TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP
KHOA ĐIỆN TỬ

TÀI LIỆU THỰC HÀNH

HỌC PHẦN: HỆ THỐNG NHÚNG

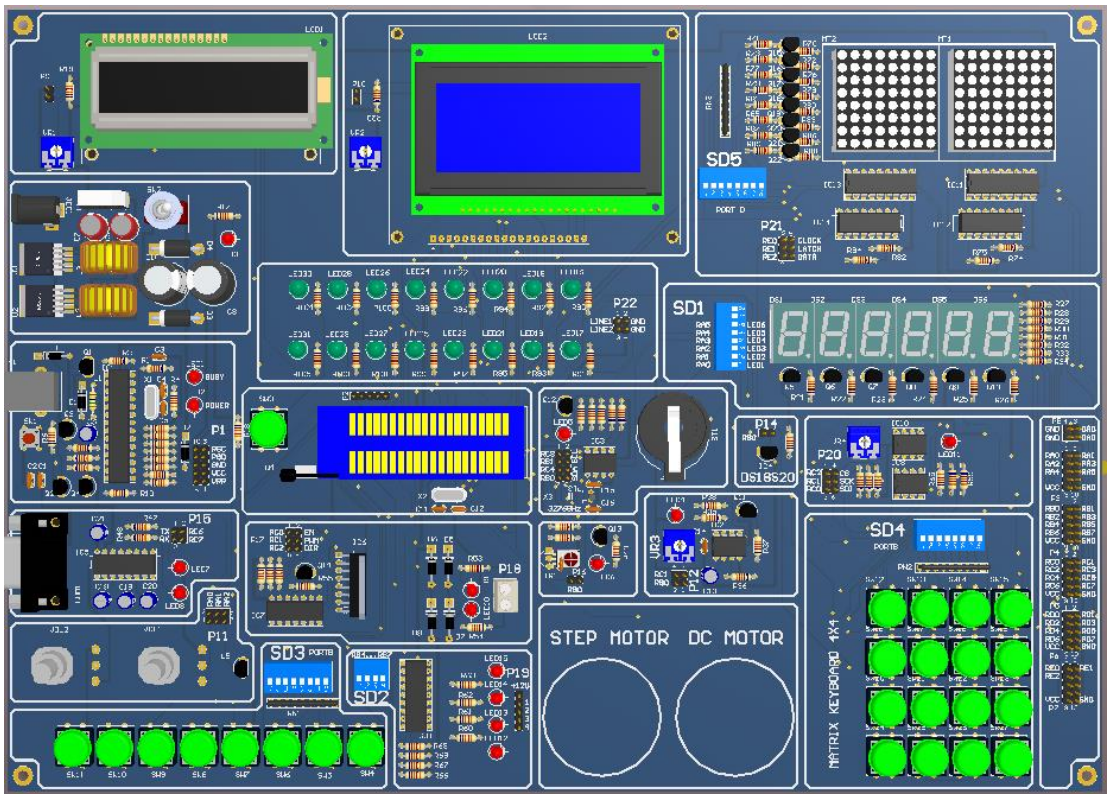
MỤC LỤC

PHẦN 1: GIỚI THIỆU MUDULE VI ĐIỀU KHIỂN PIC

1.1. Giới thiệu

Module vi điều khiển PIC là kit phát triển hoàn chỉnh, đầy đủ tính năng và dễ sử dụng cho MicroChip PIC Microcontroller. Với Module vi điều khiển PIC Người dùng không phải bận tâm về phần cứng, mà tập trung vào các ứng dụng phần mềm.

Module vi điều khiển PIC Thiết kế đặc biệt không dùng dây cắm. Các module được kết nối hoặc ngắt khỏi MCU một cách linh hoạt bằng Jumper hoặc DipSW rất tiện lợi và gọn gàng.



Hình 1: Sơ đồ tổng thể Module đào tạo vi điều khiển PIC.

1.2. Nhưng tính năng của Kit vi điều khiển PIC

1. Power Supply: Sử dụng nguồn ngoài AC/DC 7 ~ 12V có công tắc nguồn ON/OFF.

2. Tích hợp sẵn mạch nạp PICKIT2 để nạp và debug chương trình cho vi điều khiển,ngoài ra còn có đường ICSP programmer sẵn sàng sử dụng cho mạch nạp sử dụng chuẩn ICSP .

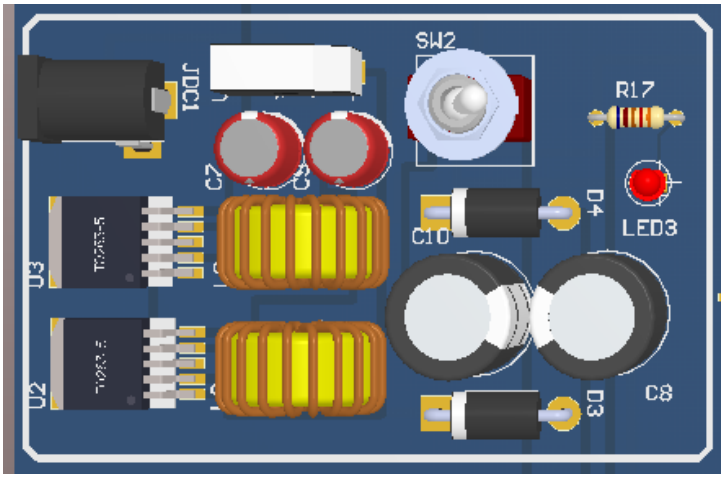
3. Suport dòng PIC16,PIC18 8bitMCU/DIP40 (có thể dùng cho DIP20 bằng cách sử dụng Adapter chuyển chân).
4. LCD 16x2 4 BIT có biến trở chỉnh độ tương phản.
5. Khối phím đơn gồm 8 nút bấm up/down .
6. Ma trận bàn phím4x4 (16 nút) bấm ma trận kết nối qua DipSW.
7. RS-232 Comunication giao tiếp truyền dữ liệu với PC.
8. Khối thời gian thực (RTC DS1307 Real time clock) kết nối hoặc ngắt khỏi Vi điều khiển qua DIPSW.
9. DS1820 Digital thermometer dùng để đo nhiệt độ từ -55ºC to 125ºC.
10. Điều khiển thu phát hồng ngoại - IR.
11. Bộ chuyển đổi ADC 3 kênh – 8 bit hoặc 10 bit.
12. 6 Led 7seg Anode chung multiplex mode. kết nối hoặc ngắt khỏi Vi điều khiển qua DIPSW.
13. LED ma trận hiển thị số - ký tự .
14. 16 led đơn anod chung.
15. Điều khiển động cơ bước.
16. Điều khiển động cơ một chiều.
17. Graphic LCD 128x64 / Charracter LCD16x2 8-bit có biến trở chỉnh độ tương phản.
18. Xtal sử dụng socket dễ dàng thay đổi.
19. RESET Switch - loại lớn cho phép dễ dàng Reset mạch bằng tay .
20. Mạch in xuyên lớp chất lượng cao kích thước 350x250 mm.
21. Có Mica bảo vệ bên dưới chống ngắn mạch.

1.3. Cấu hình chi tiết các khối module

1.3.1. Khối nguồn

- Cung cấp nguồn DC5V cho toàn bộ các module của KIT VDK PIC

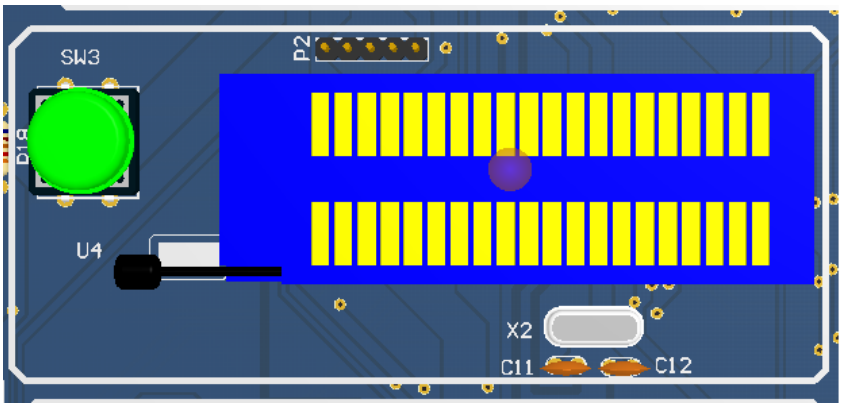
- Cung cấp nguồn DC12V cho khối công suất của module điều khiển DC Motor và STEP Motor



Hình 2: Khối nguồn

1.3.2. Khối Vi điều khiển

- Sử dụng đường nạp ICSP từ mạch nạp ngoài hoặc từ mạch nạp đã tích hợp sẵn trên KIT PIC (có thể dùng cho các dòng PIC16Fxxx,PIC18Fxxx 8bitMCU/DIP40).

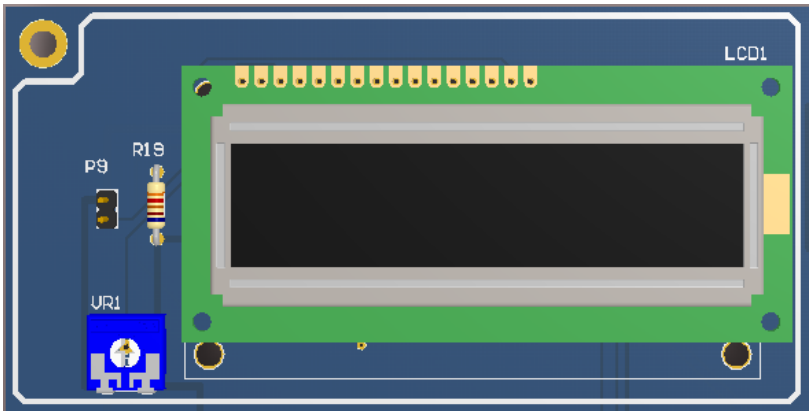


Hình 3: Khối vi điều khiển trung tâm

1.3.3. Khối hiển thị LCD 16x2

- Màn hình LCD 16x2 được thiết kế ở chế độ giao tiếp 4bit (Giao tiếp với VDK qua Port P2).

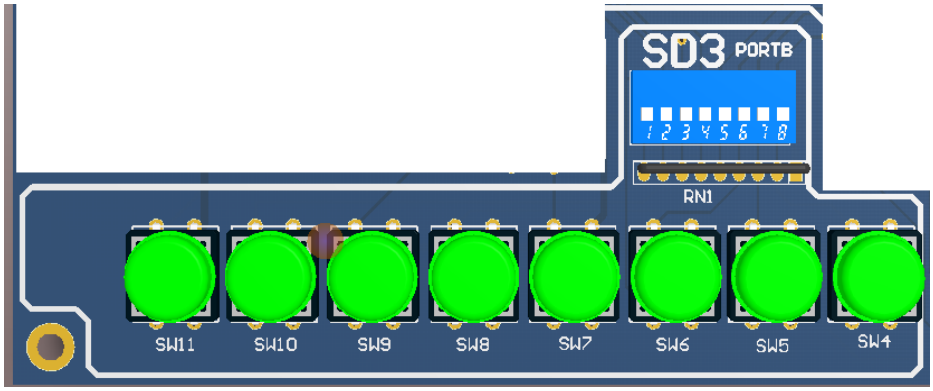
- Biến trở VR1 điều chỉnh độ sang LCD.



