ngắt CCPxIF được set, các pin output không thay đổi trạng thái, CCP1 reset Timer1, CCP2 reset Timer1 và khởi động khối ADC.

11xx CCPx hoạt động ở chế độ PWM.

24. Thanh ghi RCSTA: địa chỉ 18h

Thanh ghi chứa các bit trạng thái và các bit điều khiển quá trình nhận dữ liệu qua chuẩn giao tiếp USART.

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
Bit 7							Bit 0

Bit 7 SPEN Serial Port Enable bit

SPEN = 1 Cho phép cổng giao tiếp USART (pin RC7/RX/DT và RC6/TX/CK).

SPEN = 0 không cho phép cổng giao tiếp USART.

Bit 6 RX9 9-bit Receive Enable bit

RX9 = 1 nhận 9 bit dữ liệu.

RX9 = 0 nhận 8 bit dữ liệu.

Bit 5 SREN Single Receive Enable bit

Ở chế độ USART bất đồng bộ: bit này không cần quan tâm.

Ở chế độ USART Master đồng bộ:

SREN = 1 cho phép chức năng nhận 1 byte dữ liệu (8 bit hoặc 9 bit).

SREN = 0 không cho phép chức năng nhận 1 byte dữ liệu.

Bit 4 CREN Continuous Receive Enable bit

Ở chế độ bất đồng bộ:

CREN = 1 cho phép nhận 1 chuỗi dữ liệu liên tục.

CREN = 0 không cho phép nhận 1 chuỗi dữ liệu liên tục.

Ở chế độ bất đồng bộ:

CREN = 1 cho phép nhận dữ liệu cho tới khi xóa bit CREN.

CREN = 0 không cho phép nhận chuỗi dữ liệu.

Bit 3 ADDEN Address Detect Enable bit

Ở chế độ USART bất đồng bộ 9 bit

ADDEN = 1 cho phép xác nhận địa chỉ, khi bit RSR<8> được set thì ngắt được cho phép thực thi và giá trị trong buffer được nhận vào.

ADDEN = 0 không cho phép xác nhận địa chỉ, các byte dữ liệu được nhận vào và bit thứ 9 có thể được sử dụng như là bit parity.

Bit 2 FERR Framing Error bit

FERR = 1 xuất hiện lỗi "Framing" trong quá trình truyền nhận dữ liệu.

FERR = 0 không xuất hiện lỗi "Framing" trong quá trình truyền nhận dữ liệu.

Bit 1 OERR Overrun Error bit,

OERR = 1 xuất hiện lỗi "Overrun"

OERR = 0 không xuất hiện lỗi "Overrun"

Bit 0 RX9D

Bit này chứa bit dữ liệu thứ 9 của dữ liệu truyền nhận.

25. Thanh ghi TXREG: địa chỉ 19h

Thanh ghi đóng vai trò là buffer đệm 8 bit trong quá trình truyền dữ liệu thông qua chuẩn giao tiếp USART.

26. Thanh ghi RCREG: địa chỉ 1Ah

Thanh ghi đóng vai trò là buffer đệm trong quá trình nhận dữ liệu qua chuẩn giao tiếp USART.

27. Thanh ghi CCPR2L: địa chỉ 1Bh

Thanh ghi chứa 8 bit thấp của khối CCP2.

28. Thanh ghi CCPR2H: địa chỉ 1Ch

Thanh ghi chứa 8 bit cao của khối CCP2.

29. Thanh ghi ADRESH: địa chỉ 1Eh

Thanh ghi chứa byte cao của kết quả quá trình chuyển đổi ADC.

30. Thanh ghi ADCON0: địa chỉ 1Fh

Đây là một trong hai thanh ghi điều khiển khối chuyển đổi ADC. Thanh ghi còn lại là thanh ghi ADCON1 (địa chỉ 9Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

Bit 7:6 **ADCS1:ADCS0**: Các bit lựa chọn tần số chuyển đổi A/D

00 = FOSC/2

01 = FOSC/4

10 = FOSC/32

11 = FRC (xung clock được lấy từ dao động nội RC)

Bit 5:3 **CHS2:CHS0**: Các bit lựa chọn kênh Analog

000: Kênh 0, (AN0)

001: Kênh 1, (AN1)

010: Kênh 2, (AN2)

011: Kênh 3, (AN3)

100: Kênh 4, (AN4)

101: Kênh 5, (AN5)

110: Kênh 6, (AN6)

111: Kênh 7, (AN7)

Bit 2 **GO/ DONE**: Bit báo trạng thái chuyển đổi A/D

Khi bit ADON = 1

1: Quá trình A/D đang thực hiện (Khi chúng ta set bit này lên thì quá trình chuyển đổi sẽ xảy ra, khi quá trình kết thúc nó sẽ tự động được xóa bằng phần mềm).

