CCS C for PIC16F877A

Thang8831
http://www.picvietnam.com

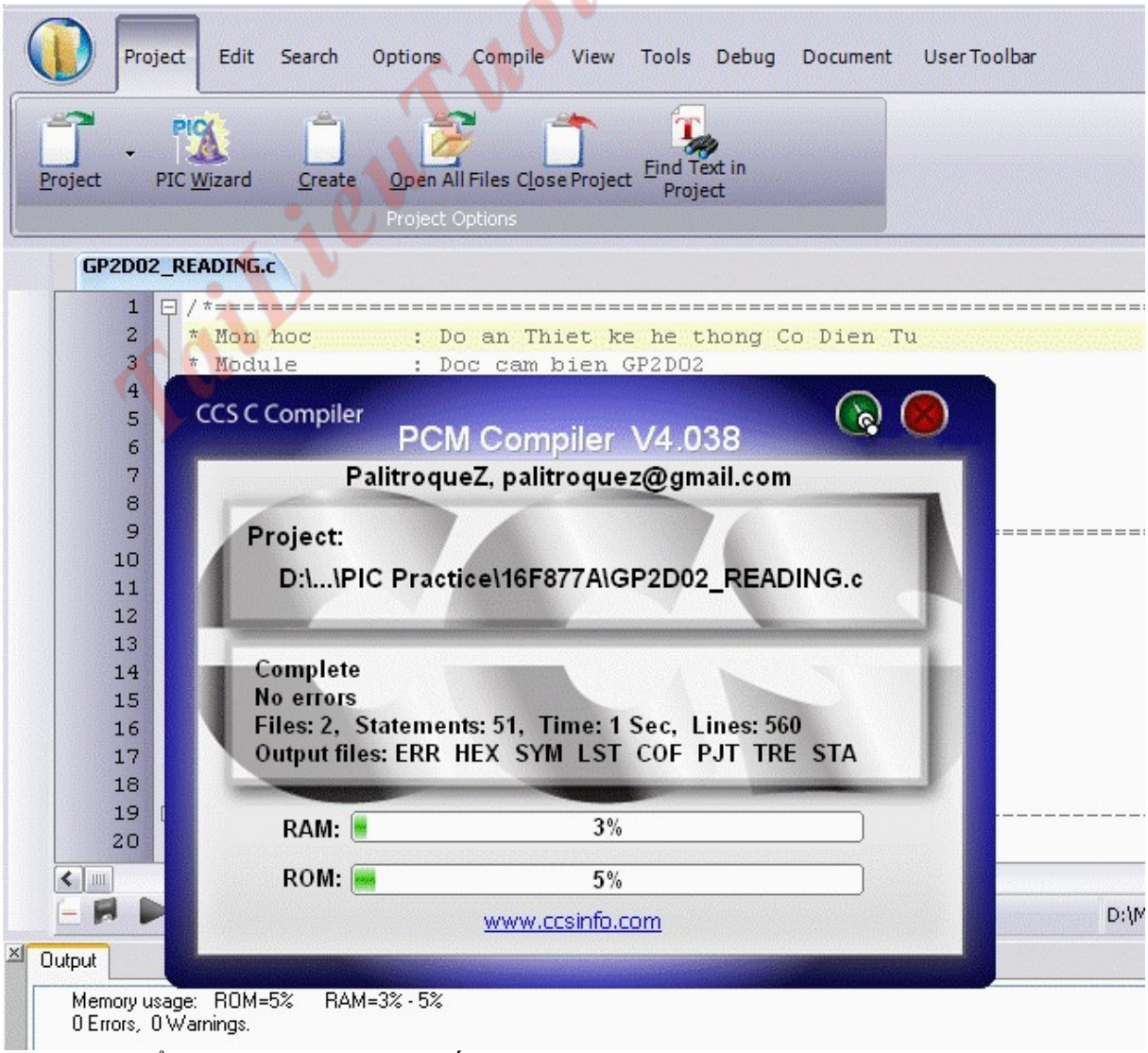
Mục lục

Mở Đầu

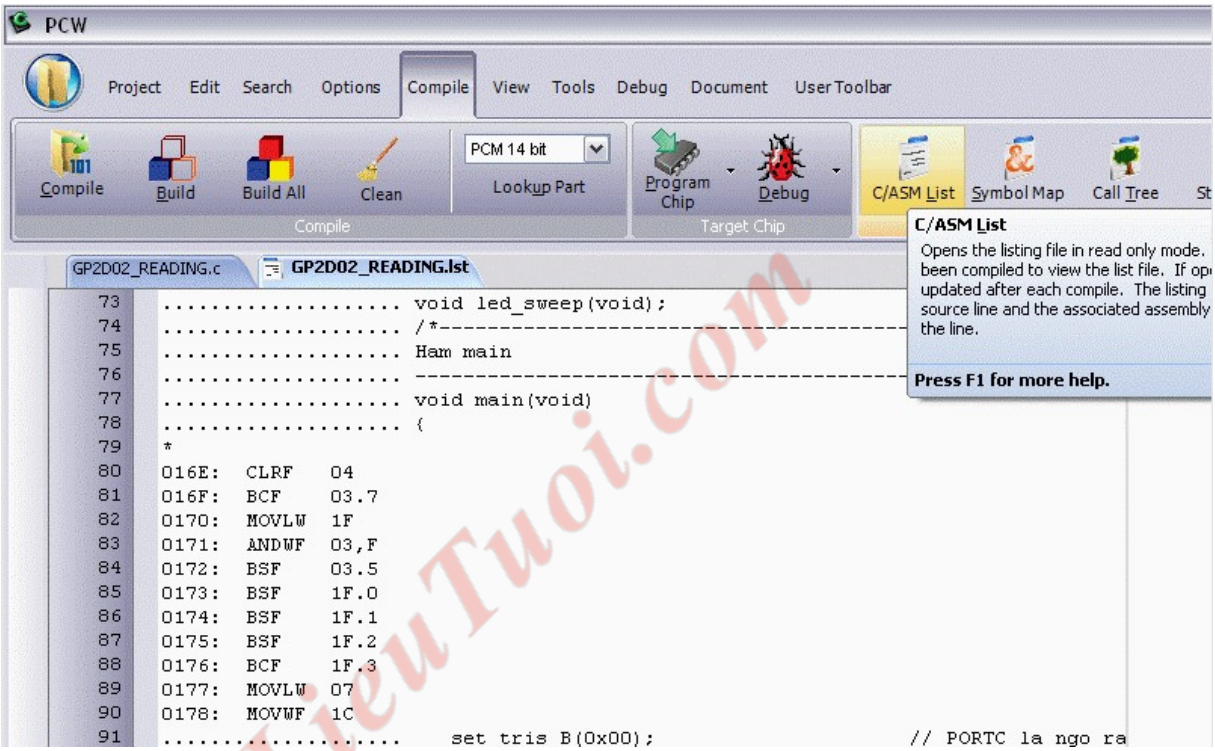
Để lập trình cho PIC, mọi người có thể chọn cho mình những ngôn ngữ lập trình khác nhau như ASM, CCS C, HT-PIC, pascal, basic,...
Với nhh, đầu tiên tìm hiểu và viết chương trình cơ bản bằng ASM để hiểu rõ cấu trúc sau đó thì viết bằng CCS C cũng viết lại những cái cơ bản và đi dần lên, tốc độ lúc này nhanh hơn khi viết bằng ASM rất nhiều.

Khi viết bằng CCS C thông thường thì dịch ra file.hex có dài hơn so với khi viết bằng ASM. Hai ngôn ngữ CCS C và HT-PIC được ưa chuộng hơn cả, CCS C dễ học,gần gũi với ASM còn HT-PIC là dạng ANSI C.

Để lập trình và biên dịch CCS C, dùng chương trình PIC C Complier,sau khi soạn thảo các bạn ấn F9, để dịch,nếu thành công sẽ có thông báo như sau:



Ngoài ra, để xem code ASM như thế nào,sau khi dịch bạn chọn mục C/ASM List như hình dưới đây:



Link download trình biên dịch CCS C ở đây:
http://www.tailieuvietnam.net/downlo...CCSC_3.249.rar
Phiên bản mới hơn tải tại: www.kho.tailieuvietnam.net, vào Download Home > Điện tử tàn thư > Vi điều khiển - Vi xử lý – PLC

Sử dụng CCS cho việc lập trình PIC là rất hay và tiện lợi.Truoc day ban noisepic có đề cập cách lập trình cho ccs khá hay. Ta sẽ khai báo thêm một file định nghĩa các thanh ghi của pic kiểu như :

// 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
#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
```

Thang8831
http://www.picvietnam.com

```
#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

//DINH NGHIA BIT
#bit RA5       =0x05.5
#bit RA4       =0x05.4
#bit RA3       =0x05.3
#bit RA2       =0x05.2
#bit RA1       =0x05.1
#bit RA0       =0x05.0

#bit RB7       =0x06.7
#bit RB6       =0x06.6
#bit RB5       =0x06.5
#bit RB4       =0x06.4
#bit RB3       =0x06.3
#bit RB2       =0x06.2
#bit RB1       =0x06.1
#bit RB0       =0x06.0

#bit RC7       =0x07.7
#bit RC6       =0x07.6
#bit RC5       =0x07.5
#bit RC4       =0x07.4
#bit RC3       =0x07.3
#bit RC2       =0x07.2
#bit RC1       =0x07.1
#bit RC0       =0x07.0

#bit RD7       =0x08.7
#bit RD6       =0x08.6
#bit RD5       =0x08.5
```

```
#bit RD4        =0x08.4
#bit RD3        =0x08.3
#bit RD2        =0x08.2
#bit RD1        =0x08.1
#bit RD0        =0x08.0

#bit RE2        =0x09.2
#bit RE1        =0x09.1
#bit RE0        =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
```

Sau đó ta có thể sử dụng lệnh gán PortB = 0x00 để xuất sẽ tiện hơn nhiều. Mình lập trình cho CCS đều dùng kiểu này. Khi đó ta sẽ vừa tận dụng được các hàm có sẵn của CCS vừa thao tác trực tiếp các thanh ghi như bên ASM.

I. 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 để 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 được 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.

1.3. 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 nhiệt 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.

Các bác ơi cho em hỏi, vậy em muốn nhúng một đoạn ASM vào trong 1 function của CCS thì em phải nhúng như thế nào ạ?

Dùng các directive #ASM và #ENDASM để bọc đoạn code đó. Đọc thêm hướng dẫn về hai directive này trong tài liệu hướng dẫn của CCS, ở đó đã có ví dụ.

Em thật sự không hiểu câu này:" (nếu dùng hai thì chèn dấu "|"ở giữa) " anh NHH có thể minh họa cho em được không? em mới tìm hiểu về Pic được một tuần, nhưng chắc chắn là anh chỉ rõ hơn thì em sẽ hiểu! cảm ơn anh và chúc anh vui!

Chọn ví dụ như vầy nè : Ví dụ chọn Timer0, chia prescaler 1:2

Code:

```
setup_timer_0(CC_INTERNAL|RTCC_DIV_2);
```

II.Chúng ta cùng nhau tìm hiểu lần lượt các phần sau:

1.I/O + Delay
2.Timer và ngắt Timer
3.Ngắt ngoài
4.ADC, PWM,... (tập trung mổ xẻ nhiều)
5.....
Tạm thời cứ như vậy đã, sau này sẽ tính tiếp!

1. I/O_Delay

1.3. Input_output

```
//================================================ =======
// Ten chuong trinh     : Thuc hien vao ra
// Nguoi thuc hien      : linhnc308
// Ngay thuc hien         : 1/09/2006
// Phien ban     : 1.0
```

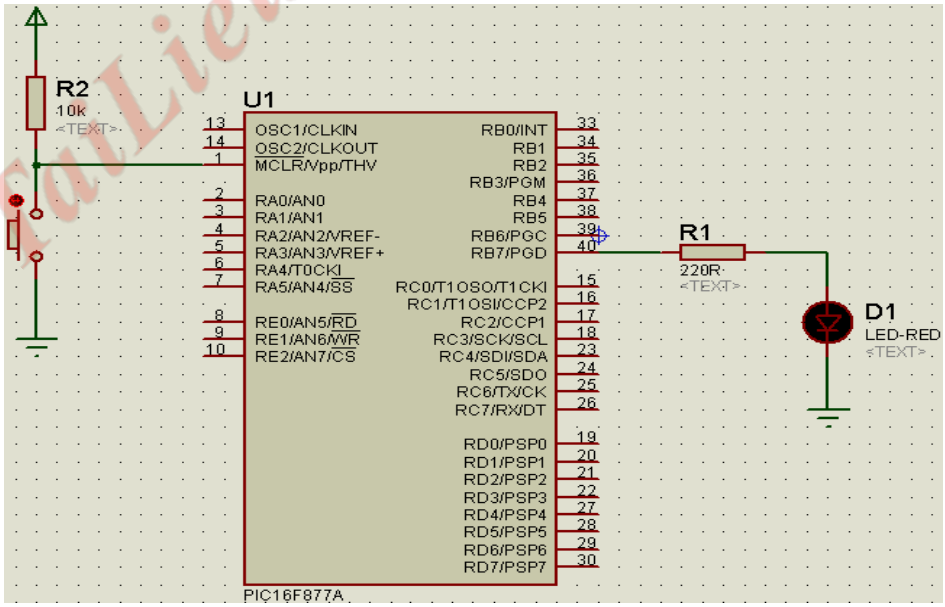
Thang8831
http://www.picvietnam.com

```
// Mo ta phan cung      : Dung PIC16F877A - thach anh 20MHz
//================================================ =======
#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)
#CASE
// Dinh nghia ten cac cong ra
#define Relay1  RD0
#define Relay2  RD1
#define Relay3  RD2
#define Relay4  RD3
#define Relay5  RD4
#define Relay6  RD5
#define Relay7  RD6
#define Relay8  RD7
#define Relay9  RC4
#define Relay10 RC5
#define Relay11 RC6
#define Relay12 RC7

#define In1    RA0
#define In2    RA1
#define In3    RA2
#define In4    RA3

#define AllRelay1 PORTD    // PIN D0 : D7
#define AllRelay2 PORTC    // PIN C4 : C7
#define Step      PORTB
#Define AllInput  PORTA

#define OFF 0
#define ON  1

#define OutEnable1 TRISD   // Relay Output
#define OutEnable2 TRISC   // Relay Output
#define InEnable   TRISA   // Input
#define StepEnable TRISB   // Step Motor
#define PWM_Enable TRISC2  // PWM, PIN_C2
void main()
{
int16 DutyCycle;

  delay_ms(250);
// Khoi tao che do vao ra
  OutEnable1 = 0x00;
  OutEnable2 = 0x0F;
  InEnable   = 0x0F;
```

```
  StepEnable = 0;
  PWM_Enable = 1;   // Khong cho phep xuat PWM
//=====================================
// Khoi tao cho bo PWM
  setup_ccp1(CCP_PWM);   //  CCP1_PINC2 as a PWM
       //   CycleTime = (1/clock)*4*t2div*(period+1)
            //   Clock=20000000 and period=127 (below)
       //   Tinh toan tan so PWM phat ra:
                 //    (1/10000000)*4*1*128 =  51.2 us or 19.5 khz
                 //    (1/20000000)*4*2*128 =  51.2 us or 19.5 khz

                 //    (1/10000000)*4*4*128 = 204.8 us or 4.9 khz
                 //    (1/10000000)*4*16*128= 819.2 us or 1.2 khz
 //setup_timer_2(T2_DIV_BY_1, 31, 1);    // 78.12KHz
 //setup_timer_2(T2_DIV_BY_1, 255, 1);     // 19.53KHz duty = 0..1023
 setup_timer_2(T2_DIV_BY_4, 255, 1);   // 4.5KHz
 //setup_timer_2(T2_DIV_BY_16, 127, 1);  // 1.2KHz
 //set_pwm1_duty(value);         // This sets the time the pulse is
//=====================================
// Test Mode
 OutEnable2 = 0x0F;
 delay_ms(10);
 Step = 0x00;                  // Motor Stop
 AllRelay1 = 0x00;
 AllRelay2 = 0x00;  // Mo toan bo cac Role
 DutyCycle = 1023;
 set_pwm1_duty(DutyCycle);          // Chay bo PWM
 PWM_Enable = 1;

while (TRUE) {
 AllRelay1 = 0x00;
 Relay12 = ON;  // Mo toan bo cac Role
 delay_ms(1000);
 AllRelay1 = 0xFF;
 Relay12 = OFF;  // Dong toan bo cac Role
 delay_ms(750);

  if (DutyCycle == 1024) DutyCycle = 1024;
  if (In1 == 0)
  {
    DutyCycle += 64;
    set_pwm1_duty(DutyCycle);          // Chay bo PWM
  }
  if (DutyCycle == 0) DutyCycle = 0;
  if (In2 == 0)
  {
    DutyCycle -= 64;
    set_pwm1_duty(DutyCycle);          // Chay bo PWM
  }
  delay_ms(250);
```