Hình 4: Khối hiển thị LCD 16x2

1.3.4. Khối phím đơn

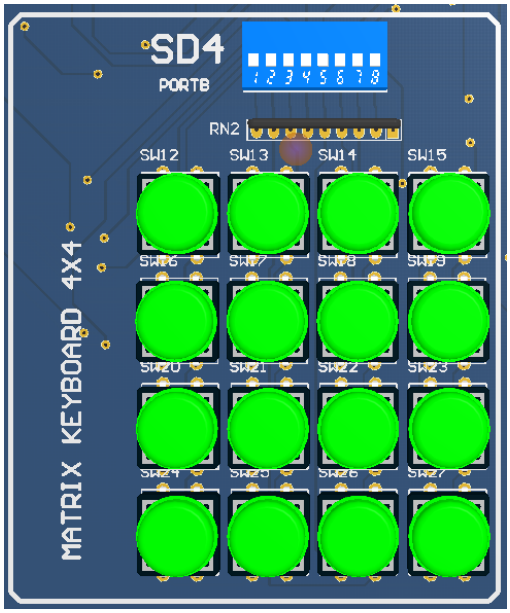
- 8 phím đơn giao tiếp trực tiếp với VDK qua PORT P1
- Thực hiện điều khiển kết nối thông qua Dip S8



Hình 5: Khối phím đơn

1.3.5. Khối phím ma trận

- Ma trận phím bấm là ma trận 4x4.
- Thực hiện điều khiển kết nối thông qua Dip S5.

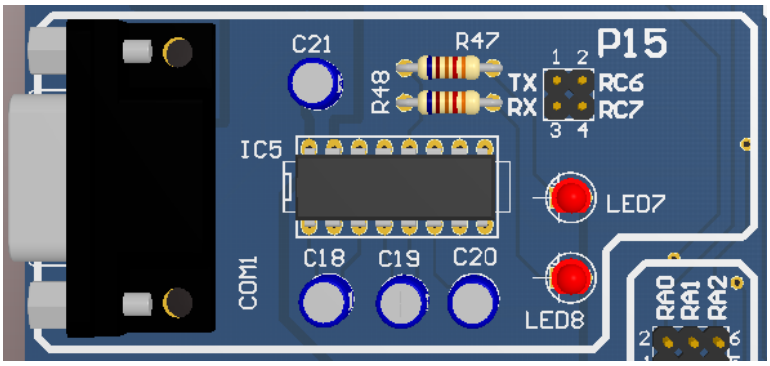


Hình 6: Khối ma trận bàn phím

1.3.6. Khối giao tiếp RS232

Giao tiếp giữa VDK với thiết bị ngoài thông qua chuẩn giao tiếp RS232

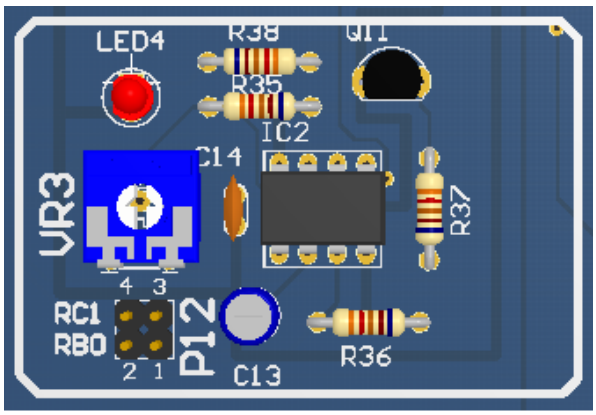
Ngoài ra còn có các chân cắm để kết nối giữa Module điều khiển trung tâm với các Module khác



Hình 7: Khối giao tiếp RS232

1.3.7. Khối tạo xung (Bộ đếm)

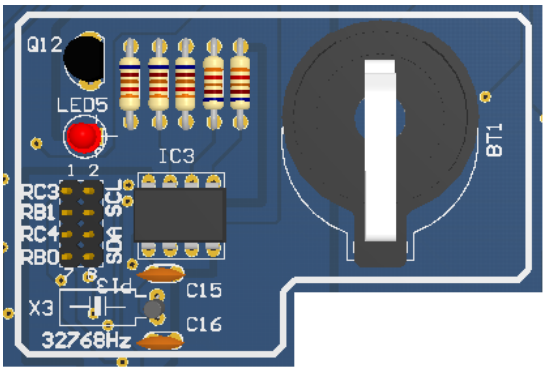
- Khối tạo xung vuông có thể điều chỉnh tần số bằng biến trở để tạo các tín hiệu như: ngắt theo chu kỳ, .. *sử* dụng làm nguồn ngắt ngoài cho VDK trong các ứng dụng lập trình External Interrupts,Counter
- Giao tiếp I^2C với khối điều khiển trung tâm thông qua P3.4 và P3.2



Hình 8: Khối tạo xung vuông

1.3.8. Khối thời gian thực (Real Time Clock)

- Sử dụng DS1307 với nguồn Pin dự phòng
- Giao tiếp với IC thời gian thực qua chuẩn I^2C sử dụng RC1 và RB0



Hình 9: Khối thời gian thực

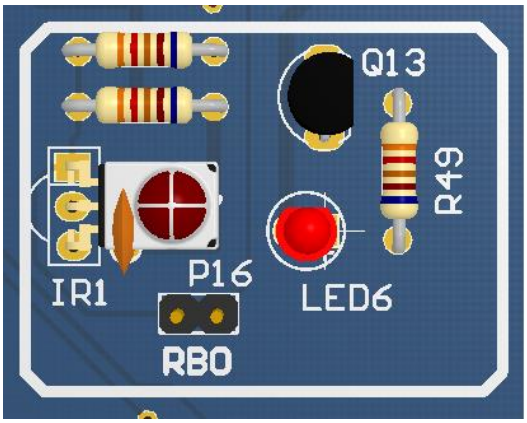
1.3.9. Khối đo nhiệt độ - DS18S20

- *Cho phép VDK giao tiếp với cảm biến nhiệt độ DS18S20 qua chuẩn giao tiếp ONE-WIRE*

Hình 10: Khối đo nhiệt độ

1.3.10. Khối thu tín hiệu hồng ngoại – IR

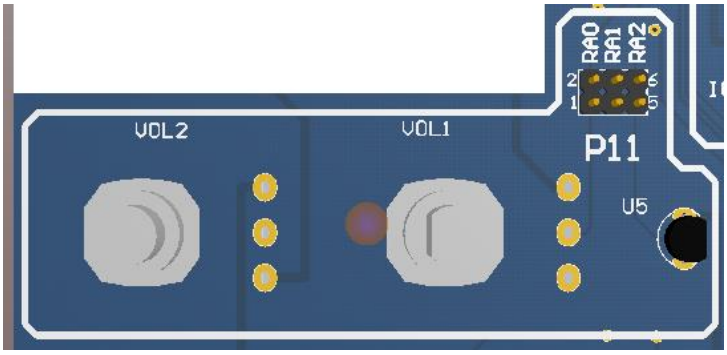
- Cho phép VDK nhận tín hiệu điều khiển từ xa dùng hồng ngoại để xử lý
- Kết nối với VĐK trung tâm thông qua RB0.



Hình 11: Khối thu tín hiệu hồng ngoại

1.3.11. Khối chuyển đổi analog – digital (ADC)

- Trong Module có sẵn 2 biến trở VOL1 , VOL2 để điều chỉnh độ lớn tín hiệu Analog, cảm biến nhiệt độ LM35



Hình 12: Khối chuyển đổi ADC

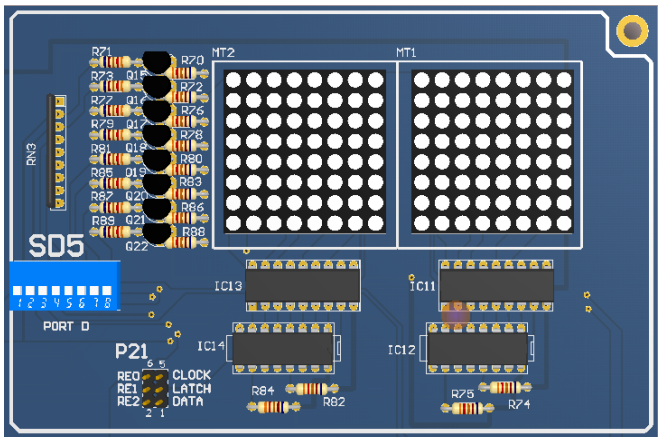
1.3.12. Khối hiển thị LED 7 thanh

Bao gồm 6 led 7 thanh được mắc chung đường dữ liệu thực hiện các bài toán hiển thị số từ vi điều khiển.

Hình 13: Khối hiển thị LED 7 thanh

1.3.13. Khối led Matrix

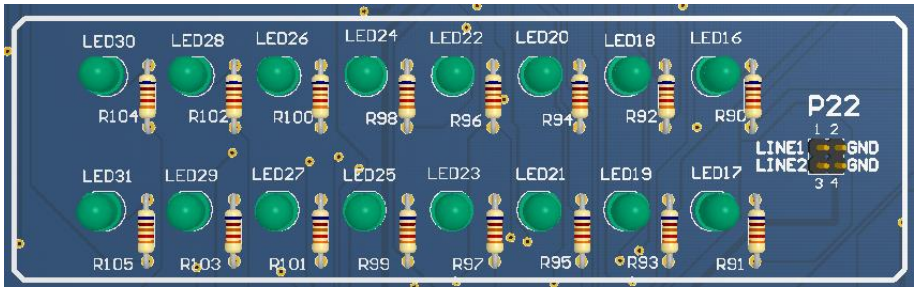
Gồm 2 led matrix được thiết kế sẵn để hiển thị theo phương pháp quét cột



Hình 14: Khối LED ma trận

1.3.14. Khối led đơn

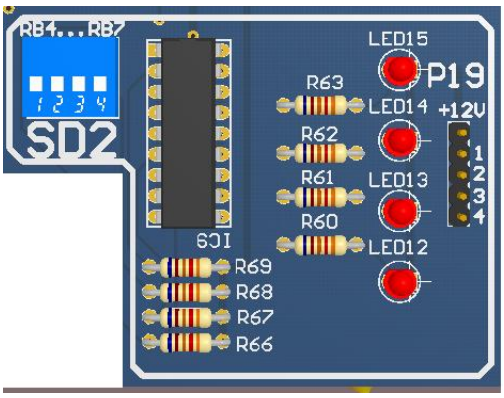
Gồm 2 hàng ,mỗi hàng 8 led đơn mắc chung chân Katot giúp hiển thị trạng thái đầu ra vi điều khiển



Hình 15: Khối Led đơn

1.3.15. Khối điều khiển STEP Motor.

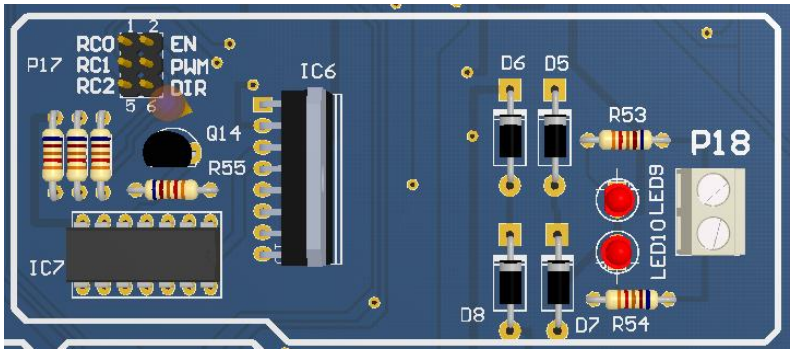
- Sử dụng mạch khuếch đại công suất cho động cơ bước 12V.
- Kết nối với vi điều khiển qua chân RB4, RB5, RB6,RB7 .



