TUT01.01.PVN

Gửi đến:	www.picvietnam.com
Nội dung:	Lập trình cho PIC bằng CCS ver3.242
	MICROSOFT WORD

Tóm tắt:

Tài liệu hướng dẫn sử dụng trình biên dịch CCS cho lập trình PIC. Tìm hiểu tông quan về CCS và cách tạo một Project trong CCS.

Chương trình mẫu cho PIC16F877

Các ví dụ lập trình đơn giản: quét LED, ADC, RS232…

1. Tổng quan về CCS

1.1. Vì sao ta sử dung CCS ?

Sự ra đời của một loại vi điều khiển đi kèm với việc phát triển phần mềm ứng dụng cho việc lập trình cho con vi điều khiển đó. Vi điều khiển chỉ hiểu và làm việc với hai con số 0 và 1. Ban đầu để việc lập trình cho VĐK là làm việc với dãy các con số 0 và 1. Sau này khi kiến trúc của Vi điều khiển ngày càng phức tạp, số lượng thanh ghi lệnh nhiều lên, việc lập trình với dãy các số 0 và 1 không còn phù hợp nữa, đòi hỏi ra đời một ngôn ngữ mới thay thế. Và ngôn ngữ lập trình Assembly. Ở đây ta không nói nhiều đến Assmebly. Sau này khi ngôn ngữ C ra đời, nhu cầu dùng ngôn ngữ C đề thay cho ASM trong việc mô tả các lệnh lập trình cho Vi điều khiển một cách ngắn gọn và dễ hiểu hơn đã dẫn đến sự ra đời của nhiều chương trình soạn thảo và biên dịch C cho Vi điều khiển : Keil C, HT-PIC, MikroC, CCS…

Tôi chọn CCS cho bài giới thiệu này vì CCS là một công cụ lập trình C mạnh cho Vi điều khiển PIC. Những ưu và nhược điểm của CCS sẽ được đề cập đến trong các phần dưới đây.

1.2. Giới thiệu về CCS ?

CCS là trình biên dịch lập trình ngôn ngữ C cho Vi điều khiển PIC của hãng Microchip. Chương trình là sự tích hợp của 3 trình biên dich riêng biết cho 3 dòng PIC khác nhau đó là:

- PCB cho dòng PIC 12-bit opcodes
- PCM cho dòng PIC 14-bit opcodes
- PCH cho dòng PIC 16 và 18-bit

Tất cả 3 trình biên dich này đuợc tích hợp lại vào trong một chương trình bao gồm cả trình soạn thảo và biên dịch là CCS, phiên bản mới nhất là PCWH Compiler Ver 3.227

Giống như nhiều trình biên dich C khác cho PIC, CCS giúp cho người sử dụng nắm bắt nhanh được vi điều khiển PIC và sử dụng PIC trong các dự án. Các chương trình diều khiển sẽ được thực hiện nhanh chóng và đạt hiệu quả cao thông qua việc sử dụng ngôn ngữ lạp trình cấp cao – Ngôn ngữ C

Tài liệu hướng dẫn sử dụng có rất nhiều, nhưng chi tiết nhất chính là bản Help đi kèm theo phần mềm (tài liệu Tiếng Anh). Trong bản trợ giúp nhà sản xuất đã mô tả rất nhiều về hằng, biến, chỉ thị tiền xủa lý, cấu trúc các câu lệnh trong chương trình, các hàm tạo sẵn cho người sử dụng… Ngoài ra về Tiếng Việt cũng có bản dịch của tác giả **Trần Xuân Trường, SV K2001 DH BK HCM**. Tài liệu này dịch trên cơ sở bản Help của CCS, tuy rằng chưa đầy đủ nhưng đây là một tài liệu hay, nếu bạn tìm hiểu về PIC và CCS thì nên tìm tài liệu này về đọc. Địa chỉ Download tài liệu: www.picvietnam.com -> Mục nói về CCS.

2. Tạo PROJECT đầu tiên trong CCS

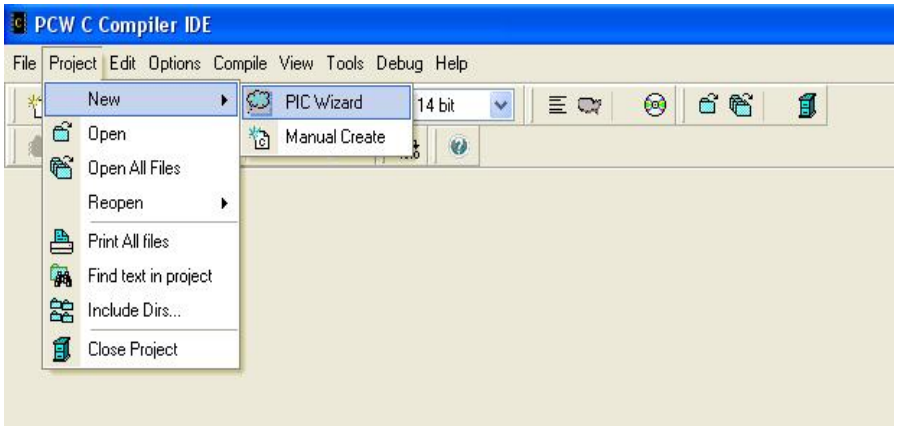
Để tạo một Project trong CCS có nhiều cách, có thể dùng Project Wizard, Manual Creat, hay đơn giản là tạo một Files mới và thêm vào đó các khai báo ban đầu cần thiết và "bắt buộc".

Dưới đây sẽ trình bày cách tạo một project hợp lệ theo cả 3 phương pháp. Một điều ta cần chú ý khi tạo một Project đó là: khi tạo bắt cứ một Project nào mới thì ta nên tạo một thư mục mới với tên liên quan đến Project ta định làm, rồi lưu các files vào đó. Khi lập trình và biên dịch, CCS sẽ tạo ra rất nhiều files khác nhau, do đó nếu để chung các Project trogn một thư mục sẽ rất mất thời gian trong việc tìm kiếm sau này. Đây cũng là quy tắc chung khi ta làm việc với bất kỳ phần mềm nào, thiết kế mạch hay lập trình.

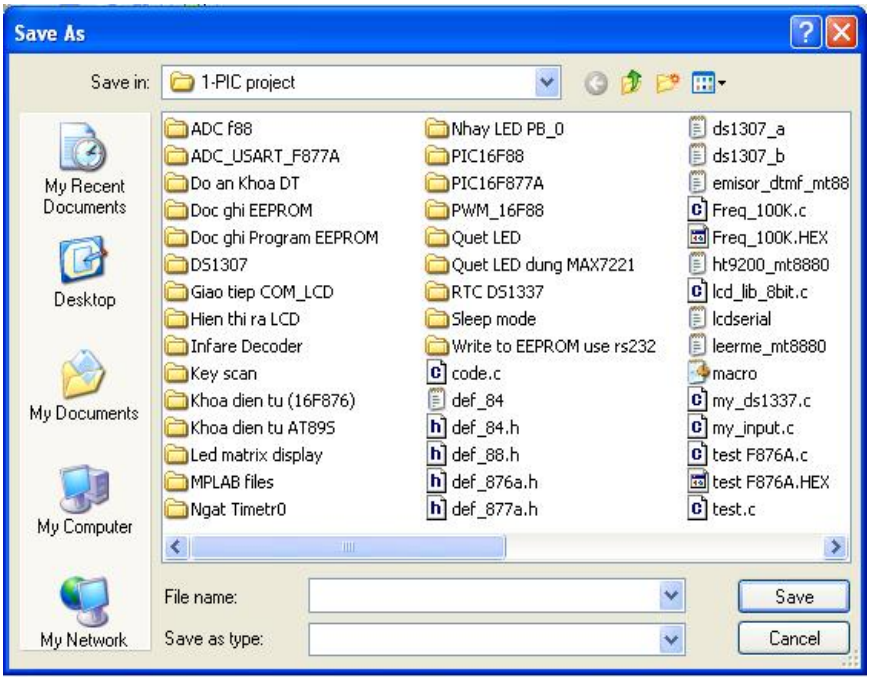
Việc đầu tiên bạn cần làm là khởi động máy tính và bật chương trình PIC C Compiler.

2.1. Tạo một PROJECT sử dụng PIC Wizard

Trước hết bạn khởi động chương trình làm việc PIC C Compiler. Từ giao diện chương trình bạn di chuột chọn ***Project -> New -> PIC Wizard*** nhấn nút trái chuột chọn.



Sau khi nhấn chuột, một cửa sổ hiện ra yêu cầu ban nhập tên Files cần tạo. Bạn tạo một thư mục mới, vào thư mục đó và lưu tên files cần tạo tại đây.



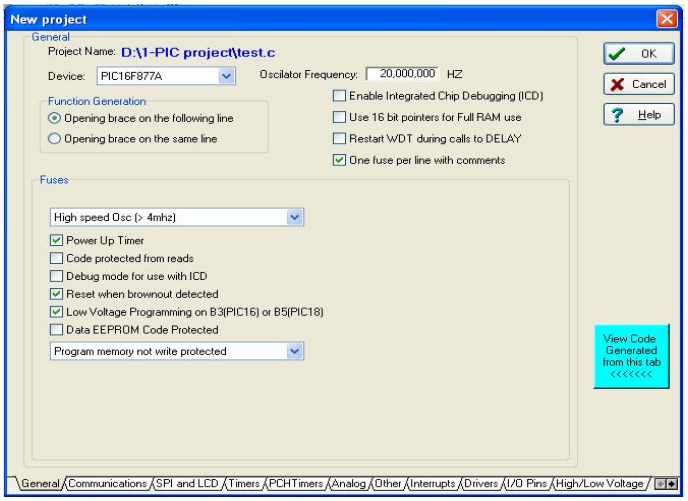
Hình2.0: Cửa sổ Save As

Như vậy là xong bước đầu tiên. Sau khi nhấn nút ***Save,*** một cửa sổ ***New Project*** hiện ra. Trong của sổ này bao gồm rất nhiều Tab, mỗi Tab mô tả về một vài tính năng của con PIC. Ta sẽ chọn tính năng sử dụng tại các Tab tương ứng.

