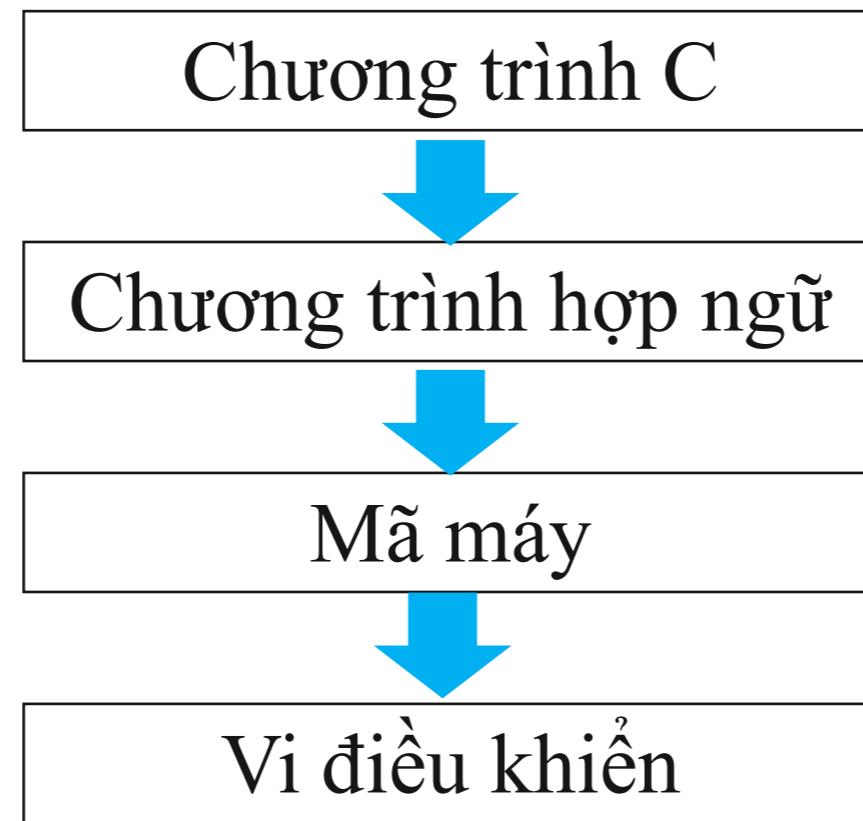


Phản Lý thuyết

1. Giới thiệu chung - Mô hình hệ VXL - Nguyên tắc hoạt động
2. Cấu trúc và hoạt động của vi xử lý 8085
3. Quá trình thực hiện 1 lệnh trong VXL 8085
4. Giới thiệu về vi điều khiển PIC
5. **Bộ công cụ nạp chương trình, công cụ mô phỏng vi điều khiển**
6. Bộ định thời Timer
7. Ghép nối với bộ hiển thị
8. ADC
9. Giao tiếp truyền dữ liệu
10. Ngắt
11. PWM

Nạp chương trình cho PIC



- Nên học ngôn ngữ nào để lập trình cho vi điều khiển ?
- Làm thế nào để cài đặt chương trình vào vi điều khiển ?

Ngôn ngữ lập trình cho PIC

- Ngôn ngữ cấp thấp, được cung cấp bởi nhà sản xuất Microchip (MPLAB)
- Ngôn ngữ cấp cao, thường là C,... (KeilC, MikroC, CCS,...)
- Ngôn ngữ riêng, PICBasic, MikroBasic...



- Lựa chọn sử dụng CCS vì tính đơn giản, dễ sử dụng, chương trình nhẹ

Mạch nạp cho PIC

- Mạch nạp chính hãng Microchip: PICSTART Plus, MPLAB ICD 2, MPLAB PM3, PRO MATE II,... Giá thành cao, nạp nhiều chế độ
- Mạch nạp do bên thứ 3 sản xuất: P16PRO40, mạch nạp Universal của Williem
- Tự làm mạch nạp

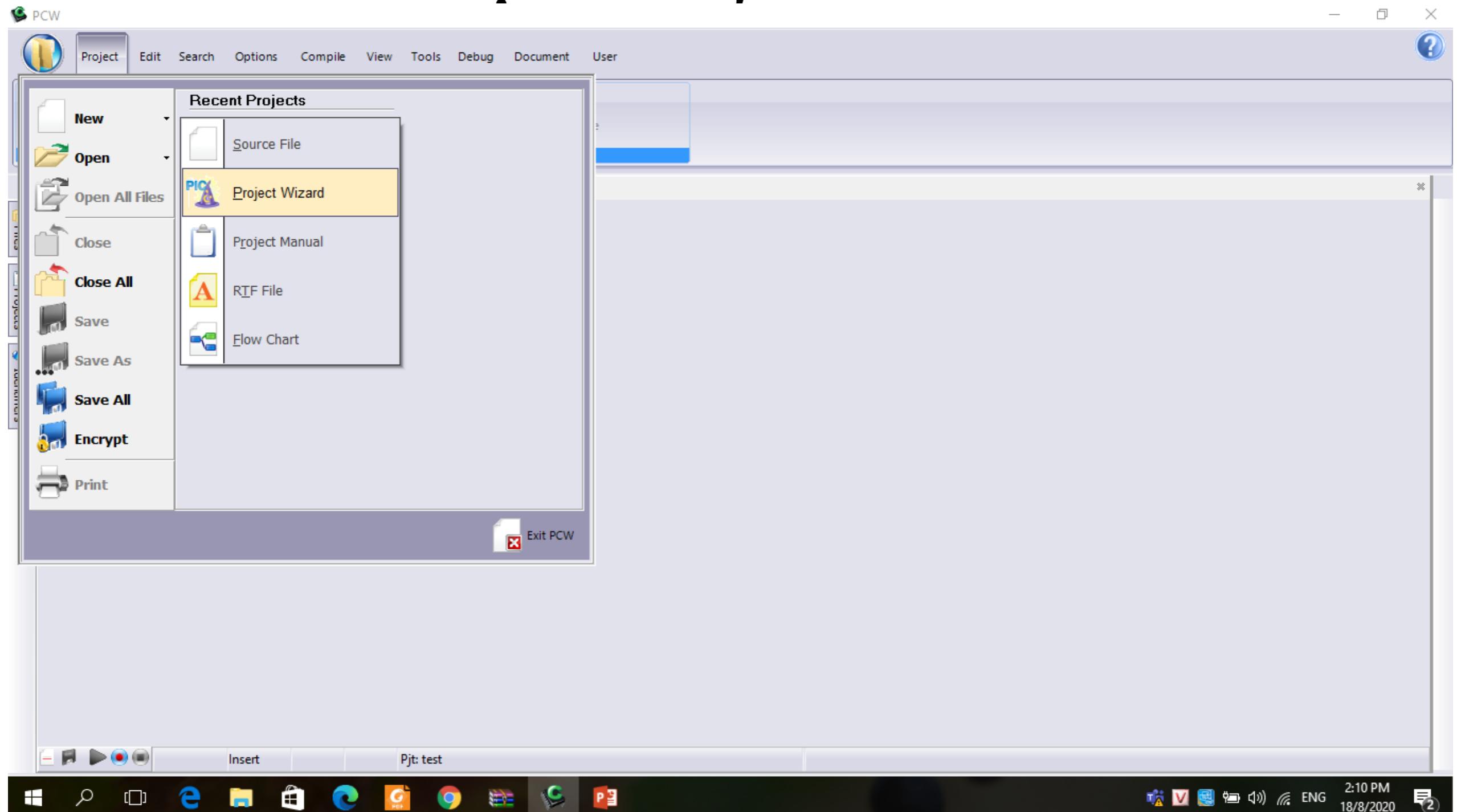


Để thực hiện mô phỏng thì chưa
cần sử dụng tới mạch nạp

Sử dụng CCS

- Có thể download CCS ở link
[https://drive.google.com/file/d/
0B61acQhFWDNwTGdLX3VNkU4Rm8/view](https://drive.google.com/file/d/0B61acQhFWDNwTGdLX3VNkU4Rm8/view)

Tạo Project mới



File → New → Project Wizard
Đặt tên project

Chọn PIC, chọn cấu hình

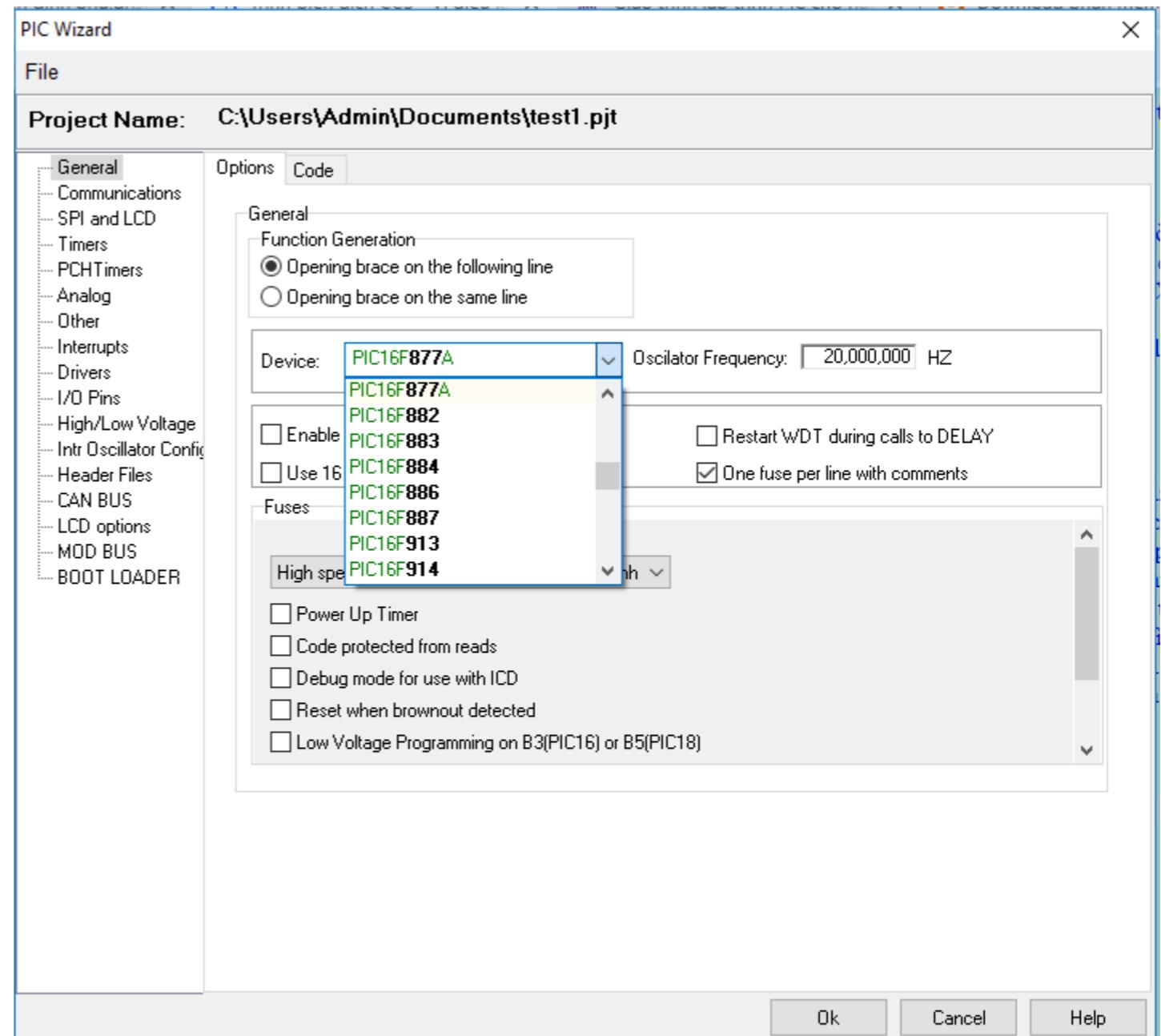
Tab General

Device: liệt kê các loại PIC. Ta lựa chọn PIC16F877A

Oscilator frequency: Chọn tần số thạch anh mà ta sử dụng

Fuse: Thiết lập các bit Config như chế độ dao động, chế độ bảo vệ Code, Reset Brown-out,