Hình 16: Khối điều khiển động cơ bước

1.3.16. Khối điều khiển động cơ một chiều (DC motor)

Gồm 1 động cơ DC 12V và mạch điều khiển , mạch công suất chuyên dụng cho động cơ DC



Hình 17: Khối điều khiển động cơ một chiều

PHẦN 2: HƯỚNG DẪN SỬ DỤNG MODULE VI ĐIỀU KHIỂN PIC VÀ PHẦN MỀM VÀ PIC C Compiler.

2.1. Hướngdẫn sử dụng Module vi điều khiển PIC

A. Khi kết nối mạch:

- Tên của từng port xuất nhập trên mô hình.
- Thứ tự các bit (từ LSB đến MSB) tại các port xuất nhập trên mô hình.
- Khi kết nối phải đảm bảo sao cho bit 0 của port vi điều khiển đúng vị trí bit 0 của đối tượng cần điều khiển.
- Khi kết nối đúng vị trí bit 0 thì các bit còn lại sẽ đúng vị trí.
- Tất cả các chương trình mẫu trong hệ thống này đều được kiểm tra rất kỹ theo đúng như kết nối mạch được trình bày.
- Nếu một yêu cầu nào đó không đúng thì hãy xem lại phần kết nối và chương trình.

B. Khi viết chương trình:

- Số 0 thường được đánh nhầm là chữ O.
- Thường đánh thiếu tiền tố # và hậu tố H đi kèm trong một số trường hợp.
- Sau lệnh END thì không còn một hàng hay một ký tự nào (kể cả ký tự trắng) nếu không chương trình biên dịch sẽ báo lỗi. Lỗi này có thể bỏ qua.
- Hãy dùng phím TAB để viết chương trình cho thẳng hàng. Điều này rất có ích cho bạn khi cần xem lại và kiểm tra lỗi chương trình được nhanhchóng.
- Nếu nhập một chương trình nào đó trong tài liệu mà chương trình chạy không đúng như yêu cầu thì hãy xem kỹ lại có đánh đầy đủ tất cả các lệnh trong chương trình hay chưa? Có thiếu sót gì không? Kết nối mạch có theo như hướng dẫn hay không? Tất cả các chương trình trong tài liệu hướng dẫn đã được tets thử.

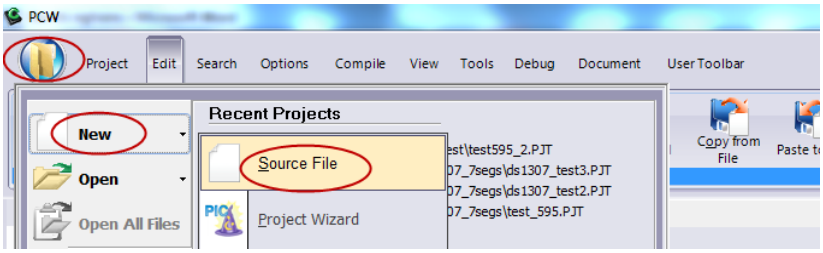
2.2. Hướng dẫn sử dụng phần mềm

2.2.1. *Viết chương trình với PIC C Compliler*

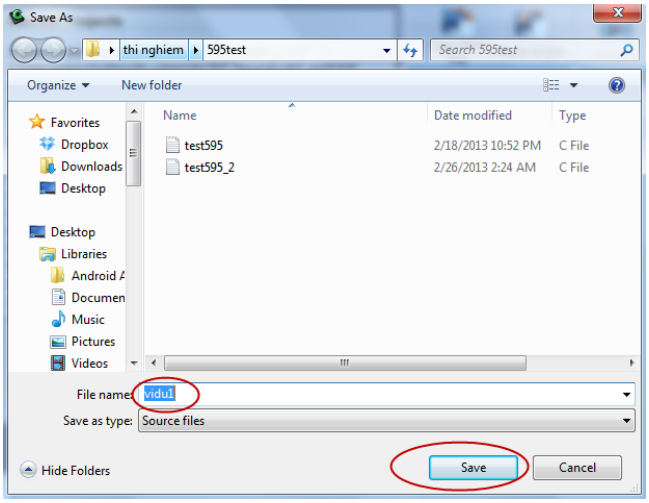
Bước 1 : Chạy phần mềm PWCHD.

Start_|All Program_|PIC-C_|PIC C Compliler.

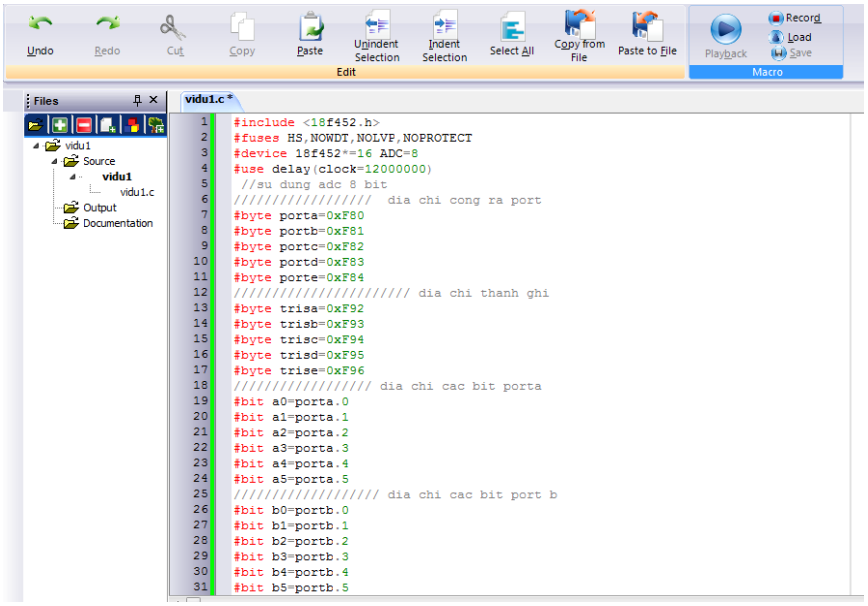
Bước 2 : Chọn menu:



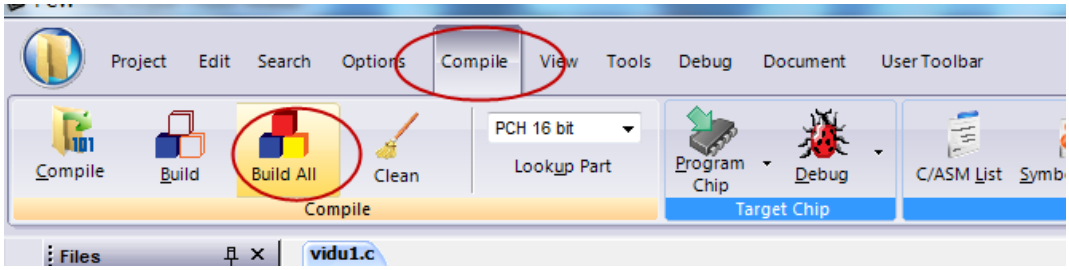
Bước 3 :



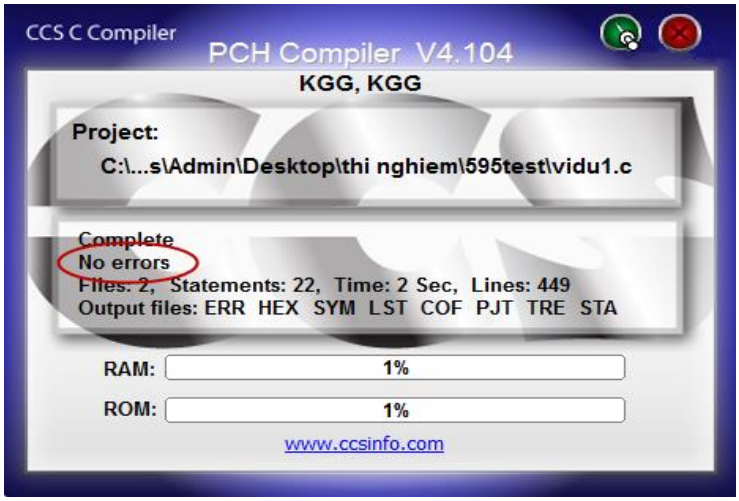
Bước 4: Viết chương trình vào cửa số chương trình có tến là vidu1.c vừa tạo.



Bước 5: Biên dịch chương trình ra file HEX, bằng cách Menu/Compile/ Build All.



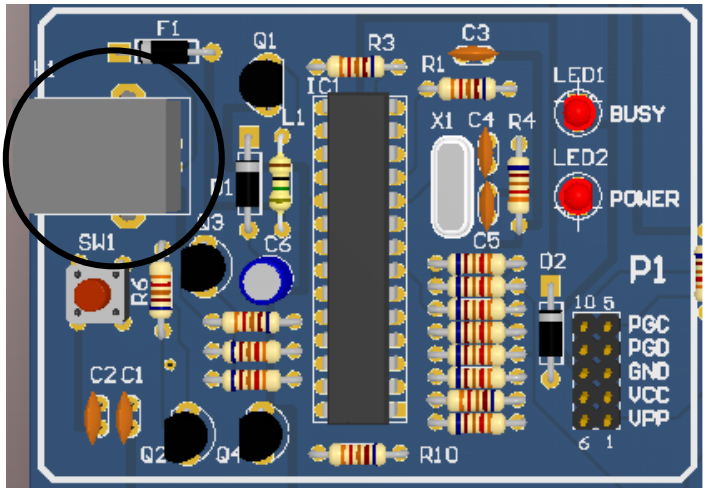
Bước 6: Nếu chương trình không báo lỗi màn hình sẽ hiện lên thông báo.



Sau các bước trên ta đã có file Hex dùng để nạp chip, File này sẽ trong cùng thư mục với file vidu1.c .