Dưới đây sẽ trình bày ý nghĩa từng mục chọn trong mỗi Tab. Các mục chọn này chính là đề cập đến các tính năng của một con PIC, tùy theo từng loại mà sẽ có các Tab tương ứng. Đối với từng dự án khác nhau, khi ta cần sử dụng tính năng nào của con PIC thì ta sẽ chọn mục đó. Tổng cộng có 13 Tab đẻ ta lưa chọn. Tôi giới thiệu những Tab chính thường hay được sử dụng.

2.1.1. Tab General

Tab General cho phép ta lựa chọn loại PIC mà ta sử dụng và một số lựa chọn khác như chọn tần số thạch anh dao động, thiết lập các bit CONFIG nhằm thiết lập chế độ hoạt động cho PIC.



Hình 2.1: Tab General

- **Device**: Liệt kê danh sách các loại PIC 12F, 16F, 18F… Ta sẽ chọn tên Vi điều khiển PIC mà ta sử dụng trong dự án. Lấy ví dụ chọn PIC16F877A
- **Oscilator Frequency:** Tần số thạch anh ta sử dụng, chọn 20 MHz (tùy từng loại)
- **Fuses:** Thiết lập các bit Config như: Chế độ dao động (HS, RC, Internal), chế độ bảo vệ Code, Brownout detected…
- Chọn kiểu con trỏ RAM là 16-bit hay 8-bit

2.1.2. Tab Communications

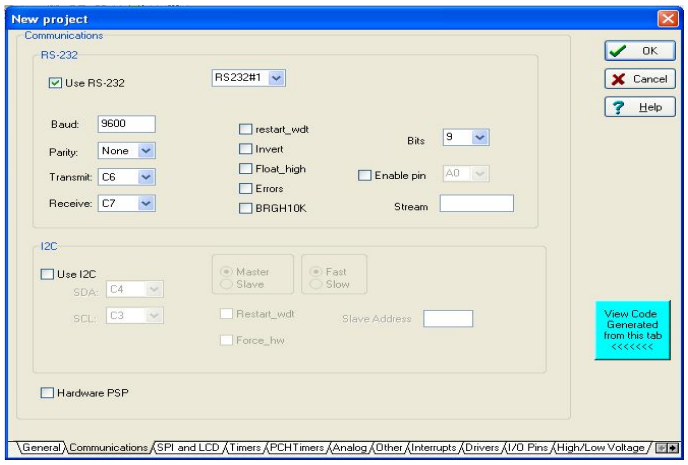
Tab Communications liệt kê các giao tiếp nối tiếp mà một con PIC hỗ trợ, thường là RS232 và I2C, cùng với các lựa chọn để thiết lập chế độ hoạt động cho từng loại giao tiếp.

Giao tiếp RS232

Mỗi một Vi điều khiển PIC hỗ trợ một cổng truyền thông RS232 chuẩn. Tab này cho phép ta lựa chọn chân Rx, Tx, tốc độ Baud, Data bit, Bit Parity…

Giao tiếp I2C

Để sử dụng I2C ta tích vào nút chọn Use I2C, khi đó ta có các lựa chọn: Chân SDA, SCL, Tốc độ truyền (Fast - Slow), chế độ Master hay Slave, địa chỉ cho Salve.

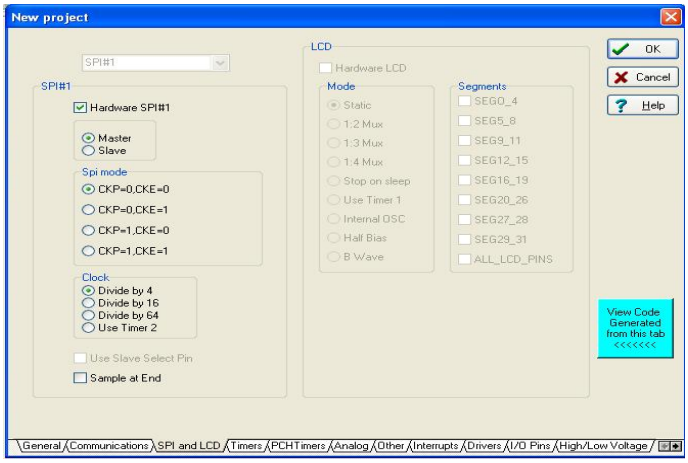


Hình 2.2: Tab Communications

2.1.3. Tab SPI and LCD

Tab này liệt kê cho người dùng các lựa chọn đối với giao tiếp nối tiếp SPI, chuẩn giao tiếp tốc độ cao mà PIC hỗ trợ về phần cứng. Chú ý khi ta dùng I2C thì không thể dùng SPI và ngược lại. Để có thể sử dụng cả hai giao tiếp này cùng một lúc thì buộc một trong 2 giao tiếp phải lập trình bằng phần mềm (giống như khi dùng I2C cho các chip AT8051, không có hỗ trợ phần cứng SSP).

Phần cấu hình cho LCD dành cho các chíp dòng 18F và 30F.

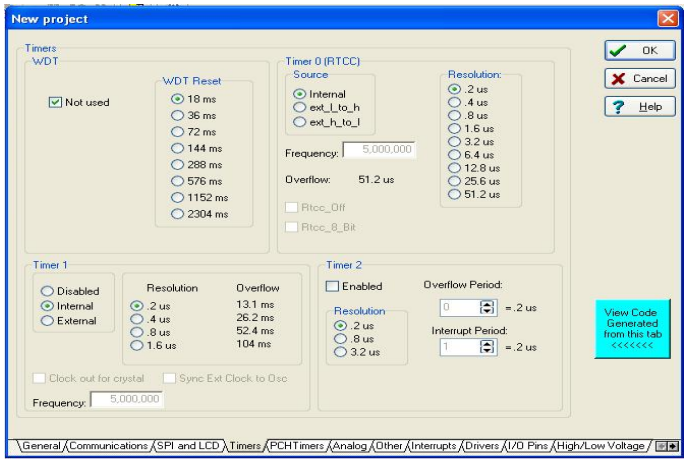


Hình 2.3: Tab SPI and LCD

2.1.4. Tab Timer

Liệt kê các bộ ***đếm/định thời*** mà các con PIC dòng Mid-range có: Timer0, timer1, timer2, WDT…

Trong các lựa chọn cấu hình cho các bộ đếm /định thời có: chọn nguồn xung đồng hồ (trong/ngoài), khoảng thời gian xảy ra tràn…

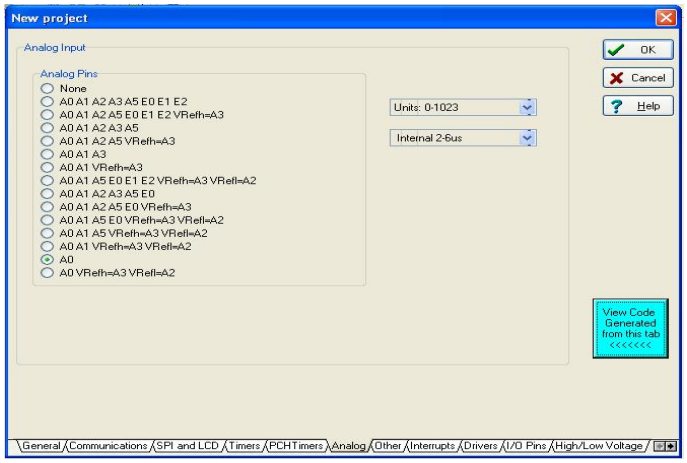


Hình 2.4: Tab Timer

2.1.5. Tab Analog

Liệt kê các lựa chọn cho bộ chuyển đổi tương tự/số (ADC) của PIC. Tùy vào từng IC cụ thể mà có các lựa chọn khác nhau, bao gồm:

- Lựa chọn cổng vào tương tự
- Chọn chân điện áp lấy mẫu (Vref)
- Chọn độ phân giải: 8-bit = 0 ~ 255 hay 10-bit = 0~1023
- Nguồn xung đồng hồ cho bộ ADC (trong hay ngoài), từ đó mà ta có được tốc độ lấy mẫu, thường ta chọn là ***internal 2-6 us.***
- Khi không sử dụng bộ ADC ta chọn **none**



Hình 2.5: Tab Analog

2.1.6. Tab Other

Tab này cho phép ta thiết lập các thông số cho các bộ **Capture/Comparator/PWM.**

Capture - Bắt giữ

- Chọn bắt giữ xung theo sườn dương (**rising edge**) hay sườn âm (**falling edge**) của xung vào
- Chọn bắt giữ sau 1, 4 hay 16 xung (copy giá trị của TimerX vào thanh ghi lưu trữ CCCPx sau 1, 4 hay 16 xung).