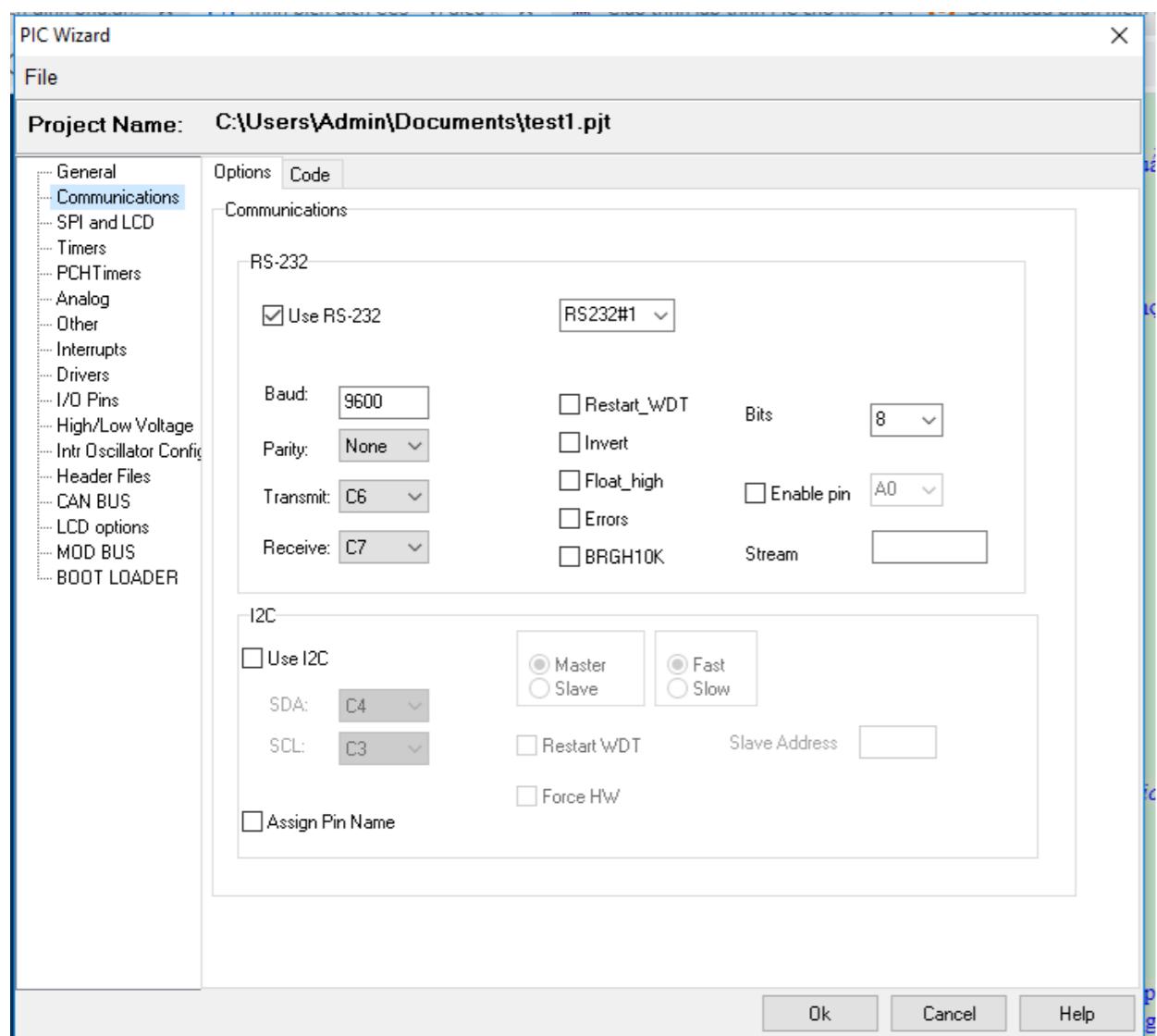
...