0: Quá trình A/D không xảy ra hoặc đã hoàn tất.

Bit 1 Không sử dụng, giá trị là 0

Bit 0 **ADON** : Bit cho phép module A/D hoạt động.

1: Nguồn được cung cấp cho A/D

0: Ngưng cung cấp nguồn cho A/D

31. Thanh ghi ADCON1: địa chỉ 9Fh

Thanh ghi chứa các bit điều khiển bộ chuyển đổi ADC (ADC có hai thanh ghi điều khiển là ADCON1 và ADCON0).

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Bit 7 **ADFM**: Bit lựa chọn định dạng kết quả A/D

1: Canh phải, 6 bit cao nhất của thanh ghi ADRESH có giá trị 0

0: Canh trái, 6 bit thấp nhất của thanh ghi ADRESL có giá trị 0

Bit 6 **ADCS2**: Bit lựa chọn clock chuyển đổi A/D

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

Bit 5,4 không sử dụng

Bit 3:0 **PCFG3:PCFG0**: Các bit điều khiển cấu hình các chân ADC

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	Vss	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	Vss	7/1
0010	D	D	D	A	A	A	A	A	VDD	Vss	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	Vss	4/1
0100	D	D	D	D	A	D	A	A	VDD	Vss	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	Vss	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	Vss	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	Vss	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	Vss	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

32. Thanh ghi OPTION_REG: địa chỉ 81h, 181h

Thanh ghi này cho phép nối vaø ghi, cho phép nhiều khiên chòu naêng pull-up của các chân trong PORTB, xác lập các tham số về xung taùcđoäng, cãnh taùc ñoäng của ngàét ngoaïi vi vaø boả ñeám Timer0.

Thanh ghi tùy chọn chứa các bit điều khiển để cấu hình cho các chức năng như: ngắt ngoài, Timer 0 chức năng kéo lên Vdd của các chân Port B, và thời gian chờ của WDT.

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
RBPƯ	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Bit **7 RBPƯ** : Bit cho phép PORTB được kéo lên nguồn.

1: Không cho phép PORTB kéo lên nguồn.

0: Cho phép PORTB kéo lên nguồn.

Bit 6 **INTEDG**: Bit lựa chọn cạnh tác động ngắt (**INTERRUPT EDGE**)

1: Ngắt sẽ được tác động bởi cạnh lên của chân **RB0/INT**

0: Ngắt sẽ được tác động bởi cạnh xuống của chân **RB0/INT**

Bit 5 **T0CS**: Bit lựa chọn nguồn xung Clock cho **Timer 0**

1: Xung Clock cung cấp bởi nguồn ngoài qua chân **RA4/T0CKI**

0: Xung Clock cung cấp bởi nguồn dao động nội.

Bit 4 **T0SE**: Bit lựa chọn cạnh nào của xung clock tác động lên **timer 0**

1: Cạnh xuống

0: Cạnh lên

Bit 3 **PSA**: Bit quyết định tốc độ đếm **PS2:PS0** sẽ tác động lên Timer 0 hay **WDT**

1: Tốc độ đếm **PS2:PS0** sẽ tác động lên **WDT**

0: Tốc độ đếm **PS2:PS0** sẽ tác động lên **Timer 0**

Bit 2-0 **PS2:PS0**: Dùng để lựa chọn tốc độ đếm của timer hay **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

33. Thanh ghi TRISA: địa chỉ 85h

Thanh ghi điều khiển xuất nhập của các pin trong PORTA.

34. Thanh ghi TRISB: địa chỉ 86h, 186h

Thanh ghi điều khiển xuất nhập của các pin trong PORTB.

35. Thanh ghi TRISC: địa chỉ 87h

Thanh ghi điều khiển xuất nhập của các pin trong PORTC.