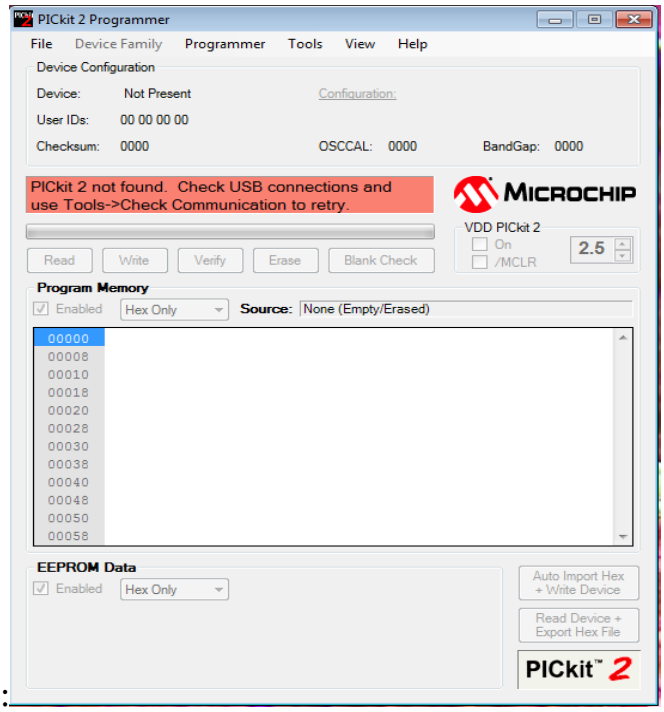
2.2.2. *Nạp chương trình cho Vi điều khiển dùng PICkit2*

-Cắm dây USB vào mạch nạp:



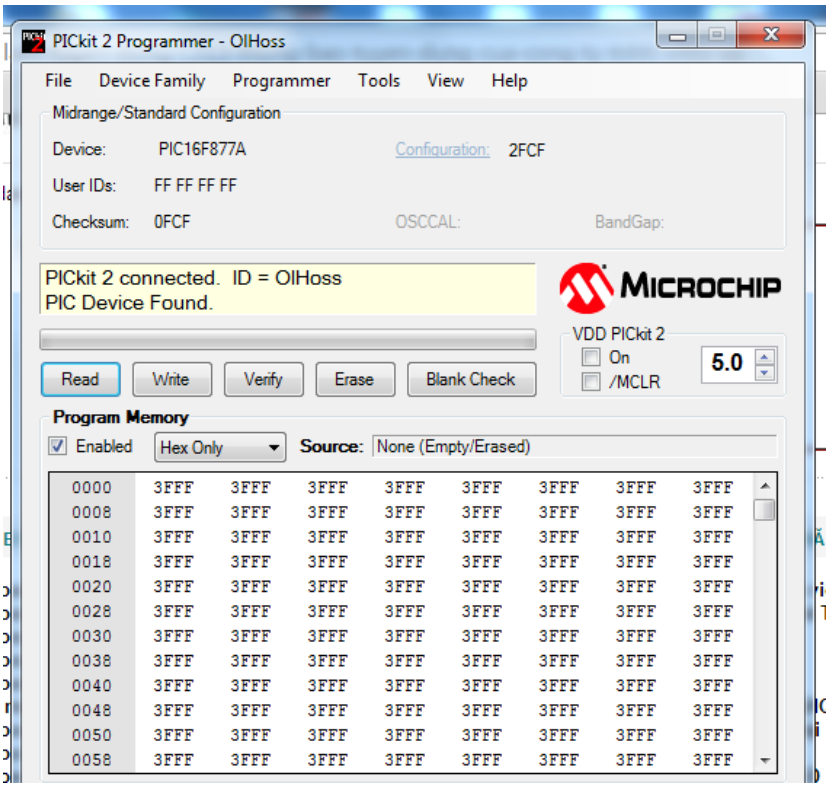
-Gạt công tắc nguồn sang vị trí bật để cấp nguồn từ USB đến KIT.

- Nếu phần mềm suất hiện lỗi

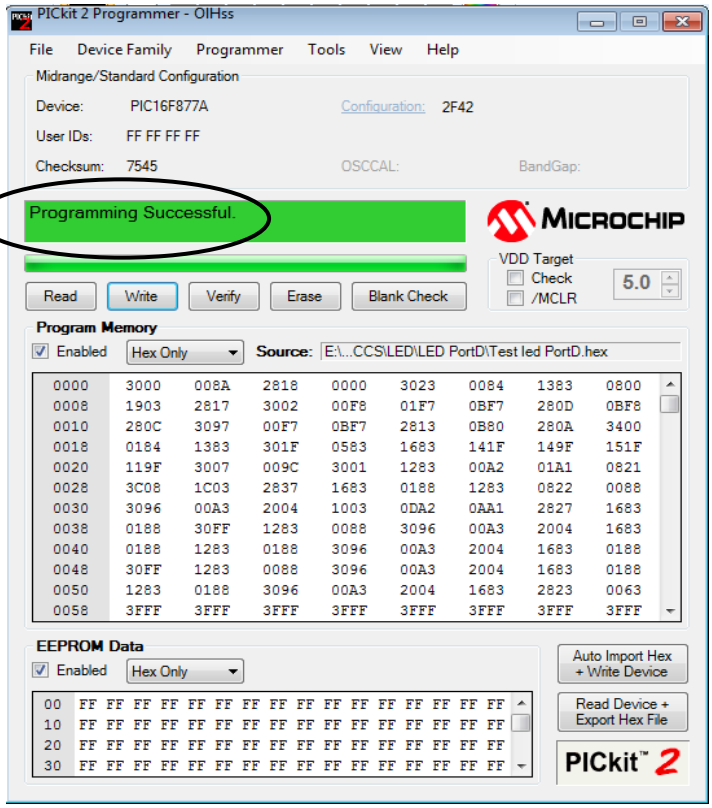


::

- Vào Tool > check Communication to retry.



- Sau khi Mạch nạp đã nhận ra vi điều khiển như hình trên, tiến hành import file HEX của chương trình bằng cách file > import.
- Sau đó click write để nạp chương trình vào vi điều khiển.
- Nếu nạp thành công phần mềm pickit 2 sẽ báo:



PHẦN 3: HƯỚNG DẪN THỰC HÀNH TRÊN MODULE VI ĐIỀU KHIỂN PIC

BÀI 1: LẬP TRÌNH PHÍM BẤM ĐƠN

A. MỤC ĐÍCH:

- Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào vi điều khiển và sử dụng mô hình thí nghiệm để kiểm chứng.
- Điều khiển thiết bị ngoại vi bằng các port của vi điều khiển.
- Điều khiển các thiết bị ngoại vi bằng phím bấm
- Trình bày kỹ thuật quét phím .
- Trình bày một số ứng dụng trong kỹ thuật điều khiển phím đơn

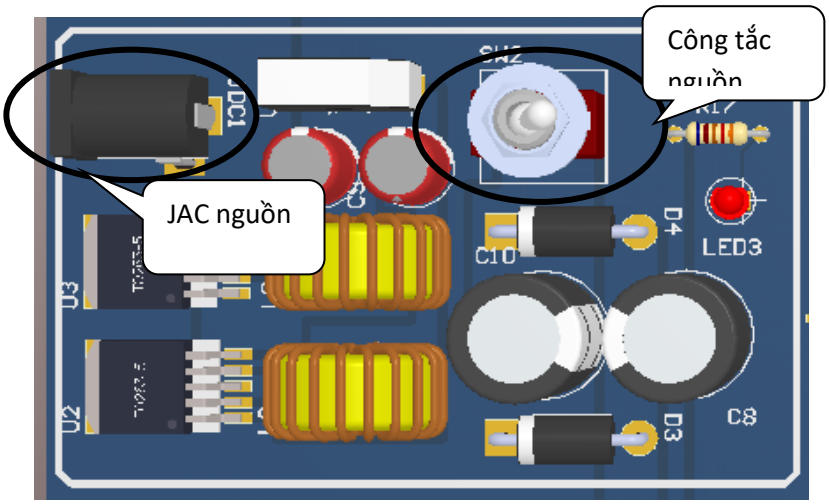
B. YÊU CẦU:

- Nắm vững tập lệnh của vi điều khiển PIC.
- Biết cách viết các chương trình điều khiển nhận tín hiệu từ phím bấm
- Nắm được sơ đồ và nguyên lý hoạt động của khối phím bấm đơn trên mô hình thí nghiệm.
- Nắm được nguyên lý và kỹ thuật quét phím.
- Biết cách viết các chương trình ứng dụng có sử dụng phím bấm đơn để điều khiển các thiết bị ngoại vi khác nhau

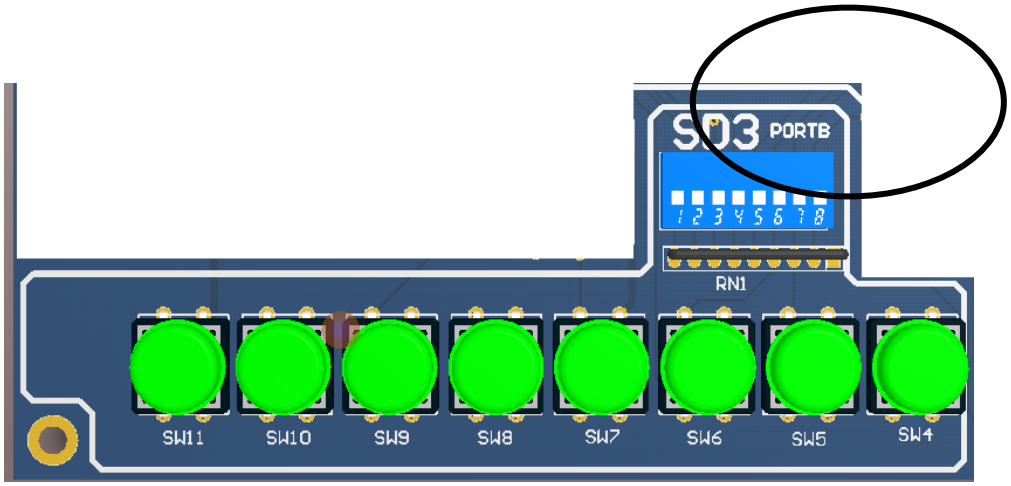
C. TRÌNH TỰ THÍ NGHIỆM

1. Kết nối thiết bị

- Cắm jac nguồn cho Module thí nghiệm PIC
- Để công tắc ở vị trí OFF

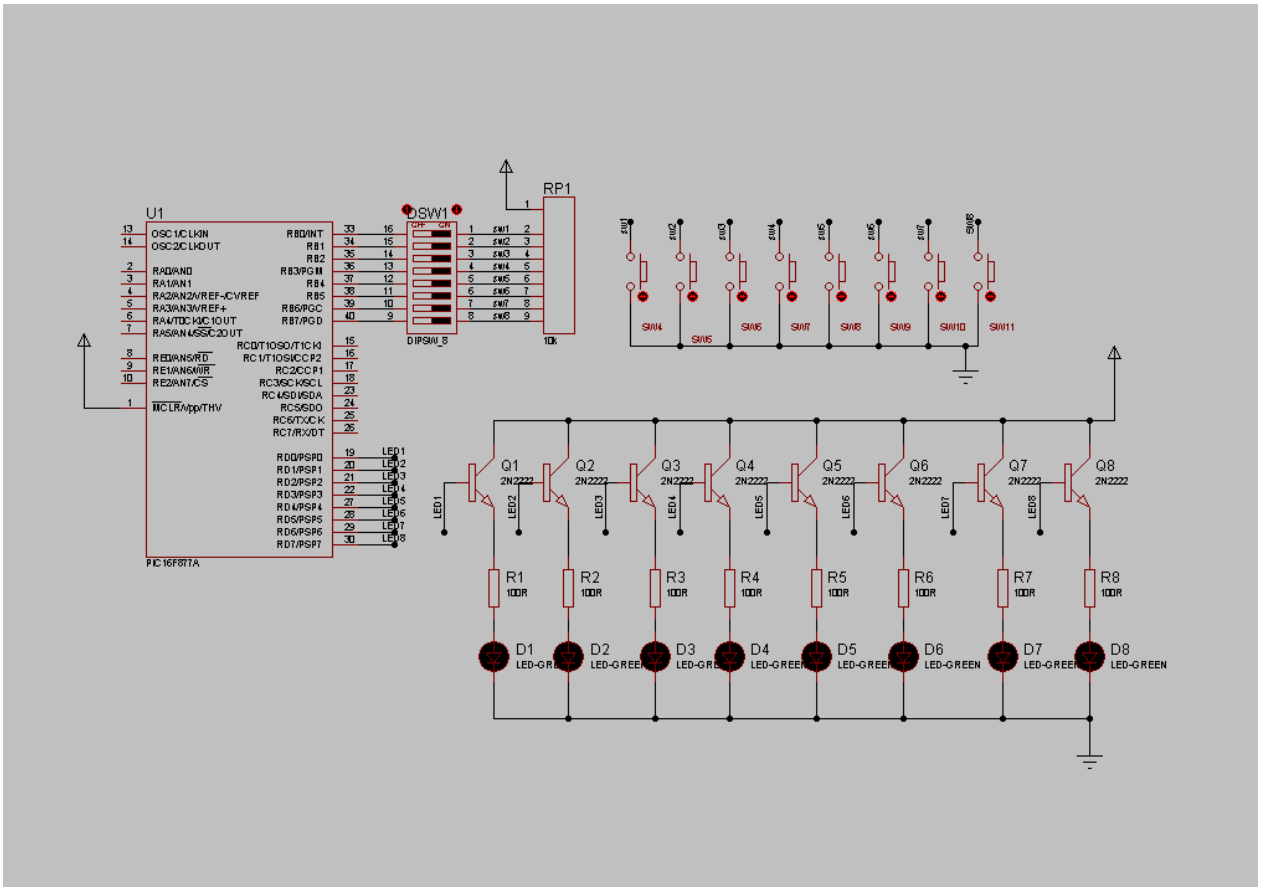


- Chuyển **SD3** về vị trí **ON** (Tương đương với việc kết nối các chân dữ liệu với PORTB).