Compare - So sánh

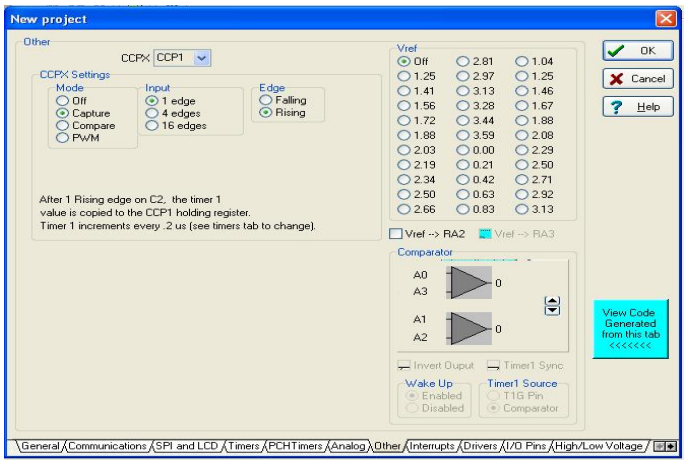
- Ta có các lựa chọn thực hiện lệnh khi xayư ra bằng nhau giữa 2 đối tượng so sánh là giá trị của Timer1 với giá trị lưu trong thanh ghi để so sánh. Bao gồm:
  - Thực hiện ngắt và thiết lập mức 0
  - Thực hiện ngắt và thiết lập mức 1
  - Thực hiện ngắt nhưng không thay đổi trạng thái của chân PIC.
  - Đưa Timer1 về 0 nhưng không thay đổi trạng thái chân.

PWM - Điều chế độ rộng xung

- Lựa chọn về tần số xung ra và duty cycle. Ta có thể lựa chọn sẵn hay tự chọn tần số, tất nhiên tần số ra phải nằm trong một khoảng nhất định.

Comparator - So sánh điện áp

- Lựa chọn mức điện áp so sánh Vref. Có rất nhiều mức điện áp để ta lựa chọn. Ngoài ra ta còn có thể lựa chọn cho đầu vào của các bộ so sánh.

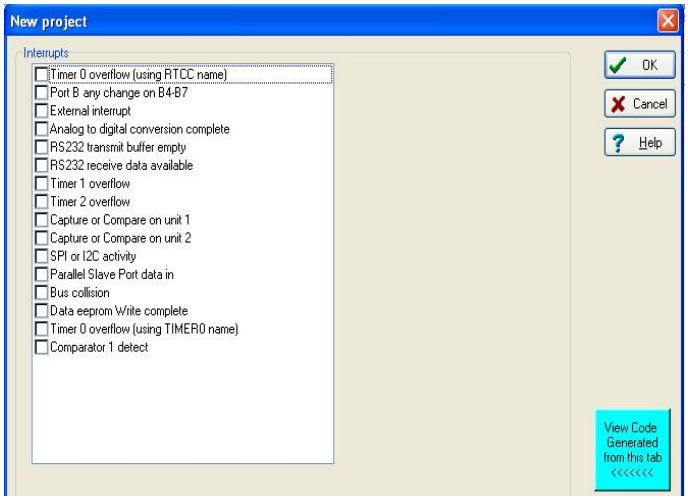


Hình 2.6: Tab Other

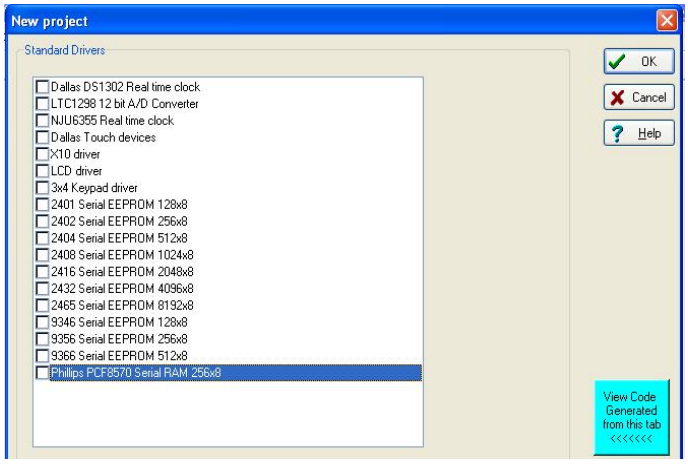
2.1.7. Tab Interrupts và Tab Driver

Tab Interrupts cho phép ta lựa chọn nguồn ngắt mà ta muốn sử dụng. Tùy vào từng loại PIC mà số lượng nguồn ngắt khác nhau, bao gồm: ngắt ngoài 0(INT0), ngắt RS232, ngắt Timer, ngắt I2C-SPI, ngắt onchange PORTB.v.v…

Tab Drivers được dùng để lựa chọn những ngoại vi mà trình dịch đã hỗ trợ các hàm giao tiếp. Đây là nhưng ngoại vi mà ta sẽ kết nối với PIC, trong các IC mà CCS hỗ trợ, đáng chú ý là các loại EEPROM như 2404, 2416, 2432, 9346, 9356…Ngoài ra còn có IC RAM PCF8570, IC thời gian thực DS1302, Keypad 3x4, LCD, ADC… Chi tiết ta có thể xem trong thư mục Driver của chương trình**: \...\PICC\Drivers**



Hình 2.7: Tab Interrupts



Hình 2.8: Tab Driver

Sau các bước chọn trên, ta nhấn OK để kết thúc quá trình tạo một Project trong CCS, một Files ***ten_project.c*** được tạo ra, chứa những khai báo cần thiết cho PIC trong một Files ***ten_project.h.*** Dưới đây là nội dung một files chương trình mẫu.

Chuong_trinh_mau.c

```
#include "D:\1-PIC project\chuong trinh test.HEX.h"
#int_EXT
EXT_isr()
{
// Code here
}
Void Chuong_trinh_con()
{
// Code here
}
void main()
{
  setup_adc_ports(AN0);
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);
  enable_interrupts(INT_EXT);
  enable_interrupts(INT_TBE);
  enable_interrupts(INT_RDA);
  enable_interrupts(GLOBAL);
// Enter your code here
}
```

Chuong_trinh_mau.h

```
#include <16F877A.h>
#device adc=8

#FUSES NOWDT,HS,NOPUT,NOPROTECT,NODEBUG,
#use delay(clock=20000000)
#define SRAM_SCL   PIN_C3
#define SRAM_SDA   PIN_C4
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
```

2.2. Mẫu chương trình chuẩn cho lập trình CCS

Phần trên ta đã tìm hiểu cách tạo một Project trong CCS, tuy nhiên theo cách đó mất khá nhiều thời gian, mặt khác mỗi người lập trình sẽ tạo ra nhưng form tài liệu theo cách riêng khác nhau, không đồng nhất. Tài liệu không được chuẩn hóa sẽ gây một số khó khăn cho người đọc, người đọc có thể không hiểu hết những gì mà người lập trình muốn diễn đạt. Với mục đích đưa ra một form tài liệu chuẩn cho việc lập trình bằng CCS, qua tham khảo bản mẫu cho lập trình bằng ASM của anh Falleaf trên diễn đàn WWW.PICVIETNAM.COM tôi đưa ra đây một form tài liệu cho việc viết lập trình bằng CCS. Đi kèm văn bản này còn có các files nguồn cho văn bản mẫu, bao gồm files cho PIC16F877A, 16F876A, 16F88. Về sau khi lập trình bạn chỉ việc copy tài liệu này vào thư mục chứa Project của bạn, sửa đổi tên files. Khi cần thay đổi nội dung cấu hình cho PIC bạn chi việc tham khảo qua PIC Wizard , xem code và copy đưa vào Project.

Mô tả nội dung chương trình.

- **#include 16f877a.h** : Đi kèm chương trình dịch, chứa khai báo về các thanh ghi trong mỗi con PIC, dùng cho việc cấu hình cho PIC.
- **#include def_877a.h**: Files do người lập trình tạo ra, chứa khai báo về các thanh ghi trong PIC giúp cho viêc lập trình được dễ dang hơn ví dụ ta co thể gán PORTB = 0xAA (chi tiết files này sẽ trình bày trong phần dưới đây)
- **#device *=16 ADC = 10**: Khai báo dùng con trỏ 8 hay 16 bit, bộ ADC là 8 hay 10 bit
- **#FUSES NOWDT, HS:** Khai báo về cấu hình cho PIC
- **#use delay(clock=20000000):** Tần số thạch anh sử dụng
- **#use rs232 (baud=9600,…):** Khai báo cho giao tiếp nối tiếp RS232
- **#use i2c(master, SDA=PIN_C4,…):** Khai báo dùng I2C, chế độ hoạt động
- **#include <tên_file.c>:**Khai báo các files thư viện được sử dụng ví dụ LCD_lib_4bit.c
- **#INT_xxx :** Khai báo địa chỉ chương trình phục vụ ngắt
- **Void tên_chương_trình (tên_biến) {}:** Chương trình chính hay chương trình con

Chương trình mẫu cho PIC16F877A

```
//=========================================================  =======
// Ten chuong trinh    : Mach test den LED_1
// Nguoi thuc hien     : Falleaf
// Ngay thuc hien      : 23/05/2005
// Phien ban     : 1.0
// Mo ta phan cung     : Dung PIC16F877A - thach anh 20MHz
//           : LED giao tiep voi PORTB
//           : Cuc am cua LED noi voi GND
//           : RB0 - RB7 la cac chan output
//-----------------------------------------------------------------
// Ngay hoan thanh     : 23/05/2005
// Ngay kiem tra       : 23/05/2005
// Nguoi kiem tra      : Doan Hiep
//-----------------------------------------------------------------
// Chu thich     : Mo ta cac diem khac nhau cua cac phien ban khac nhau
//           : hoac cac chu thich khac
//           : vd, dung che do Power On Reset, PORTB = 00000000
//           : hoac, chuong trinh viet cho PIC Tutorial
//           : hoac, chuong trinh nay hoan toan mien phi va co the dung cho
//           : moi muc dich khac nhau
//=========================================================  =======
#include <16f877a.h>
#include <def_877a.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)
#use i2c(Master,Fast,sda=PIN_B1,scl=PIN_B4)
#int_xxx      // Khai bao chuong trinh ngat
xxx_isr()  {
// Code here
}
void Ten_chuong_trinh_con(Ten_Bien) {
// Code here
}
void main() {
// Enter code here!
}
```