36. Thanh ghi TRISD: địa chỉ 88h

Thanh ghi điều khiển xuất nhập của các pin trong PORTD.

37. Thanh ghi TRISE: địa chỉ 89h

Thanh ghi điều khiển xuất nhập của các pin trong PORTE, điều khiển cổng giao tiếp song song PSP (Parallel Slave Port).

IBF	OBF	IBOV	SPPMODE	-	2	1	0
Bit 7							Bit 0

Bit 7 BIF Input Buffer Full Status bit

BIF = 1 một Word dữ liệu vừa được nhận và đang chờ CPU đọc vào.

BIF = 0 chưa có Word dữ liệu nào được nhận.

Bit 6 OBF Output Buffer Full Status bit

OBF = 1 Buffer truyền dữ liệu vẫn còn chứa dữ liệu cũ và vẫn chưa được đọc.

OBF = 0 Buffer truyền dữ liệu đã được đọc.

Bit 5 IBOV Input Buffer Overflow Detect bit

IBOV = 1 dữ liệu được ghi lên buffer trong khi dữ liệu cũ vẫn chưa được đọc.

IBOV = 0 buffer chưa bị tràn.

Bit 4 PSPMODE Parallel Slave Port Mode Select bit

PSPMODE = 1 Cho phép PSP, PORTD đóng vai trò là cổng giao tiếp song song PSP.

PSPMODE = 0 Không cho phép PSP.

Bit 3 Không cần quan tâm và mặc định mang giá trị 0.

Bit 2 Bit2 Direction Control for pin .

Bit2 = 1 Input

Bit2 = 0 Output

Bit 1 Bit1 Direction Control for pin

Bit1 = 1 Input

Bit1 = 0 Output

Bit 0 Bit0 Direction Control for pin

Bit0 = 1 Input

Bit0 = 0 Output

38. Thanh ghi PIE1: địa chỉ 8Ch

Thanh ghi chứa các bit cho phép các ngắt ngoại vi.

PSPIE	ADIE	RCIE	TXIE	SSPIE	CCPIE1	TMR2IE	TMR1IE
Bit 7							Bit 0

Bit 7 PSPIE Parallel Slave Port Read/Write Interrupt Enable bit

PSPIE = 1 cho phép ngắt PSP read/write.

PSPIE = 0 không cho phép ngắt PSP read/write.

Bit 6 ADIE ADC (A/D converter) Interrupt Enable bit

ADIE = 1 cho phép ngắt ADC.

ADIE = 0 không cho phép ngắt ADC.

Bit 5 RCIE USART Receive Interrupt Enable bit

RCIE = 1 cho phép ngắt nhận USART

RCIE = 0 không cho phép ngắt nhận USART

Bit 4 TXIE USART Transmit Interrupt Enable bit

TXIE = 1 cho phép ngắt truyền USART

TXIE = 0 không cho phép ngắt truyền USART

Bit 3 SSPIE Synchronous Serial Port Interrupt Enable bit

SSPIE = 1 cho phép ngắt SSP

SSPIE = 0 không cho phép ngắt SSP

Bit 2 CCP1IE CCP1 Interrupt Enable bit

CCP1IE = 1 cho phép ngắt CCP1

CCP1IE = 0 không cho phép ngắt CCP1

Bit 1 TMR2IE TMR2 to PR2 Match Interrupt Enable bit

TMR2IE = 1 cho phép ngắt.

TMR2IE = 0 không cho phép ngắt.

Bit 0 TMR1IE TMR1 Overflow Interrupt Enable bit

TMR1IE = 1 cho phép ngắt.

TMR1IE = 0 không cho phép ngắt.

39. Thanh ghi PIE2: địa chỉ 8Dh

Thanh ghi chứa các bit cho phép các ngắt ngoại vi.

-	CMIE	-	EEIE	BCLIE	-	-	CCP2IE
Bit 7							Bit 0

Bit 7, 5, 2, 1 Không cần quan tâm và mặc định mang giá trị 0.

Bit 6: CMIE Comparator Interrupt Enable bit

CMIE = 1 Cho phép ngắt của bộ so sánh.

CMIE = 0 Không cho phép ngắt.

Bit 4: EEIE EEPROM Write Operation Interrupt Enable bit

EEIE = 1 Cho phép ngắt khi ghi dữ liệu lên bộ nhớ EEPROM.