- Chuyển **SD1, SD2, SD4, SD4, SD5** ở vị trí **OFF.**

2. Sơ đồ nguyên lý mạch thí nghiệm bàn phím đơn.



3. Ví dụ mẫu

VD 1.

Viết chương trình liên tục kiểm tra trạng thái các nút bấm từ SW4-SW9 trong sơ đồ nguyên lý ở trên. Nếu nút bấm SWx thay đổi trạng thái thì thay đổi trạng thái của đèn led Dx tương ứng. Trong đó Dx với X từ 0 đến 7 được điều khiển bởi các chân tương ứng Dx trên cổng RDx:

Code + Sơ đồ mạch thảo khảo tại: *https://github.com/huynguyen82/TNUT-PICExamples*

Chương trình mẫu:

```
#include <16f877a.h>
#include "def_16f877a.h"
/*
Có thể download file này tron link trên https://github.com/huynguyen82/TNUT-PICExamples
*/
#fuses HS,NOLVP,NOWDT,NOPROTECT
#device *=16 ADC=10
#use delay(clock=12M)
//////////////////  dia chi cong ra port
```

```
#define LED1 D0
#define LED2 D1
#define LED3 D2
#define LED4 D3
#define LED5 D4
#define LED6 D5
#define LED7 D6
#define LED8 D7
#define SW1 B0
#define SW2 B1
#define SW3 B2
#define SW4 B3
#define SW5 B4
#define SW6 B5
#define SW7 B6
#define SW8 B7

void init()
{
 TRISB=0xff;
 TRISD=0x00;
 PORTD=0x00;
}

//////////////////////
void LD1()
 {
  if(SW1==0)
    {while(SW1==0){};
     delay_ms(10);
     LED1=~LED1;
     }
     }
//////////////////
void LD2()
 {
  if(SW2==0)
    {while(SW2==0){};
     delay_ms(10);
     LED2=~LED2;}
     }
//////////////////
void LD3()
 {
  if(SW3==0)
    {while(SW3==0){};
     delay_ms(10);
     LED3=~LED3;}
     }
```

```
//////////////////
void LD4()
 {
  if(SW4==0)
    {while(SW4==0){};
     delay_ms(10);
     LED4=~LED4;}
     }
//////////////////
void LD5()
 {
  if(SW5==0)
    {while(SW5==0){};
     delay_ms(10);
     LED5=~LED5;}
     }
//////////////////
void LD6()
 {
  if(SW6==0)
    {while(SW6==0){};
     delay_ms(10);
     LED6=~LED6;}
     }
//////////////////
void LD7()
 {
  if(SW7==0)
    {while(SW7==0){};
     delay_ms(10);
     LED7=~LED7;}
     }
//////////////////
void LD8()
 {
  if(SW8==0)
    {while(SW8==0){};
     delay_ms(10);
     LED8=~LED8;}
     }
//////////////////
void main(){
init();
while(1){
LD1();
LD2();
LD3();
LD4();
LD5();
```

```
LD6();
LD7();
LD8();}}
```

4. Lưu chương trình và biên dịch chương trình.

5. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.

6. Gắn chip vi điều khiển thí nghiệm vào socket

7. Nạp chương trình vào vi điều khiển.

Bật nguồn cho mô hình thí nghiệm. Quan sát kết quả hoạt động, nếu kết quả hoạt động không đúng yêu cầu của đề bài thì phải quay lại kiểm tra việc kết nối mạch, hiệu chỉnh chương trình.

8. **Bài tập sinh viên tự làm**

Viết chương trình kiểm tra nếu SW1 được bấm thì các LED đơn liên tục sáng tuần tự từ trái qua phải, mỗi led sáng trong 100ms. Nếu các các nút khác được bấm thì tắt tất cả các LED.

BÀI 2: LẬP TRÌNH MA TRẬN BÀN PHÍM

A. MỤC ĐÍCH:

- Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào vi điều khiển và sử dụng mô hình thí nghiệm để kiểm chứng.
- Điều khiển thiết bị ngoại vi bằng các port của vi điều khiển.
- Điều khiển các thiết bị ngoại vi bằng bàn phím (bànphím thiết kế theo kiểu ma trận).
- Trình bày kỹ thuật quét phím cho dạng bàn phím ma trận 16 phím (4 hàng x 4 cột).
- Trình bày một số ứng dụng trong kỹ thuật điều khiểnbàn phím.

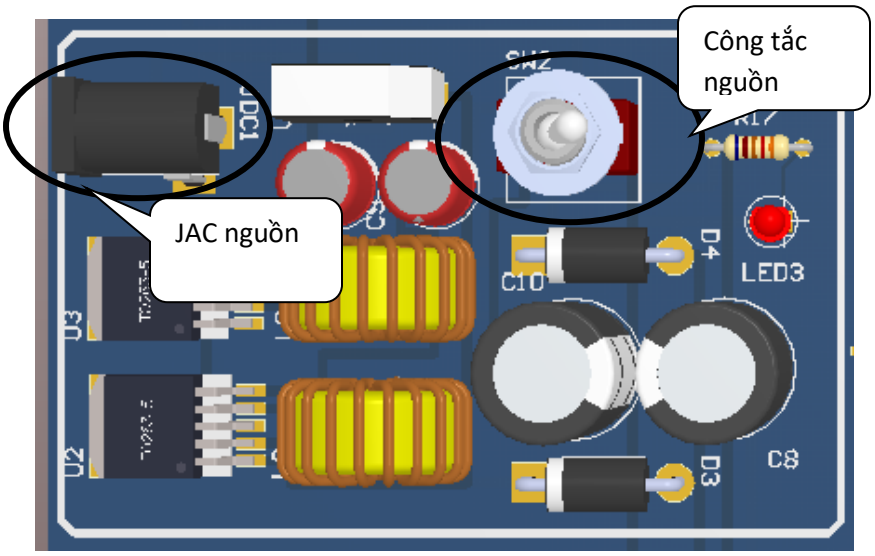
B. YÊU CẦU:

- Nắm vững tập lệnh của vi điều khiển PIC.
- Biết cách viết các chương trình điều khiển bàn phím ma trận.
- Nắm được sơ đồ và nguyên lý hoạt động của khối bàn phím ma trận trên mô hình thí nghiệm.
- Nắm được nguyên lý và kỹ thuật quét phím cho các dạng bàn phím ma trận.
- Biết cách viết các chương trình ứng dụng có sử dụngbàn phím ma trận để điều khiển các thiết bị ngoại vi khác nhau

C. TRÌNH TỰ THÍ NGHIỆM

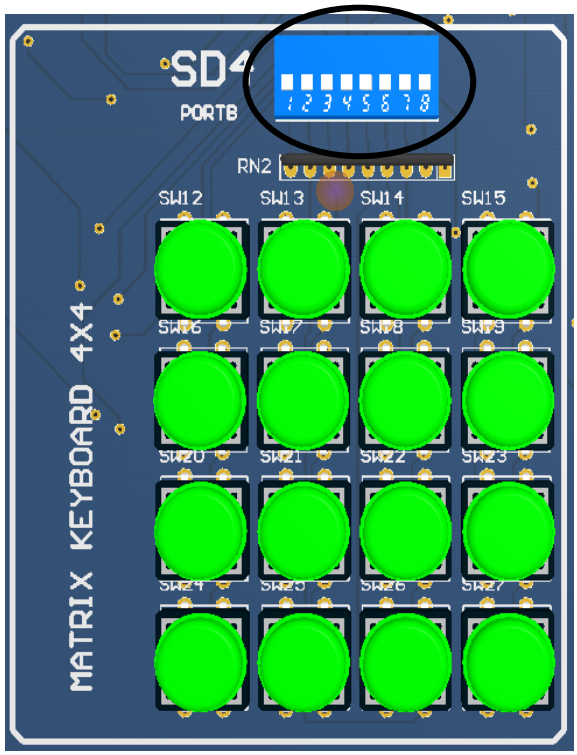
1. Kết nối thiết bị

- Cắm jac nguồn cho Module thí nghiệm PIC
- Để công tắc ở vị trí OFF



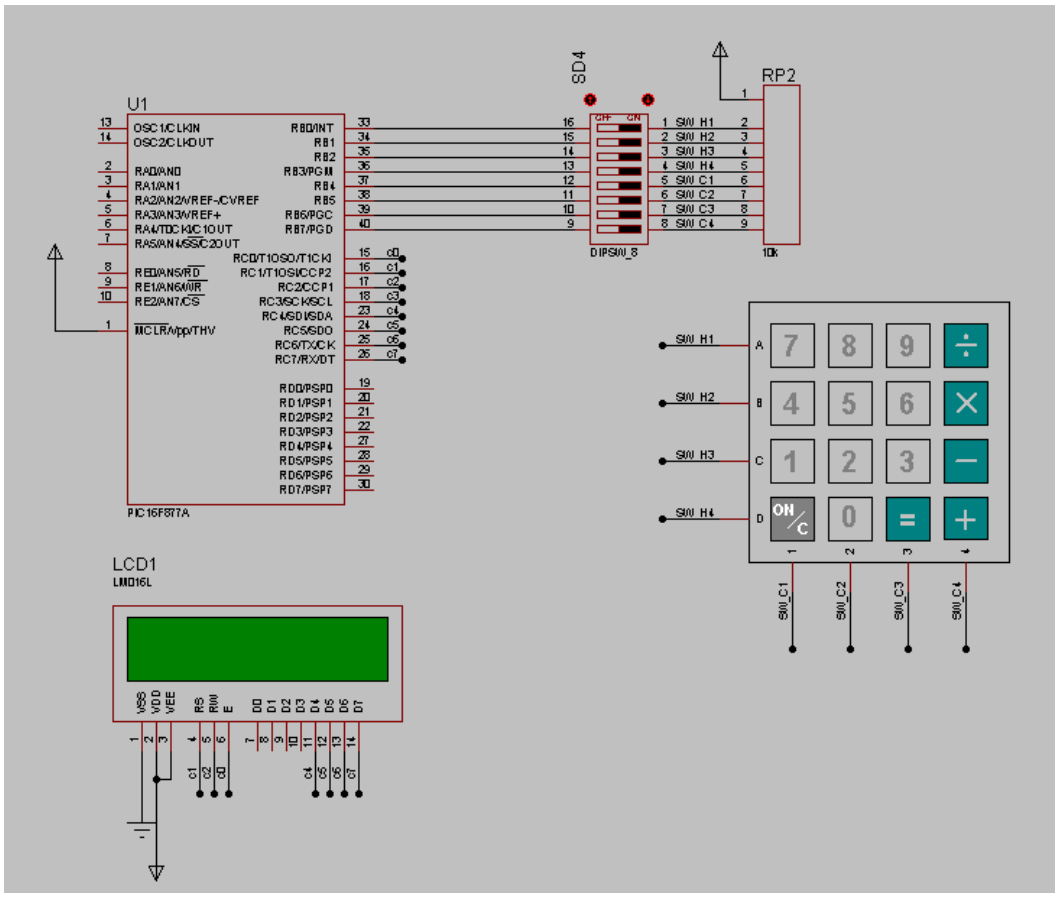
- Chuyển **SD4** về vị trí **ON** (Tương đương với việc kết nối các chân dữ liệu với PORTB).