Chọn PIC, chọn cấu hình

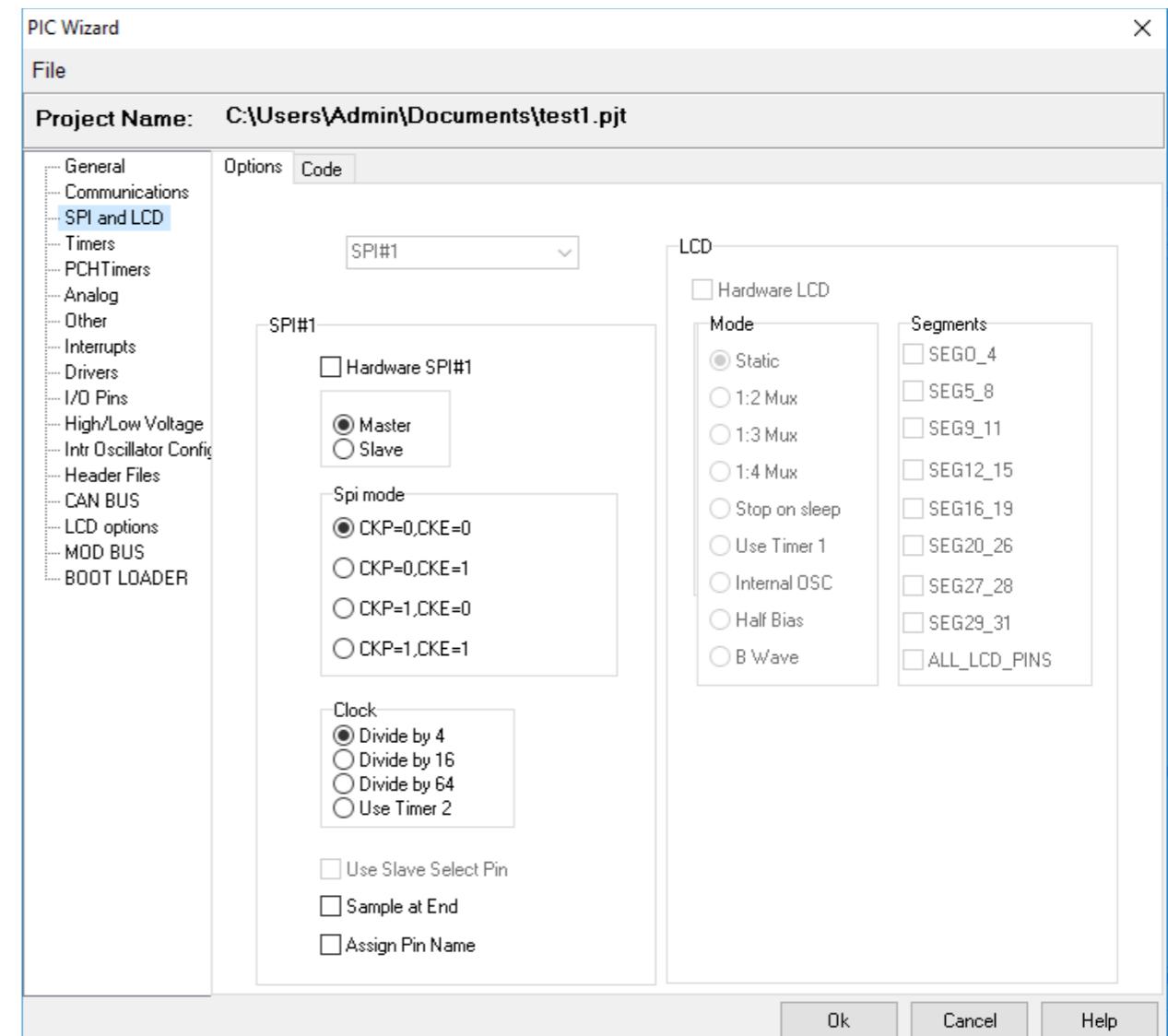
Tab Communication

Liệt kê, khai bao các giao tiếp RS232, I2C nếu sử dụng



Tab SPI và LCD

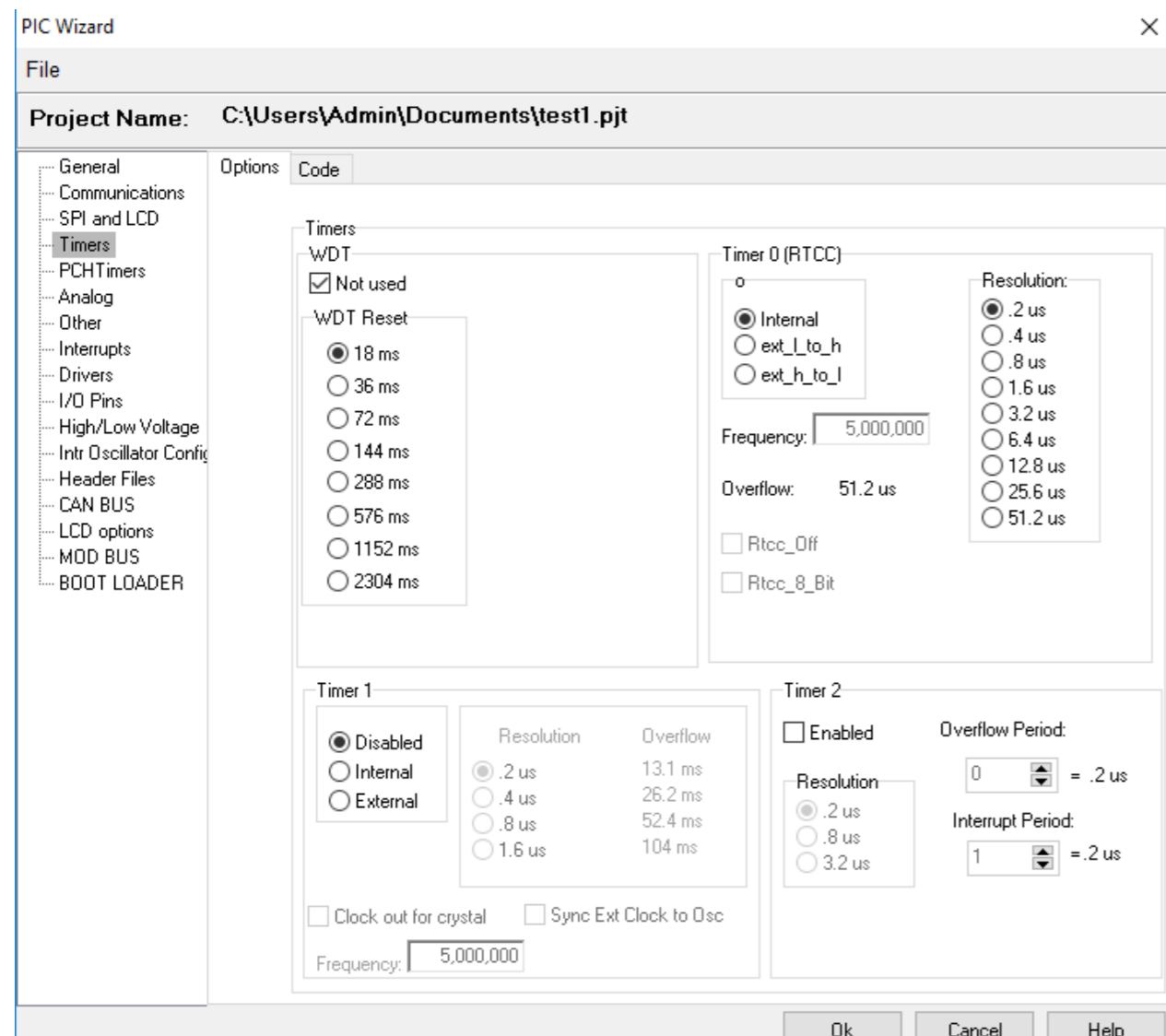
Khai báo lựa chọn giao tiếp SPI (khi không dùng I2C)
Phần LCD dành cho 18F và 30 F



Chọn PIC, chọn cấu hình

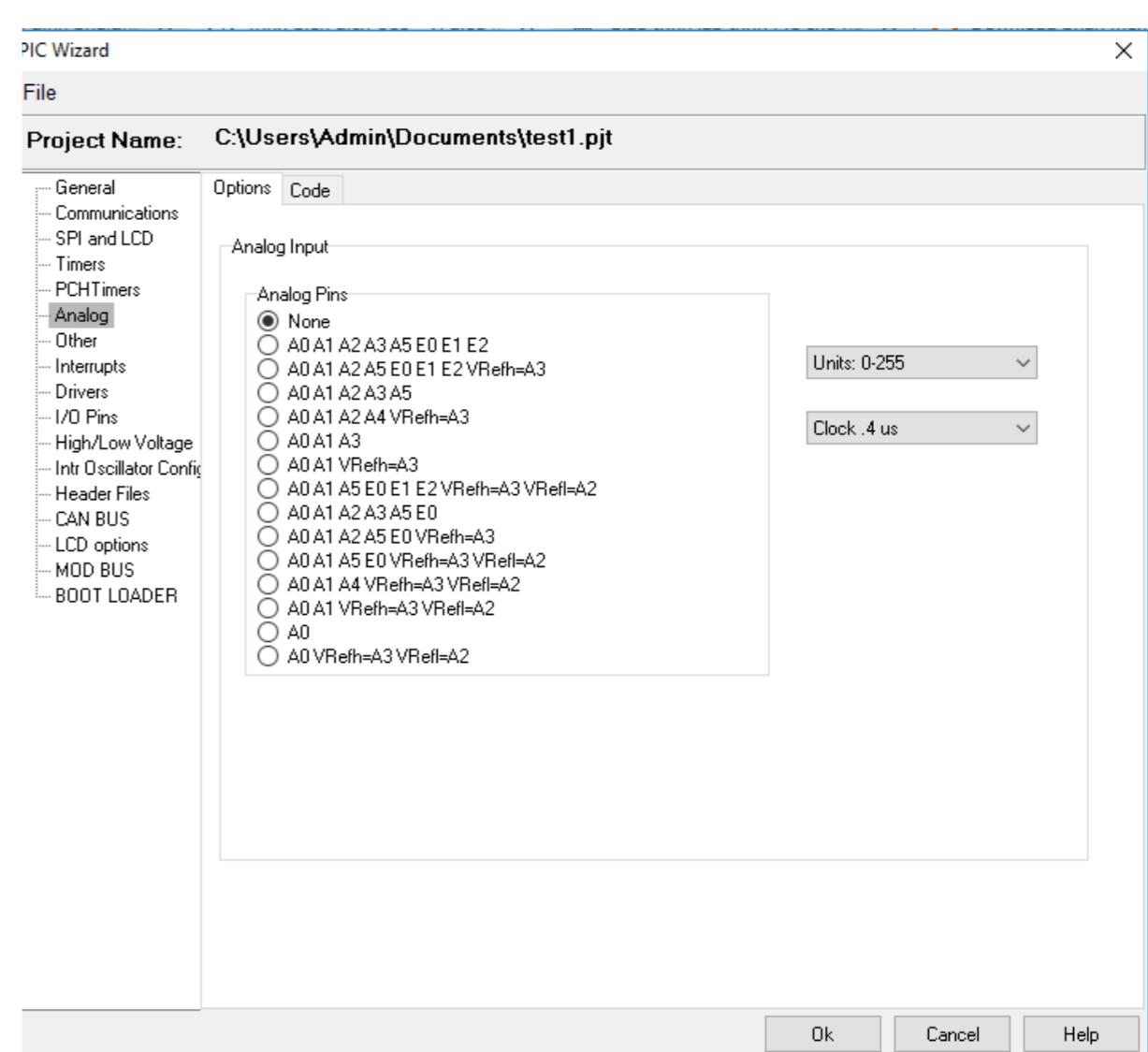
Tab Timer

Liệt kê các bộ Timer mà PIC sử dụng, WDT,..
Chú ý lựa chọn nguồn xung



Tab Analog

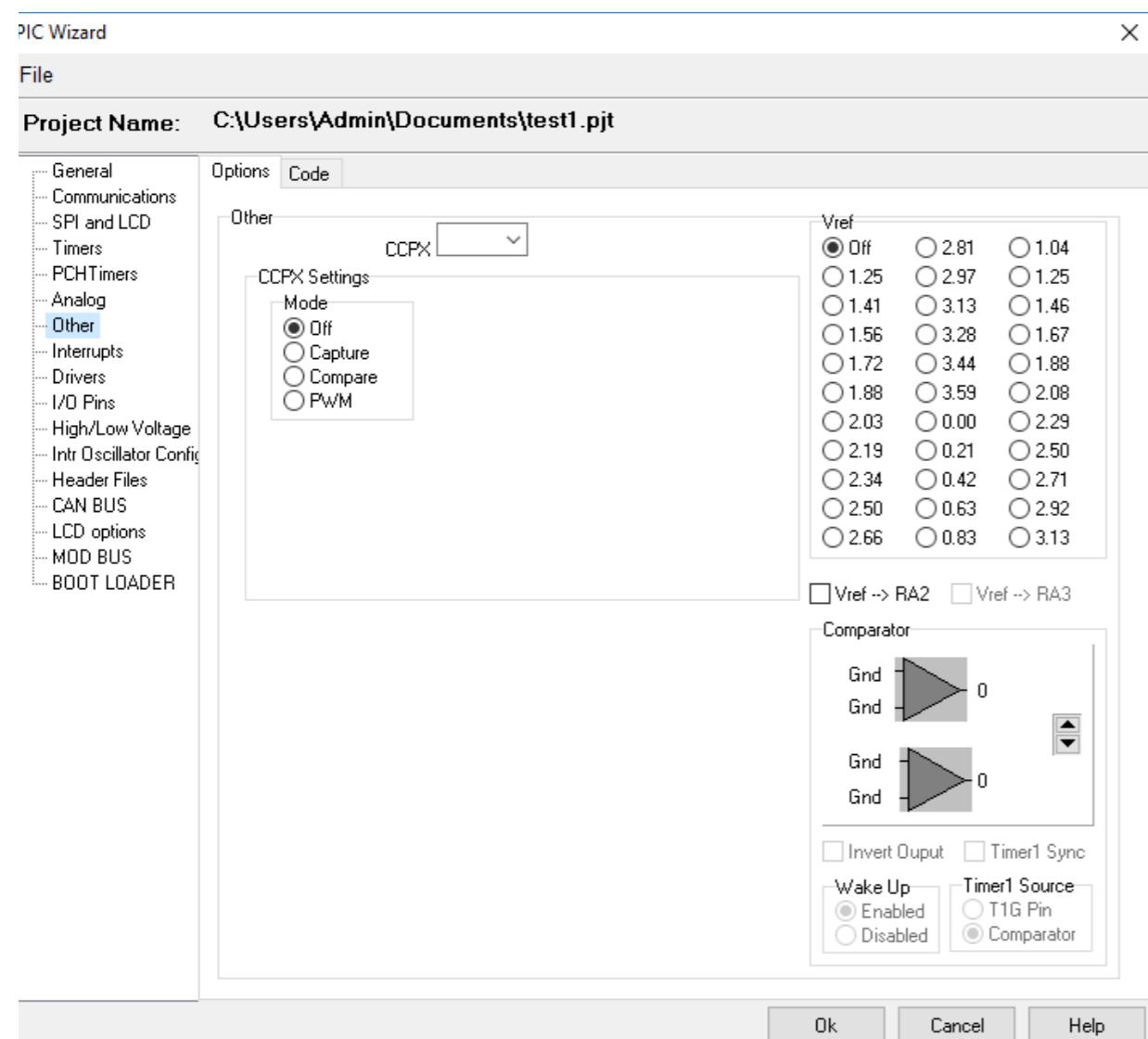
Liệt kê, lựa chọn các bộ chuyển đổi ADC, chân vào, điện áp tham chiếu, độ phân giải,...