EEIE = 0 Không cho phép ngắt khi ghi dữ liệu lên bộ nhớ EEPROM.

Bit 3: BCLIE Bus Collision Interrupt Enable bit

BCLIE = 1 Cho phép ngắt.

BCLIE = 0 Không cho phép ngắt.

Bit 0: CCP2IE CCP2 Interrupt Enable bit

CCP2IE = 1 Cho phép ngắt.

CCP2IE = 0 Không cho phép ngắt

40. Thanh ghi PCON: địa chỉ 8Eh

Thanh ghi điều khiển chứa các cờ hiệu cho biết trạng thái các chế độ reset của vi điều khiển.

Bit 7, 6, 5, 4, 3, 2 Không cần quan tâm và mặc định mang giá trị 0.

Bit 1 Power-on Reset Status bit

= 1 không có sự tác động của Power-on Reset.

= 0 có sự tác động của Power-on reset.

Bit 0 Brown-out Reset Status bit

= 1 không có sự tác động của Brown-out reset.

= 0 có sự tác động của Brown-out reset.

41. Thanh ghi SSPCON2: địa chỉ 91h

Thanh ghi điều khiển các chế độ hoạt động của chuẩn giao tiếp I2C.

GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
Bit 7							Bit 0

Bit 7 GCEN General Call Enable bit

GCEN = 1 Cho phép ngắt khi địa chỉ 0000h được nhận vào thanh ghi SSPSR (địa chỉ của chế độ General Call Address).

GCEN = 0 Không cho phép chế độ địa chỉ trên.

Bit 6 ACKSTAT Acknowledge Status bit (bit này chỉ có tác dụng khi truyền dữ liệu ở chế độ I2C Master mode).

ACKSTAT = 1 nhận được xung từ I2C Slave.

ACKSTAT = 0 chưa nhận được xung .

Bit 5 ACKDT Acknowledge Data bit (bit này chỉ có tác dụng khi nhận dữ liệu ở chế độ I2C Master mode).

ACKDT = 1 chưa nhận được xung .

ACKDT = 0 Đã nhận được xung .

Bit 4 ACKEN Acknowledge Sequence Enable bit (bit này chỉ có tác dụng khi nhận dữ liệu ở chế độ I2C Master mode)

ACKEN = 1 cho phép xung xuất hiện ở 2 pin SDA và SCL khi kết thúc quá trình nhận dữ liệu.

ACKEN = 0 không cho phép tác động trên.

Bit 3 RCEN Receive Enable bit (bit này chỉ có tác dụng ở chế độ I2C Master mode).

RCEN = 1 Cho phép nhận dữ liệu ở chế độ I2C Master mode.

RCEN = 0 Không cho phép nhận dữ liệu.

Bit 2 PEN Stop Condition Enable bit

PEN = 1 cho phép thiết lập điều kiện Stop ở 2 pin SDA và SCL.

PEN = 0 không cho phép tác động trên.

Bit 1 RSEN Repeated Start Condition Enable bit

RSEN = 1 cho phép thiết lập điều kiện Start lặp lại liên tục ở 2 pin SDA và SCL.

RSEN = 0 không cho phép tác động trên.

Bit 0 SEN Start Condition Enable/Stretch Enable bit

Ở chế độ Master mode:

SEN = 1 cho phép thiết lập điều kiện Start ở 2 pin SDA và SCL.

SEN = 0 không cho phép tác động trên.

Ở chế độ Slave mode:

SEN = 1 cho phép khóa xung clock từ pin SCL của I2C Master.

Không cho phép tác động trên.

42. Thanh ghi PR2: địa chỉ 92h

Thanh ghi dùng để ấn định trước giá trị đếm cho Timer2. Khi vi điều khiển được reset, PR2 mang giá trị FFh. Khi ta đưa một giá trị vào thanh ghi PR2, Timer2 sẽ đếm từ 00h cho đến khi giá trị bộ đếm của Timer2 bằng với giá trị của bộ đếm trong thanh ghi PR2. Như vậy mặc định Timer2 sẽ đếm từ 00h đến FFh.