Chương trình mẫu cho PIC16F876A

```
//================================================  =======
// Ten chuong trinh    : Mach test den LED_1
// Nguoi thuc hien     : Falleaf
// Ngay thuc hien      : 23/05/2005
// Phien ban     : 1.0
// Mo ta phan cung     : Dung PIC16F876A - thach anh 20MHz
//          : LED giao tiep voi PORTB
//          : Cuc am cua LED noi voi GND
//          : RB0 - RB7 la cac chan output
//-------------------------------------------------------------
// Ngay hoan thanh     : 23/05/2005
// Ngay kiem tra       : 23/05/2005
// Nguoi kiem tra      : Doan Hiep
//-------------------------------------------------------------
// Chu thich     : Mo ta cac diem khac nhau cua cac phien ban khac nhau
//          : hoac cac chu thich khac
//          : vd, dung che do Power On Reset, PORTB = 00000000
//          : hoac, chuong trinh viet cho PIC Tutorial
//          : hoac, chuong trinh nay hoan toan mien phi va co the dung cho
//          : moi muc dich khac nhau
//================================================  =======
#include <16f876a.h>
#include <def_876a.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)
#use i2c(Master,Fast,sda=PIN_B1,scl=PIN_B4)
#int_xxx      // Khai bao chuong trinh ngat
xxx_isr()  {
// Code here
}
void Ten_chuong_trinh_con(Ten_Bien) {
// Code here
}
void main() {
// Enter code here!
}
```

Chương trình mẫu cho PIC16F88

```
//================================================  =======A
// Ten chuong trinh    : Mach test den LED_1
// Nguoi thuc hien     : Falleaf
// Ngay thuc hien      : 23/05/2005
// Phien ban     : 1.0
// Mo ta phan cung     : Dung PIC16F88 - thach anh 20MHz
//           : LED giao tiep voi PORTB
//           : Cuc am cua LED noi voi GND
//           : RB0 - RB7 la cac chan output
//---------------------------------------------------------------
// Ngay hoan thanh     : 23/05/2005
// Ngay kiem tra       : 23/05/2005
// Nguoi kiem tra      : Doan Hiep
//---------------------------------------------------------------
// Chu thich     : Mo ta cac diem khac nhau cua cac phien ban khac nhau
//           : hoac cac chu thich khac
//           : vd, dung che do Power On Reset, PORTB = 00000000
//           : hoac, chuong trinh viet cho PIC Tutorial
//           : hoac, chuong trinh nay hoan toan mien phi va co the dung cho
//           : moi muc dich khac nhau
//================================================  =======
#include <16f88.h>
#include <def_88.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, MCLR, NOBROWNOUT, NOLVP, NOCPD, NOWRT, NODEBUG
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)
#use i2c(Master,Fast,sda=PIN_B1,scl=PIN_B4)

#int_xxx      // Khai bao chuong trinh ngat
xxx_isr() {
// Code here
}
void Ten_chuong_trinh_con(Bien) {
// Code here
}
void main() {
// Enter code here!
}
```

3. Một số ví dụ cho lập trình CCS

Với mục tiêu giúp người đọc nhanh chóng lắm bắt được cách lập trình C cho PIC thông qua chương trình dịch CCS. Dưới đây tôi giới thiệu một vài bài lập trình đơn giản cho PIC, các bài mẫu này dựa theo tài liệu tutorial của Nigel như quét LED, LED 7 thanh, LCD, bàn phím…, cách dùng các giao tiếp của PIC để giao tiếp với thiết bị ngoại vi như Real Time IC, ADC, EEPROM…

- **Yêu cầu về phần cứng tối thiểu cần có để thực hành:**

- PIC16F877A (hoặc 16F876A hay 16F88) = 50K (Tốt nhất là PIC16F877A)
- 1 Board cắm linh kiện (tối thiểu) = 40K
- Thạch anh 20MHz, tụ 22pF, 10uF, trở 10K, 4K7, 330Ω, nút bấm = 10K
- 10 LED đơn xanh hay đỏ, 4 LED 7 thanh (loại 4 LED liền một đế) = 15K
- MAX232 để giao tiếp máy tính () = 10K
  Tổng cộng là: 125K

- **Phần cứng mở rộng**

- LCD 1602A loại 2 dòng 16 ký tự (Nếu có LCD 2002 càng tốt) = 65K (Minh Hà có bán)
- Real Time IC DS1307 hay DS1337 = 25K (có thể xin sample của Maxim-IC)
- EEPROM AT24Cxx
- ADC/DAC IC loại 12-bit trở nên (ADC 10-bit thì PIC cũng có)
- Sensor nhiêt LM335 hay LM35 = 13K
- Động cơ bước, động cơ một chiều

Mục đích chính của tôi trong việc giới thiệu các ví dụ dưới đây là nhằm giúp mọi người nhanh chóng nắm được kỹ thuật lập trình bằng CCS, thông qua các ví dụ mọi người sẽ hiểu các hàm của CCS, cách sử dụng trong từng ứng dụng cụ thể. Về chi tiết của mỗi hàm tôi sẽ không trình bày kỹ tại đây, để biết rõ ta có thể xem trong phần Trợ giúp của CCS hay tài liệu của tác giả Trần Xuân Trường, trong đó đã nói khá đầy đủ. Tôi nhấn mạnh một điều khi mọi người tìm hiểu về PIC và CCS đó là hãy tự mình tìm hiểu là chính, từ việc nghiên cứu tài liệu, tìm tài liệu cho đến thiết kế mạch và viết chương trình. Những gì tại đây chỉ là cơ bản, còn việc phát triển, sử dụng hết điểm mạnh của PIC và CCS là ở phía mọi người. Chúc thành công!

Một điều chú ý là tất cả các mạch điện và code tôi trình bày dưới đây tôi đều đã lắp mạch thật trên bo cắm và chạy tốt.

3.1. Chương trình nhấp nháy LED

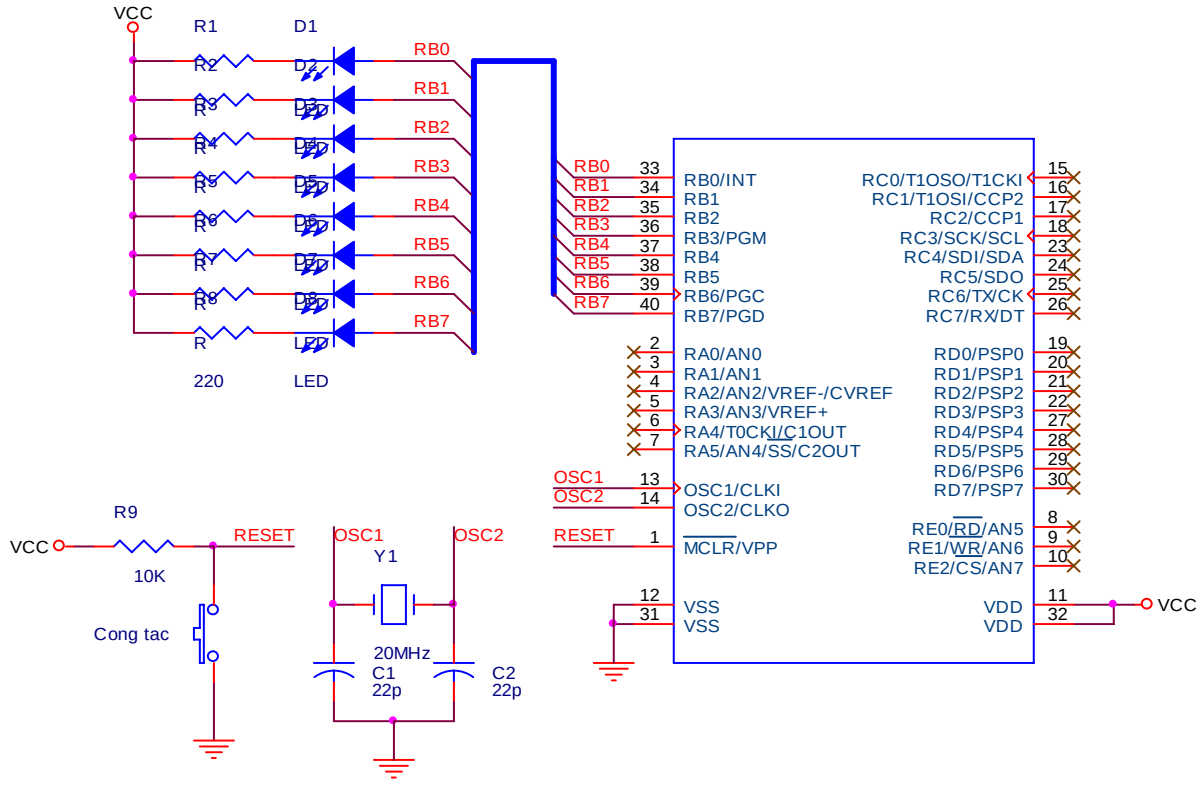
Nhấp nháy LED có thể coi là một chương trình "Kinh điển". Mỗi người khi bắt tay vào học VĐK thì bài học đầu tiên là làm nhấp nháy một hay vài con LED trên chân VĐK. Trong tài liệu này tôi cũng chọn bài tập đó để bắt đầu. Bản thân tôi cũng vậy, bài học đầu tiên là nháy LED và quét LED 7 thanh

Mục đích của bài như trên đã nói: Làm nhấp nháy 8 LED tại PORTB của PIC 16F877A, thời gian trễ do người lập trình định trước.