Chọn PIC, chọn cấu hình

Tab Other

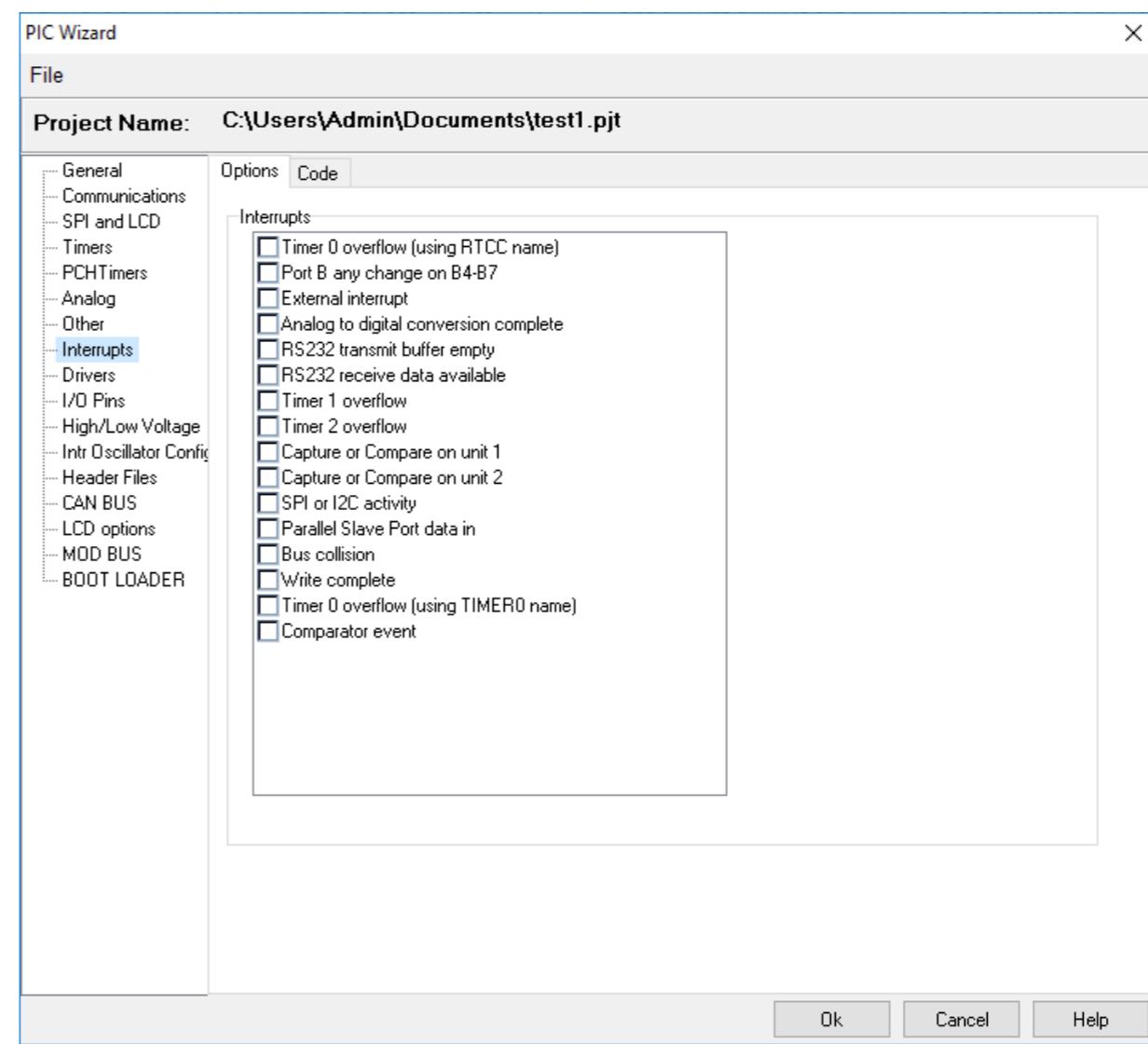
Lựa chọn các thông số cho bộ Capture/So sánh/PWM



Tab Interrupt và Tab Driver

Lựa chọn loại và kiểu ngắt

Lựa chọn ngoại vi mà CCS hỗ trợ



Cấu trúc chương trình

Khai báo biến, hằng, mảng

Biến

Hằng

Mảng

Int8 a;

Int const a=20;

Int16 a[3]={2,5,7}

int1	Số 1 bit = T hoặc F	float	Số thực 32 bit
int8	Số nguyên 1 byte	short	Mặc định kiểu int1
int16	Số nguyên 16 bit	byte	Mặc định kiểu int8
int32	Số nguyên 32 bit	int	Mặc định kiểu int8
char	Ký tự 8 bit	Long	Mặc định kiểu int16

- Có thể thêm signed hoặc unsigned để xác định số có dấu hay không
- Đối với PIC16F877A thì chỉ số mảng có kích thước tối đa là 256 byte

Các tiền tố

#include

Thêm các thư viện vào chương trình

```
#include<PIC16F877A.h>
#include “PIC16F877A.h”
```

Thư viện định nghĩa địa chỉ các cổng và thanh ghi

#bit, #byte, #locate và #define

#bit ten_bien = x,y	Biến ten=bien là bit thứ y trong biến (dữ liệu x)
#byte ten_bien = x	x là địa chỉ thanh ghi
#locate ten_bien=x	Giống #byte nhưng có thêm chức năng bảo vệ không cho CCS sử dụng vào mục đích khác
#define ten_bien text	Text là chuỗi hay số

Các tiền tố

#device

#device chip

Chip là tên vi điều khiển sử dụng, không dùng tiền tố này nên
đã khai báo tên chip ở #include

#use

#use delay(clock=speed)

Khai báo hàm delay cho vi điều khiển

Các hàm xử lý số

- Sin(), cos(), tan(), asin(), acos(), atan()
- Abs(),
- Exp()
- Log(), log10()
- Pow()
- Sqrt()

Các hàm xử lý bit

- shift_right(address,byte,value) Dịch phải (trái) 1 bit vào 1 mảng hay một cấu trúc
- Shift_left(address,byte,value)
- Bit_clear(var,bit) Xóa bit được định bởi vị trí trong var
- Bit_set(var,bit) set bit được định bởi vị trí trong var
- Bit_test(var,bit) Kiểm tra xem bit trong var là 0 hay 1
- swap(var) Đảo vị trí 4 bit cao và thấp của var
- make8(var,offset) Trích 1 byte từ biến var với vị trí offset
- make16(varhigh,varlow) Tạo 2 byte từ biến varhigh và varlow
- make32(var1,var2,var3,var4) Tương tự make16()

Hàm delay

Xuất nhập dữ liệu

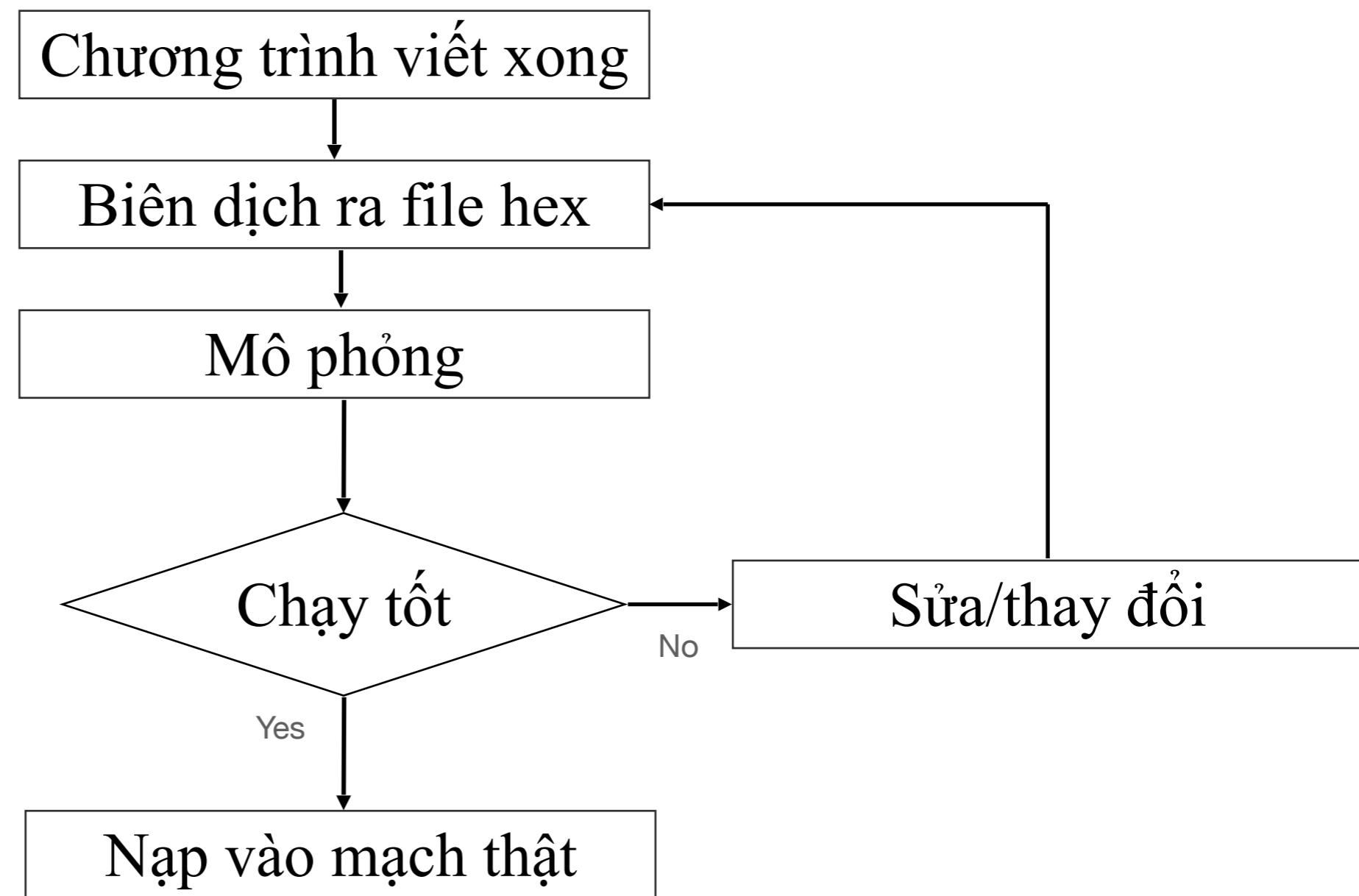
- **Output_low(pin_B0)** Thiết lập mức 0 và 1 cho chân IC. Có thể xuất xung dùng set_tris_X() và #use fast_io
- **Output_high(pin_B0)**
- **Output_bit(pin,value)** pin: tên chân value: giá trị 0 hoặc 1
- **Input_bit(pin)** Hàm trả về giá trị 0 hay 1 là trạng thái chân IC
- **Output_X(value)** Value có giá trị 1 byte, X là tên port
- **Input_X()** Hàm trả về giá trị 8 bit đang có của port
- **Port_B_pullups(value)** Thiết lập ngõ portB pullup. Value=1 sẽ kích hoạt tính năng này và value=0 sẽ ngừng
- **Set_tris_X(value)** Hàm định nghĩa chân vào ra cho cổng X, value là giá trị 8 bit