```
}
}

/*
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
#use i2c(Master,Fast,sda=PIN_C4,scl=PIN_C3)

#int_xxx       // Khai bao chuong trinh ngat
xxx_isr()
{
// Code here
}

void Ten_chuong_trinh_con(Bien)
{
// Code here
}
*/
```

1.3. Nháy LED PortB7



CODE:

```
#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)

//#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
//#use i2c(Master,Fast,sda=PIN_C4,scl=PIN_C3)

//#int_xxx       // Khai bao chuong trinh ngat
//xxx_isr()
//{
// Code here
```

```
//}
 main()
{
//thiet lap che do cho portb
trisb=0x00;
portb=0xff;
while (true)
{
 portb=0;
 delay_ms(500);
 portb=0x80;
 delay_ms(500);
}
}
```

1.4. Nháy Led nhiều chế độ

```
#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(L_TO_H);
   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() {
  temp = 0xFF;
  for (i=0;i<=8;i++) {
     portd = temp;
     delay_ms(250);
     temp >>= 1;
  }
}
void program3() {
   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);
}
```

1.5. Điều khiển led sáng dồn

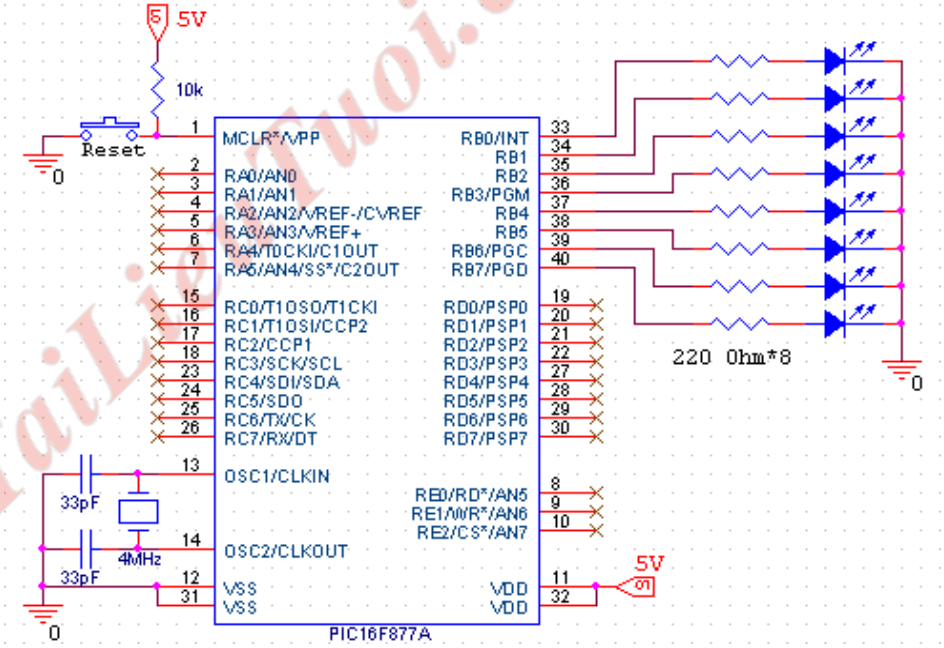
```
//Chuong trinh led sang don o PORTB
/*ket qua o PORTB
là:00000001,00000010,00000100,00001000,00010000,00100000,01000000,10000000,
10000001,10000010,10000100,10001000,.....cuoi cung thi PORTB=0xFF */
#include<16F877A.h>
#include<def_16f877a.h>
#fuses NOWDT,PUT,HS,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use fast_io(b)
int8 sck,bienxoay;
main()
   {
      trisb=0;
    while(true){
      sck=8;
      portb=0;
      delay_ms(100);
      bienxoay=1;
      while(sck>0)
         {
               portb=bienxoay;
               bienxoay=bienxoay<<1;
               delay_ms(100);
               sck--;
         }
      }
   }
```


1.6. I/O + Delay _ Delay 1s RB0

```
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#byte PORTB = 0x06
   int16 count;
   int8 a;
//Chuong trinh ngat TMR0
#int_timer0
void interrupt_timer0()
{
   set_timer0(6);
   ++count;
   if(count == 2000)   // 2000*500us = 500000us = 1s
     {
       count=0;
       rotate_left(&a,1);
     }
}
//Chuong trinh chinh
void main(void)
```

Thang8831
http://www.picvietnam.com

```
{
  set_tris_b(0);
  enable_interrupts(int_timer0);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);
  enable_interrupts(global);
  set_timer0(6);// T_dinhthi = 2*(256 - 6)*1us = 500us
  a = 0x01;
while(true)
  {
    PORTB = a;
  }
}
```



1.7. Nháy Led RB0

Chương trình này làm nhấp nháy con led ở chân RB0 1s sáng, 1s tắt.

Code:

```
#include<16F877A.h>
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=10000000)
main()
{
while(true)
        {
        output_high(PIN_B0);
        delay_ms(1000);
        output_low(PIN_B0);
        delay_ms(1000);
        }
}
```

Trên đây:

Code:

```
#include<16F877A.h>
```

Khai báo con PIC bạn sử dụng, file này chương trình viết sẵn nhằm khai báo các bit, thanh ghi quan trọng trong con pic này.Các bạn có thể vào thư mục cài đặt **C:\Program Files\PICC\Devices\16F877A.h** để xem nó khai báo được những gì trong đó!
Code:

```
#fuses NOWDT,PUT,HS,NOPROTECT
```

Thiết lập các config
Code:

```
#use delay(clock=10000000)
```

Khai báo tần số dao động cấp cho PIC
Code:

```
output_high(PIN_B0)
```

Xuất ra chân RB0 mức logic 1 (tức 5V), do khi thực hiện hàm này đã bao hàm luôn việc tác động lên thanh ghi TRISB (dùng chọn chân I/O) do vậy ta không cần viết lệnh chọn chân I/O nữa.
Code:

```
output_low(PIN_B0)
```

Ngược lại
Code:

```
delay_ms(1000)
```

Tạo trễ khoảng thời gian theo mili giây là 1000 (tức 1s)
***Chú ý** hàm này chỉ có tác dụng khi có khai báo tần số dao động cấp cho PIC
Và bây giờ thử làm cho tất cả 8 led nối với portB chớp tắt 1s xem nào!Phải chăng ta sẽ làm như sau (Viết trong vòng lặp while):
Code:

```
{
output_high(PIN_B0);
output_high(PIN_B1);
output_high(PIN_B2);
output_high(PIN_B3);
output_high(PIN_B4);
output_high(PIN_B5);
output_high(PIN_B6);
output_high(PIN_B7);
delay_ms(1000);
output_low(PIN_B0);
output_low(PIN_B1);
output_low(PIN_B2);
output_low(PIN_B3);
output_low(PIN_B4);
output_low(PIN_B5);
output_low(PIN_B6);
output_low(PIN_B7);
delay_ms(1000);
}
```

Viết như thế này thì quá dài và thiếu chính xác nữa, có cách nào khác hay hơn không ? Sao ta không xuất đẩy hẳn portB lên mức cao,tạo trễ 1s rồi ép cho nó xuống mức thấp,tạo trễ 1s cùng một lúc nhỉ !
Bài tiếp theo sẽ đưa ra câu trả lời....

Thang8831
http://www.picvietnam.com

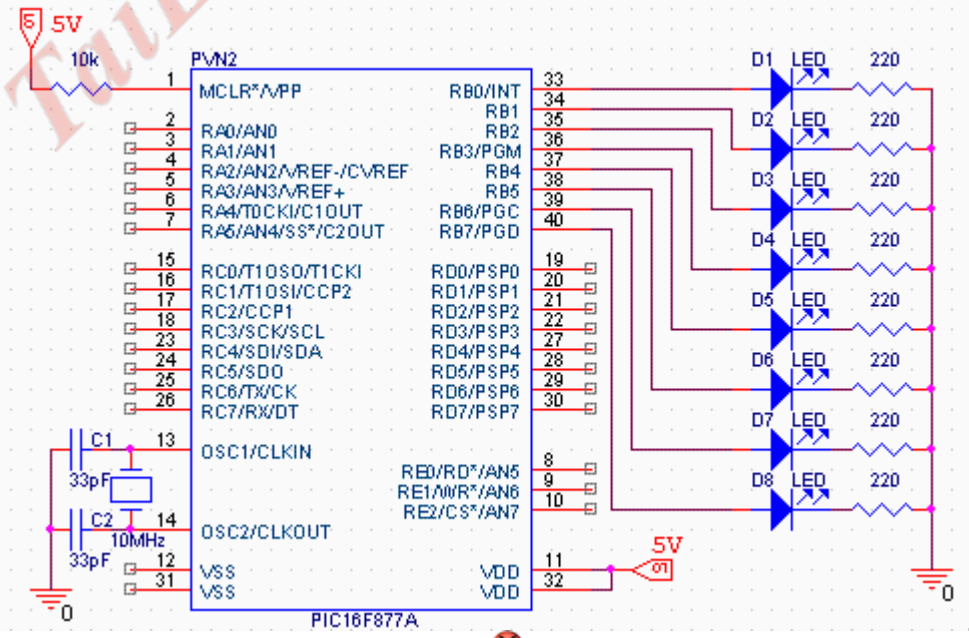
output_high(pin_xx);
output_low (pin_xx);

Hai câu lệnh trên chỉ làm cho chân ra xx là cao hay thấp, ứng với mức logic 1 hoặc 0.trong bài trên ta muốn cho sáng tất một port thì chỉ càn câu lệnh :

Code:

```
void main (void )
{
   set_tris_b(0);       // cả port B là port ra
   set_tris_c(0);      // cả port C là port ra
   port_b(0x00);     // khởi tạo giá trị đầu port B là 0 ( Tất cả led đều
tắt )
   port_c(0x00);    // khởi tạo giá trị đầu port B là 0 ( Tất cả led đều tắt
)
   delay_ms(100);
   while(1)
  {
        port_b(0xff);
        delay_ms(1000);
        port_c(0xff);
        delay_ms(1000);
   }
}
```

1.8. Delay 1s portB



Ăc..ặc..đang post thì bị cúp điện,bực cả mình...

Và đây là câu trả lời cho việc delay led ở portB 1s

Code:

```
#include<16F877A.h>
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=10000000)
#use fast_io(b)
#byte portb=0x6
main()
```

Thang8831
http://www.picvietnam.com

```
{
set_tris_b(0);
while(true)
{
portb=255;      //all led on
delay_ms(1000);
portb=0;                //all led off
delay_ms(1000);
}
}
```

Code:

```
#byte portb=0x6
```

Khai báo địa chỉ portB, không như trong MPLAB đã định nghĩa luôn cái này cho ta, nếu không có dòng này chương trình sẽ báo lỗi chưa định nghĩa portB

Code:

```
set_tris_b(0)
```

Tất cả các chân của portB là output, muốn set tất cả các chân là input thì trong ngoặc là 255,....Trong HELP hướng dẫn lệnh này như sau:

"These functions allow the I/O port direction (TRI-State) registers to be set. This must be used with FAST_IO and when I/O ports are accessed as memory such as when a #BYTE directive is used to access an I/O port. Using the default standard I/O the built in functions set the I/O direction automatically."

Rõ ràng khi set byte làm I/O nếu ta thêm khai báo:

Code:

```
#use fast_io(b)
```

Dùng khai báo này thì CCS sẽ chỉ thực hiện đúng một lệnh cho việc I/O các chân trên port tương ứng, nếu ko nó phải mất khoảng 3 hay 4 lệnh gì đó.

Phần I/O + Delay tạm thời xong như vậy, bác nào có phản hồi thì tiếp tục thảo luận.Xong phần này có thể viết về xuất led 7 đoạn, quét phím, LCD,....Bác nào viết thì cứ post lên cho bà con thảo luận nhé !

+Một phương án khác:

```
#include <16F877A.h>
#fuses HS, NOWDT, NOLVP, XT
#use delay(clock=4000000)
void main() {
byte leds = 0xff;
set_tris_b(0x00); // configure pins of PORTB as output
while (true) {
output_b(leds);
leds = ~leds; // toggle leds
delay_ms(1000);
}
}
```

1.9. Delay_Timer0

```
//****************************************
// CHUONG TRINH SU DUNG TMR0
//date:23/08/2005
//author:noisepic@gmail.com
```

```
//status: OK!
//****************************************
#include<16F877A.h>
#include<def_877A.h>
#fuses NOWDT,PUT,HS,NOPROTECT,NOLVP
#use delay(clock=20000000)

void tre_ms(unsigned int time);

void main()
{
TRISB=0;
setup_timer_0();
while(1)
{
PORTB = 0;
tre_ms(250);
PORTB =0xff;
tre_ms(250);
}
}
//****************************************
void tre_ms(unsigned int time)
{
int8 i,j;
GIE=0;
T0CS=0;// Chon internal
T0SE=0;// rising edge
PSA =0;// Timer mode
PS2=0;PS1=1;PS0=1;   // 1:8
for(i=0;i<time;i++)
{
for(j=0;j<5;j++)
{
TMR0=132;
while(tmr0if==0);
tmr0if = 0;
}
}
}
//**********
```

cho em hỏi, em đang làm thí nghiệm pic16f84a, làm led chớp tắt theo ý muốn nhưng em muốn dùng 1 biến trở để chỉnh tốc độ delay thì phải làm sao? nhờ các anh chỉ giáo! và em muốn dùng time 0 được không?
(xin lỗi mấy anh, mục này mà em hỏi F84a)

Người ta thường dùng bộ ADC trong PIC (tất nhiên đang nói đến những chip có module ADC) để đọc giá trị của biến trở rồi điều chỉnh thời gian/tốc độ chớp tắt. Với PIC16F84A (không có module ADC), bạn vẫn có thể làm được điều đó, nhưng dùng một cách khác, được

đề cập đến trong app. note 863 của Microchip. Bạn tìm đọc app. note đó trên web site của Microchip.

+cho em hoi trong ccs co ho tro vong lap " for" ko vay cac bac .. sao em lam hoai ma ko duoc vay .. bien dich van dung ma chay thi ko duoc...

Code:

```
 #include<16F877.h>
#device *=16 ADC=8
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=10000000)
#use fast_io(b)
#byte p3=0x06
#byte porta=0x05

void main()
{
   int8 const led[]={1,2,4,8,16,32,64,128},a;
   set_tris_b(0);
   set_tris_a(0);
   set_tris_c(0);
   set_tris_d(0);

   while(true)
    {
      for( a=0;a<8;a++)
      p3=led[a];
      delay_ms(100);
    }
}
```


Như thế nào là chạy không được?

Theo code của bạn, trong vòng while quá trình sau sẽ được thực hiện, nếu a là biến:

1. Xuất 8 lần dữ liệu ra p3 (địa chỉ 0x06)
2. Làm trễ 100 ms
3. Quay lại bước 1

Em biet no sai o dau roi. Sau moi lan gia tri a tang len 1 . phai delay mot khoan thoi gian, neu ko no se chay den gia tri cuoi cung roi xuat ra portb , vay p3=led[7]=128; nhu vay xem nhu vong lap ko co tac dung(co vay thoi cugn so y thiet(~_~).. Vay doan code dung la 8 con led se chay duoi nhau hoai (^_^)..)

Code:

```
  #include<16F877.h>
#device *=16 ADC=8
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=10000000)
#use fast_io(b)
#byte p3=0x06

void main()
{
   int8 const led[]={1,2,4,8,16,32,64,128},a;
   set_tris_b(0);
   while(true)
    {
      for(a=0;a<8;a++)
      {
      p3=led[a];
      delay_ms(20);
```

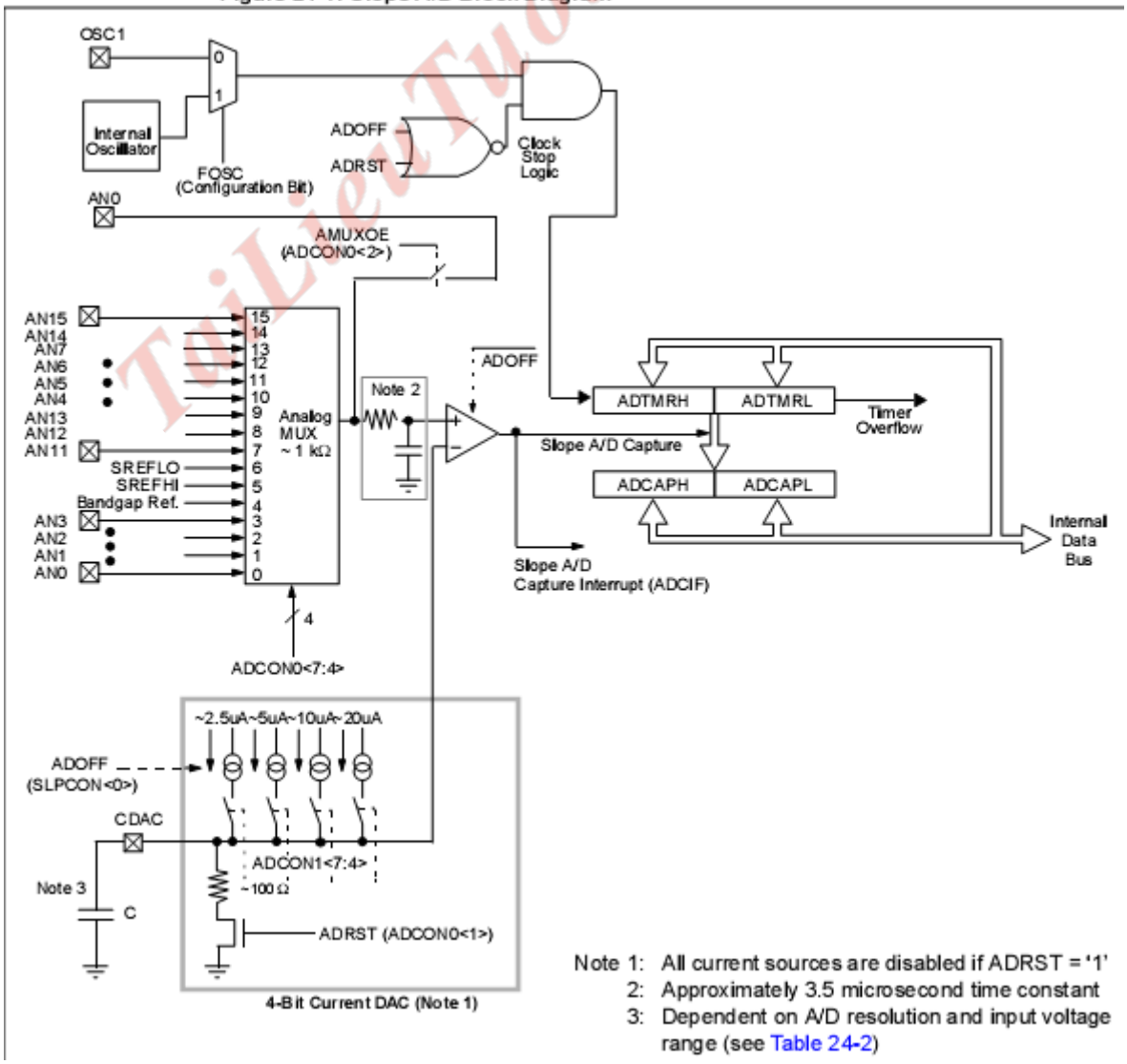
```
        }
    }
}
```

namqn: bạn nên đặt phần code của bạn giữa hai tag [code] và [/code] để định dạng cho phần code (hai tag không có khoảng trắng nào hết chứ không phải như được hiển thị ở đây).

2. ADC

A. Sơ đồ:

Figure 24-1: Slope A/D Block Diagram



Note 1: All current sources are disabled if ADRST = '1'
2: Approximately 3.5 microsecond time constant
3: Dependent on A/D resolution and input voltage range (see Table 24-2)

B.Code

B.1. ADC reading voltage

--reading Voltage

```
#include <16f877A.h>
#include <def_877a.h>
#device *=16 ADC=10
```

Thang8831

http://www.picvietnam.com

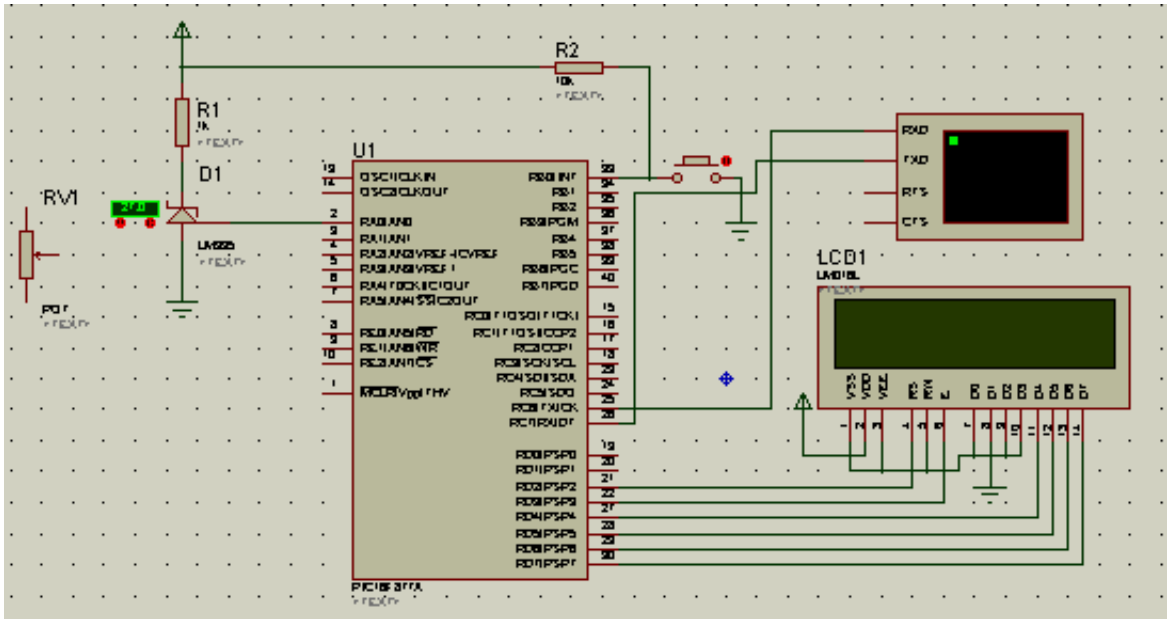
```
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, BROWNOUT, LVP, NOCPD,
NOWRT
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
#include <lcd_lib_4bit.c>

int16 temp,high,low;
int8 nghin,tram,chuc,donvi;
int1 mili_volt;
float volt;
int8 const a1[10] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};   // Khong co
cham
int8 const a2[10] = {0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10};   // Co cham

#INT_EXT
void ngatngoai()
{

}
//====================================
void convert_bcd(int16 x)
  {
   nghin = x / 1000 + 0x30;
   temp  = x % 1000;
   tram  = temp / 100 + 0x30;
   temp  = temp % 100;
   chuc  = temp / 10 + 0x30;
   donvi = temp % 10 + 0x30;
  }
//====================================
void main() {

lcd_init();
printf(lcd_putchar,"CT Do dien ap");

   setup_adc_ports(AN0);
   chs0=0;chs1=0;chs2=0;
   setup_adc(ADC_CLOCK_INTERNAL);
   delay_us(10);

do {
   temp = read_adc();
// Dang so thap phan
   volt = (float)(temp*5)/1023;
   if (volt < 1) mili_volt=1;
   else mili_volt = 0;
// Dang so nguyen
   high = (temp*5)/1023;
   low = (temp*5)%1023;
// ====Truyen len may tinh
```

Thang8831
http://www.picvietnam.com

```
  printf("\r\nGia tri ADC = %lu",read_adc());
  Printf("\r\nGia tri dien ap = %f",volt);
//================
  volt = volt * 1000;

  convert_bcd((int16)volt);
  lcd_putcmd(0xC0);

  printf(lcd_putchar, "V = ");
  if (!mili_volt) {lcd_putchar(nghin);lcd_putchar(".");}
  lcd_putchar(tram);
  lcd_putchar(chuc);
  lcd_putchar(donvi);
  if (mili_volt) printf(lcd_putchar,"  mV");
  else   printf(lcd_putchar," V");

  printf("\r\n V = %lu",high);
  if(low < 100)
    printf(".0%2lu",low);
  else
    printf(".%lu",low);
}  while(true);

}
```


B.2. LM335_LCD

a. Sơ đồ:



b. lcd_lib_4bit

```
#include <stddef.h>

#define LCD_RS          PIN_D2
//#define LCD_RW          PIN_A1
#define LCD_EN          PIN_D3
```

```
#define LCD_D4         PIN_D4
#define LCD_D5         PIN_D5
#define LCD_D6         PIN_D6
#define LCD_D7         PIN_D7

// misc display defines-
#define Line_1         0x80
#define Line_2         0xC0
#define Clear_Scr      0x01

// prototype statements
#separate void LCD_Init ( void );// ham khoi tao LCD
#separate void LCD_SetPosition ( unsigned int cX );//Thiet lap vi tri con tro
#separate void LCD_PutChar ( unsigned int cX );// Ham viet1kitu/1chuoi len LCD
#separate void LCD_PutCmd ( unsigned int cX) ;// Ham gui lenh len LCD
#separate void LCD_PulseEnable ( void );// Xung kich hoat
#separate void LCD_SetData ( unsigned int cX );// Dat du lieu len chan Data
// D/n Cong
#use standard_io ( B )
#use standard_io (A)

//khoi tao LCD*********************************************
#separate void LCD_Init ( void )
   {
   LCD_SetData ( 0x00 );
   delay_ms(200);       /* wait enough time after Vdd rise >> 15ms */
   output_low ( LCD_RS );// che do gui lenh
   LCD_SetData ( 0x03 );   /* init with specific nibbles to start 4-bit mode */
   LCD_PulseEnable();
   LCD_PulseEnable();
   LCD_PulseEnable();
   LCD_SetData ( 0x02 );   /* set 4-bit interface */
   LCD_PulseEnable();      /* send dual nibbles hereafter, MSN first */
   LCD_PutCmd ( 0x2C );    /* function set (all lines, 5x7 characters) */
   LCD_PutCmd ( 0b00001100);    /* display ON, cursor off, no blink */
   LCD_PutCmd ( 0x06 );    /* entry mode set, increment & scroll left */
   LCD_PutCmd ( 0x01 );    /* clear display */
   }
#separate void LCD_SetPosition ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
   LCD_SetData ( swap ( cX ) | 0x08 );
   LCD_PulseEnable();
   LCD_SetData ( swap ( cX ) );
   LCD_PulseEnable();
   }
#separate void LCD_PutChar ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
```

```
      output_high ( LCD_RS );
      LCD_PutCmd( cX );
      output_low ( LCD_RS );
   }
#separate void LCD_PutCmd ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
   LCD_SetData ( swap ( cX ) );     /* send high nibble */
   LCD_PulseEnable();
   LCD_SetData ( swap ( cX ) );     /* send low nibble */
   LCD_PulseEnable();
   }
#separate void LCD_PulseEnable ( void )
   {
   output_high ( LCD_EN );
   delay_us ( 3 );         // was 10
   output_low ( LCD_EN );
   delay_ms ( 3 );         // was 5
   }
#separate void LCD_SetData ( unsigned int cX )
   {
   output_bit ( LCD_D4, cX & 0x01 );
   output_bit ( LCD_D5, cX & 0x02 );
   output_bit ( LCD_D6, cX & 0x04 );
   output_bit ( LCD_D7, cX & 0x08 );
   }
```

B.3. LM335_F877A_LCD1602

```
/* Mach do nhiet do
- MCU    = PIC16F877A
- Sensor = LM335 (co the thay the bang LM35D)
- MAX232 giao tiep may tinh
- LCD1602A de hien thi gia tri nhiet do
Mo ta phan cung:
- Mach cho sensor mac nhu trong Datasheet cua LM335
  Chan V_out noi qua dien tro 1K voi +5V. Chan nay cung duoc noi voi kenh AN0 cua PIC
  Chan Adj noi voi dien tro 10K de tinh chinh
  Chan GND noi dat
- Mach VDK gom co LCD va max232
  LCD noi voi PORTD cua PIC
    RS -> RD2, RW -> GND, E -> RD3
    D4-D7 -> RD4-RD7
  Max232:
    chan10 -> RC6, chan9 -> RC7
    chan8  -> chan3 DB9, chan7 -> chan2 DB9, chan5 DB9 -> GND
  Kenh AN0 cua PIC noi den chan V_out LM335
  Nut bam noi tai chan RB0 -> nhan ngat ngoai
  Thach anh loai 20MHz, tu 22pF
==============================================
- Designer: linhnc308@yahoo.com
- Chuc thanh cong cung VDK PIC
```

```
*/
#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=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
#include <lcd_lib_4bit.c>
int8 low,high,key,mode,min,max,mode1,i;
int1 blink,on_off,blink_min,blink_max;
int1 do_F;
void convert_bcd(int8 x);
void set_blink();
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 = 0 ;
 mode   =0 ;
 mode1 = 0 ;
 blink=0  ;

  trisa = 0xFF;
  trisb = 0x01;
  trisd = 0x00;

    printf("Chuong trinh do nhiet do\n");
    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);  // Chuyen doi tach so tram, chuc, donvi de hien thi len LED 7
    delay_ms(1000);
    LCD_putcmd(0xC0);
    Printf(LCD_putchar," Init OK");

    while(1)
    {
     if (i==15)
       {
        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);
        LCD_putcmd(0xC0);
        printf(LCD_putchar," T = ");
        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 set_blink()
{
  switch(mode)
  {
```

```
     case 1: blink_min=1; break;
     case 2: {blink_max=1; blink_min=0;}  break;
     case 3: {mode=0; blink=0; blink_min=0; blink_max=0;}   break;
   }
}
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(" ");
    }
}
```

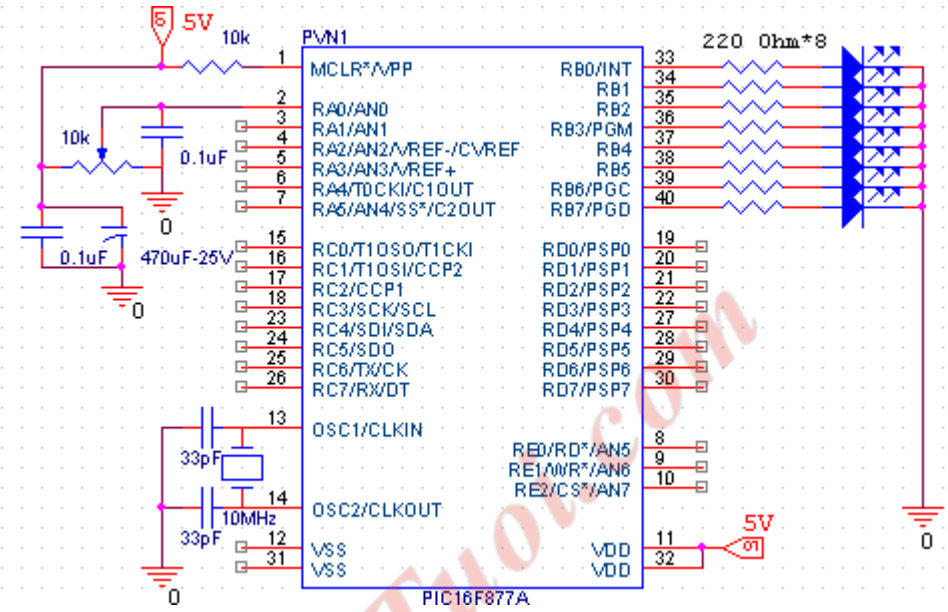
Cho toi hoi tai sao khi khai bao:
#device PIC16F877 *=16 ADC=10
thi CCSC bao loi: "Can not change device type this far into the code"
Cach khac phuc

Đây là 1 ví dụ nhỏ về ADC,chân RA0 lấy tín hiệu Analog từ biến trở và xuất giá trị số biến đổi tương ứng qua tám led nối ở portB
Code:

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#device 16F877*=16 ADC=8
#use delay(clock=10000000)
Int8 adc;
main()
{
setup_adc(adc_clock_internal);
setup_adc_ports(AN0);
set_adc_channel(0);
delay_ms(10);
while(true)
{
adc=read_adc();
output_B(adc);
}
}
```

Hình Kèm Theo

Thang8831
http://www.picvietnam.com



Khai báo:

Code:

```
"#fuses HS,NOWDT,NOPROTECT,NOLVP,NOBROWNOUT
```

-HS :sử dụng thạch anh tần số cao
-NOWDT:tắt WDT
-NOPROTECT:tắt PROTECT
-NOLVP:không dùng LVP
-NOBROWNOUT:ko BROWNOUT

Còn cụ thể ý nghĩa thế nào,bạn vào help của CCS C gõ :"#fuses"

Cái luồng bên này nhh vẽ hình cẩu thả quá!

Cái chân MCLR*, bạn phải nối thên cái công tắc ấn vào. nhh nối như vậy thì không reset được con PIC đâu

Hai cái chân Vss phải được nối mass.