43. Thanh ghi SSPADD: địa chỉ 93h

Thanh ghi chứa địa chỉ của vi điều khiển khi hoạt động ở chuẩn giao tiếp I2C Slave mode. Khi không dùng để chứa địa chỉ (I2C Master mode) SSPADD được dùng để chứa giá trị tạo ra xung clock đồng bộ tại pin SCL.

44. Thanh ghi SSPSTAT: địa chỉ 94h

Thanh ghi chứa các bit trạng thái của chuẩn giao tiếp MSSP.

SMP	CKE	D/A	P	S	R/W	UA	BF
Bit 7							Bit 0

Khi MSSP hoạt động ở chế độ SPI:

Bit 7 SMP Sample bit

SPI Master mode:

SMP = 1 dữ liệu được lấy mẫu (xác định trạng thái logic) tại thời điểm cuối xung clock.

SMP = 0 dữ liệu được lấy mẫu tại thời điểm giữa xung clock.

SPI Slave mode: bit này phải được xóa về 0.

Bit 6 CKE SPI Clock Select bit

CKE = 1 SPI Master truyền dữ liệu khi xung clock chuyển từ trạng thái tích cực đến trạng thái chờ.

CKE = 0 SPI Master truyền dữ liệu khi xung clock chuyển từ trạng thái chờ đến trạng thái tích cực. (trạng thái chờ được xác định bởi bit CKP (SSPCON<4>).

Bit 5 bit.

Bit này chỉ có tác dụng ở chế độ I2C mode.

Bit 4 P Stop bit

Bit này chỉ sử dụng khi MSSP ở chế độ I2C.

Bit 3 S Start bit

Bit này chỉ có tác dụng khi MSSP ở chế độ I2C.

Bit 2 bit information

Bit này chỉ có tác dụng khi MSSP ở chế độ I2C.

Bit 1 UA Update Address bit

Bit này chỉ có tác dụng khi MSSP ở chế độ I2C.

Bit 0 BF Buffer Status bit

BF = 1 thanh ghi đệm SSPBUF đã có dữ liệu.

BF = 0 thanh ghi đệm SSPBUF chưa có dữ liệu.

Khi hoạt động ở chế độ I2C

Bit 7 SPM Slew Rate Control bit

SPM = 1 dùng tốc độ chuẩn (100 KHz và 1 MHz).

SPM = 0 dùng tốc độ cao (400 KHz).

Bit 6 CKE MSBus Select bit

CKE = 1 cho phép MSBus.

CKE = 0 không cho phép MSBus.

Bit 5 bit

I2C Master mode: không quan tâm.

= 1 byte vừa truyền đi hoặc nhận được là dữ liệu.

= 0 byte vừa truyền đi hoặc nhận được là địa chỉ.

Bit 4 P Stop bit

P = 1 vừa nhận được bit Stop.

P = 0 chưa nhận được bit Stop.

Bit 3 S Start bit

S = 1 vừa nhận được bit Start.

S = 0 chưa nhận được bit Start.

Bit 2 bit information

I2C Slave mode:

= 1 đọc dữ liệu.

= 0 ghi dữ liệu.

I2C Master mode:

= 1 đang truyền dữ liệu.

= 0 không truyền dữ liệu.

Bit 1 UA Update Address

Bit này chỉ có tác dụng đối với chế độ I2C Slave mode 10 bit địa chỉ.

UA = 1 vi điều khiển cần cập nhật thêm địa chỉ từ thanh ghi SSPADD.

UA = 0 không cần cập nhật thêm địa chỉ.

Bit 0 BF Buffer Full Status bit

BF = 1 Thanh ghi SSPBUF đang chứa dữ liệu truyền đi hoặc nhận được.

BF = 0 thanh ghi SSPBUF không có dữ liệu.

45. Thanh ghi TXSTA: địa chỉ 98h

Thanh ghi chứa các bit trạng thái và điều khiển việc truyền dữ liệu thông qua chuẩn giao tiếp USART.

CSRC	TX-9	TXEN	SYNC	-	BRGH	TRMT	TX9D
Bit 7							Bit 0

Bit 7 CSRC Clock Source Select bit

Ở chế độ bất đồng bộ: không cần quan tâm. Ở chế độ đồng bộ:

CSRC = 1 Master mode (xung clock được lấy từ bộ tạo xung BRG).

CSRC = 0 Slave mode (xung clock được nhận từ bên ngoài).