- Chú ý khi sử dụng ma trận bàn phím, phải rút GLCD ra khỏi KIT, vì đường dữ liệu của GLCD trùng với đường dữ liệu của GLCD dẫn đến đọc phím ma trận khong chính xác.



- Chuyển **SD1, SD2, SD3, SD5** ở vị trí **OFF.**

2. Sơ đồ nguyên lý mạch thí nghiệm ma trận bàn phím



3. Ví dụ mẫu

VD1: Viết chương trình hiển thị phím được bấm trên LCD trong sơ đồ mạch trên.

Code + Sơ đồ mạch thảo khảo tại: *https://github.com/huynguyen82/TNUT-PICExamples*

Chương trình:

```
#include <16f877a.h>
#fuses HS,NOLVP,NOWDT,NOPROTECT
#device *=16 ADC=10
#use delay(clock=12M)

#define ROW1 PIN_B0
#define ROW2 PIN_B1
#define ROW3 PIN_B2
#define ROW4 PIN_B3
#define COL1 PIN_B4
#define COL2 PIN_B5
#define COL3 PIN_B6
```

```
#define COL4 PIN_B7

#define LCD_ENABLE_PIN  PIN_C0
#define LCD_RS_PIN      PIN_C1
#define LCD_RW_PIN      PIN_C2
#define LCD_DATA4       PIN_C4
#define LCD_DATA5       PIN_C5
#define LCD_DATA6       PIN_C6
#define LCD_DATA7       PIN_C7

#include <lcd.c>
#include "key_4x4.c"
/*
Có thể download file này tron link trên https://github.com/huynguyen82/TNUT-PICExamples
*/

void main()
  {
   unsigned int8 key;
   key_4x4_init();
   lcd_init();
   while(true)
       {
         key=get_key_4x4();
         if(key)
          {
             lcd_gotoxy(0,0);
             printf(lcd_putc,"Key pressed:%c",key);
          }
         delay_ms(10);
       }
}
```

4. Lưu chương trình và biên dịch chương trình.

5. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.

6. Gắn chip vi điều khiển thí nghiệm vào socket

7. Nạp chương trình vào vi điều khiển.

Bật nguồn cho mô hình thí nghiệm. Quan sát kết quả hoạt động, nếu kết quả hoạt động không đúng yêu cầu của đề bài thì phải quay lại kiểm tra việc kết nối mạch, hiệu chỉnh chương trình.

8. Bài tập sinh viên tự làm: Máy tính cầm tay

Viết chương trình cho mạch trên thực hiện các phép tính +,-,*,/ với các số tự nhiên (tập Z) khi người dùng nhập vào từ bàn phím.

VD: người dùng nhập vào 9/3 thì màn hình LCD phải hiện thị kết quả là 9/3=3

BÀI 3: LẬP TRÌNH VỚI ADC

A. *MỤC ĐÍCH:*

- Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào vi điều khiển và sử dụng mô hình thí nghiệm để kiểm chứng.
- Điều khiển thiết bị ngoại vi bằng các port của vi điều khiển.
- Thực hiện việc biến đổi từ tín hiệu tương tự (Analog) sang tín hiệu số (Digital) chế độ 8 bits và 16 bits.

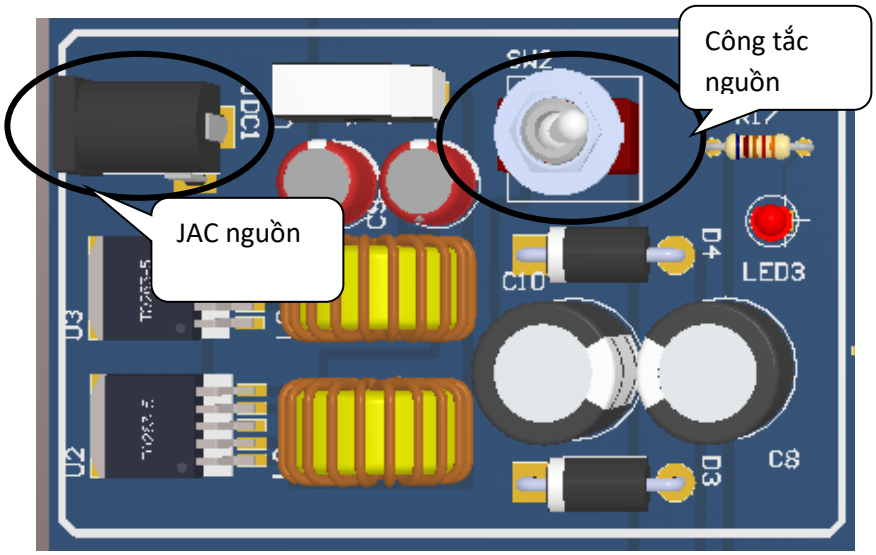
B. *YÊU CẦU:*

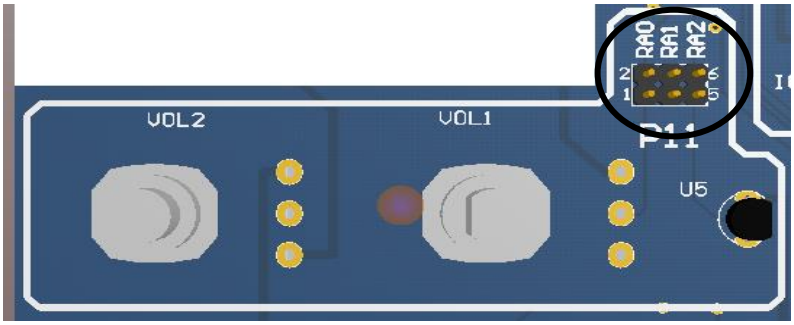
- Nắm vững tập lệnh của vi điều khiển PIC
- Tham khảo trước và nắm được hoạt động của các thanh ghi ADC của PIC

C. *TRÌNH TỰ THÍ NGHIỆM*

1. Kết nối thiết bị

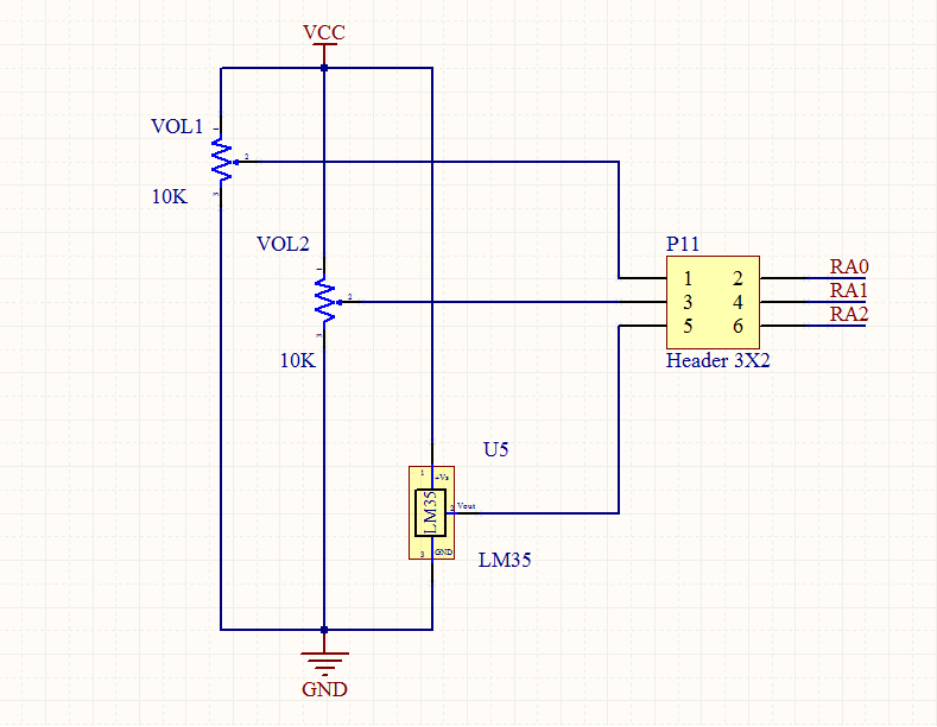
- Cắm jac nguồn cho Module thí nghiệm PIC
- Để công tắc ở vị trí OFF





- Cắm jump 1-2: nối biến trở VOL1 vào chân RA0.
- Cắm jump 3-4: nối biến trở VOL2 vào chân RA1.
- Cắm jump 5-6: nối cảm biên LM35 vào chân RA2.
- Chuyển switch SD1 về vi trí OFF.

2. Sơ đồ nguyên lý mạch ADC



3. Ví dụ mẫu:

Bài tập:

Chương trình:

```
#include <16f877A.h>
#fuses HS, NOLVP, NOWDT, NOPROTECT
#device 16f877*=16, ADC=10
#use   delay (clock=12m) //Use built-in function: delay_ms() & delay_us()
#include "def_16f877a.h"
#include "LCD_4bits.c" //use module function


float value_ADC1,value_ADC2,value_ADC3;

void init()
 {trisa=0xff;
  lcd_init();
  delay_ms(20);
  setup_adc_ports(AN0_AN1_AN2_AN3_AN4);// CHON 5 KENH ADC
VREF =5v
  setup_adc(ADC_CLOCK_DIV_32);// xung clock chia 32
 }

void main()
 {
  init();
  while(true)
   {
    set_adc_channel(0);delay_us(100);value_ADC1=read_adc();
    set_adc_channel(1);delay_us(100);value_ADC2=read_adc();
    set_adc_channel(2);delay_us(100);value_ADC3=read_adc();
    lcd_gotoxy(0,1);
    printf(lcd_putc," AD1 AD2  LM35");
    lcd_gotoxy(1,2);
```

```
printf(lcd_putc,"%4.0f|%4.0f|%4.0f",value_ADC1,value_ADC2,value_ADC3)
;
   delay_ms(250);
  }
  }
```

4. Lưu chương trình và biên dịch chương trình.
5. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.
6. Gắn chip vi điều khiển thí nghiệm vào socket
7. Nạp chương trình vào vi điều khiển.