Chân nào không xài, nhh nên đánh dấu bỏ đi (trong thanh công cụ của ORCAD có cái dấu này đó).

Một cách viết khác để tham khảo:

```
#include <16F877A.h>
#fuses HS, NOWDT, NOPROTECT, NOLVP
#device 16F877*=16, ADC=8
#use delay(clock=4000000)

void main() {
    setup_adc(adc_clock_internal);
    setup_adc_ports(ALL_ANALOG);
    set_adc_channel(0); // TM Board: VR3=0, VR2=1, VR1=2
delay_ms(10);
   while (true)
   output_b(read_adc());
}
```

B.4. ADC_186

```
//   ANALOGUE TO DIGITAL CONVERTER HARDWARE INTERFACE PROGRAM
//   COPYRIGHT PROPERTY OF ALPHADATA DESIGNS LIMITED (c) 1999
//   WRITTEN FOR THE MAX186 12 BIT 8-CHANNEL MICRO-WIRE A TO D
```

```
//   published by permission of Alphadata designs on
//   Hi-Tech C website,  http://www.workingtex.com/htpic.  Thanks!
//-----------------------------------------------------
// Version History
//-----------------------------------------------------
// Issue 1.0 : 21/12/1999 : First Officially Released
//-----------------------------------------------------
#include "ioh8314.h"
#include "h8genlib.h"
#include "spi.h"

extern byte p4dr;

/* call this to set up the hardware ports */

void adc_initialise(void)
{
}
/* this table makes up the muddled way the device is addressed */
/* see the data sheet for the MAX186 to see why !            */
const byte command_table[]={        0b10001111,
                                    0b11001111,
                                    0b10011111,
                                    0b11011111,
                                    0b10101111,
                                    0b11101111,
                                    0b10111111,
                                    0b11111111};
/* call this to read one channel of data from the a to d */
word adc_read_channel(byte channel_number)
{
        byte    bits;
        byte    command;
        int     data;

        spi_adc_select();
        /* Start by sending command byte */
        command = command_table[channel_number];

        for (bits=0;bits<8;bits++) {
                if ((command & 0x80) == 0) spi_lodata(); else spi_hidata();
                command = command << 1;

                spi_hiclock();
                spi_loclock();
        }
        spi_lodata();
        spi_hiclock();
        spi_loclock();
```

```
        data = 0;
        for (bits=0;bits<12;bits++) {
                data = (data << 1);
                if (P4DR & 0b10000000) data = data | 0x0001;
                spi_hiclock();
                spi_loclock();
        }
        spi_adc_deselect();
        return (data);
}
// does an average reading
word adc_filtered_read(byte channel,int readings)
{
        long    av;
        word    lp;
        av = 0;
        for (lp=0;lp<readings;lp++) {
                av = av + adc_read_channel(channel);
        }
        return (word)(av / readings);
}
// end
```

+Mình mới học về PIC.Cho hỏi ADC module có chức năng gì,công dụng ntn trên PIC?
Mình nghĩ cho một điện thế analog thì nó xuất ra tín hiệu digital.Mình đã có đọc đoạn code này nhưng chưa hiểu rõ :
Code:

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#device 16F877*=16 ADC=8
#use delay(clock=10000000)
Int8 adc;
main()
{
setup_adc(adc_clock_internal);
setup_adc_ports(AN0);
set_adc_channel(0);
delay_ms(10);
while(true)
{
adc=read_adc();
output_B(adc);
}
}
```

TL: ADC là chức năng chuyển đổi tín hiệu analog -> digital (Analog Digital Converter). Nó dùng 1 điện áp để so sánh (Vref) tùy độ phân giải mà điện áp này được chia là nhiều mức khác nhau (2^n) với 16F877A ADC 10 bit tức Vref dc chia thành 2^10 = 1024 mức. Mỗi mức ADC cách nhau tương ứng với Vref/số mức. Vref = 5V thì mỗi mức ADC của 16F877 ứng với 5V/1024 = 4.88mV. Điện áp trên cổng AN sẽ được so sánh để ra được mức ADC tương ứng. 0V ứng với ADC =0, 4.88mV ứng với ADC =1...Kết quả được chứa trong 2 thanh ghi ADRESH:ADRESL.

Còn mạch cho code trên thì bạn mắc 1 cầu chia áp để đưa điện áp vào chân AN0 là được.

Thang8831
http://www.picvietnam.com

Dùng biến trở cho gọn.

Còn PWM thì bạn đọc trong datasheet đó chương 8 phần CAPTURE/COMPARE/PWM MODULES và xem thêm bên ứng dụng điều khiển PID cho động cơ DC.

+Em có đoạn code sau nhưng em không hiểu là lệnh adcHI và adcLO làm gì (với khai báo
char adcHI,adcLO).Anh nào giải thích hộ em

Code:

```
   while(TRUE)
        {
                adcValue = read_adc(); // Get ADC reading

                adcHI = (char)((adcValue >> 5)& 0x1f);
                adcLO = (char)((adcValue & 0x1f)|0x80);

                putc(adcHI);
                putc(adcLO);

                delay_ms(10);  // Preset delay, repeat every 10ms
        }
}
```

TL: adcHI và adcLO là các biến kiểu char (8-bit), không phải là lệnh.

Trong đoạn lệnh mà bạn đã đưa ra, kết quả của việc biến đổi A/D được tách thành 2 phần, phần cao chứa trong adcHI, còn 5 bit thấp hơn của kết quả được chứa trong adcLO. Bit 7 của adcLO được bật.

Sau đó adcHI và adcLO được xuất ra thông qua hàm putc().

3. DAC

3.1. DAC_1446

```
//   DIGITAL TO ANALOGUE CONVERTER HARDWARE INTERFACE PROGRAM
//   COPYRIGHT PROPERTY OF ALPHADATA DESIGNS LIMITED (c) 1999
//   WRITTEN FOR THE LTC1446 12 BIT 2-CHANNEL MICRO-WIRE D TO A
//   published by permission of Alphadata designs on
//   Hi-Tech C website, http://www.workingtex.com/htpic. Thanks!
//-----------------------------------------------------
// Version History
//-----------------------------------------------------
// Issue 1.0 : 21/12/1999 : First Officially Released
//-----------------------------------------------------
#include <stdio.h>

#include "h8genlib.h"
#include "ioh8314.h"
#include "sci.h"
#include "spi.h"
#include "dac_1446.h"
extern byte p4dr;
/* call this to write one channel of data to the d to a */
void dac_write(word output)
{
```

```
byte count;
byte block;
byte ops[3];
//char   s[50];
        /* Select ADC device */
        spi_dac_select();
        //Format output data
        if (output<0x1000) {
                ops[0]=0;
                ops[1]=output >> 8;
                ops[2]=output;
        } else {
                ops[0] = output >> 4;
                ops[1] = (output << 4) + 0x0F;
                ops[2] = 0xFF;
        }
//      ops[0]= ((byte*)&output)[1];
//      ops[1]= ((byte*)&output)[2];
//      ops[2]= ((byte*)&output)[3];
//      sprintf(s,"%02X %02X %02X\r\n",ops[0],ops[1],ops[2]);
//      sci_put_string(0,s);
        /* output three blocks of eight bits */
        for (block=0;block<3;block++) {
                for (count=0;count<8;count++) {
                        if ((ops[block] & 0x80) == 0) spi_lodata(); else spi_hidata();
                        spi_hiclock();
                        spi_loclock();
                        ops[block] = ops[block] << 1;
                }
        }
        spi_dac_deselect();
}
/* call this to set up the hardware ports */
void dac_initialise(void)
{
        dac_write(0);
}
// end
```

4. Timer

Trong Pic16f877a có 3 timer :
+ Timer0 : 8 bit
+ Timer1 : 16 bit
+ Timer2 : 8 bit

Timer dùng cho nhiều ứng dụng : định thời, capture, pwm, ...

Timer có nhiều kiểu chia tần, dùng chia trước và sau (prescale và postscale) là chia trước và chia sau, có nhiều cách đặt tỉ lệ cho Timer từ 1:1 - 1:256 tức là cách chia này giúp cho ta nhận được xung kích vào Timer sẽ được chậm đi n lần (1:n) so với 1Tcy (Fosc/4), và như vậy ta sẽ được xung kich chậm hơn:

setup_timer_0(RTCC_INTERNAL|RTCC_DIV_4); // 4Mhz => dùng dao động nội, chế độ prescal 1:4 => clock cho Timer0 la Fosc/4/4 = Fosc/16.

Khi này ta có Fosc/4 => Tcy = 1us. Ttimer0 = Fosc/16 => 4us
Timer0 tràn 8 bit => 4x8bit = 1024us.
8 bit có 256 trạng thái chứ không phải 255 trạng thái, do đó x256,

Còn cách thứ 2 Postscale (Only Timer2) thì nó sẽ đếm số lần tràn của Timer2, Nghĩa là: 1:2 - 2 làn tràn cho ra 1 lần xung .....
1:16 - 16 làn Timer2 tràn

Giả sử Áp dụng với Timer2 thay Timer0 ở trên và với postscale 1:8 ta sẽ thu được đầu ra là : 4x256x8 = 8192 us

4.1. Timer0

Thanh ghi tác động:
Các lệnh:
Code:

```
setup_TIMER_0(mode);
setup_COUNTERS (rtcc_state, ps_state); // hay setup_WDT()
set_TIMER0(value); // hay set_RTCC(value) :xác định giá trị ban đầu (8bit)
cho Timer0
get_TIMER0(); // hay get_RTCC() :trả về số nguyên (8bit) của Timer0
```

Trong đó mode là một hoặc hai constant (nếu dùng hai thì chèn dấu "|"ở giữa) được định nghĩa trong file 16F877A.h gồm :
RTCC_INTERNAL : chọn xung clock nội
RTCC_EXT_L_TO_H : chọn bit cạnh lên trên chân RA4
RTCC_EXT_H_TO_L : chọn bit cạnh xuống trên chân RA4
RTCC_DIV_2 :chia prescaler 1:2
RTCC_DIV_4 1:4
RTCC_DIV_8 1:8
RTCC_DIV_16 1:16
RTCC_DIV_32 1:32
RTCC_DIV_64 1:64
RTCC_DIV_128 1:128
RTCC_DIV_256 1:256
rtcc_state là một trong những constant sau:
RTCC_INTERNAL
RTCC_EXT_L_TO_H
RTCC_EXT_H_TO_L
ps_state là một trong những constant sau:
RTCC_DIV_2
RTCC_DIV_4
RTCC_DIV_8
RTCC_DIV_16
RTCC_DIV_32
RTCC_DIV_64
RTCC_DIV_128
RTCC_DIV_256
WDT_18MS
WDT_36MS
WDT_72MS
WDT_144MS
WDT_288MS
WDT_576MS
WDT_1152MS

WDT_2304MS

Mình cũng chưa hiểu ý nghĩa của hàm WDT_..., ko biết có phải khai báo như trên thì sau khoảng thời gian ms bao nhiêu đó đặt sau WDT_ thì sẽ reset lại Pic ?????

4.2. Timer1

Thanh ghi tác động:
Các lệnh:
Code:

```
setup_TIMER_1(mode);
set_TIMER1(value);              // xác định giá trị ban đầu (16bit) cho
Timer1
get_TIMER1();                   // trả về số nguyên (16bit) của Timer1
```

mode gồm (có thể kết hợp bằng dấu "|"):

T1_DISABLED : tắt Timer1
T1_INTERNAL : xung clock nội (Fosc/4)
T1_EXTERNAL : xung clock ngoài trên chân RC0
T1_EXTERNAL_SYNC : xung clock ngoài đồng bộ
T1_CLK_OUT
T1_DIV_BY_1
T1_DIV_BY_2
T1_DIV_BY_4
T1_DIV_BY_8

4.3. Timer2

Thanh ghi tác động:
Các lệnh:
Code:

```
setup_TIMER_2(mode, period, postscale);
set_TIMER2(value);              // xác định giá trị ban đầu (8bit) cho Timer2
get_TIMER2();                   // trả về số nguyên  8bit
```

Với mode gồm (co the ket hop bang dau "|"):
T2_DISABLED
T2_DIV_BY_1
T2_DIV_BY_4
T2_DIV_BY_16
period là số nguyên từ 0-255, xác định giá trị xung reset
postscale là số nguyên 1-16, xác định reset bao nhiêu lần trước khi ngắt.
Code:

```
////////////////////////////////////////////////////////////////////////
////                    EX_STWT.C                                   ////
////                                                                ////
//// This program uses the RTCC (timer0) and interrupts to keep a  ////
//// real time seconds counter.  A simple stop watch function is   ////
//// then implemented.                                             ////
////                                                                ////
//// Configure the CCS prototype card as follows:                  ////
////     Insert jumpers from: 11 to 17 and 12 to 18.               ////
////////////////////////////////////////////////////////////////////////
#include<16f877a.h>
#fuses HS,NOLVP,NOWDT,PUT
```

```
#use delay(clock=20000000)
#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)

#define high_start 76
byte seconds, high_count;

#INT_RTCC                        //Interrupt procedure
clock_isr() {                    //called every time RTCC
  high_count -= 1;               //flips from 255 to 0
  if(high_count==0) {
    ++seconds;
    high_count=high_start;       //Inc SECONDS counter every
  }                              //76 times to keep time
}
void main() {                    //A simple stopwatch program
  byte start, time;

  high_count = high_start;
  setup_timer_0( RTCC_INTERNAL | RTCC_DIV_256 );
  set_timer0(0);
  enable_interrupts(INT_RTCC);
  enable_interrupts(GLOBAL);
  do {
    printf("Press any key to begin.\n\r");
    getc();
    start = seconds;
    printf("Press any key to stop.\r\n");
    getc();
    time = seconds - start;
    printf("%U seconds.\n\r", time);
  } while (TRUE);
}
```

4.4. frequencymeter

Code:

```
#include <16f628.H>
#use   delay(clock=4000000)//
#fuses NOPROTECT,NOWDT,PUT,noBROWNOUT,noLVP,NOMCLR,xt
#BYTE     PORT_A=0X05
#BYTE     PORT_B=0X06
/***entegreterbiyecisi@yahoo.com***//***entegreterbiyecisi@yahoo.com***/
// LCD STUFF
#define LCD_RS      PIN_b0
#define LCD_EN      PIN_b1
#define LCD_D4      PIN_b2
#define LCD_D5      PIN_b3
#define LCD_D6      PIN_b4
#define LCD_D7      PIN_b5
#define FIRST_LINE  0x00
#define SECOND_LINE 0x40
#define CLEAR_DISP  0x01
#define CURS_ON     0x0e
#define CURS_OFF    0x0c
```

```
/***entegreterbiyecisi@yahoo.com***/
#use standard_io ( a )
#use standard_io ( b )
/***entegreterbiyecisi@yahoo.com***/
// proto statements
void LCD_Init        ( void );
void LCD_SetPosition ( unsigned int cX );
void LCD_PutChar     ( unsigned int cX );
void LCD_PutCmd      ( unsigned int cX );
void LCD_PulseEnable ( void );
void LCD_SetData     ( unsigned int cX );
/***entegreterbiyecisi@yahoo.com***//***entegreterbiyecisi@yahoo.com***/
/********************entegreterbiyecisi@yahoo.com************************/
int32 ab=0,hz=0;
int1 stept_say=0,data_bitti=0,step,aa=0;
int16 sayi=0,tr=20;
/*********************entegreterbiyecisi@yahoo.com**********************/
#int_timer1
tas(){
ab++;
}
#int_timer0
sn(){sayi=0;
 set_timer0(61);//(255-60)*195*20=1000000us=dahili 1sn icin
 if(tr){ tr--;}
 else{delay_us ( 698 );
      output_low(pin_a0);
      disable_interrupts (global);
      disable_interrupts(int_timer0);
      disable_interrupts(int_timer1);
      sayi=get_timer1();
      aa=1;
      hz=sayi+(65536*ab);
      tr=20;
                                }}
/*********************entegreterbiyecisi@yahoo.com************************/
   void main() {
      setup_timer_1(t1_external|t1_div_by_1);
      setup_timer_0 (RTCC_INTERNAL|RTCC_DIV_256);
      enable_interrupts(int_timer0); // timer0
      enable_interrupts(int_timer1);
      enable_interrupts(global);
      lcd_init();
      SET_TRIS_A(0b00100000);
      SET_TRIS_B(0b11000000);
      set_timer0(61);
      set_timer1(0);

      LCD_SetPosition(first_LINE+0);
      printf(lcd_putchar,"\NECATi KIYLIOGLU ");
      LCD_SetPosition(second_LINE+1);
      printf(lcd_putchar,"\ 0532 613 65 87");
      delay_ms (500);
      LCD_PutCmd ( CLEAR_DISP );
      sayi=0;
      hz=0;
/*********************entegreterbiyecisi@yahoo.com************************/
while(true){
   if(aa==1){
      //LCD_PutCmd ( CLEAR_DISP );
```