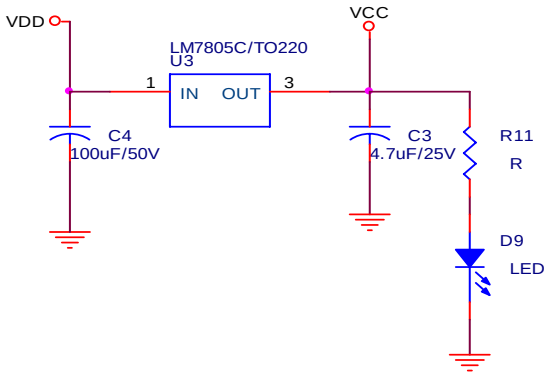
Những điều thu được qua bài học:

- Vẽ một mạch điện tử hoàn chỉnh dùng OrCad 9.2
- Tạo một Dự án trong CCS (cái này đã nói trong phần 2)
- Tệp định nghĩa các thanh ghi của PIC do người dùng tạo ra
- Thiết lập chế độ vào ra cho một cổng của PIC
- Sử dụng hàm tạo trễ thời gian

Dưới đây là sơ đồ phần cứng. Trong sơ đồ các LED được mắc chung lên dương nguồn thông qua điện trở. Gia trị điện trở thay đổi trong khoảng 100Ω cho đến 560Ω tùy theo độ sáng của LED mà ta muốn và cũng để đảm bảo dòng qua mỗi LED không quá 20mA khi nguồn cấp là 5V. Như vậy để làm sáng LED ta chỉ việc đưa mức 0 ra các chân PIC và ngược lại để tắt ta đưa mức 1.



Hình 3.1. Sơ đồ mạch nháy 8 LED tại PORTB



Hình 3.2. Sơ đồ mach nguồn cho PIC

Mã nguồn chương trình nạp vào PIC

```
//================================================ =======
// Ten chuong trinh : Mach nhay den LED
// Nguoi thuc hien  : linhnc308
// Ngay thuc hien   : 13/03/2006
// Phien ban  : 1.0
// Mo ta phan cung  : Dung PIC16F877A - thach anh 20MHz
//                  : LED giao tiep voi PORTB
//                  : Cuc am cua LED noi voi PORTB
//                  : RB0 - RB7 la cac chan output
//-------------------------------------------------------------
// Ngay hoan thanh : 13/06/2006
// Ngay kiem tra   : 13/06/2006
// Nguoi kiem tra  : linhnc308
//-------------------------------------------------------------
// Chu thich  : dung che do Power On Reset, PORTB = 00000000
//                  : chuong trinh viet cho PIC Tutorial
//                  :  chuong trinh nay hoan toan mien phi va co the dung cho
//                  : moi muc dich khac nhau
//================================================ =======

#include <16f877a.h>
#include <def_877a.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT,
NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)

void main()
```

```
{
// Thiet lap che do cho PORTB
TRISB = 0x00;        // Tat ca PORTB deu la cong xuat du lieu
PORTB = 0xFF;        // Tat het cac LED
While(1)
{
        PORTB = 0;   // Cho các LED sáng
        delay_ms(250);       // Tạo thời gian trễ 250ms
        PORTB = 0xFF;
        delay_ms(250);
}
}
```

Qua ví dụ đơn giản trên bạn hiểu cách xuất dữ liệu ra một cổng của PIC và dùng các hàm tạo trễ.

Thủ tục thiết lập vào ra cho một cổng hay một chân của PIC

- Ghi giá trị vào thanh ghi điều khiển chế độ của cổng tương ứng là TRISx
    - Bit 0 ứng với chân xuất dữ liệu
    - Bit 1 ứng với nhận dữ liệu
    - Thanh ghi TRISx có thể câu hình theo từng bit
- Khi muốn xuất dữ liệu, ví dụ ra PORTB, câu lệnh là: PORTB = gia_tri;
- Khi muôn nhận dữ liệu từ PORTB, câu lệnh là: data_in = PORTB;

Về các hàm tạo trễ, trong CCS hỗ trọ sẵn 3 loại hàm tạo trễ là:

- **delay_cycles(gia_tri)**: gia_tri là thời gian trễ tính theo số chu kỳ máy
- **delay_us(gia_tri)**: Tạo trễ Micro giây
- **delay_ms(gia_trị)**: Tạo trễ Mili giây

Bản chất của các hàm tạo trễ là đưa Vi điều khiển vào một vòng lặp chẳng làm gì cả cho đủ số thời gian mà ta cần. Ngoài việc dùng hàm tạo trễ có sẵn ta có thể tự viết hàm tạo trễ dùng bộ Timer

3.2. Bộ ADC trong PIC và ứng dụng

Bộ chuyển đổi từ tương tự sang số là một khối mạch điện tử quan trọng, có mặt trong rất nhiều thiết kế điện tử. Các bộ ADC thực tế được đóng gói trong những IC chuyên dụng, do nhiều hãng sản xuất. Điểm quan trong cần lưu ý ở các bộ ADC này là độ phân giải và tốc độ lấy mẫu tìn hiệu. Độ phân giải của bộ ADC có thể là 8-bít, 10-bít, 12-bít, 16-bít, 24-bít… Tốc độ lấy mẫu của ADC có thể nhanh hay chậm, tùy từng ứng dụng mà ta chọn tốc độ thích hợp.

Vi điều khiển PIC là một trong những dòng Vi điều khiển có phần giao tiếp ngoại vi mạnh và đa dạng. Bên trong PIC đã được tích hợp sẵn một bộ ADC có độ phân giải tối đa

là 10-bít (tùy chon là 8-bit hay 10-bit). Với bộ ADC trong PIC ta có thể làm được khá nhiều công việc, dưới đây tôi trình bày một ứng dụng của bộ ADC trong việc thiết kế mạch đo nhiệt độ sử dụng sensor nhiệt LM335.

Dưới đây là phần code mạch đo nhiệt dộ, hiển thị trên LCD.

```
//=================================================  =======
// Ten chuong trinh       : Mach do nhiet do
// Nguoi thuc hien: linhnc308
// Ngay thuc hien  : 28/03/2006
// Phien ban: 1.0
// Mo ta phan cung        : Dung PIC16F877A - thach anh 20MHz
//                : LCD giao tiep voi PORTD
//                : Dau ra LM335 dua vao chan AN0
//-------------------------------------------------------------
// Ngay hoan thanh        : 28/03/2006
// Ngay kiem tra    : 28/03/2006
// Nguoi kiem tra  : Linhnc308
//-------------------------------------------------------------
// Chu thich : hoac cac chu thich khac
//                : dung che do Power On Reset
//                : chuong trinh viet cho PIC Tutorial
//=================================================  =======
#include <16F877A.h>
#include <def_877a.h>
#device *=16 adc=10
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT,
NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
#use rs232(baud=115200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
#include <lcd_lib_4bit.c>        // Thu vien ham cho LCD
int8 low,high,key,mode,min,max,mode1,i;
int1 do_F;
void convert_bcd(int8 x);
void bao_dong();
void test();
//-------------------------------------------------------------
void main()
{
```

```
  float value;
  on_off =1;
  min    =15;   //nhiet do min default
  max    =35;   //nhiet do max default
  do_F   =0 ;
  i = 50 ;
  mode   =0 ;
  mode1 = 0 ;
   trisa = 0xFF;
   trisb = 0x01;
   trisd = 0x00;
     LCD_init();
     Printf(LCD_putchar,"Lop DT8 - BKHN");
     LCD_putcmd(0xC0);
     Printf(LCD_putchar,"Khoi tao...");
// Khoi tao cho ngat ngoai
     enable_interrupts (INT_EXT);
     ext_int_edge(H_TO_L);
     enable_interrupts (GLOBAL);
// Khoi tao che do cho bo ADC
     setup_adc_ports(AN0);
     setup_adc(ADC_CLOCK_INTERNAL);
     delay_us(10);
// Lay mau nhiet do lan dau tien
     value=(float)read_adc();
     value = (value - 558.5)/2.048;       // For 5V supply
     //    value = (value - 754.8)/2.048;  // For 3.7V Supply
     //    value = (value - 698.2)/2.048;  // For 4V supply
     convert_bcd((int8)value);  // Tach so tram, chuc, donvi de hien thi len LED 7
     delay_ms(1000);
     LCD_putcmd(0xC0);
     Printf(LCD_putchar,"Khoi tao xong");
     while(1)
     {
      if (i==50)
        {
          value = read_adc();
```