Ví dụ chương trình nháy led

Chương trình nháy 1 led tại cổng B0 với chu kì 500ms

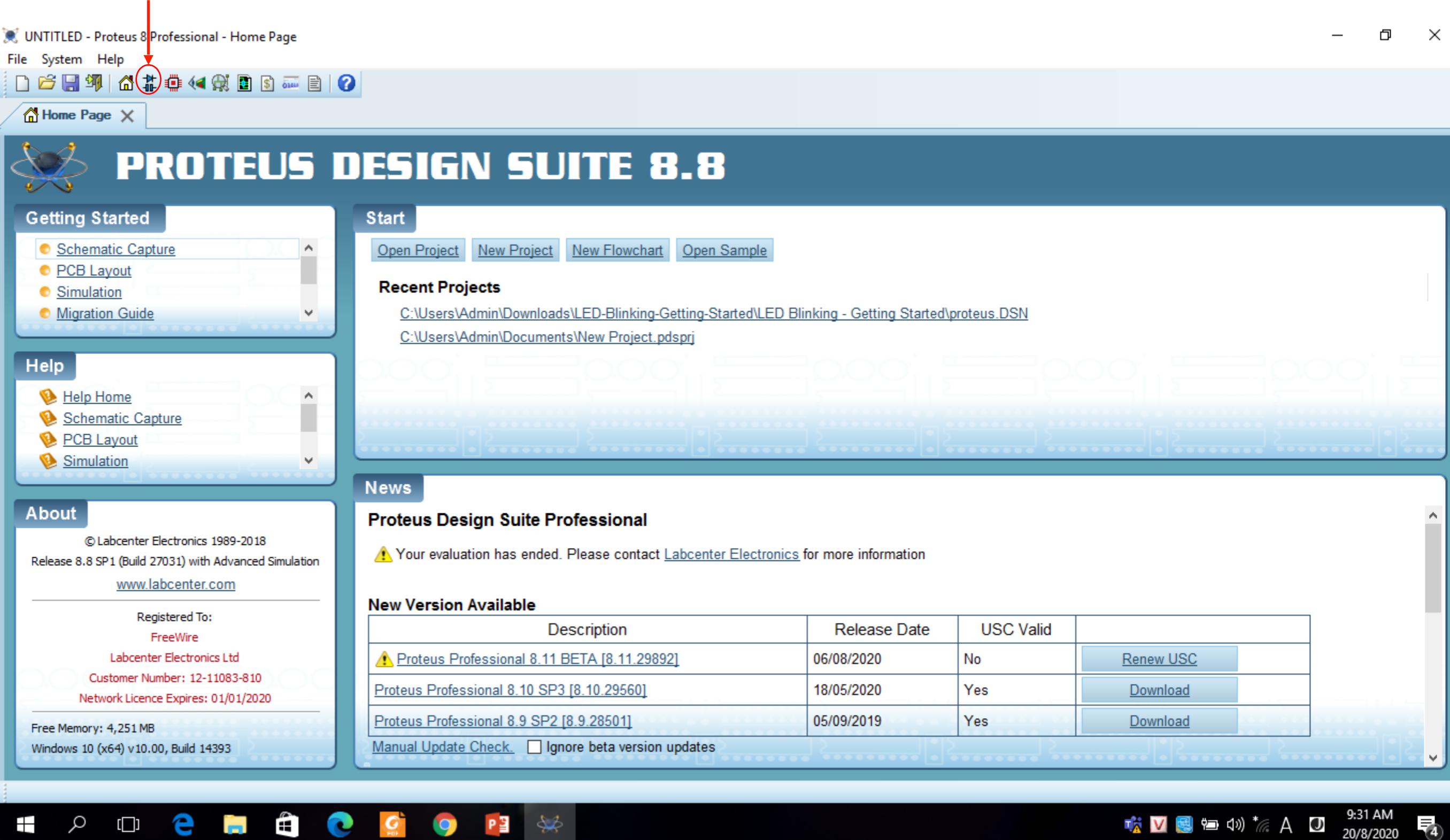
```
#include<16F877A.h>
#use delay(clock=20000000)
Void main()
{ while(1)
    {output_high(pin_B0);
     delay_ms(250);
     output_low(pin_B0);
     delay_ms(250);
    }
}
```

Viết chương trình xong thì làm gì ?

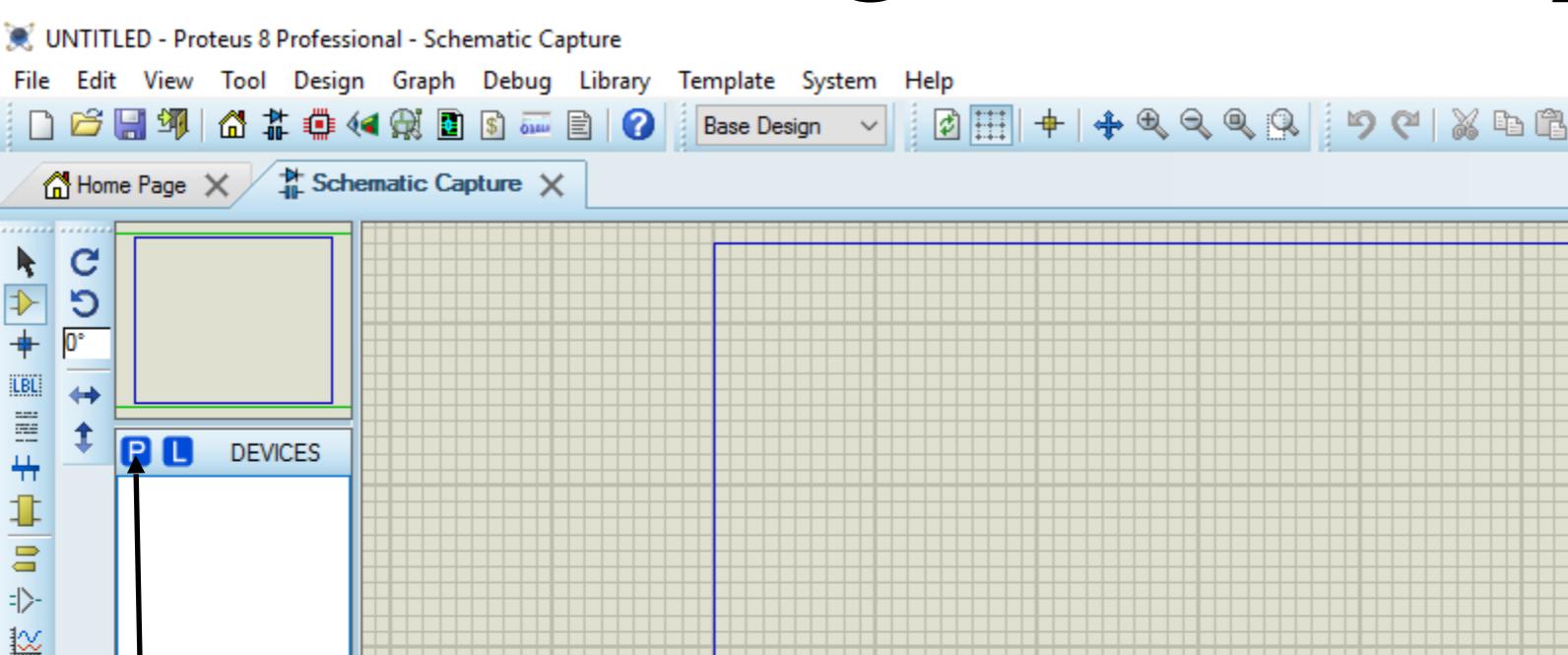


Chương trình mô phỏng Protues

Schematic Capture



Chương trình mô phỏng Protues



Ấn vào chữ P để mở
thư viện linh kiện

Có thể nhập tên linh
kiện mục keywords

PICK DEVICES

Keywords: PIC16F877A

Results (1205):