```
      LCD_SetPosition(first_LINE+0);
      printf(lcd_putchar,"\FREQUENCYMETER     ");
    if(999>=hz){
      LCD_SetPosition(second_LINE+0);
      printf(lcd_putchar,"\FRQ=%ldHz          ",hz);}
      ////////////////////////////////////////
    if(hz>=1000){
    if(999999>=hz){
      LCD_SetPosition(second_LINE+0);
      printf(lcd_putchar,"\FRQ=%3.3wKhz         ",hz);}}
      ////////////////////////////////////////////
    if(hz>=1000000){
      LCD_SetPosition(second_LINE+0);
      printf(lcd_putchar,"\FRQ=%2.6wMhz          ",hz);}
      ////////////////////////////////////////////
      delay_ms (1);

      set_timer1(0);
      enable_interrupts(int_timer0);
      enable_interrupts(int_timer1);
      enable_interrupts (global);
      aa=0;
      ab=0;
      }
            }}
/**********************entegreterbiyecisi@yahoo.com************************/
/***entegreterbiyecisi@yahoo.com***/ //lcd basla
void LCD_Init ( void ){
    LCD_SetData ( 0x00 );
    output_low ( LCD_RS );
    LCD_SetData ( 0x03 );   // init with specific nibbles to start 4-bit
mode
    LCD_PulseEnable();
    LCD_PulseEnable();
    LCD_PulseEnable();
    LCD_SetData ( 0x02 );   // set 4-bit interface
    LCD_PulseEnable();      // send dual nibbles hereafter, MSN first
    LCD_PutCmd ( 0x2C );    // function set (all lines, 5x7 characters)
    LCD_PutCmd ( 0x0C );    // display ON, cursor off, no blink
    LCD_PutCmd ( 0x01 );    // clear display
    LCD_PutCmd ( 0x06 );    // entry mode set, increment
    }
/***entegreterbiyecisi@yahoo.com***/
void LCD_SetPosition ( unsigned int cX )
    {
    // this subroutine works specifically for 4-bit Port A
    LCD_SetData ( swap ( cX ) | 0x08 );
    LCD_PulseEnable();
    LCD_SetData ( swap ( cX ) );
    LCD_PulseEnable();
    }
/***entegreterbiyecisi@yahoo.com***/
void LCD_PutChar ( unsigned int cX )
    {
    // this subroutine works specifically for 4-bit Port A
    output_high ( LCD_RS );
    LCD_SetData ( swap ( cX ) );     // send high nibble
    LCD_PulseEnable();
    LCD_SetData ( swap ( cX ) );     // send low nibble
    LCD_PulseEnable();
```

```
    output_low ( LCD_RS );
    }
/***entegreterbiyecisi@yahoo.com***/
void LCD_PutCmd ( unsigned int cX )
    {
    // this subroutine works specifically for 4-bit Port A
    LCD_SetData ( swap ( cX ) );     // send high nibble
    LCD_PulseEnable();
    LCD_SetData ( swap ( cX ) );     // send low nibble
    LCD_PulseEnable();
    }
/***entegreterbiyecisi@yahoo.com***/
void LCD_PulseEnable ( void )
    {
    output_high ( LCD_EN );
    delay_us ( 100 );
    output_low ( LCD_EN );
    delay_ms ( 5 );
    }
/***entegreterbiyecisi@yahoo.com***/
void LCD_SetData ( unsigned int cX )
    {
    output_bit ( LCD_D4, cX & 0x01 );
    output_bit ( LCD_D5, cX & 0x02 );
    output_bit ( LCD_D6, cX & 0x04 );
    output_bit ( LCD_D7, cX & 0x08 );
    }
/***entegreterbiyecisi@yahoo.com***/ //lcd son
////////////////////////////////////////////////////////////////////////
```


Cảm ơn bạn **Necati** đã Post lên chương trình mà tôi và nhiều anh em đang quan tâm. Tôi muốn hỏi thêm là nếu muốn đo 1 tần số sóng mang nằm trong tín hiệu điều chế thì giải quyết như thế nào. Giả sử có 1 tín hiệu cần điều chế có f = 2KHz độ rộng xung là 2us, tín hiệu sóng mang có f0 = 20MHz, nghĩa là trong 2us của tín hiệu điều chế sẽ có 20*2 = 40 chu kỳ xung của sóng mang trong đó. Bài toán ở đây là đo được tần số 20MHz từ tín hiệu đã điều chế đó.

Tôi đọc kỹ thì thấy chương trình của anh **Necati** cũng đo tần số liên tục, nhưng sử dụng thêm Timer0 để định thời 1 giây, và Timer1 cũng để lấy giá trị đếm số lần xuất hiện xung vào. Tất nhiên, chương trình viết rất chuyên nghiệp, đó cũng là điều mà tôi và nhiều anh em cần học hỏi thêm rất nhiều.

5. INTERRUPT

Các lệnh dùng cho ngắt:

Code:

```
enable_interrupts(level);                    //cho phép ngắt kiểu level
disable_interrupts(level);                   //cấm ngắt kiểu level
ext_int_edge(edge);                          // chọn cách lấy xung loại  edge
```

level bao gồm:

GLOBAL : ngắt toàn cục
INT_RTCC : tràn TMR0
INT_RB : có thay đổi trạng thái một trong các chân RB4 đến RB7
INT_EXT : ngắt ngoài
INT_AD : chuyển đổi AD đã hoàn tất
INT_TBE : bộ đệm chuyển RS232 trống
INT_RDA : data nhận từ RS232 sẵn sàng

INT_TIMER1 : tràn TMR1
INT_TIMER2 : tràn TMR2
INT_CCP1 : có capture hay compare trên CCP1
INT_CCP2 : có capture hay compare trên CCP2
INT_SSP : có hoạt động SPI hay I2C
INT_PSP : có data vào cổng parallel slave
INT_BUSCOL : xung đột bus
INT_EEPROM : ghi vào eeprom hoàn tất
INT_TIMER0 : tràn TMR0
INT_COMP : kiểm tra bằng nhau comparator

edge bao gồm:

L_TO_H : cạnh lên
H_TO_L : cạnh xuống

Sau khai báo trên để vào đoạn chương trình ngắt, khai báo:

#INT_.....

Ví dụ vào thực thi ngắt ngoài, ta có đoạn code:

Code:

```
#INT_EXT
void ngat_ngoai()
{
//Chương trình ngắt tại đây
}
```

5.1. Ngắt Timer0

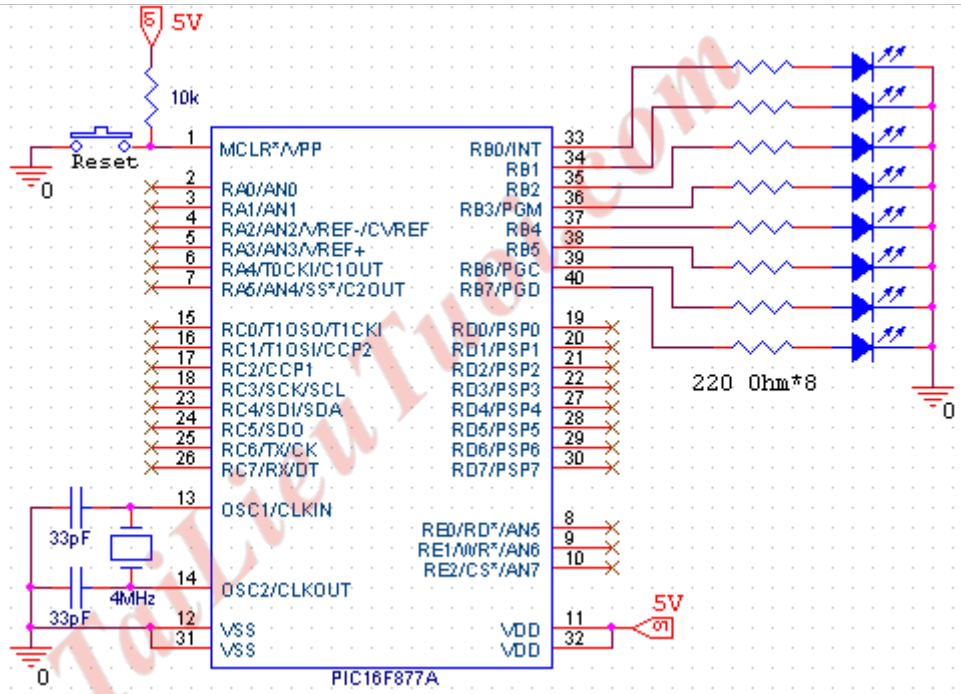
Đây là chương trình dùng ngắt Timer0 định thì 1s.

Đầu tiên led ở chân RB0 sáng, sau 1s sẽ dịch sang trái, nghĩa là led 1 trên chân RB1 sáng , lần lượt như vậy cho các led trên portB và lặp lại mãi mãi.

Code:

```
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#byte PORTB = 0x06

int16 count;
int8 a;
//Chuong trinh ngat TMR0
#int_timer0
void interrupt_timer0()
{
   set_timer0(6);
   ++count;
   if(count == 2000)   // 2000*500us = 500000us = 1s
      {
         count=0;
         rotate_left(&a,1);
      }
}
//Chuong trinh chinh
void main(void)
{
   set_tris_b(0);
   enable_interrupts(int_timer0);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);
   enable_interrupts(global);
   set_timer0(6);// T_dinhthi = 2*(256 - 6)*1us = 500us
```

```
    a = 0x01;

    while(true)
    {
        PORTB = a;
    }
}
```



Trích:

> Mình cũng chưa hiểu ý nghĩa của hàm WDT_..., ko biết có phải khai báo như trên thì sau khoảng thời gian ms bao nhiêu đó đặt sau WDT_ thì sẽ reset lại Pic ?????

WDT là "chó giữ nhà" (Watchdog Timer). Bộ phận này có nhiệm vụ reset lại PIC sau một khoảng thời gian định trước. WDT sẽ reset vi điều khiển khi bộ đếm của WDT bị tràn. Mục đích của nó là tránh trường hợp vi điều khiển bị "treo" khi phải hoạt động liên tục trong một khoảng thời gian lâu dài.

Thời gian định trước này phụ thuộc vào tần số loại thạch anh sử dụng và bộ chia tần số trước (prescaler) của WDT.

Ta thấy WDT chỉ liên quan đến Timer 0, còn các Timer khác không có liên quan. Đó là tại vì WDT có bộ chia tần số (prescaler) dùng chung với Timer 0. Lưu ý là muốn sử dụng WDT cần chú ý đến phần khai báo các "fuse" ở đầu chương trình.

Trích:

> rtcc_state là một trong những constant sau:
> RTCC_INTERNAL
> RTCC_EXT_L_TO_H
> RTCC_EXT_H_TO_L

Mỗi Timer đều có 2 tác dụng:

Tác dụng định thời: Timer sẽ dựa vào các xung tạo ra bởi bộ dao động (thạch anh, dao động RC, ...) cung cấp cho vi điều khiển để đếm. Và dựa vào tần số bộ dao động, giá trị các bộ chia tần số và giá trị của Timer, ta có thể xác định được thời gian thực. Như vậy trong

trường hợp muốn Timer hoạt động ở chế độ định thời, ta phải khai báo rtcc_state là "RTCC_INTERNAL" (xử dụng tần số dao động nội).

Tác dụng đếm: Timer sẽ dựa vào các xung lấy từ môi trường bên ngoài để đếm. Tùy theo Timer mà ta sử dụng chân lấy xung tương ứng (Timer 0 là chân RA4, Timer1 là chân RC0). Các xung này có tác dụng phản ánh các hiện tượng trong thực tế, và việc đếm các xung cũng đồng nghĩa với việc đếm các hiện tượng đó. Và để linh động hơn trong quá trình xử lí, Timer còn cho phép chọn cạnh tác động lên bộ đếm (chế độ này chỉ có ở Timer 0). Như vậy muốn Timer hoạt động ở chế độ đếm, ta phải khai báo rtcc_state là một trong 2 trường hợp còn lại (sử dụng dao động ngoài).

Trích:

ps_state là một trong những constant sau:
RTCC_DIV_2
RTCC_DIV_4
RTCC_DIV_8
RTCC_DIV_16
RTCC_DIV_32
RTCC_DIV_64
RTCC_DIV_128
RTCC_DIV_256
WDT_18MS
WDT_36MS
WDT_72MS
WDT_144MS
WDT_288MS
WDT_576MS
WDT_1152MS
WDT_2304MS

Ở đây có đến 2 hàm dùng để ấn định tỉ số chia của prescaler, một hàm là "RTCC_DIV_...", một hàm là "WDT_ ...". Đó là bởi vì Timer 0 và WDT dùng chung bộ chia tần số. Khi bộ chia được Timer 0 sử dụng thì WDT không đươc hỗ trợ với bộ chia này nữa. Như vậy sự khác biệt về thao tác giữa 2 hàm này có thể là như sau:

Hàm "RTCC_DIV_..." : cho phép Timer 0 sử dụng bộ chia tần số, không cho phép WDT sử dụng và ấn định tỉ số chia của nó.

Hàm "WDT_ ..." : cho phép WDT 0 sử dụng bộ chia tần số, không cho phép Timer 0 sử dụng và ấn định tỉ số chia của nó.

Trích:

T2_DISABLED
T2_DIV_BY_1
T2_DIV_BY_4
T2_DIV_BY_16
period là số nguyên từ 0-255, xác định giá trị xung reset
postscale là số nguyên 1-16, xác định reset bao nhiêu lần trước khi ngắt.
hôm nay 09:30 AM

Ta có thể nhận thấy là Timer 2 có đến 2 bộ chia tần số trước và sau, một bộ prescaler được đính kèm vào các chế độ hoạt động của Timer 2 (T2_DIV_BY_1, T2_DIV_BY_4,

T2_DIV_BY_16), một bộ là postscaler cis tỉ số chia từ 1:16. Như vậy nó cho phép việc lựa chọn tỉ số chia linh động hơn.

Timer 2 không hoạt động ở chế độ đếm. Chức năng của nó chủ yếu là tác động lên tốc độ baud cho MSSP thì phải. Không nhớ rõ lắm.

Trích:

> postscale là số nguyên 1-16, xác định reset bao nhiêu lần trước khi ngắt.

Cái này để mình coi lại đã, tại sao nó lại xác định reset bao nhiêu lần trước khi ngắt ??. Phải coi lại cái sơ đồ khối của Timer 2 mới biết được.

Một cách viết khác để tham khảo với hy vọng viết C sao cho dễ hiểu :-)

```
#include <16F877A.h>
#fuses NOWDT, PUT, XT, NOPROTECT
#use delay(clock=4000000)

#define INITIAL_VALUE 6

byte count;
byte led;

void change_led(void);

#int_timer0
void interrupt_timer0() {
set_timer0(INITIAL_VALUE);
count++;
if (count == 2000) {
count = 0;
change_led();
}
}
void main() {
set_tris_b(0);
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);
setup_timer_0(RTCC_INTERNAL | RTCC_DIV_2); // set mod
set_timer0(INITIAL_VALUE); // set initial value
count = 0;
led = 1;
while (true)
output_b(led);
}
void change_led() {
led = led << 1;
if (led == 0)
led = 1;
}
```

Thứ nhất khi nào hàm con interrupt_timer0() được gọi
Thứ hai, việc tính toán định thì 1s được tính như thế nào.
Thứ ba, biến a được khai báo là số nguyên 8bit. Phạm vi từ 0->255,làm sao bằng 256 đc.

1. Ngắt Timer0 được gọi khi Timer 0 bị tràn từ 0xff sang 0x00 với điều kiện phải có 2 khai cho phép ngắt timer 0 và ngắt toàn cục:
Code:

```
enable_interrupts(int_timer0);
enable_interrupts(global);
```

2. Việc tính toán thời gian tràn của Timer rất dễ, xem luồng "PIC6f877A từ dễ tới khó", hoặc tìm đâu đó trong diễn đàn này phần mềm "PIC Timer Calculator".
3. Biến a 8bit int, ko thể có giá trị 256 -> đúng vậy.

5.2. Ngắt ngoài

Chương trình ngắt đây

+Các bạn coi gium` mình có sai chỗ nào không mà mình delay không được: Mình muốn khi đóng RB4 thì LED sẽ nháp nháy với delay 50ms chẳng hạn. Phần ngắt chuyển chế độ thì mình làm được nhưng delay trong mỗi chế độ thì potay.

Code:

```
   #include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=1000000)
#use fast_io(b)
#byte portb=0x06
#byte intcon=0x000B
#bit RB4=portb.4
#bit RB5=portb.5
#bit RBIF=intcon.0 //dinh nghia co ngat RB
#bit RBIE=intcon.3 //dinh nghia bit cho phep ngat RB