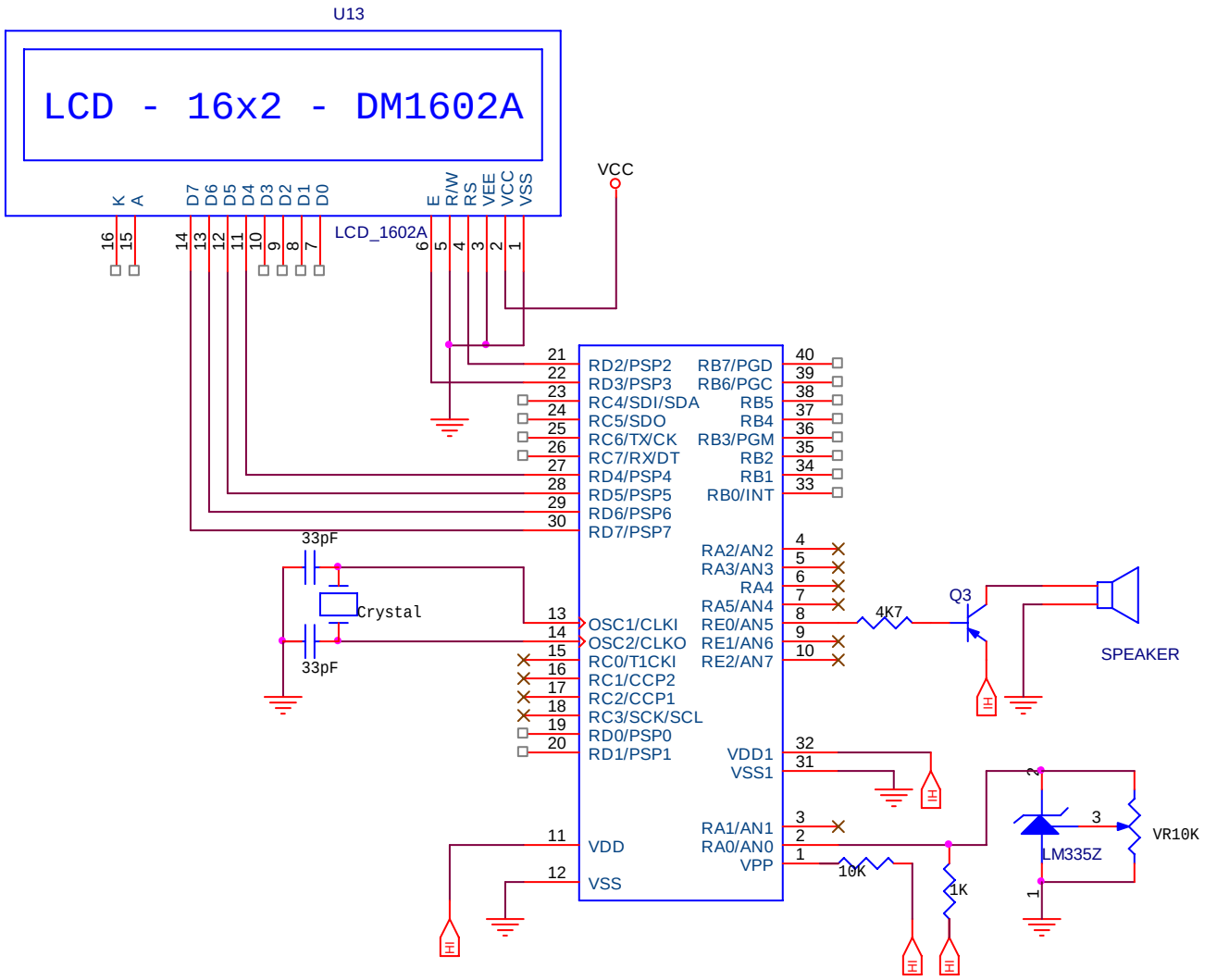
```
            value=(value-558.5)/2.048;
            if (do_F==1) value=1.8*value+32;
            convert_bcd((int8)value);
            printf("\n\rNhiet do phong: %u",value);// Gui gia tri len may tinh
            LCD_putcmd(0xC0);
            printf(LCD_putchar,"  Temp = ");
            LCD_putchar(high); LCD_putchar(low);
            if (do_F==0) printf(LCD_putchar," C");
            else printf(LCD_putchar," F");
            i=0;
           }
          i++;
          if(((int8)value > 40) || ((int8)value < 15)) on_off=1;
          else
           {
            on_off = 0;
            LCD_Putcmd(0xCF);
            LCD_putchar(" ");
            blink=0;
           }
          if (on_off==1)
          {
          if (blink==0) {
LCD_Putcmd(0xCF);LCD_putchar("!");blink=1;delay_ms(250);}
          else        {LCD_Putcmd(0xCF);LCD_putchar(" ");blink=0;delay_ms(250);}
          }
         }
       }//end main-----------------------------------------------------
      #INT_EXT
      void test()
      {
        if (do_F == 1) do_F=0;
        else        do_F=1;
      }
      void convert_bcd(int8 x)
       {
        low=x%10;  //chia lay phan du, so hang don vi
```

```
  high=x/10;   //tach hang tram va hang chuc
  low = low + 0x30;
  high = high + 0x30;
 }
void bao_dong(){
int8 i;
if (blink == 0) blink = 1;
else          blink=0;
    for(i=0;i<50;i++)
    {
     LCD_Putcmd(0xCF);
     if (blink==0) LCD_putchar("!");
     else       LCD_putchar(" ");
    }
}
```

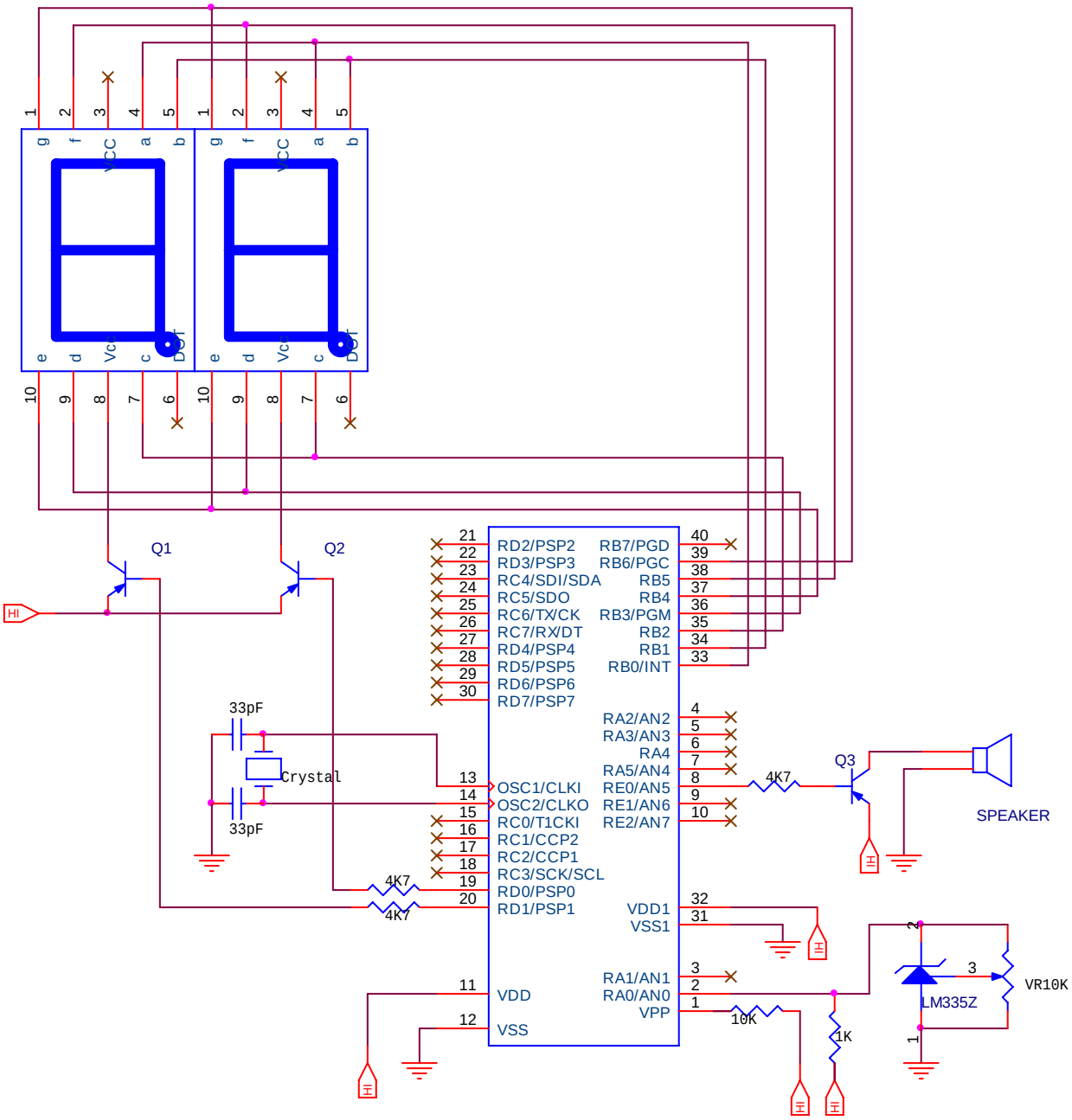
Dưới đây là một sơ đồ dùng PIC và LM335 để đo nhiệt độ, hiển thị trên LCD và trên LED 7. Trong chương trình bạn thấy có hàm chuyển đổi nhiệt độ từ ía trị độ K về độ C. Nguyên nhân có hàm đó là do con LM335 thay đổi 10mV/K, ta cần hiển thị là độ C. Nhận thấy $0^oC = 273K$, như vậy tại 0^oC con LM335 sẽ xuất ra một điện áp là 2.73V và với điện áp này, ADC trong PIC sẽ cho giá trị số là $\frac{2.73*1023}{5} = 558.558$. Như vậy khi tính toán giá trị nhiệt độ ta cần trừ đi giá trị 558.558 này. Công thức đầy đủ là:

$$Do\_C = \frac{adc\_value - 558.558}{2.048}$$

Giá trị 2.048 có là do ta dùng ADC 10-bit, điện áp lấy mẫu là 5V, như vậy mỗi mức lượng tử sẽ tương ứng với $\frac{5V}{1024} = 4.883mV$. LM335 thay dổi 10mV/K do đó ứng với sụ thay đổi 1 độ C sẽ thay đổi 2.048 mức lượng tử (10mV/4.883mV = 2.048). Công thức trên là cho ADC 10-bit, với các bộ ADC 8-bit hay 12-bit việc tính toán chuyển đổi giá trị cũng tương tự.