Bit 6 TX-9 9-bit Transmit Enable bit

TX-9 = 1 truyền dữ liệu 9 bit.

TX-9 = 0 truyền dữ liệu 8 bit.

Bit 5 TXEN Transmit Enable bit

TXEN = 1 cho phép truyền.

TXEN = 0 không cho phép truyền.

Bit 4 SYNC USART Mode Select bit

SYNC = 1 dạng đồng bộ

SYNC = 0 dạng bất đồng bộ.

Bit 3 Không cần quan tâm và mặc định mang giá trị 0.

Bit 2 BRGH High Baud Rate Select bit, Bit này chỉ có tác dụng ở chế độ bất đồng bộ.

BRGH = 1 tốc độ cao.

BRGL = 0 tốc độ thấp.

Bit 1 TRMT Transmit Shift Register Status bit

TRMT = 1 thanh ghi TSR không có dữ liệu.

TRMT = 0 thanh ghi TSR có chứa dữ liệu.

Bit 0 TX9D

Bit này chứa bit dữ liệu thứ 9 khi dữ liệu truyền nhận là 9 bit.

45. Thanh ghi SPBRG: địa chỉ 99h

Thanh ghi chứa giá trị tạo xung clock cho bộ tạo xung BRG (Baud Rate Generator).

Tần số xung clock do BRG tạo ra được tính theo các công thức trong bảng sau:

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = $F_{osc}/(64 (X + 1))$	Baud Rate = $F_{osc}/(16 (X + 1))$
1	(Synchronous) Baud Rate = $F_{osc}/(4 (X + 1))$	N/A

46. Thanh ghi CMCON: địa chỉ 9Ch

Thanh ghi điều khiển và chỉ thị các trạng thái cũng như kết quả của bộ so sánh.

C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
Bit 7							Bit 0

Bit 7 C2OUT Comparator 2 (C2) Output bit

Khi C2INV = 0

C2OUT = 1 khi (pin VIN+ của C2) > (pin VIN- của C2).

C2OUT = 0 khi (pin VIN+ của C2) < (pin VIN- của C2).

Khi C2INV = 1

C2OUT = 1 khi (pin VIN+ của C2) < (pin VIN- của C2).

$C2OUT = 0$ khi (pin VIN+ của C2) > (pin VIN- của C2).

Bit 6 C1OUT Comparator 1 (C1) Output bit

Khi C1INV = 0

$C1OUT = 1$ khi (pin VIN+ của C1) > (pin VIN- của C1).

$C1OUT = 0$ khi (pin VIN+ của C1) < (pin VIN- của C1).

Khi C1INV = 1

$C1OUT = 1$ khi (pin VIN+ của C1) < (pin VIN- của C1).

$C1OUT = 0$ khi (pin VIN+ của C1) > (pin VIN- của C1).

Bit 5 C2INV Comparator 2 Output Conversion bit

$C2INV = 1$ ngõ ra C2 được đảo trạng thái.

$C2INV = 0$ ngõ ra C2 không đảo trạng thái.

Bit 4 C1INV Comparator 1 Output Conversion bit

$C1INV = 1$ ngõ ra C1 được đảo trạng thái.

$C1INV = 0$ ngõ ra C1 không đảo trạng thái.

Bit 3 CIS Comparator Input Switch bit

Bit này chỉ có tác dụng khi CM2:CM0 = 110

$CIS = 1$ khi pin VIN- của C1 nối với RA3/AN3 và pin VIN- của C2 nối với RA2/AN2

$CIS = 0$ khi pin VIN- của C1 nối với RA0/AN0 và pin VIN- của C2 nối với RA1/AN1

Bit 2-0 CM2:CM0 Comparator Mode bit

Các bit này đóng vai trò trong việc thiết lập các cấu hình hoạt động của bộ Comparator.

47. Thanh ghi CVRCON: địa chỉ 9Dh

Thanh ghi điều khiển bộ tạo điện áp so sánh khi bộ Comparator

CVREN	CVROE	CVRR	-	CVR3	CVR2	CVR1	CVR0
Bit 7							Bit 0

Bit 7 CVREN Comparator Voltage Reference Enable bit.

$CVREN = 1$ bộ tạo điện áp so sánh được cấp điện áp hoạt động.

$CVREN = 0$ bộ tạo điện áp so sánh không được cấp điện áp hoạt động.