// Chuong trinh ngat
#int_RB
void ngat_RB()
{
if((RBIF)&&(RBIE))
   {
      //Kiem tra sw1
      {
      if(RB4==0)
         {

         portb=0b00000001;

         delay_ms(200);
         portb=0b00001111;
         delay_ms(200);

         }
      }
      //Kiem tra sw2
      {
      if(RB5==0)
         {
         portb=0b00001000;
                                    }
      }
```

```
   RBIF=0; //Xoa co ngat RB
   }
}
// Chuong trinh chinh
main()
{
set_tris_b(0b11110000);
portb=0b11110000;
enable_interrupts(global);
enable_interrupts(int_RB);
ext_int_edge(H_to_L);
while(true)
   {

   }
}
```

TL:

Code:

```
   #include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#use fast_io(b)
#byte portb=0x06
#byte intcon=0x000B
#bit RB4=portb.4
#bit RB5=portb.5
#bit RBIF=intcon.0 //dinh nghia co ngat RB
#bit RBIE=intcon.3 //dinh nghia bit cho phep ngat RB

// Chuong trinh ngat
#int_RB
void ngat_RB()
{
if((RBIF)&&(RBIE))
   {
      //Kiem tra sw1
      {
      if(RB4==0)
         {

         portb=0b00000001;

         delay_ms(50);
         portb=0b00001111;
         delay_ms(50);

         }
      }
      //Kiem tra sw2
      {
      if(RB5==0)
         {
         portb=0b00001000;
                                    }
      }

   RBIF=0; //Xoa co ngat RB
```

```
   }
}
// Chuong trinh chinh
main()
{
set_tris_b(0b11110000);
portb=0b11110000;
enable_interrupts(global);
enable_interrupts(int_RB);
ext_int_edge(H_to_L);
while(true)
   {
   }
}
```

Đã sửa lại cho phù hợp file mô phỏng của bạn. Với code trên, kết thúc ngắt tất nhiên ko còn delay nữa. 50ms hơi ít, tăng lên 1000ms, thấy kết quả.

+Chào cả nhà !

Sao không thấy bác nào post bài vào luồng này vậy kà !Trầm quá...!Trầm quá...!Hay cái CCS C này không hấp dẫn mọi người chăng!

Không ai viết gì, tớ vẫn post cho nó đỡ trầm....!

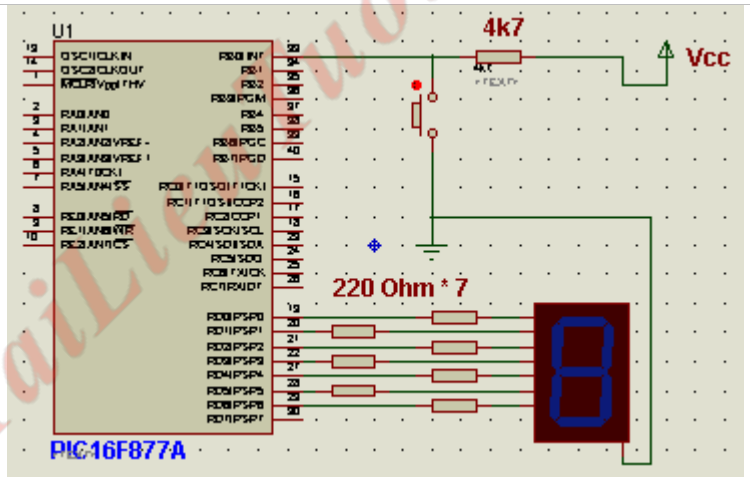
Đã ví dụ về ngắt Timer, sau đây là 2 ví dụ về ngắt ngoài trên chân RB0 và trên các chân RB4 đến RB7:

Chương trình sau dùng ngắt ngoài trên RB0 đếm số lần cái button được nhấn xuống, hiển thị lên led 7 đoạn (common cathode).Nếu số lần nhấn vượt quá 9, chương trình sẽ quay về hiển thị lên led từ số 1.

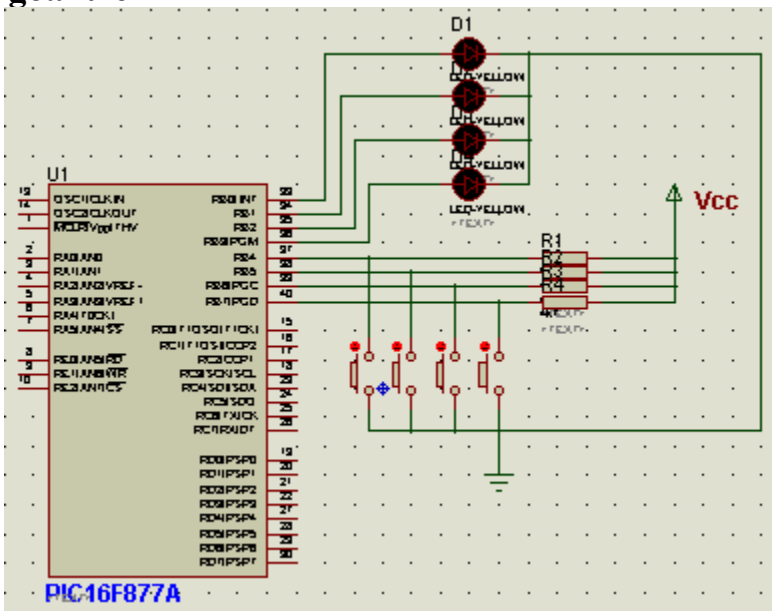
Code:

```
//***************************************************************************
// Author  : nhh
// Date    : 03/04/06
// Hardware: PIC16F877A
//***************************************************************************
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#use fast_io(b)
#use fast_io(d)
#byte portb=0x06
#byte portd=0x08
const unsigned char digital[]={0b00000110, 0b01011011, 0b01001111,
0b01100110,\
                                                0b01101101, 0b01111101,
0b00000111, 0b01111111, 0b01101111};
                  // ma hoa digital duoi dang mang
// Chuong trinh ngat
#int_ext
void ngat_RB0()
{
int i;
   if(i<9)
      {
      portd=digital[i];
      ++i;
      }
   if(i==9)
      {
      i=0;
```

```
      }
}
// Chuong trinh chinh
main()
{
set_tris_b(0b00000001);
set_tris_d(0);
enable_interrupts(global);
enable_interrupts(int_ext);
ext_int_edge(H_to_L);
portd=0b00111111;
while(true)
   {
   // chi doi ngat nen vong lap nay ko co gi ca !
   }
}
```



5.3. Ngắt ngoài trên RB4-RB7



Còn đây là ứng dụng ngắt ngoài trên RB4 đến RB7 để thay đổi kiểu cũng như tốc độ chớp nháy mấy con led chỉ để....ngắm cho vui mắt !

Ấn sw1, led1 nhấp nháy với delay 250ms

Ấn sw2, led1,2 nhấp nháy với delay 200ms

Ấn sw3, led1,2,3 nhấp nháy với delay 150ms

Ấn sw4, led1,2,3,4 nhấp nháy với delay 100ms

Thang8831

http://www.picvietnam.com

Code:

```
//************************************************************************
// Author  : nhh
// Date    : 03/04/06
// Hardware: PIC16F877A
//************************************************************************
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#use fast_io(b)
#byte portb=0x06
#byte intcon=0x000B
#bit RB4=portb.4
#bit RB5=portb.5
#bit RB6=portb.6
#bit RB7=portb.7
#bit RBIF=intcon.0 //dinh nghia co ngat RB
#bit RBIE=intcon.3 //dinh nghia bit cho phep ngat RB
int led=0,speed;
// Chuong trinh ngat
#int_RB
void ngat_RB()
{
if((RBIF)&&(RBIE))
   {
      //Kiem tra sw1
      {
      if(RB4==0)
         {
         led=0b00000001; //led1 sang
         speed=250;
         }
      }
      //Kiem tra sw2
      {
      if(RB5==0)
         {
         led=0b00000011; //led1,2 sang
         speed=200;
         }
      }
      //Kiem tra sw3
      {
      if(RB6==0)
         {
         led=0b00000111; //led1,2,3 sang
         speed=150;
         }
      }
      //Kiem tra sw4
      {
      if(RB7==0)
         {
         led=0b00001111; //led1,2,3,4 sang
         speed=100;
         }
      }
   RBIF=0; //Xoa co ngat RB
   }
}
```

```
// Chuong trinh chinh
main()
{
set_tris_b(0b11110000);
portb=0b00001111;
enable_interrupts(global);
enable_interrupts(int_RB);
ext_int_edge(H_to_L);
while(true)
   {
   portb=led;
   delay_ms(speed);
   portb=0;
   delay_ms(speed);
   }
}
```

[quote=nhh;2261]Còn đây là ứng dụng ngắt ngoài trên RB4 đến RB7 để thay đổi kiểu cũng như tốc độ chớp nháy mấy con led chỉ để....ngắm cho vui mắt !

Ấn sw1, led1 nhấp nháy với delay 250ms
Ấn sw2, led1,2 nhấp nháy với delay 200ms
Ấn sw3, led1,2,3 nhấp nháy với delay 150ms
Ấn sw4, led1,2,3,4 nhấp nháy với delay 100ms
[code]//**********
****
// Author : nhh
// Date : 03/04/06
// Hardware: PIC16F877A
//**********
****

Không như trong MPLAB, đã định nghĩa sẵng các thanh ghi và bit tương ứng, còn CCS C chỉ định nghĩa chân PIC, những thanh ghi, những bit mà CCS C cho là cần thiết, ta xem trong file PIC16F887A.h, thanh ghi, bit nào chưa định nghĩa mà muốn sử dụng thì phải định nghĩa nó. Ta có thể viết riêng 1 file.h loại này sao cho mình dễ nhớ nhất, đến khi muốn sử dụng chỉ cần khai báo #include<file.h> vào là xài thôi!

Em mới vô thôi, huynh giải thích mấy dòng code sau cho em được ko ?
" set_timer0(6) " ; " ++count; " ; "if (count==2000) "

Nó nằm trong ct nháy led dùng interrup và timer0. biến count và a có tác dụng gì ? tại sao phải lùi a "a=a<<1" ?

- set_timer0(6); đây là phương thức gọi hàm. set_timer0() là một hàm, 6 là tham số. Còn hàm này làm gì thì bạn phải tự tìm hiểu lấy.
  * ++count; là thực hiện tăng biến count lên 1 đơn vị. Nó giống lệnh count=count+1. Còn có một cú pháp nhữa là count++. Hai lệnh lệnh này có sự khác nhau về thứ tự thực hiện khi nằm trong một biểu thức so sánh.
  * if(count==2000) là phép so sánh. Nếu giá trị của biến count bằng 2000 thì kết quả của phép so sánh là True ngược lại là False.
  * a là .. mình chịu thua!?!? Còn dịch trái là để nhân đôi giá trị biết. Bạn hãy xem lại đại số bool có rất nhiều điều hay ở đó.

+Nhân tiện anh cho em hỏi thêm : giả sử em viết một ct nháy led (cho nó chạy như vòng lặp), đầu tiên kt các khóa để biết ct sẽ nháy led kiểu nào. sau đó thì ct sẽ thực hiện lệnh nháy đèn led, nếu như sau khi ta bấm nút chọn kiểu nháy, ct sẽ tiếp tục thực hiện lệnh tiếp

theo mà ta đổi ý bấm nút khác để đổi kiểu thì ct sẽ không nhận vì đang thực hiện các lệnh bật-tắt led. Bài tập này hình như có ở trang 2, 3 dùng interrup và timer0, Tại sao nhỉ ? Nếu bạn sử dụng vòng lặp thì nút bấm sẽ ko được nhận trong khi vòng lặp đang thực hiện.
Cách giải quyết bài toán như sau:
_ Bạn hãy khai báo 1 biến toàn cục. Biến này lưu kiểu chớp. Hàm chớp sẽ dựa vào giá trị biến này mà đổi kiểu chớp.
_ Sử dụng timer0 làm thời gian chuẩn để trì hoãn chớp tắt (vì nếu dùng làm thời gian trì hoãn chớp tắt thì nó quá ngắn ta không thể thấy chớp được).
_ Khai báo một biến toàn cục làm hệ số cho thời gian trì hoãn. Trong ngắt Timer0 bạn sẽ đếm biến này đến một giá trị nào đó thì gọi hàm chớp tắt.
_ Bạn cũng cần khai báo một biến toàn cục để cho hàm chớp tắt biết mình phải làm gì. Hoặc nếu ko bạn có thể truyền tham số cho hàm.
_ Trong chương trình main, bạn dùng vòng lặp để bắt phím nhấn (hoặc dùng ngắt) và thực hiện thay đổi giá trị của biến kiểu chớp.
Trên đây là giải thuật, đó là phương pháp để giải quyết bài toán. Tất nhiên trong quá trình viết bạn có thể khai báo thêm các biến toàn cục hay cục bộ để thực hiện thuật toán. Tốt nhất là bạn nên viết trước và gặp khó khăn thì đưa cả code lên để mọi người giúp bạn giải quyết.

Em mượn tạm đoạn code của nhh anh mổ xẻ nó giùm em nhé

```
//************************************************ **
// Author : nhh
// Date : 02/04/06
// Hardware: PIC16F877A
//************************************************ **
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#use fast_io(b)
#byte portb=0x06
#define led pin_B0
int16 count;
int8 a;
//Chuong trinh ngat TMR0
#int_timer0
void interrupt_timer0()
{
set_timer0(6);
++count;
if(count==2000)
{
count=0;
a=a<<1; // dich trai a 1bit
}
if(a==256)
{
a=1;
count=0;
}
}
```