Hình 3.3. Mạch đo nhiệt độ LM335 hiển thị trên LCD1602



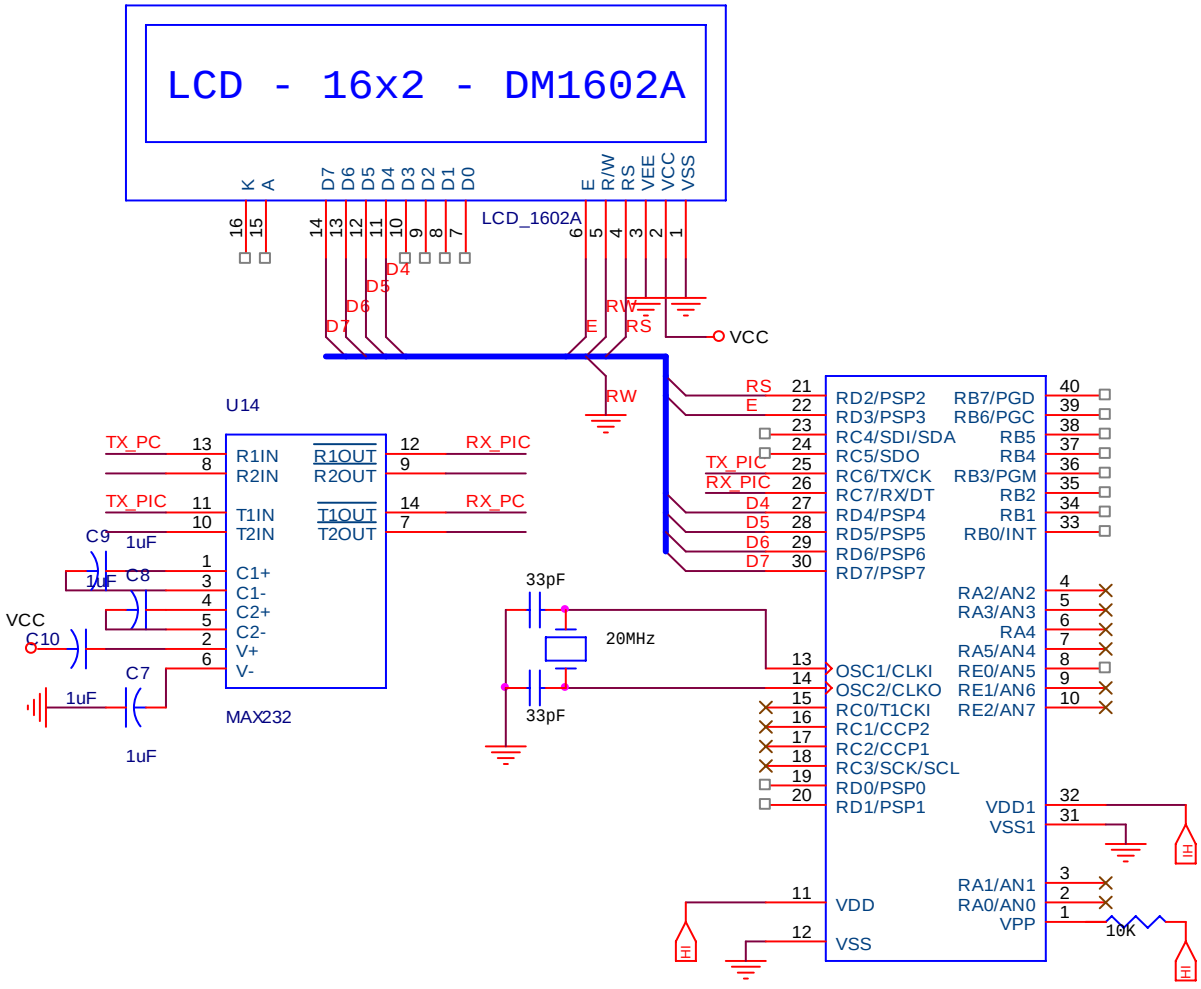
Hình 3.4. Mạch đo nhiệt độ hiển thi trên LED 7 thanh

Trong mạch trên ta dùng chính con PIC cho việc giải mã LED 7 thanh. Nguyên tắc quét cho từng LED 7 thanh là gửi giá trị cần hiển thị -> bật LED -> Tạo thời gian trễ -> tắt LED. Quá trình cứ lặp lại như vậy cho đến khi quét hết LED. Ta tính toán thời gian trễ sao cho đảm bảo các số hiển thị liên tục.

3.3. Giao tiếp máy tính RS232

Việc giao tiếp giữa Vi điều khiển và máy tính là bài lập trình khá quan trọng khi ta làm việc với các dòng Vi điều khiển khác nhau. Với Vi điều khiển PIC cũng vậy, trong mỗi IC PIC đều có tích hợp một khối giao tiếp máy tính USART. Ta sử dụng khối giao tiếp này để truyền dữ liệu lên máy tính và xử lý dữ liệu đó tùy vào mục đích của người lập trình. Để nhận dữ liệu do Vi điều khiển truyền lên máy tính ta có thể sử dụng các phần mềm giao tiếp COM có sẵn hay viết một chương trình mới, sử dụng các ngôn ngữ lập trình như C++, VB hay Delphi… Trong chương trình ví dụ dưới đây tôi sử dụng công cụ sẵn có của CCS là Serial Port Monitor để truyền và nhận dữ liệu từ PIC.

Sơ đồ mạch điện ORCAD. Mạch sử dụng IC MAX232 để kết nối đến cổng COM của máy tính. Mạch đơn giản chỉ nhằm mục đích giới thiệu khối giao tiếp máy tính của PIC và cách lập trình cho nó trong CCS.



Hình 3.5. Mạch giao tiếp máy tính, hiển thị LCD

Mã nguồn chương trình:

```
#include <16f877a.h>
#include <def_877a.h>
#use delay(clock=20000000)
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
// Khai báo sử dụng giao tiếp nối tiếp RS232
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)

#include <lcd_lib_4bit.c>

int8 count=0;
char string_in[16];

#INT_RDA        // Hàm xử lý ngắt nối tiếp
Receive_isr() {
char c;
int8 i;
count++;
c = getc();
putc(c);
if (c=='c' | c=='C')
  {
    LCD_putcmd(0x01);    //Clear Screen
    c='c';
    count=0;
  }
if ((count<=16) && (c!='c'))  LCD_putchar(c);
if (count > 16)
  {
    count=0;
    LCD_putcmd(0xC0);
  }
}
void main()
{
```

```
enable_interrupts(int_rda);
enable_interrupts(GLOBAL);

lcd_init(); // Khởi tạo cho LCD
lcd_putcmd(0x01);
lcd_putcmd(line_1);
printf("Enter a String.");
printf("Or anything you want!");
while (1) {}
}
```

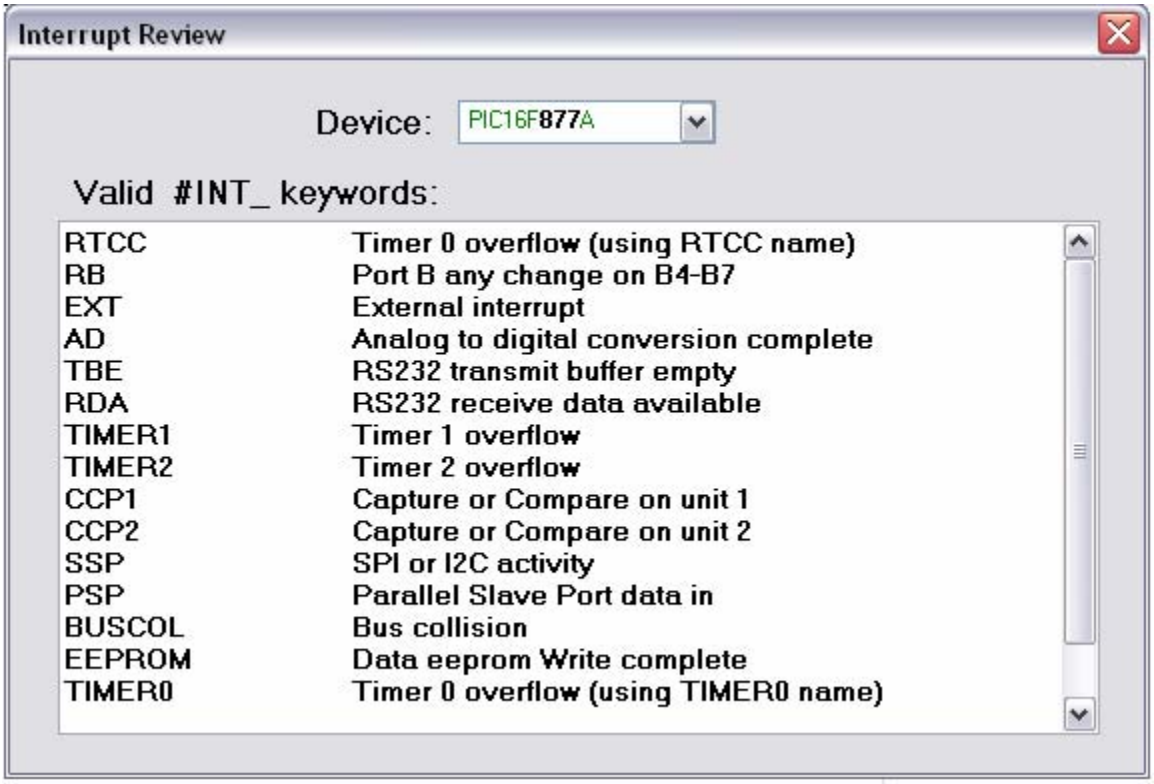
Mô tả chương trình: Trên đây là chương trình giao tiếp với máy tính, ta thấy trong CCS để sử dụng giao tiếp nối tiếp ta chỉ cần khai báo **#use rs232().** Các hàm giao tiếp với máy tính mà CCS hỗ trợ là:

- **putc(char ky_tu)** : Gửi một ký tự ASCII lên máy tính
- **getc()** : Hàm trả về một ký tự nhận được từ máy tính
- **printf(string):** hàm gửi một chuỗi ký tự lên máy tính

Trong chương trình ta có sử dụng hàm xử lý ngắt nối tiếp để xử lý ký tự nhân được từ máy tính. Khi có ngắt xảy ra, ta gọi hàm **getc()** sẽ trả về ký tự vừa nhận được. Trên màn hình LCD sẽ hiển thị ký tự mà ta gõ từ bàn phím máy tính.