Bật nguồn cho mô hình thí nghiệm. Quan sát kết quả hoạt động, nếu kết quả hoạt động không đúng yêu cầu của đề bài thì phải quay lại kiểm tra việc kết nối mạch, hiệu chỉnh chương trình.

BÀI 4: LẬP TRÌNH XỬ LÝ TÍN NGẮT NGOÀI

A. MỤC ĐÍCH:

- Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào vi điều khiển và sử dụng mô hình thí nghiệm để kiểm chứng.
- Điều khiển thiết bị ngoại vi bằng các port của vi điều khiển.
- Thiết kế các ứng dụng điều khiển thực tế có sử dụngngắt (Interrupt).
- So sánh ưu và nhược điểm của các chương trình điều khiển có sử dụng ngắt và không sử dụng ngắt.

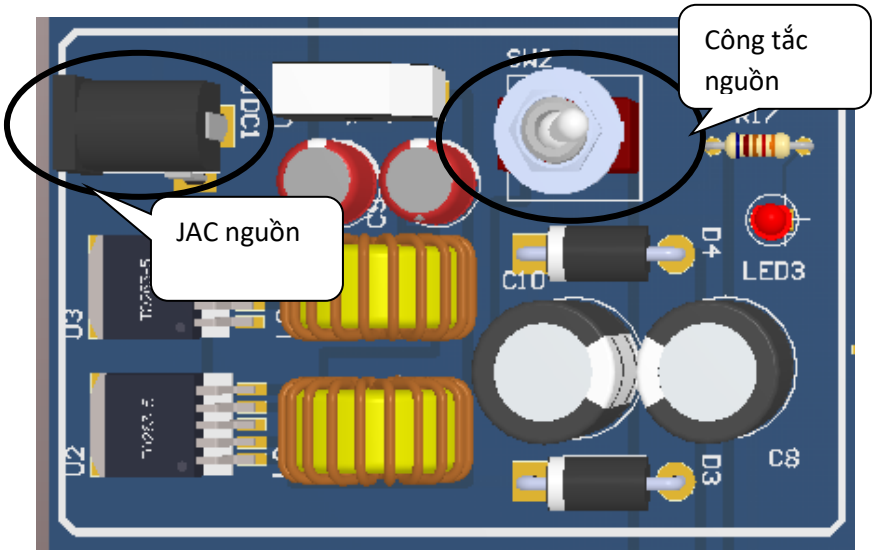
B. YÊU CẦU:

- Tham khảo trước hoạt động của ngắt (Interrupt) ở các chế độ khác nhau.
- Nắm được phương pháp lập trình và điều khiển có sử dụng các ngắt.

C. TRÌNH TỰ THÍ NGHIỆM

1. Kết nối thiết bị

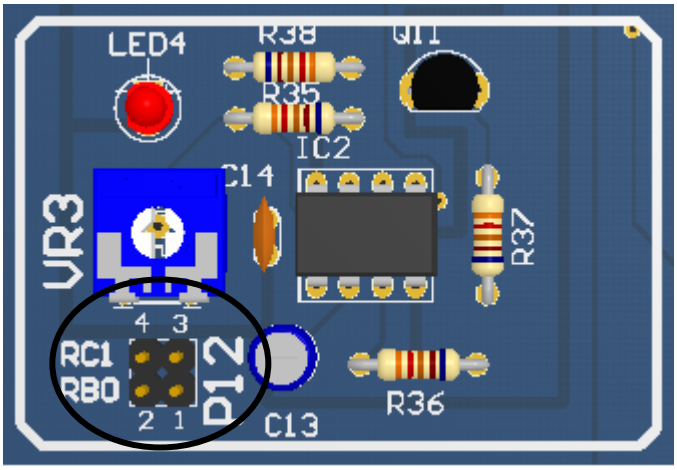
   - Cắm jac nguồn cho Module thí nghiệm PIC
   - Để công tắc ở vị trí OFF



   - Chuyển **SD1, SD2, SD3, SD4, SD5** ở vị trí **OFF.**

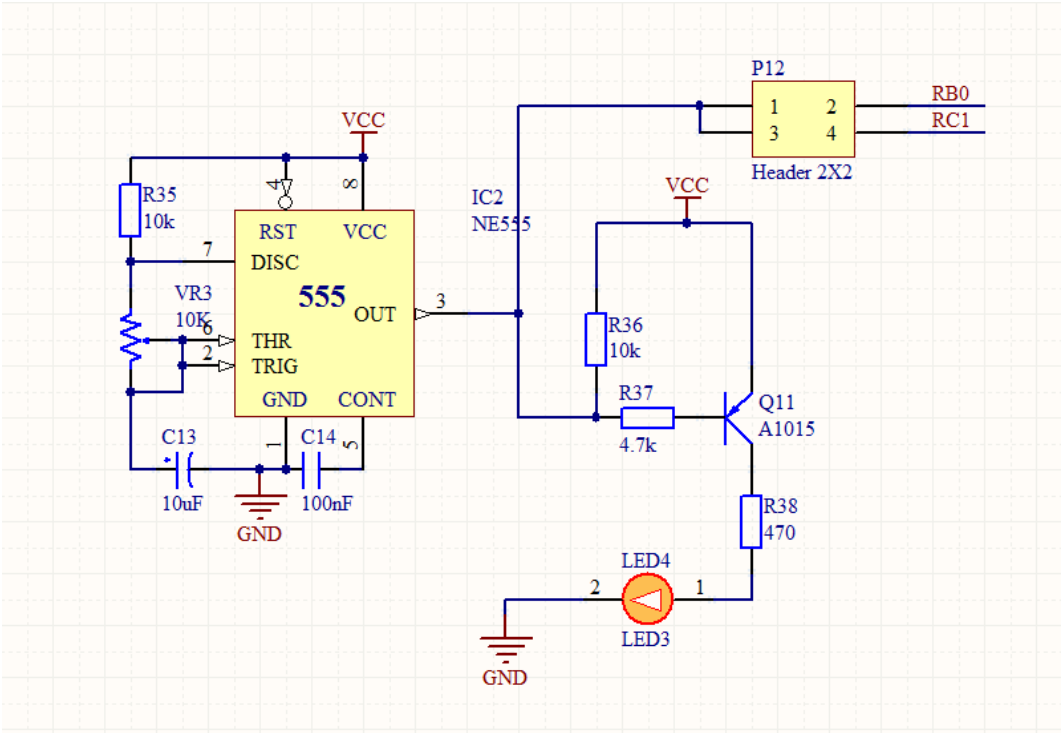
- Cắm chọn 2 Jump ở vị trí P12 để kết nối tín hiệu xung vuông từ bộ tạo xung 555 với chân RC1 và RB0 làm tín hiệu ngắt ngoài và ngắt timer.
- Hiển thị giá trị đếm lên LED 7 thanh, LCD, GLCD tham khảo các

Bài2, Bài 6, Bài 7.



- Biến trở VR3 để điều chỉnh tần số xung vuông.

2. Sơ đồ nguyên lý mạch tạo xung vuông ngắt ngoài cho vi điều khiển



3. Ví dụ mẫu:

Bài tập:

Chương trình:

```
#include <16f877a.h>
#include "def_16f877a.h"
#fuses HS,NOWDT,NOLVP,NOPROTECT
#device 16f877a*=16 ADC=8
#use delay(clock=12000000)
long int    pulse;
int8 i,N1,N2,N3,N4,N5,N6;
long dulieu;
const int8
led7seg[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
const int8 time=50;

/*************************************/
#INT_EXT
void pulse_counter()
   {
    clear_interrupt(int_ext); // clear flag
    pulse++; //nhan so xung vao
   }

/*************************************/
void init()
  {
  trisb=0b00000001;
  trisa=0x00;
  trisd=0x00;
  pulse=0;
  enable_interrupts(INT_EXT); //khoi tao ngat ngoai
  EXT_INT_EDGE(H_TO_L);       // ngat suon xuong
  enable_interrupts(GLOBAL);

  }
/*************************************/
void hienthi(int16 k)
 {
```

```
  N1=(k/1)%10;
  N2=(k/10)%10;
  N3=(k/100)%10;
  N4=(k/1000)%10;
  N5=(k/10000)%10;
  N6=(k/100000)%10;
  for(i=0;i<20;i++)
   {
    portd=(led7seg[N6]); a0=0; delay_us(time);a0=1;
    portd=(led7seg[N5]); a1=0; delay_us(time);a1=1;
    portd=(led7seg[N4]); a2=0; delay_us(time);a2=1;
    portd=(led7seg[N3]); a3=0; delay_us(time);a3=1;
    portd=(led7seg[N2]); a4=0; delay_us(time);a4=1;
    portd=(led7seg[N1]); a5=0; delay_us(time);a5=1;
    }
    portd=0xff;delay_ms(1);
   }
/*************************************/
void main()
  {
   init();
   while(true)
     {
      hienthi(pulse);
     }
   }

/*************************************/
```

4. Lưu chương trình và biên dịch chương trình.

5. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.

6. Gắn chip vi điều khiển thí nghiệm vào socket

7. Nạp chương trình vào vi điều khiển.

Bật nguồn cho mô hình thí nghiệm. Quan sát kết quả hoạt động, nếu kết quả hoạt động không đúng yêu cầu của đề bài thì phải quay lại kiểm tra việc kết nối mạch, hiệu chỉnh chương trình.

BÀI 5: LẬP TRÌNH TRUYỀN THÔNG RS232

A. MỤC ĐÍCH:

- Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào vi điều khiển và sử dụng mô hình thí nghiệm để kiểm chứng.
- Điều khiển thiết bị ngoại vi bằng các port của vi điều khiển.
- Ứng dụng port nối tiếp của vi điều khiển để mở rộng port nhập và xuất.
- Điều khiển việc thu phát nối tiếp

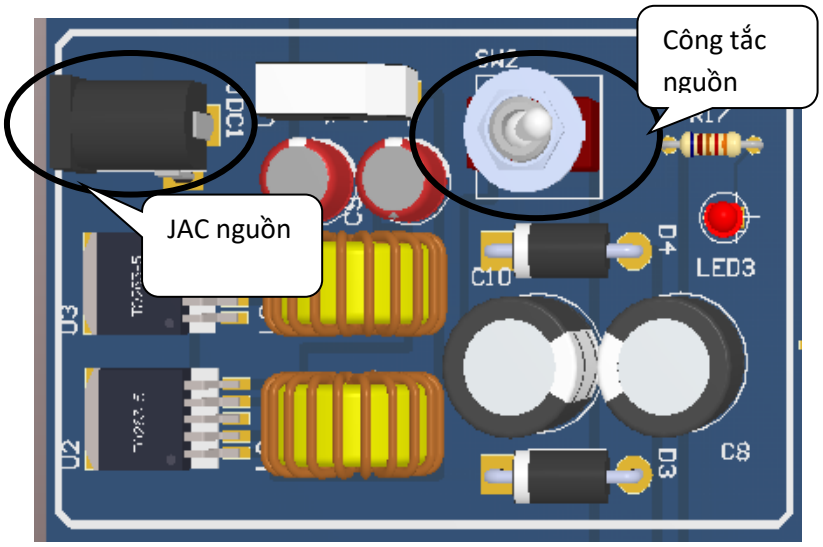
B. YÊU CẦU:

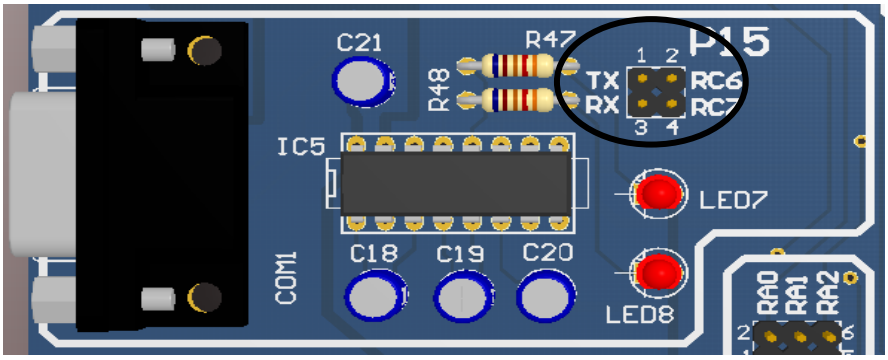
- Nắm vững tập lệnh của vi điều khiển PIC.
- Biết cách hoạt động của port nối tiếp ở các chế độ khác nhau.
- Biết cách lập trình điều khiển việc xuất nhập dữ liệu thông qua port nối tiếp ở các chế độ khác nhau.