```
//Chuong trinh chinh
main()
{
set_tris_b(0);
enable_interrupts(global);
enable_interrupts(int_timer0);
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);
set_timer0(6);
count=0;
a=1;
while(true)
{
portb=a;
}
}
```

Anh giải thích mấy cỗ có giùm em. Tại sao phải setup_timer_0(RTCC_INTERNAL| RTCC_DIV_2); mà ko Div 3, 4,... hở anh ?

*Dấu thứ nhất:

[CODE:]
a=a<<1; // dich trai a 1bit
[/code]
Như tác giả đã chú thích đó là lệnh dịch trái 1 bit.
VD: trước khi dịch, a có giá trị 0 0 0 0 0 0 0 1 (0x01) thì sau lệnh dịch này giá trị biến a sẽ là 0 0 0 0 0 0 1 0 (0x02).
Vậy lệnh dịch trái sẽ làm tăng giá trị biến bị dịch lên 2 lần: 2 thành 4. Giống như bạn dịch trong hệ thập phân số 0500 thì được 5000 tức tăng 10 lần. Hệ nhị phân (2 số) dịch trái 1 bit sẽ tăng giá trị 2 lần, hệ thập phân (10 số) dịch trái một bit, số sẽ tăng giá trị 10 lần.
Vậy nhiều lệnh dịch sẽ làm cho a thay đổi như sau
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0
* Dấu thứ 2:
[CODE:]
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);
[/code]
mode may be one or two of the constants defined in the devices .h file.
RTCC_INTERNAL, RTCC_EXT_L_TO_H or RTCC_EXT_H_TO_L
RTCC_DIV_2, RTCC_DIV_4, RTCC_DIV_8, RTCC_DIV_16, RTCC_DIV_32, RTCC_DIV_64, RTCC_DIV_128, RTCC_DIV_256
Bạn hãy đọc Help và các ví dụ của nó thì bạn sẽ hiểu được cách tính.
Bạn hãy tìm hiểu kỹ về bộ chia tần trong DataSheet của chip sẽ hiểu tại sao chỉ làm 2 mũ n.
* Dấu thứ 3:

[CODE:]
portb=a;
[/code]
Đây là lệnh xuất giá trị biến a ra PortB. a có 8 bit, PortB có 8 chân B7 đến B0.
Lệnh trên sẽ áp các bit của a vào PortB theo đúng trọng số.

Tôi copy nguyen bai delay1s_RB0 của bác về nạp thử, nhưng không hiểu sao nó không chạy, mà phải cắm chân RB3 (chân PGM) xuống đất thì nó mới chạy, bác có thể cho tôi biết tại sao không?
Cám ơn bác!

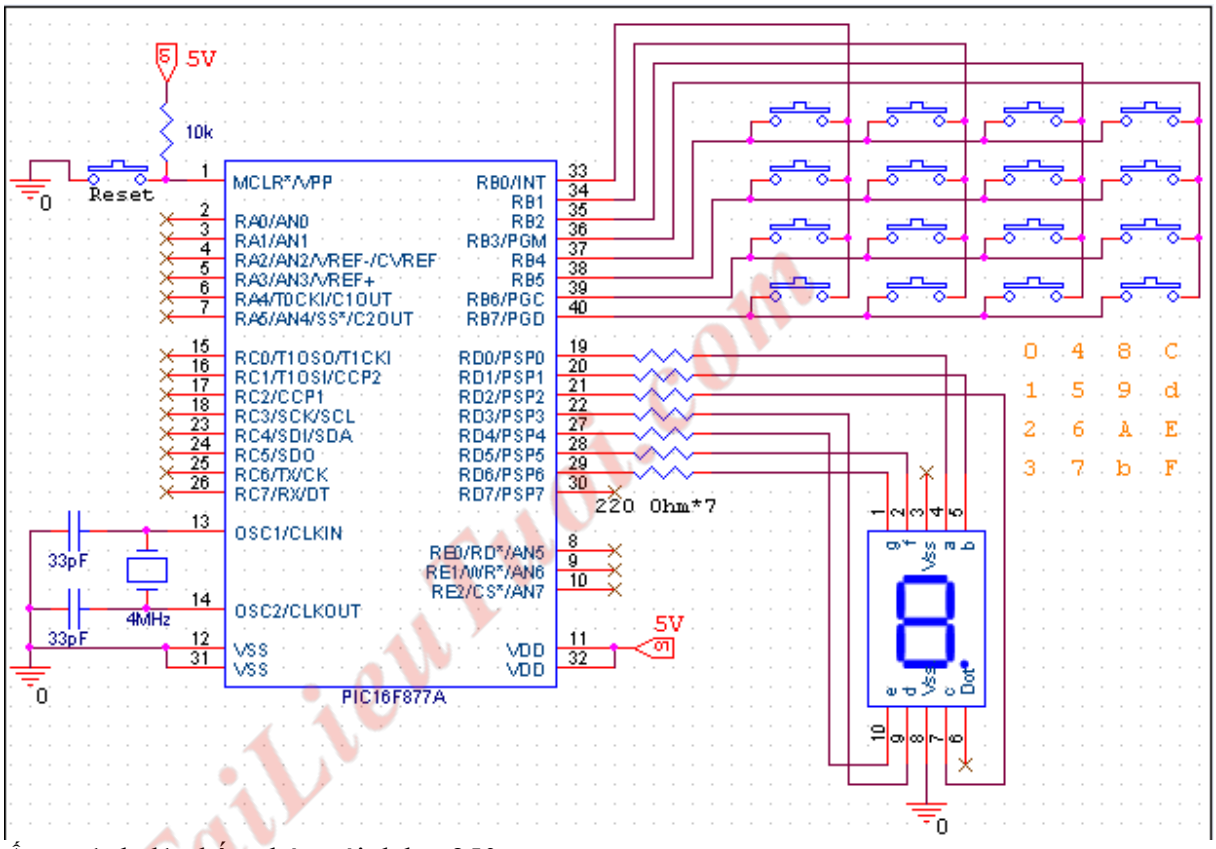
Ở một số chip, chân RB3 còn là chân PGM dùng để cấp nguồn cho chế độ nạp chip điện áp thấp (Low-Voltage Programming). Nếu chip của bạn còn mới, có thể nó đang được bật chế độ LVP (mặc định khi xuất xưởng), do đó bạn để hở RB3 sẽ không cho phép chip chạy chương trình, và bạn phải nối RB3 xuống 0V (nên nối qua một điện trở khoảng 1k) thì chip mới không đi vào chế độ LVP và mới chạy được chương trình đã có trong chip.

Bây giờ thì tôi đã hiểu, nhưng khi phải nối mass chân RB3 rồi thì coi như chân đó không thể dùng được nữa hả?

Bạn vẫn có thể dùng chân RB3 như bình thường khi chip đã ở chế độ chạy chương trình bình thường. Nhưng nếu bạn nối thẳng RB3 vào mức 0V và sau đó dùng RB3 như ngõ ra, thì khi bạn xuất mức cao ra RB3 sẽ gây ngắn mạch giữa các chân Vdd và Vss (vì bạn đã nối thẳng RB3 vào Vss, và sau đó lại yêu cầu ngõ ra RB3 được nối lên Vdd trong chương trình). Đó chính là lý do của lời khuyên nên nối RB3 xuống mức 0V thông qua một điện trở khoảng 1k, nếu bạn muốn RB3 khi dùng như ngõ ra được nhẹ tải hơn thì có thể dùng giá trị 4.7k hay 10k cho điện trở kéo xuống 0V đó.

5.4. Giải mã bàn phím

Mạch quét 16 phím, hiện kết quả lên led 7 đoạn.



Ấn sw1, led1 nhấp nháy với delay 250ms
Ấn sw2, led1,2 nhấp nháy với delay 200ms
Ấn sw3, led1,2,3 nhấp nháy với delay 150ms
Ấn sw4, led1,2,3,4 nhấp nháy với delay 100ms
Code:

```
 //*************************************************************************
*
// Author  : nhh
// Date    : 03/04/06
// Hardware: PIC16F877A
//*************************************************************************
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=4000000)
#use fast_io(b)
#use fast_io(c)
#byte portb = 0x06
#byte portc = 0x07
#bit RB0 = 0x06.0
#bit RB1 = 0x06.1
#bit RB2 = 0x06.2
#bit RB3 = 0x06.3
#bit RB4 = 0x06.4
#bit RB5 = 0x06.5
#bit RB6 = 0x06.6
#bit RB7 = 0x06.7
#bit RBIF=intcon.0 //dinh nghia co ngat RB
#bit RBIE=intcon.3 //dinh nghia bit cho phep ngat RB
int a;
const unsigned char dig[]={0b00111111,0b00000110, 0b01011011,0b01001111,\
```

```
0b01100110,0b01101101,0b01111101,0b00000111,0b01111111,0b01101111,0b0111011
1,\
0b01111100,0b00111001,0b01011110,0b11111001,0b11110001};
                  // ma hoa digital duoi dang mang
// Chuong trinh ngat
#int_RB
void ngat_RB()
{
if((RBIF)&&(RBIE))
   {
          {
          if(RB4&&RB0)
          a=dig[0];
          }
          {
          if(RB4&&RB1)
          a=dig[4];
          }
          {
          if(RB4&&RB2)
          a=dig[8];
          }
          {
          if(RB4&&RB3)
          a=dig[12];
          }
//.......
          {
          if(RB5&&RB0)
          a=dig[1];
          }
          {
          if(RB5&&RB1)
          a=dig[5];
          }
          {
          if(RB5&&RB2)
          a=dig[9];
          }
          {
          if(RB5&&RB3)
          a=dig[13];
          }
//........
           {
          if(RB6&&RB0)
          a=dig[2];
          }
          {
          if(RB6&&RB1)
          a=dig[6];
          }
          {
          if(RB6&&RB2)
          a=dig[10];
          }
          {
          if(RB6&&RB3)
          a=dig[14];
          }
```

```
//........
          {
         if(RB7&&RB0)
         a=dig[3];
         }
         {
         if(RB7&&RB1)
         a=dig[7];
         }
         {
         if(RB7&&RB2)
         a=dig[11];
         }
         {
         if(RB7&&RB3)
         a=dig[15];
         }
   RBIF=0; //Xoa co ngat RB
   }
}
// Chuong trinh chinh
main()
{
set_tris_b(0b11110000);
set_tris_c(0);
enable_interrupts(global);
enable_interrupts(int_RB);
ext_int_edge(H_to_L);
portb=0;
portc=0;
while(true)
   {
   portb=1;
   portb=2;
   portb=4;
   portb=8;
   portc=a;
   }
}
```


anh có thể nói rõ hơn được không a? số: 1,2,4,6,8 nó mặc định cho cổng đấy a?

- Xuất 1 ra PORTA thì chỉ có bit 0 (tức là chân RA0) ở mức 1, các bit (chân) khác là 0.
- Xuất 2 ra PORTA thì chỉ có bit 1 (tức là chân RA1) ở mức 1, các bit (chân) khác là 0.
- Xuất 4 ra PORTA thì chỉ có bit 2 (tức là chân RA2) ở mức 1, các bit (chân) khác là 0.
- Xuất 8 ra PORTA thì chỉ có bit 3 (tức là chân RA3) ở mức 1, các bit (chân) khác là 0.

Bạn nhìn vào sơ đồ mạch của nhh sẽ thấy các chân RA0 .. RA3 nối với các cột của bàn phím.

TL: Không như trong MPLAB, đã định nghĩa sẵng các thanh ghi và bit tương ứng, còn CCS C chỉ định nghĩa chân PIC, những thanh ghi, những bit mà CCS C cho là cần thiết, ta xem trong file PIC16F887A.h, thanh ghi, bit nào chưa định nghĩa mà muốn sử dụng thì phải định nghĩa nó.Ta có thể viết riêng 1 file.h loại này sao cho mình dễ nhớ nhất, đến khi muốn sử dụng chỉ cần khai báo #include<file.h> vào là xài thôi!

#fuses NOWDT,PUT,HS,NOPROTECT

#use delay(clock=4000000)

Bạn chú ý, clock = 4MHz là chế đọ dao động XT chứ không phải HS.

Thang8831

http://www.picvietnam.com

Bài : Giải mã bàn phím
Mạch quét 16 phím, hiện kết quả lên led 7 đoạn.
Trong chuong trinh thieu dinh nghia
#byte intcon=0x000B
và port xuất ra led 7 đoạn là PORTC không phải portd như hình vẽ.
+Chào bạn mình thay led 7 đoạn bằng LCD thì chương trình chạy không đúng, bạn có thể hướng dẫn cho mình dùng ngắt để giải mã phím xuất ra lcd được không, cám ơn bạn nhiều. LCD và led 7 đoạn tất nhiên là khác nhau rồi.
Để thực hiện tốt giải mã matrix phím, bác phải có giải pháp chống nhiễu (run phím) bằng phần cứng, hoặc phần mềm, thông thường là dùng phần mềm. Code bên trên chưa có chống nhiễu

5.5. Chương trình gửi ký tự ra 2x16 LCD dùng CCS C

Chương trình gửi ký tự ra 2x16 LCD dùng CCS C

```
#include "16F877A.h" // PIC16F877A header file
#use delay(clock=4000000) // for 4Mhz crystal
#fuses XT, NOWDT, NOPROTECT, NOLVP // for debug mode

#define WRITE_DATA 0
#define WRITE_COMMAND 1

#define NCHAR_PER_LINE 16 // max char numbers per line
#define MS10 10 // 10 milliseconds
#define US400 400 // 400 microseconds

#define LCD_RS PIN_A1
#define LCD_RW PIN_A2
#define LCD_E PIN_A3
////////////////////////////////////////////////////////
//
/* private */ void lcd_write(byte dat, int1 option) {
delay_us(US400);
if (option == WRITE_DATA)
output_high(LCD_RS);
else // option == WRITE_COMMAND
output_low(LCD_RS);
output_low(LCD_RW);
output_b(dat);

output_high(LCD_E);
delay_us(US400);
output_low(LCD_E);
}
////////////////////////////////////////////////////////
//
void lcd_init(void) {
output_low(LCD_E); // Let LCD E line low

lcd_write(0x38, WRITE_COMMAND); // Set LCD 16x2, 5x7, 8bits data
delay_ms(15);
```

```
lcd_write(0x01, WRITE_COMMAND); // Clear LCD display
delay_ms(MS10);
lcd_write(0x0f, WRITE_COMMAND); // Open display & current
delay_ms(MS10);
lcd_write(0x06, WRITE_COMMAND); // Window fixed (Character Entry Mode?)
delay_ms(MS10);
}
//////////////////////////////////////////////////////////
//
void lcd_display_char(int8 line, int8 pos, char ch) {
line = (line == 0) ? 0 : 1;
pos = (pos > NCHAR_PER_LINE) ? NCHAR_PER_LINE : pos;

lcd_write(0x80 + 0x40 * line + pos, WRITE_COMMAND);
lcd_write(ch, WRITE_DATA);
}
//////////////////////////////////////////////////////////
void lcd_display_str(int8 line, char str[], int8 nchars) {
int8 i;
for (i = 0; i < nchars; i++)
lcd_display_char(line, i, str[i]);
}
//////////////////////////////////////////////////////////
/**
* Display characters to a 2x16 LCD
*
* (1) LCD1 to GND
* (2) LCD2 to VDD 5 volts
* (3) LCD4 (RS) - LCD5 (RW) - LCD6 (E) to A1, A2, A3
* (4) LCD7-LCD14 to B0-B7 (bus data)
*
* Ref: http://pic16.com/bbs/dispbbs.asp?boa...ID=5879&page=1
*/
void main(void) {
int8 i;
char LINE1[] = { "SGN Tech" };
char LINE2[] = { "Xin chao" };

lcd_init();

// use of lcd_display_char()
for (i = 0; i < 8; i++)
lcd_display_char(0, i, LINE1[i]);

// use of lcd_display_str
lcd_display_str(1, LINE2, 8);
}
```

+CCS C có một ví dụ hay hơn: Chỉ cần dùng 4 bits D4-D7 của LCD:
Examples\EX_LCDKB.C

5.7. Ví dụ nhỏ về ngắt ngoài

Thang8831
http://www.picvietnam.com

Bình thường thì LED6 sáng, LED7 tối. Khi nhấn phím, LED6 tối, LED7 sáng trong vòng 0,5 giây, rồi trở về trạng thái ban đầu (LED6 sáng, LED7 tối)

```
#include <16F877A.h>
#fuses NOWDT, XT
#use delay(clock=4000000)

void high_b6_low_b7() {
output_high(PIN_B6);
output_low(PIN_B7);
}

void low_b6_high_b7() {
output_low(PIN_B6);
output_high(PIN_B7);
}
////////////////////////////////////////////////////////
#INT_EXT
void RB0_handler() {
low_b6_high_b7();
delay_ms(500);
high_b6_low_b7();
}
////////////////////////////////////////////////////////
/**
* Keep B6 on and B7 off. Pressing the button causes interrupt:
* B6 off and B7 on, delay half second, then B6 on and B7 off
*
* Wiring (TM Board)
* (1) PIC's B0 to Matrix Key R0
* Matrix Key C0 to GND
* (2) PIC's B6-B7 to LED6-LED7
*
* Ref: Interrupt Mechanism
* http://www.mikroelektronika.co.yu/en...sicbook/06.htm
*/
void main() {
enable_interrupts(GLOBAL); // enable all interrupts
enable_interrupts(INT_EXT); // enable external interrupt from pin RB0/INT

high_b6_low_b7();
while (true) {
// do nothing
}
}
```

5.8. Ngắt ngoài và đèn 7 đoạn

Một phương án khác:

```
#include <16F877A.h>
#fuses NOWDT, XT
```

```
#fuses NOLVP // important
#use delay(clock=4000000)

// 0 1 2 3 4 5 6 7 8 9
byte const DIGITS[] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f };
int8 i = 0;
////////////////////////////////////////////////////////
/* private */void off_on_led_transistor() {
output_low(PIN_D1);
delay_ms(1);
output_high(PIN_D1);
}
////////////////////////////////////////////////////////
/* private */void display(int8 digit) {
output_c(DIGITS[digit] ^ 0xff);
off_on_led_transistor();
}
////////////////////////////////////////////////////////
#INT_EXT
void ngat_RB0() {
i = (i < 9) ? i+1 : 1;
delay_ms(200); // switch debounce period
}
////////////////////////////////////////////////////////
/**
* Count number of key presses and display it on a 7-segment LED.
* If the number is 9, the next count will be 1
*
* Wiring (TM Board)
* (1) PIC's B0 to R0
* Matrix Key C0 to GND
* (2) PIC's C0-C6 to 7-segment LED's A-G
* PIC's D1 to 7-segment LED's C2
*/
void main() {
enable_interrupts(GLOBAL);
enable_interrupts(INT_EXT);

while (true)
display(i);
}
```

5.9. Chương trình hiển thị phím số ra đèn 7 đoạn (không dùng interrupt)

Code:

```
////////////////////////////////////////////////////////
/* private */void off_on_led_transistor() {
output_low(PIN_D1);
delay_ms(1);
output_high(PIN_D1);
}
```

```
////////////////////////////////////////////////////////
void display(int8 digit) {
output_c(DIGITS[digit] ^ 0xff);
off_on_led_transistor();
}
////////////////////////////////////////////////////////
int8 char_to_digit(char c) {
return c & 0b00001111; // first 4 bits only
}
////////////////////////////////////////////////////////
int1 digit_key_pressed(char key) {
byte pattern;
pattern = 0b00110000;
return (key & pattern) == pattern;
}
////////////////////////////////////////////////////////
/**
* Echo digit-key presses (0 to 9) of a 4x3 keypad to the 7-segment LED
*
* Configuration:
* Use PORTB for keypad by uncommenting the following line in
PICC\Drivers\KBDD.c
* #define use_portb_kbd TRUE
*
* Wiring: (TM Board)
* (1) PIC's B1-B7 to Matrix Keypad's R3-R0&C2-C0 (notice the reverse order)
* (2) PIC's C0-C6 to 7-segment LED's A-G
* PIC's D1 to 7-segment LED's C2
*/
void main() {
int8 i, digit;
char key;

kbd_init();
while (true) {
key = kbd_getc();
if (digit_key_pressed(key)) {
digit = char_to_digit(key);
for (i = 0; i < 200; i++) // repeat the display for human eyes
display(digit);
}
}
}
```


5.10. Chương trình hiển thị phím số ra đèn 7 đoạn (DÙNG INTERRUPT)

Chương trình hiển thị phím số trên bàn phím 4x3 ra đèn 7 đoạn (DÙNG INTERRUPT).