3.4. Ngắt của PIC và cách sử dụng

Trong Vi điều khiển PIC có nhiều nguồn ngắt. Để biết cụ thể ta có thể vào mục View >> Valid Interrupts . Khi đó một của sổ sẽ hiện ra liệt kê đầy đủ các nguồn ngắt của từng con PIC.



Hình 3.6 Các nguồn ngắt trong PIC

Để viết một hàm phục vụ ngắt ta chỉ việc thêm khai báo **#INT_tên_ngắt** vào trước hàm phục vụ cho ngắt đó. Khi đó trình dich sẽ hiểu đó là địa chỉ hàm cho ngắt, khi có ngắt tương ứng xảy ra thì nó sẽ nhảy đến vị trí đó

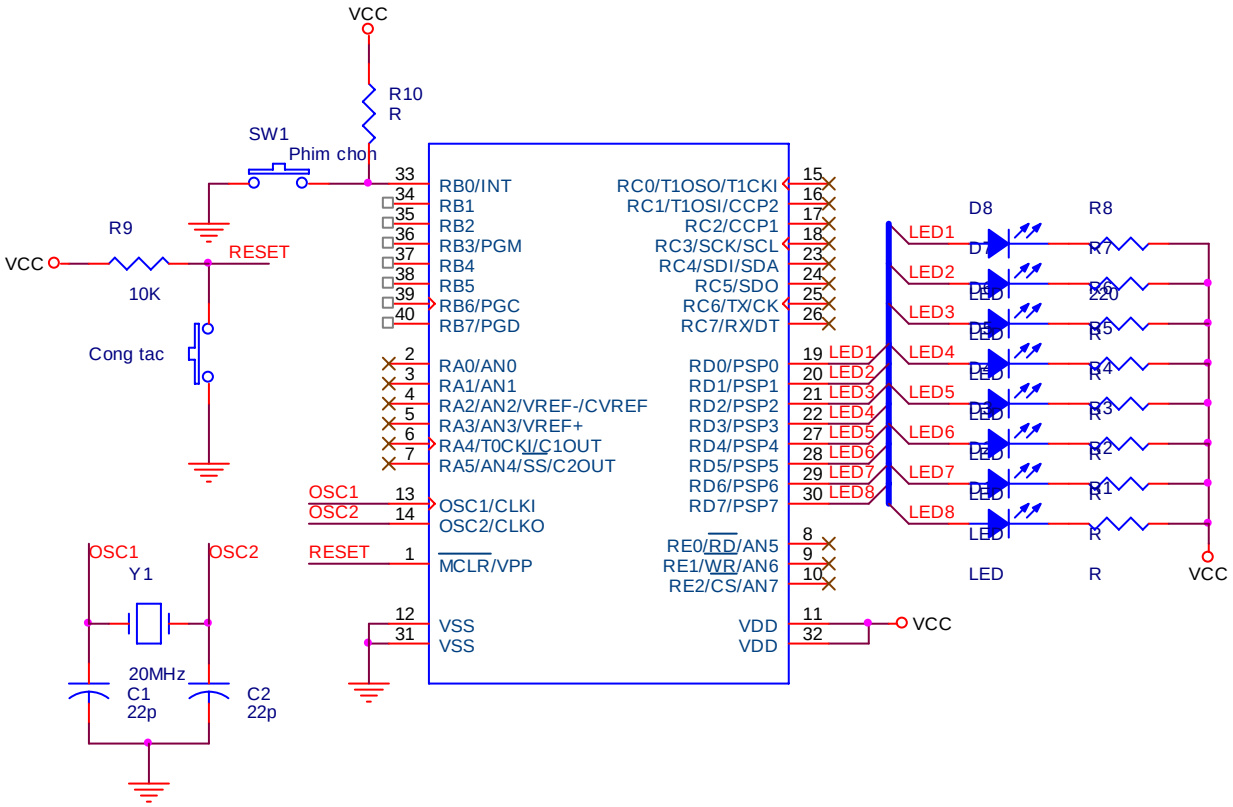
Lấy ví dụ khi ta muốn xử lý ngắt ngoài, hàm sẽ được viết như sau:

```
#INT_EXT
Ext_isr()
{
// Nhập mã tại đây
}
```

Dưới đây là chương trình nháy led theo nhiều kiểu khác nhau, sử dụng 1 phím bấm nối với chân ngắt ngoài RB0 để chọn kiểu nháy. Có 8 kiểu nháy LED khác nhau, Khi đến kiểu nháy thứ 8, nếu ta nhấn thì sẽ trở về chế độ ban đầu. Ban đầu biến **mode = 0**

và tất cả các LED đều tắt Mỗi khi nhấn phím bấm, biến **mode** sẽ tăng lên 1 đơn vị. Giá trị biến mode tương ứng với chương trình nháy được thực hiện. Khi **mode = 9** thì sẽ được gán về **mode = 0.** Các kiểu nháy khác nhau là do ta bật tắt các LED trên cổng D theo các cách khác nhau. Lấy ví dụ khi ta muốn các LED nháy xen kẽ nhau ta chỉ việc gửi ra cổng D giá trị AAh (10101010) và 55h (01010101).

Sơ đồ mạch điện:



Hình 3.7. Nháy LED nhiều chế độ

Phần mã nguồn chương trình:

```
#include <16F877A.h>
#include <def_877a.h>
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT,
NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)

int8 mode,i;
byte temp;
```

```
#INT_EXT
EXT_ISR() {

mode++;
if (mode==9) mode = 0;
}
// End of INT

void program1();
void program2();
void program3();
void program4();
void program5();
void program6();
void program7();
void program8();

void main() {

  trisd = 0x00;
  trisb = 0xFF;
  portd=0xff;
  enable_interrupts(int_EXT);
  ext_int_edge(H_TO_L);  // Chọn ngắt theo sườn âm
  enable_interrupts(GLOBAL);
  mode = 0;
while (1) {
  switch(mode) {
     case 1: program1(); break;
     case 2: program2(); break;
     case 3: program3(); break;
     case 4: program4(); break;
     case 5: program5(); break;
     case 6: program6(); break;
     case 7: program7(); break;
     case 8: program8(); break;
  }
}
}

void program1() {
```

```
  PortD = 0x00;
  delay_ms(250);
  Portd = 0xFF;
  delay_ms(250);
}
void program2() { // LED sáng chạy từ trái qua phải
  temp = 0xFF;
  for (i=0;i<=8;i++) {
    portd = temp;
    delay_ms(250);
    temp >>= 1;
  }
}
void program3() { // LED sáng chạy từ phải qua trái
   temp = 0xFF;
  for (i=0;i<=8;i++) {
    portd = temp;
    delay_ms(250);
    temp <<= 1;
  }
}
void program4() {
  portd = 0xAA;
  delay_ms(500);
  portd = 0x55;
  delay_ms(500);
}
void program5() {
  Portd = 0x7E;   delay_ms(150);
  Portd = 0xBD;   delay_ms(250);
  Portd = 0xDB;   delay_ms(150);
  Portd = 0xE7;   delay_ms(150);
  Portd = 0xDB;   delay_ms(150);
  Portd = 0xBD;   delay_ms(150);
  Portd = 0x7E;   delay_ms(150);
}
void program6() {
 temp = 0xFF;
  for (i=0;i<=8;i++) {
    portd = temp;
    delay_ms(250);
```

```
    temp = temp >> 1;
  }
}
void program7() {

  Portd = 0xFE;   delay_ms(150);
  Portd = 0xFD;   delay_ms(150);
  Portd = 0xFB;   delay_ms(150);
  Portd = 0xF7;   delay_ms(150);
  Portd = 0xEF;   delay_ms(150);
  PortD = 0xDF;   delay_ms(150);
  Portd = 0xBF;   delay_ms(150);
  Portd = 0x7F;   delay_ms(150);
}

void program8() {
  Portd = 0x7F;   delay_ms(150);
  Portd = 0xBF;   delay_ms(150);
  PortD = 0xDF;   delay_ms(150);
  Portd = 0xEF;   delay_ms(150);
  Portd = 0xF7;   delay_ms(150);
  Portd = 0xFB;   delay_ms(150);
  Portd = 0xFD;   delay_ms(150);
  Portd = 0xFE;   delay_ms(150);
}
```

3.5. Bộ Đếm/Định thời (Timer)

3.6. Giao tiếp I2C, SPI

3.7. PWM, Capture, Comparator