Device	Library	Description
PIC16F819	PICMICRO	PIC16 Microcontroller (4kB code, 256B data, 256B EPROM, Ports A-B, 3xTimers, CCP1, MSSP, 5x10-bit ADC)
PIC16F83	PICMICRO	PIC16 Microcontroller (512B code, 36B data, 64B EPROM, Ports A-B, 1xTimers)
PIC16F84A	PICMICRO	PIC16 Microcontroller (1024B code, 68B data, 64B EPROM, Ports A-B, 1xTimers)
PIC16F87	PICMICRO	PIC16 Microcontroller (7168B code, 368B data, 256B EPROM, Ports A-B, 1xCCP, 3xTimers, USART)
PIC16F870	PICMICRO	PIC16 Microcontroller (2kB code, 128B data, 64B EPROM, Ports A-C, 1xCCP, 3xTimers, USART, 5x10-bit ADC)
PIC16F871	PICMICRO	PIC16 Microcontroller (2kB code, 128B data, 64B EPROM, Ports A-E, 1xCCP, PSP, 3xTimers, USART, 8x10-bit ADC)
PIC16F873	PICMICRO	PIC16 Microcontroller (4kB code, 192B data, 128B EPROM, Ports A-C, 2xCCP, 3xTimers, MSSP, USART, 5x1C)
PIC16F873A	PICMICRO	PIC16 Microcontroller (4kB code, 192B data, 128B EPROM, Ports A-C, 2xCCP, 3xTimers, MSSP, USART)
PIC16F874	PICMICRO	PIC16 Microcontroller (4kB code, 192B data, 128B EPROM, Ports A-E, 2xCCP, PSP, 3xTimers, MSSP, USART)
PIC16F874A	PICMICRO	PIC16 Microcontroller (4kB code, 192B data, 128B EPROM, Ports A-E, 2xACMP, 2xCCP, PSP, 3xTimers, MSSP)
PIC16F876	PICMICRO	PIC16 Microcontroller (8kB code, 368B data, 256B EPROM, Ports A-C, 2xCCP, 3xTimers, MSSP, USART, 5x1C)
PIC16F876A	PICMICRO	PIC16 Microcontroller (8kB code, 368B data, 256B EPROM, Ports A-C, 2xACMP, 2xCCP, 3xTimers, MSSP, USART)
PIC16F877	PICMICRO	PIC16 Microcontroller (8kB code, 368B data, 256B EPROM, Ports A-E, 2xCCP, PSP, 3xTimers, MSSP, USART)
PIC16F877A	PICMICRO	PIC16 Microcontroller (8kB code, 368B data, 256B EPROM, Ports A-E, 2xACMP, 2xCCP, PSP, 3xTimers, MSSP)
PIC16F88	PICMICRO	PIC16 Microcontroller (7168B code, 368B data, 256B EPROM, Ports A-B, 1xCCP, 3xTimers, USART)
PIC16F882	PICMICRO	PIC16 Microcontroller (2kB code, 128B data, 128B EPROM, Ports A-C,E, 2xACMP, ECCP/CCP, 3xTimers, MSSP)
PIC16F883	PICMICRO	PIC16 Microcontroller (4kB code, 352B data, 256B EPROM, Ports A-C,E, 2xACMP, ECCP/CCP, 3xTimers, MSSP)
PIC16F884	PICMICRO	PIC16 Microcontroller (4kB code, 352B data, 256B EPROM, Ports A-E, 2xACMP, ECCP/CCP, 3xTimers, MSSP)
PIC16F886	PICMICRO	PIC16 Microcontroller (8kB code, 352B data, 256B EPROM, Ports A-C,E, 2xACMP, ECCP/CCP, 3xTimers, MSSP)
PIC16F887	PICMICRO	PIC16 Microcontroller (8kB code, 352B data, 256B EPROM, Ports A-E, 2xACMP, ECCP/CCP, 3xTimers, MSSP)
PIC16F913	PICMICRO	PIC16 Microcontroller (4kB code, 256B data, 256B EPROM, Ports A-C,E, 2xACMP, 2xCCP, LCD, 3xTimers, MSSP)
PIC16F914	PICMICRO	PIC16 Microcontroller (4kB code, 256B data, 256B EPROM, Ports A-E, 2xACMP, 2xCCP, LCD, 3xTimers, MSSP)
PIC16F916	PICMICRO	PIC16 Microcontroller (8kB code, 352B data, 256B EPROM, Ports A-C,E, 2xACMP, 2xCCP, LCD, 3xTimers, MSSP)
PIC16F917	PICMICRO	PIC16 Microcontroller (8kB code, 352B data, 256B EPROM, Ports A-E, 2xACMP, 2xCCP, LCD, 3xTimers, MSSP)
PIC16F946	PICMICRO	PIC16 Microcontroller (8kB code, 336B data, 256B EPROM, Ports A-F,G, 2xACMP, 2xCCP, LCD, 3xTimers, MSSP)
PIC16HV610	PICMICRO	PIC16 Microcontroller (1kB code, 64B data, Ports A & C, 2xACMP, 2xTimers)
PIC16HV616	PICMICRO	PIC16 Microcontroller (2kB code, 128B data, Ports A & C, 2xACMP, 3xTimers, 1xECCP, 8x10-bit ADC)
PIC16HV785	PICMICRO	PIC16 Microcontroller (2kB code, 128B data, 256B EPROM, Ports A-C, 2xACMP, 3xTimers, CCP, 12x10-bit ADC)
PIC16HV14A	PICMICRO	PIC16 Microcontroller (4kB code, 1024B data, Ports A-C, 2xTimers, 2xPWM, ECLAPTR, MSSP, I2C)

PIC16F877A Preview:

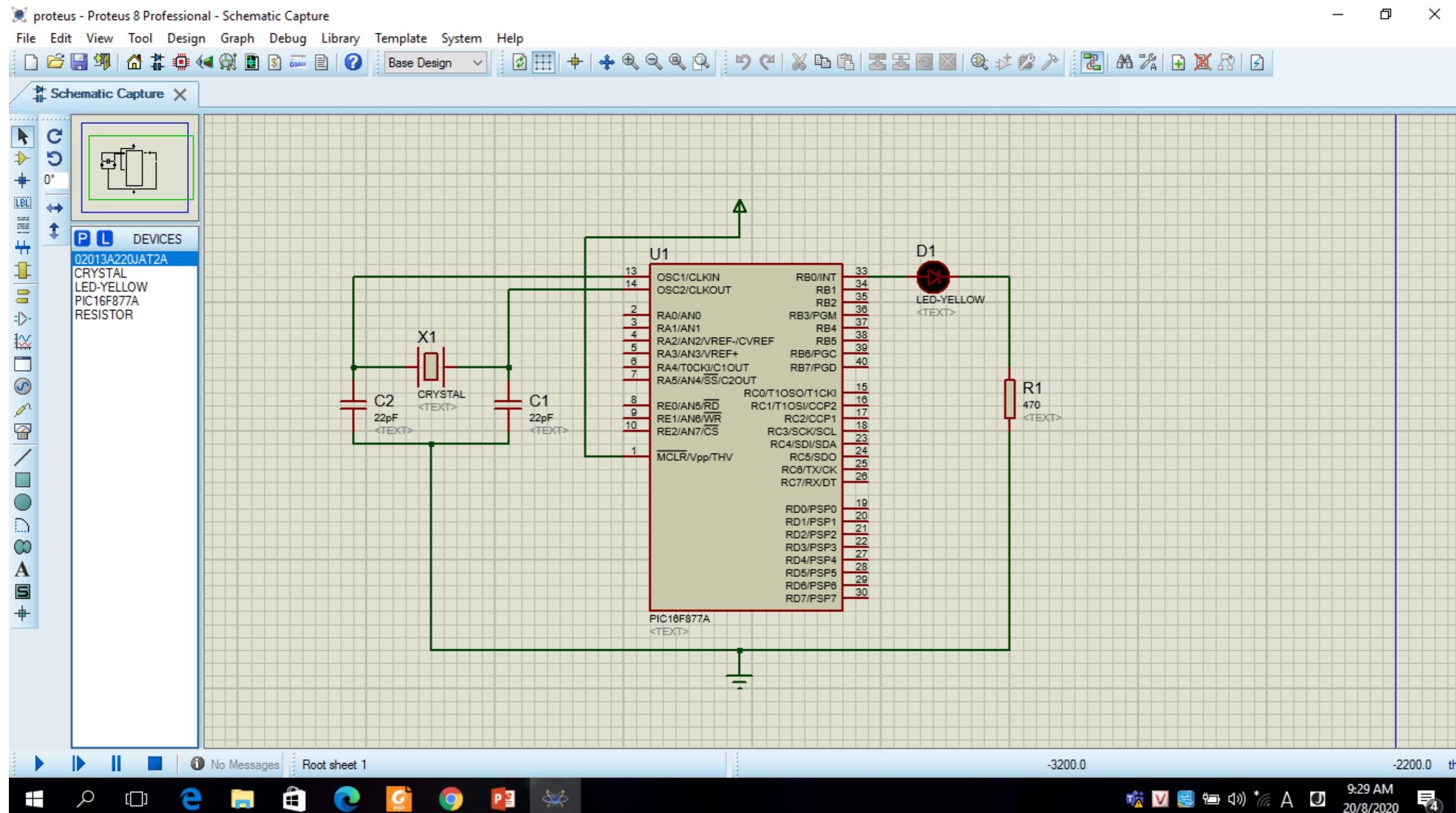
VSM DLL Model [PIC16]

PCB Preview:

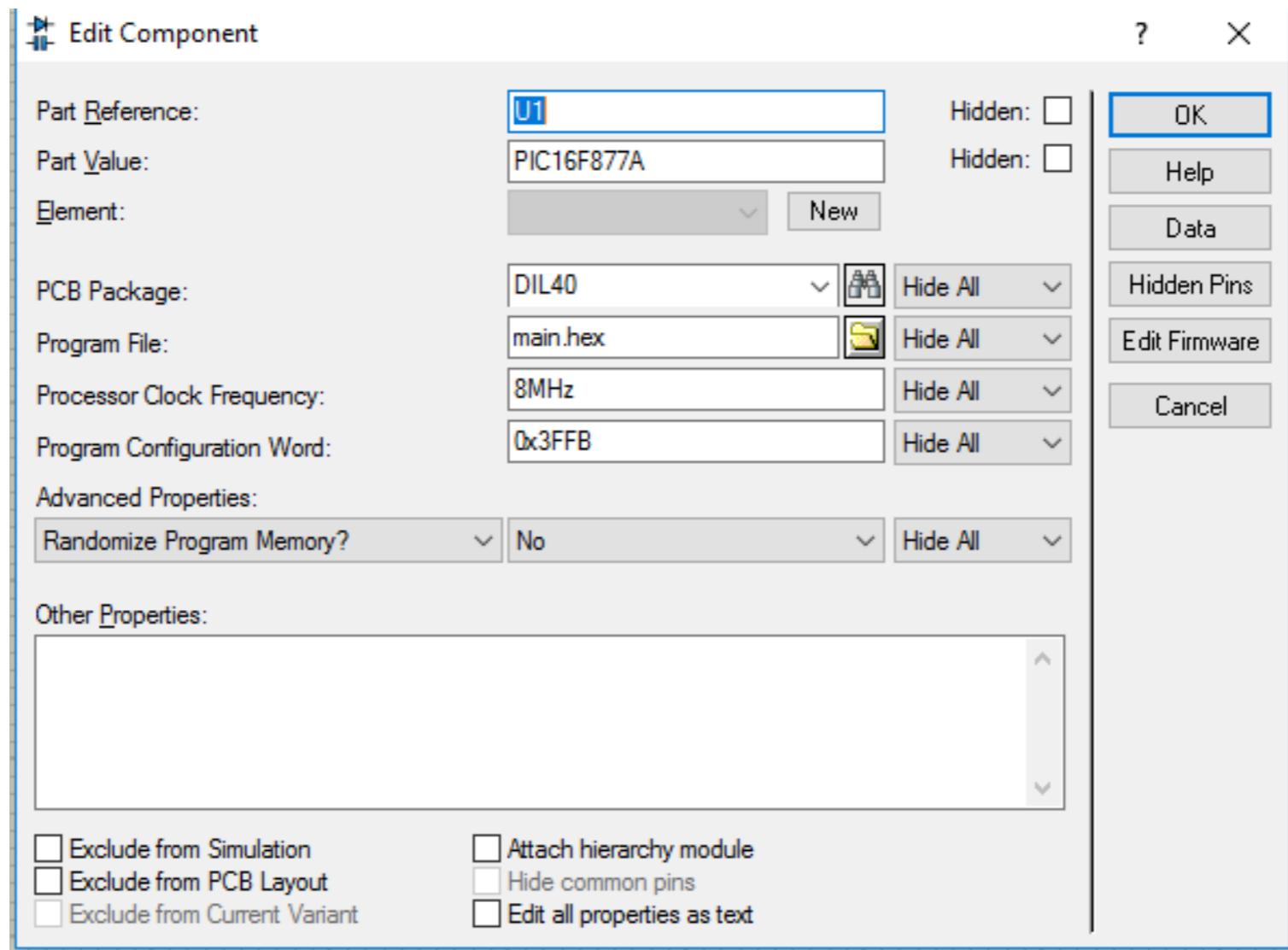
DIL40

OK Cancel

Chương trình mô phỏng Protues

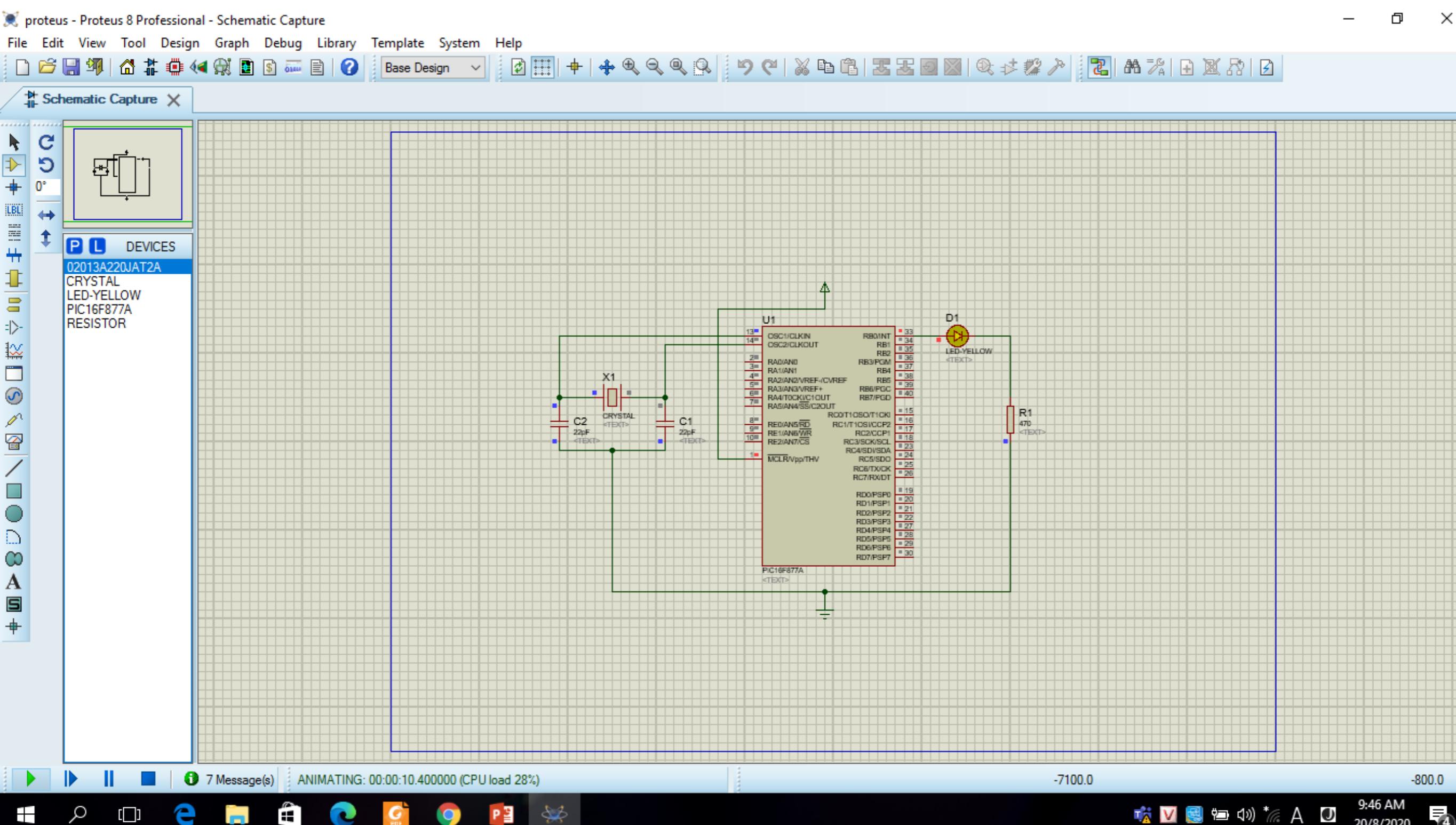


Chương trình mô phỏng Protues

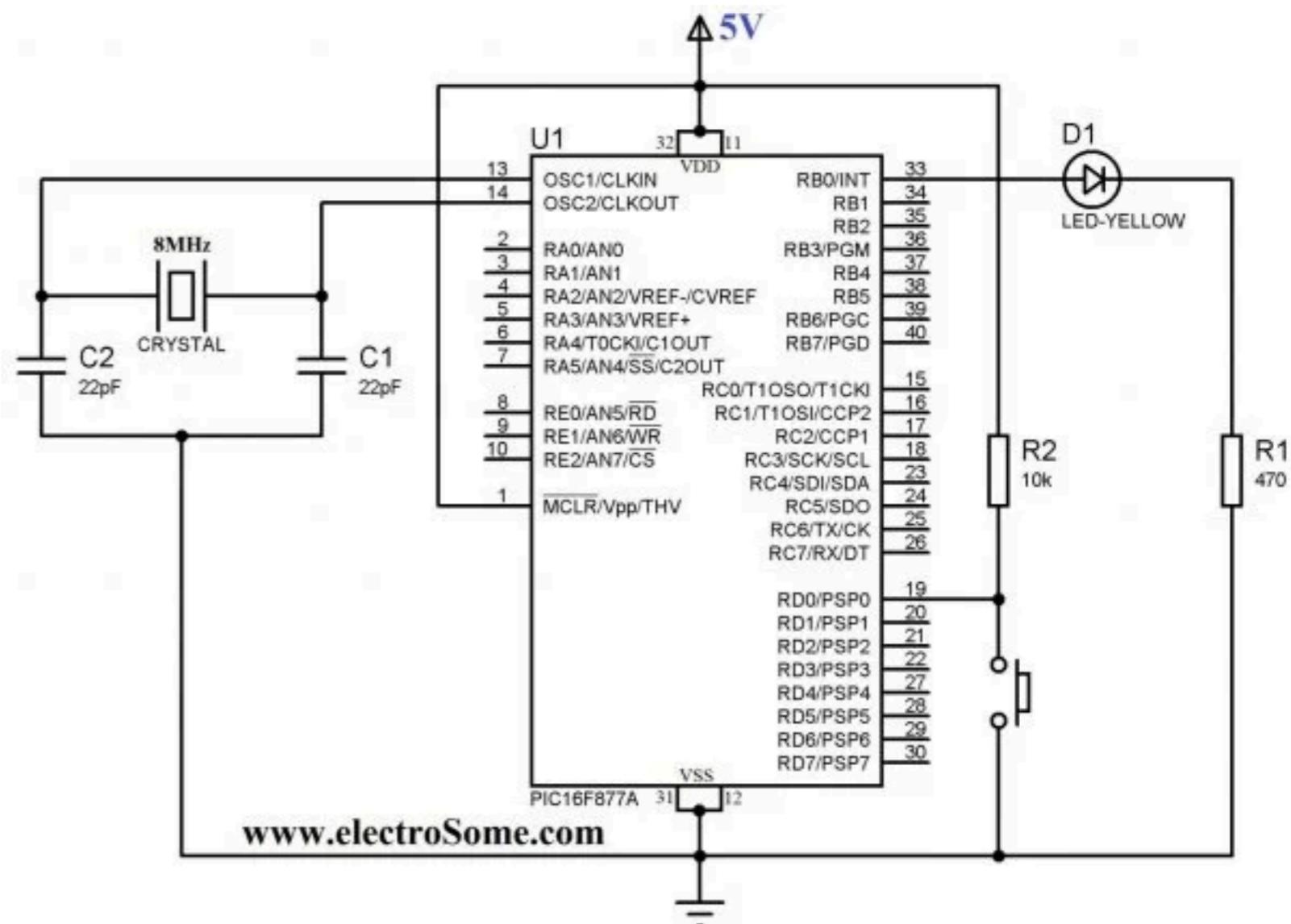
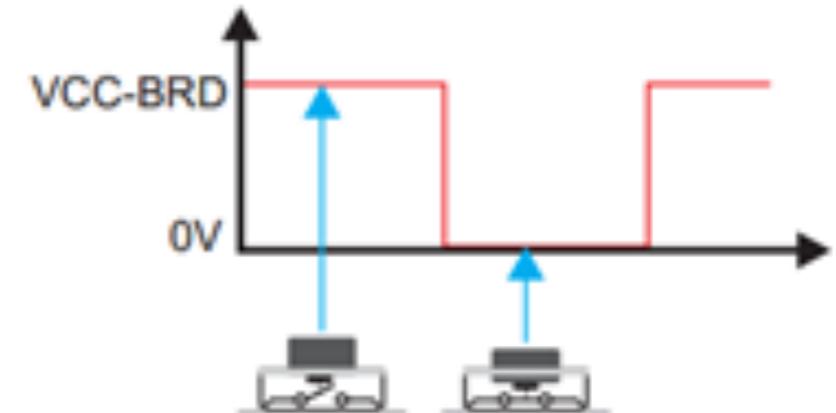
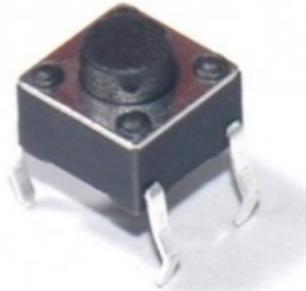
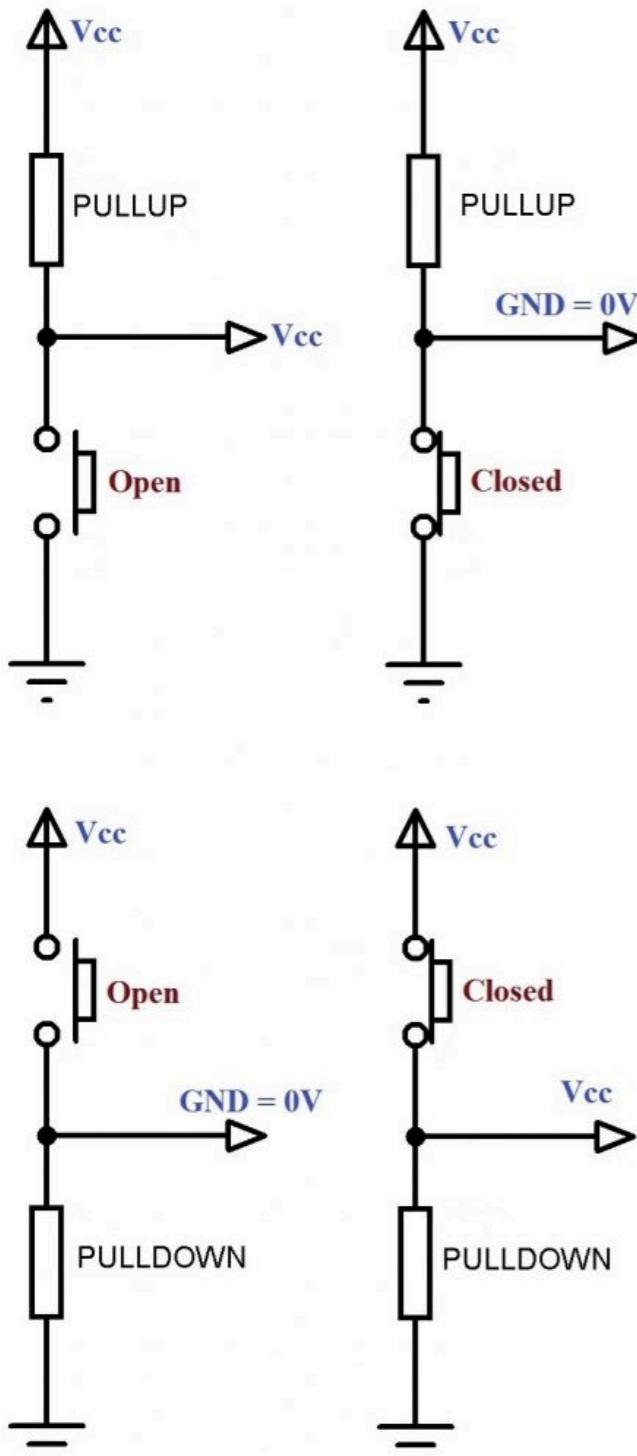