Bit 6 CVROE Comparator VREF Output Enable bit

$CVROE = 1$ điện áp do bộ tạo điện áp so sánh tạo ra được đưa ra pin RA2.

$CVROE = 0$ điện áp do bộ tạo điện áp so sánh tạo ra không được đưa ra ngoài.

Bit 5 CVRR Comparator VREF Range Selection bit

$CVRR = 1$ một mức điện áp có giá trị $VDD/24$ (điện áp do bộ tạo điện áp so sánh tạo ra có giá trị từ 0 đến $0.75VDD$).

CVRR = 0 một mức điện áp có giá trị VDD/32 (điện áp do bộ tạo điện áp so sánh tạo ra có giá trị từ 0.25 đến 0.75VDD).

Bit 4 Không cần quan tâm và mặc định mang giá trị 0.

Bit 3-0 CVR3:CVR0 Các bit chọn điện áp ngõ ra của bộ tạo điện áp so sánh.

Khi CVRR = 1:

Điện áp tại pin RA2 có giá trị $CVREF = (CVR<3:0>/24)*VDD$.

Khi CVRR = 0

Điện áp tại pin RA2 có giá trị $CVREF = (CVR<3:0>/32)*VDD + \frac{1}{4}VDD$.

48. Thanh ghi ADRESL: địa chỉ 9Eh

Thanh ghi chứa các bit thấp của kết quả bộ chuyển đổi A/D (8 bit cao chứa trong thanh ghi ADRESH địa chỉ 1Eh).

50. Thanh ghi EEDATA: địa chỉ 10Ch

Thanh ghi chứa byte thấp của dữ liệu trong quá trình ghi đọc trên bộ nhớ dữ liệu EEPROM.

51. Thanh ghi EEADR: địa chỉ 10Dh

Thanh ghi chứa byte thấp của địa chỉ trong quá trình ghi đọc trên bộ nhớ dữ liệu EEPROM.

52. Thanh ghi EEDATH: địa chỉ 10Eh

Thanh ghi chứa byte cao của dữ liệu trong quá trình ghi đọc trên bộ nhớ dữ liệu EEPROM (thanh ghi này chỉ sử dụng 6 bit thấp).

53. Thanh ghi EEADRH: địa chỉ 10Fh

Thanh ghi chứa byte cao của địa chỉ trong quá trình ghi đọc trên bộ nhớ dữ liệu EEPROM (thanh ghi này chỉ sử dụng 4 bit thấp).

54. Thanh ghi EECON1: địa chỉ 18Ch

Thanh ghi điều khiển bộ nhớ EEPROM.

EEPGD	-	-	-	WRERR	WREN	WR	RD
Bit 7							Bit 0

Bit 7 EEGPD Program/Data EEPROM Select bit

EEPGD = 1 truy xuất bộ nhớ chương trình.

EEPGD = 0 truy xuất bộ nhớ dữ liệu.

Bit 6-4 Không cần quan tâm và mặc định mang giá trị 0.

Bit 3 WRERR EEPROM Error Flag bit

WRERR = 1 quá trình ghi lên bộ nhớ bị gián đoạn và không thể tiếp tục (do các chế độ Reset WDT hoặc).

WRERR = 0 quá trình ghi lên bộ nhớ hoàn tất.

Bit 2 WREN EEPROM Write Enable bit

WREN = 1 cho phép ghi.

WREN = 0 không cho phép ghi.

Bit 1 WR Write Control bit

WR = 1 ghi dữ liệu. Bit này chỉ được set bằng chương trình và tự động xóa về 0 khi quá trình ghi dữ liệu hoàn tất.

WR = 0 hoàn tất quá trình ghi dữ liệu.

Bit 0 RD Read Control bit

RD = 1 đọc dữ liệu. Bit này chỉ được set bằng chương trình và tự động xóa về 0 khi quá trình đọc dữ liệu hoàn tất.

RD = 0 quá trình đọc dữ liệu không xảy ra.

55.Thanh ghi EECON2: địa chỉ 18Dh.

Đây là một trong 2 thanh ghi điều khiển bộ nhớ EEPROM. Tuy nhiên đây không phải là thanh ghi vật lí thông thường và không cho phép người sử dụng truy xuất dữ liệu trên thanh ghi.