C. TRÌNH TỰ THÍ NGHIỆM

1. Kết nối thiết bị

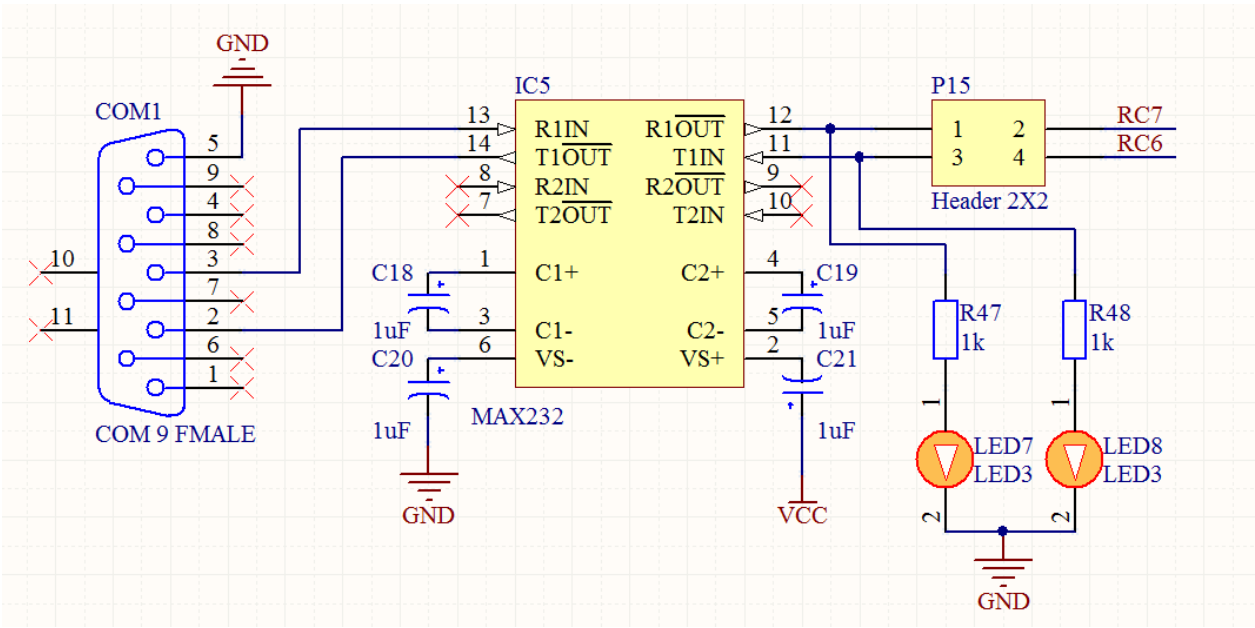
   - Cắm jac nguồn cho Module thí nghiệm PIC.
   - Để công tắc ở vị trí OFF.





Cắm chọn 2 Jam ở vị trí P15 để kết nối tín hiệu TX và RX với 2 chân RC6 và RC7 của vi điều khiển.

2. Sơ đồ nguyên lý mạch truyền thông RS232



3. Ví dụ mẫu:

Bài tập:

Chương trình:

```
#include <16f877a.h>
#fuses HS,NOWDT,NOLVP,NOPROTECT
#device 16F877*=16 ADC=8
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

void main()
  {
    while(true)
     {
        printf("  TNUT  ");
        delay_ms(20);
     }
   }
```

4. Lưu chương trình và biên dịch chương trình.
5. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.
6. Gắn chip vi điều khiển thí nghiệm vào socket
7. Nạp chương trình vào vi điều khiển.

Bật nguồn cho mô hình thí nghiệm. Quan sát kết quả hoạt động, nếu kết quả hoạt động không đúng yêu cầu của đề bài thì phải quay lại kiểm tra việc kết nối mạch, hiệu chỉnh chương trình.

PHỤ LỤC

1. Def_16f877a.h

```
// register definitions
#define W 0
#define F 1

// register files
#byte INDF =0x00
#byte TMR0 =0x01
#byte PCL =0x02
#byte STATUS =0x03
#byte FSR =0x04
//#device *=14 adc=8
#byte PORTA =0x05
#byte PORTB =0x06
#byte PORTC =0x07
#byte PORTD =0x08
#byte PORTE =0x09

#byte EEDATA =0x10C
#byte EEADR =0x10D
#byte EEDATH =0x10E
#byte EEADRH =0x10F
#byte ADCON0 =0x1F
#byte ADCON1 =0x9F
#byte ADRESH =0x9F
#byte ADSESL =0x9F

#byte PCLATH =0x0a
#byte INTCON =0x0b
#byte PIR1 =0x0c
#byte PIR2 =0x0d
#byte PIE1 =0x8c
#byte PIE2 =0x8d
#byte OPTION_REG =0x81
#byte TRISA =0x85
#byte TRISB =0x86
#byte TRISC =0x87
#byte TRISD =0x88
#byte TRISE =0x89

#byte EECON1 =0x18C
#byte EECON2 =0x18D
```

```
#byte SSPBUF =0x13
#byte SSPCON =0x14
#byte SSPCON2 =0x91
#byte SSPADD =0x93
#byte SSPSTAT =0x94
// SSPCON bit
#bit SSPWCOL = 0x14.7
#bit SSPOV = 0x14.6
#bit SSPEN = 0x14.5
#bit SSPCKP = 0x14.4
#bit SSPM3 = 0x14.3
#bit SSPM2 = 0x14.2
#bit SSPM1 = 0x14.1
#bit SSPM0 = 0x14.0
// SSPSTAT bit
#bit SSPSMP = 0x94.7
#bit SSPCKE = 0x94.6
#bit SSPDA = 0x94.5
#bit SSPP = 0x94.4
#bit SSPS = 0x94.3
#bit SSPRW = 0x94.2
#bit SSPUA = 0x94.1
#bit SSPBF = 0x94.0

// BIT DEFINITION
#bit A7 =0x05.5
#bit A6 =0x05.5
#bit A5 =0x05.5
#bit A4 =0x05.4
#bit A3 =0x05.3
#bit A2 =0x05.2
#bit A1 =0x05.1
#bit A0 =0x05.0

#bit B7 =0x06.7
#bit B6 =0x06.6
#bit B5 =0x06.5
#bit B4 =0x06.4
#bit B3 =0x06.3
#bit B2 =0x06.2
#bit B1 =0x06.1
#bit B0 =0x06.0

#bit C7 =0x07.7
```

```
#bit C6 =0x07.6
#bit C5 =0x07.5
#bit C4 =0x07.4
#bit C3 =0x07.3
#bit C2 =0x07.2
#bit C1 =0x07.1
#bit C0 =0x07.0

#bit D7 =0x08.7
#bit D6 =0x08.6
#bit D5 =0x08.5
#bit D4 =0x08.4
#bit D3 =0x08.3
#bit D2 =0x08.2
#bit D1 =0x08.1
#bit D0 =0x08.0
#bit E2 =0x09.2
#bit E1 =0x09.1
#bit E0 =0x09.0


#bit trisa5 =0x85.5
#bit trisa4 =0x85.4
#bit trisa3 =0x85.3
#bit trisa2 =0x85.2
#bit trisa1 =0x85.1
#bit trisa0 =0x85.0

#bit trisb7 =0x86.7
#bit trisb6 =0x86.6
#bit trisb5 =0x86.5
#bit trisb4 =0x86.4
#bit trisb3 =0x86.3
#bit trisb2 =0x86.2
#bit trisb1 =0x86.1
#bit trisb0 =0x86.0
#bit trisc7 =0x87.7
#bit trisc6 =0x87.6
#bit trisc5 =0x87.5
#bit trisc4 =0x87.4
#bit trisc3 =0x87.3
#bit trisc2 =0x87.2
#bit trisc1 =0x87.1
#bit trisc0 =0x87.0
```

```
#bit trisd7 =0x88.7
#bit trisd6 =0x88.6
#bit trisd5 =0x88.5
#bit trisd4 =0x88.4
#bit trisd3 =0x88.3
#bit trisd2 =0x88.2
#bit trisd1 =0x88.1
#bit trisd0 =0x88.0

#bit trise2 =0x89.2
#bit trise1 =0x89.1
#bit trise0 =0x89.0

// INTCON Bits for C
#bit gie = 0x0b.7
#bit peie = 0x0b.6
#bit tmr0ie = 0x0b.5
#bit int0ie = 0x0b.4
#bit rbie = 0x0b.3
#bit tmr0if = 0x0b.2
#bit int0if = 0x0b.1
#bit rbif = 0x0b.0

// PIR1 for C
#bit pspif = 0x0c.7
#bit adif = 0x0c.6
#bit rcif = 0x0c.5
#bit txif = 0x0c.4
#bit sspif = 0x0c.3
#bit ccp1if = 0x0c.2
#bit tmr2if = 0x0c.1
#bit tmr1if = 0x0c.0
//PIR2 for C
#bit cmif = 0x0d.6
#bit eeif = 0x0d.4
#bit bclif = 0x0d.3
#bit ccp2if = 0x0d.0
// PIE1 for C
#bit adie = 0x8c.6
#bit rcie = 0x8c.5
#bit txie = 0x8c.4
#bit sspie = 0x8c.3
#bit ccp1ie = 0x8c.2
#bit tmr2ie = 0x8c.1
#bit tmr1ie = 0x8c.0
```

```
//PIE2 for C
#bit osfie = 0x8d.7
#bit cmie = 0x8d.6
#bit eeie = 0x8d.4

// OPTION Bits
#bit not_rbpu = 0x81.7
#bit intedg = 0x81.6
#bit t0cs = 0x81.5
#bit t0se = 0x81.4
#bit psa = 0x81.3
#bit ps2 = 0x81.2
#bit ps1 = 0x81.1
#bit ps0 = 0x81.0
// EECON1 Bits
#bit eepgd = 0x18c.7
#bit free = 0x18C.4
#bit wrerr = 0x18C.3
#bit wren = 0x18C.2
#bit wr = 0x18C.1
#bit rd = 0x18C.0

//ADCON0
#bit CHS0 =0x1F.3
#bit CHS1 =0x1F.4
#bit CHS2 =0x1F.5
```