- Có thể mô phỏng với tần số hoạt động khác
- Nạp chương trình mô phỏng bằng cách chọn phần program file → chọn tiếp đến file hex mà CCS đã biên dịch ra

Chương trình mô phỏng Protues



Giao tiếp phím bấm



Kiểm tra trạng thái phím bấm

Dùng vòng lặp và điều kiện kiểm tra

- Nối phím vào chân VĐK
- Sử dụng vòng quét chính để kiểm tra trạng thái
- Dùng lệnh if để kiểm tra trạng thái cồng

```
while(1){ //Chờ khi nào chân nối tới phím bấm xuống mức thấp
    if(phim_bam==0){
        //code ctr
    }
}
hoặc
while(phim_bam); //khi p1.0 xuống 0 làm điều kiện sai vòng lặp sẽ thoát
{
//code ctr
}
```

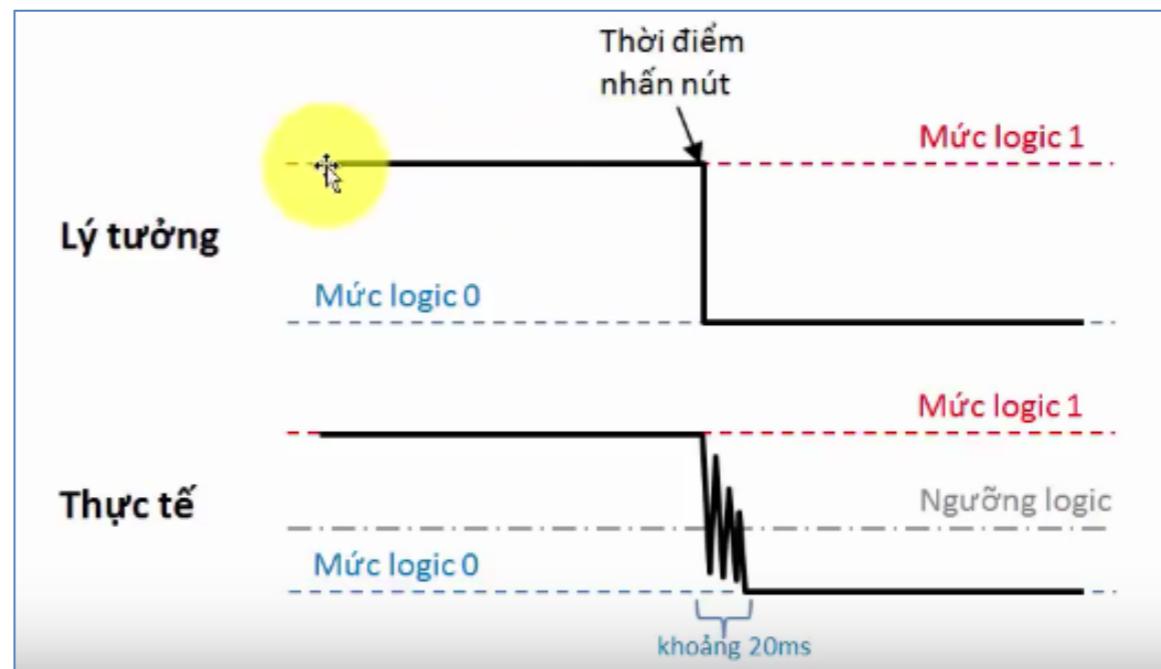
Ví dụ

```
#include <main.h>

#use delay (clock=8000000)

void main()
{
    output_low(PIN_B0);      //LED OFF
    output_float(PIN_D0);    //Set RD0 as Input Pin
                            //OR
    set_tris_x(0b00000001)
    while(TRUE)
    {
        if(input_state(PIN_D0) == 0)
        {
            output_high(PIN_B0); //LED ON
            delay_ms(2000);     //2 Second Delay
            output_low(PIN_B0); //LED OFF
        }
    }
}
```

Hiện tượng dội phím



Không ổn định khi
chuyển trạng thái



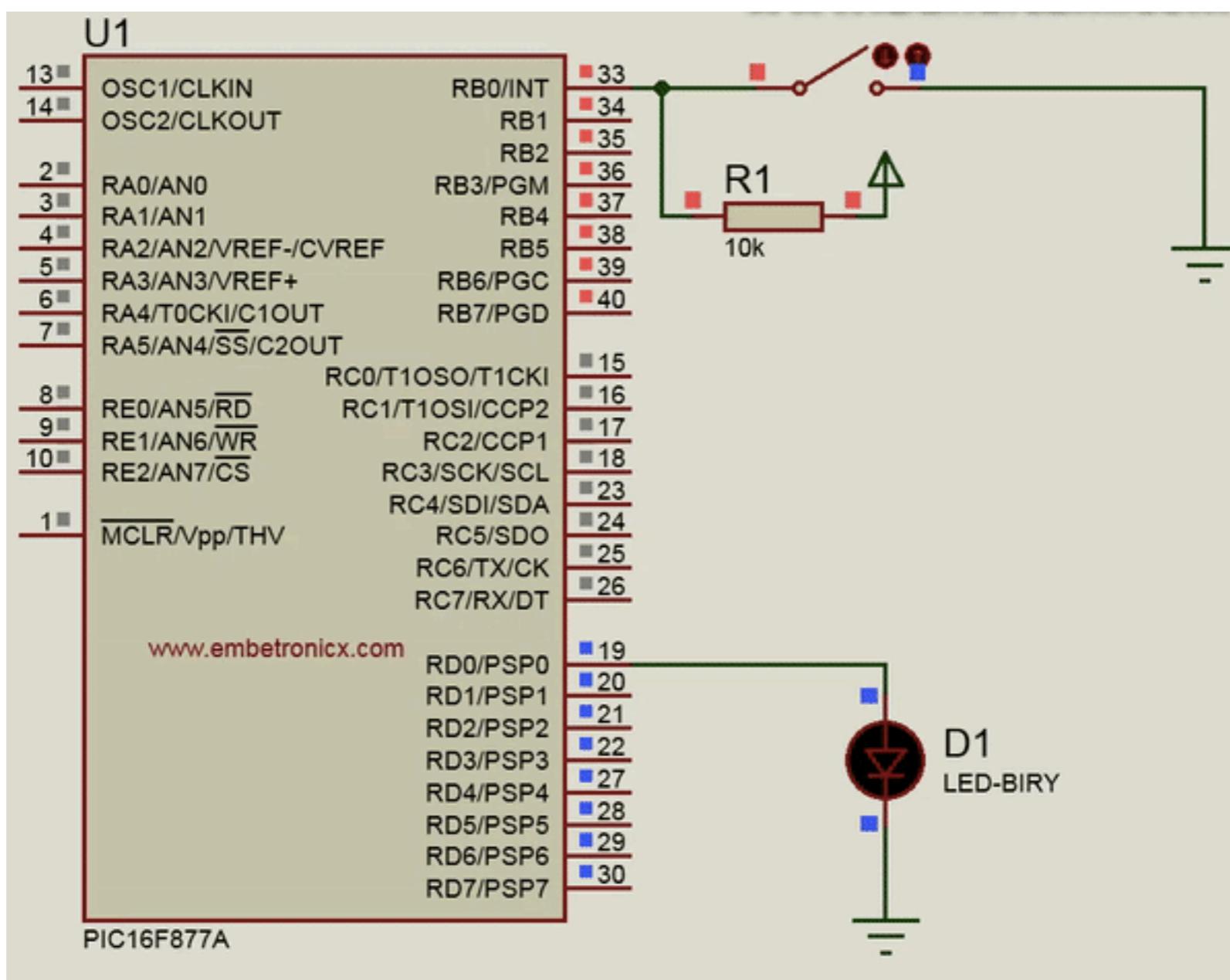
Cần một khoảng thời
gian trễ ~20ms trước khi
kiểm tra lần 2

```
while(1){  
if(BTN1 == 0) //Nút nhấn được nhấn  
{  
delay_ms(20); //Trễ 20 ms, phòng trường hợp dội phím  
if (BTN1 == 0) //Kiểm tra lại xem nút nhấn được nhấn  
{ // chương trình...  
}
```

Kiểm tra trạng thái phím bấm

Dùng ngắt ngoài

Phím bấm được nối tới đầu vào của ngắt RB/INT (chân 33)



Kiểm tra trạng thái phím bấm

Dùng ngắt ngoài

```
#include<pic16f877a.h>
#define LED PORTD

void interrupt ISR(void)
{
    unsigned int i,j;
    LED=0X55;
    for(i=0;i<600;i++)
        for(j=0;j<200;j++);           // Tạo ra trễ
    INTF=0;                           // Xoá cờ ngắt
}

void main(void)
{
    LED=0;
    TRISB0=1;                      // Mở cổng ngắt
    TRISD=0;
    OPTION_REG=0X00;                //Khởi tạo ngắt
    INTCON|=0Xd0;
    while(1) {
        LED=0x00;
    }
}
```