```
#include <16F877A.h>
#fuses NOWDT, XT
#fuses NOLVP // important
#use delay(clock=4000000)

#include <kbd.c> // in PICC\Drivers

// 0 1 2 3 4 5 6 7 8 9
byte const DIGITS[] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f };
```

```
////////////////////////////////////////////////////////
/* private */void off_on_led_transistor() {
output_low(PIN_D1);
delay_ms(1);
output_high(PIN_D1);
}
////////////////////////////////////////////////////////
void display(int8 digit) {
output_c(DIGITS[digit] ^ 0xff);
off_on_led_transistor();
}
////////////////////////////////////////////////////////
int8 char_to_digit(char c) {
return c & 0b00001111; // first 4 bits only
}
////////////////////////////////////////////////////////
int1 digit_key_pressed(char key) {
byte pattern;
pattern = 0b00110000;
return (key & pattern) == pattern;
}
////////////////////////////////////////////////////////
#INT_RB
void RB_handler() {
int8 i, digit;
char key;

key = kbd_getc();
if (digit_key_pressed(key)) {
digit = char_to_digit(key);
for (i = 0; i < 200; i++) // repeat the display for human eyes
display(digit);
}
}
////////////////////////////////////////////////////////
/**
* Echo digit-key presses (0 to 9) of a 4x3 keypad to the 7-segment LED
*
* Configuration:
* Use PORTB for keypad by uncommenting the following line in PICC\Drivers\KBDD.c
* #define use_portb_kbd TRUE
*
* Wiring: (TM Board)
* (1) PIC's B1-B7 to Matrix Keypad's R3-R0&C2-C0 (notice the reverse order)
* (2) PIC's C0-C6 to 7-segment LED's A-G
* PIC's D1 to 7-segment LED's C2
*/
void main() {
enable_interrupts(GLOBAL);
enable_interrupts(INT_RB);
```

```
kbd_init();
while (true) {
// do nothing
}
}
```

Đoạn mã hiển thị led 7 thanh này, dấu ^ có ý nghĩa là vì hả các bác

```
void display(int8 digit) {
output_c(DIGITS[digit] ^ 0xff);
off_on_led_transistor();
}
```

Thân.

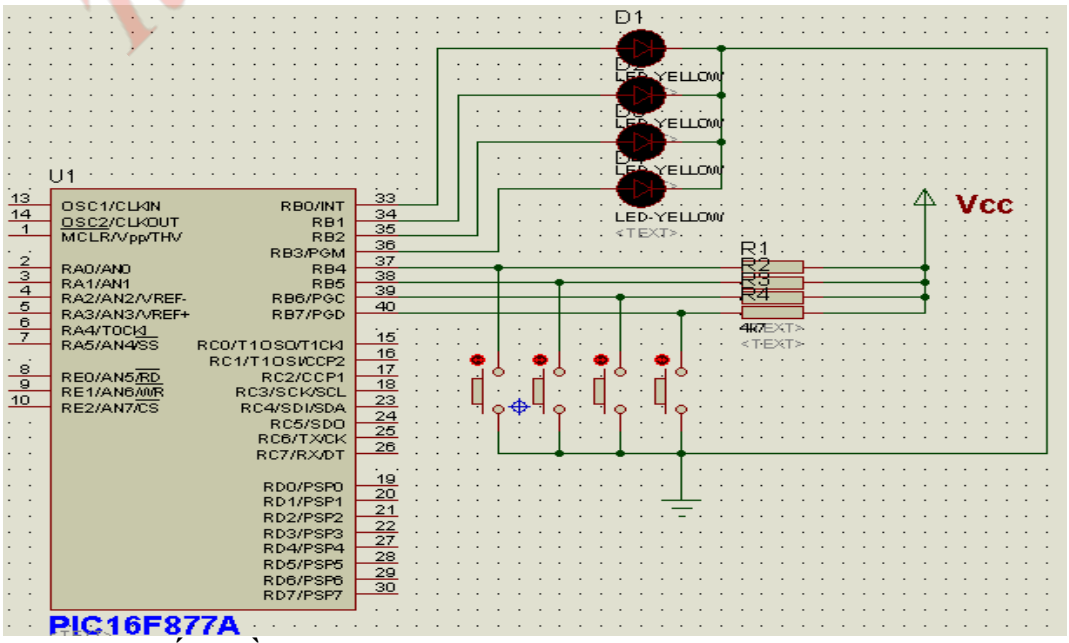
Dấu "^" là ký hiệu phép toán XOR (exclusive OR) trong C.
Phép toán && là phép **and logic**.Chắc là không phải giải thích về phép toán này.

5.11. Thay đổi tốc độ đèn led dung ngắt

Code:

```
//////////////////////////////////////////////////////////////////////////
// Author  : nhh
// Date    : 03/04/06
// Hardware: PIC16F877A
//////////////////////////////////////////////////////////////////////////
#include <16F877A.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=1000000)
#use fast_io(b)
#byte portb=0x06
#byte intcon=0x000B
#bit RB4=portb.4
#bit RB5=portb.5
#bit RBIF=intcon.0 //dinh nghia co ngat RB
#bit RBIE=intcon.3 //dinh nghia bit cho phep ngat RB

// Chuong trinh ngat
#int_RB
void ngat_RB()
{
if((RBIF)&&(RBIE))
  {
    //Kiem tra sw1
    {
    if(RB4==0)
      {

      portb=0b00000001;

      delay_ms(200);
      portb=0b00001111;
      delay_ms(200);

      }
```

```
    }
    //Kiem tra sw2
    {
    if(RB5==0)
      {
      portb=0b00001000; //led1,2 sang
                        }
    }

  RBIF=0; //Xoa co ngat RB
  }
}
// Chuong trinh chinh
main()
{
set_tris_b(0b11110000);
portb=0b11110000;
enable_interrupts(global);
enable_interrupts(int_RB);
ext_int_edge(H_to_L);
while(true)
  {

  }
}
```

Em ko dùng ngắt nhiều nên hỏi có vẽ ngu ngơ, xin thông cảm.....

hai đoạn khai báo biến dưới đây để làm gì em không hiểu????

#byte portb=0x06

#byte intcon=0x000B

Vì trong chương trình chính đã định ngõ vào ra của port B rồi, với lại cũng đã cho ngắt toàn cục rồi mà.Tại sao phải nhất thiết là 0x06 và 0x000B mà ko là giá trị khác////Em thử gán giá trị khác thì ct chạy sai...Giúp với

Thang8831

http://www.picvietnam.com

TL: Đây là khai báo địa chỉ của thanh ghi portB và thanh ghi intcon. Thường thì mỗi thanh ghi có một địa chỉ, giống như số nhà ấy, bạn khai báo nhầm địa chỉ của nó, nghĩa là bạn vào nhầm nhà rồi còn gì.
Với 16F877A
PORTB : địa chỉ 06h
INTCON : địa chỉ 0Bh
Vì trong file 16F877A.h của CCS C không có khai báo tường tận như trong các file .inc của Microchip, muốn sử dụng cho tiện thì khai báo thêm vào. Bạn mở file .h của con pic đang làm việc ra xem người ta đã khai báo những gì rồi.
+Sau khi vào trang Web của CCSC để tìm hiểu về vấn đề này mình lĩnh hội được một vài điều như sau:
- Thứ nhất, một thiết kế sẽ là không tối ưu nếu trong CTC ngắt lại gọi đến một hàm khác đã được sử dụng trong Main hay hàm khác, đặc biệt là khi bản thân hàm này lại gọi đến hàm khác nữa...(những hàm chứa trong các header thường gọi lẫn nhau như vậy).
- Thứ hai nếu sử dụng hàm như nêu trên thì chương trình chạy sẽ không theo đúng ý đồ lập trình (chẳng hạn như trong chương trình mà mình vừa Pót). Điều này xảy ra là do bộ nhớ Stack bị tràn (stack của PIC chỉ có 8 mức) khi gọi hàm chồng chéo (cả hàm Main và CTC ngắt đều gọi...).
- Thứ ba bên CCSC khuyên nếu buộc phải gọi hàm như vậy (chẳng hạn như hàm delay_ms như trên) thì hãy khai báo ở cả hàm Main và hàm ngắt ???.
TL Điều này thì mình không rõ vì đã thử nhưng không có hiệu quả.
Vấn đề của bạn như có thể giải thích một cách dễ hiểu như sau:
Nếu cả 2 hàm đều được gọi trong cả hàm ngắt và hàm main thì sẽ phát sinh ra lỗi. Tại sao lại thế. Các bạn thử suy nghĩ mà xem. khi hàm main đang chạy đến hàm mà nó và hàm ngắt cùng gọi. Nếu không có ngắt xảy ra cùng thời điểm đó thì không có vấn đề gì cả. nhưng nếu có ngắt thì nó phải lưu các thông số hiện tại và nhảy vào ngắt, và khi nó nhảy ra khỏi ngắt thì các dữ liệu đã lưu sẽ bị chồng lên trong khi thực hiện hàm đó trong ngắt. vậy chương trình sẽ không đúng nữa.
Trong Keil C của 89 thì nó chỉ là warning nhưng trong css c thì nó là error. và theo quan điểm của tôi nó phải là một error.
Nếu bạn muốn dùng 2 cái delay_ms() trong cả ngắt và main thì mình nghĩ không có cách nào đâu. nếu bạn cứ muốn dùng nó thì hãy tạo ra 2 hàm delay_ms1() và delay_ms2() và trong ngắt gọi một hàm và trong main gọi hàm còn lại.
Còn có một cách nữa mình nghĩ nó sẽ pro hơn đấy là. Mình xin viết lại một đoạn chương trình của bạn trong: bạn khai báo một biến toàn cục như thế này nhé.

```
//=======
int1 bit_timer0_status;

#INT_TIMER1
void lapngat()
{
bit_timer0_status =1;
}
//===========
// và đưa công việc của bạn muốn thực hiện khi có ngắt vào đây.
//===========
void isr_timer0(void)
{
count++;
if (count==200)
```

```
while(true)
{
output_high(PIN_C1);
delay_ms(1000);
if(!input(PIN_B3)) break;
output_low(PIN_C1);
delay_ms(1000);
}
//===========
//và trong hàm main bạn phải làm thế này
//===========
void main(void)
{
if(bit_timer0_status)
{
//có thể cấm ngắt timer0.
//gọi hàm thực hiện công việc.
isr_timer0();
bit_timer0_status=0;
//bật ngắt trở lại
}
}
```

Giải pháp của bạn rất hay, và thực ra sau khi mình tìm hiểu điều này trên chuyên trang của CCSC mình cũng có ý tưởng na ná như vậy.

Nhưng nó vẫn có những hạn chế nhất định. Mình sẽ phân tích cho bạn như sau:
Giả sử chương trình viết theo cách của bạn, khi nó kiểm tra đến lệnh 'if(bit_timer0_status)' mà chưa xảy ra ngắt thì nó sẽ không thực hiện hàm 'isr_timer0()', qua đó rồi ngắt mới xảy ra thì chương trình lại không test 'if(bit_timer0_status)' và không thực hiện hàm 'isr_timer0()', tức là không có ngắt. Còn nếu trong chương trình mà luôn kiểm tra 'if(bit_timer0_status)' thì đang từ việc sử dụng phương pháp ngắt lại thành ra phương pháp thăm dò ???

Bạn chưa đọc kỹ chương trình của mình thì phải. Sẽ không có trường hợp ngắt xẩy ra mà không thực hiện công việc mà bạn mong muốn. Có chăng thì nó chậm hơn so với khi để công việc trong ngắt một chút. thời gian này không đáng kể chỉ khoảng vài 2ms. cái này không ảnh hưởng gì phải không?

Tại sao mình lại bảo không có chuyện như bạn nói: Trong chương trình phục vụ ngắt mình đã sử dụng một bit 'bit_Timer0_Status', bit này có mục đích khi có ngắt thì bật nó lên để báo cho hàm main biết đã có ngắt xẩy ra. và kể cả khi hàm main đang làm những gì, ở đâu thì khi gặp lệnh if(bit_Timer0_Status) thì nó thực hiện công việc mong muốn. và khi thực hiện công việc này xong phải xóa bit này để dùng cho lần sau.

Như mình đã nói nếu viết theo cách của bạn thì gần như chắc chắn sẽ không thực hiện được hàm ngắt theo mong muốn. Đã đành khi timer1 báo cho(bit_Timer0_Status) =1 (timer đã ngắt), nhưng có thể nói chắc chắn rằng khi chương trình đang kiểm tra 'if(bit_timer0_status)' thì timer chưa ngắt vì nó kiểm tra ngay bắt đầu hàm main. Sau khi đã kiểm tra như vậy rồi thì chương trình có quay lại để tiếp tục kiểm tra đâu mà gọi hàm 'ngắt phụ' như bạn viết.

Nếu bạn không tin thì hãy lập thử một chương trình đơn giản sử dụng ngắt theo cách bạn viết và mô phỏng thử trên Proteus. (việc này đơn giản và không tốn nhiều thời gian,chắc không cần nói thêm!)

Tiện đây mình nói với bạn về các phương pháp trao đổi dữ liệu nói chung. Có 3 phương

pháp chủ yếu là pp thăm dò (polling), pp ngắt (interrupt) và pp trao đổi trực tiếp (pp thú ba mình không nhớ rõ lắm). Làm như cách của bạn chính là chuyển từ pp ngắt sang pp thăm dò.

Hóa ra bạn bảo mình là không cho cái if của mình vào trong cái while(true) của bạn. Thật ra cái đấy mình chỉ viết thí dụ thôi. Còn nếu bạn muốn sử dụng thì bạn phải tự làm thêm. Vì mình nghĩ trong một firmware cho vi điều khiển không có cái nào không có lệnh while(true) {} cả. Còn nếu đọc đến đây mà bạn vẫn khăng khăng nó không thực hiện được ngắt thì... mình cũng bó tay rồi. Nếu cần mình sẽ cho bạn xem một chương trình của mình, mình đã viết và thành phẩm. Phải nói nó chạy phe phé.

Và cái thứ 2 mình muốn nói thêm. Mình chỉ đưa ra những phương án như trên cho bạn, để bạn tham khảo thôi chứ không phải hay ho gì. nên bạn không cần thiết phải so sánh nó với 3 cái mà bạn nói. Nói thật mình chả hiểu đếch gì về mấy cái đấy cả... hê hê.

+Xin hỏi tại sao chương trình dùng ngắt timer0 em làm giống nhu hướng dẫn mà nó không chịu chạy?

```
 #include<18F4450.h>
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=20000000)
#use fast_io(b)
#byte portb=0x6
#define led pin_B0
int8 a;
void ngat()
{
delay_ms(1000);
a=a<<1; // dich trai a 1bit
if(a==256)
a=1;
}
main()
{
a=1;
set_tris_b(0);
while(true)
{
ngat();
output_b(a);
}
}
```

Cách này thì PIC18F4550 cũng dịch leds vô tư. Nhưng mà muốn dùng TIMER0 như của bác nhh . Khi mình làm không hiểu tại sao lại không thể chạy. Nhờ bác chỉ giáo giùm nhé!

#include <18F4550.h>
#fuses NOWDT,PUT,XT,NOPROTECT
#use delay(clock=20000000) . Mình đã sửa lại code của nhh như vậy để phù hợp với PIC18F4550, nhưng không hiểu sao vẫn không chạy đợc. Bạn có thể hướng dẫn kỹ hơn về Interrup và Timer không, cảm on bạn nhiều.

TL: Úi code của bạn sai rồi nếu muốn sử dụng được ngắt thì trong hàm main bạn phải thực hiện cho phép ngắt timer_0 và ngắt toàn cục (global) hoạt động, định xung nhịp cho timer, và hàm ngắt() phải được đặt ngay bên dưới chỉ thị #int_timer0 để trình dịch có thể hiểu được đây là hàm phục vụ ngắt, hàm ngắt() này bạn không thể gọi đến giống như một hàm thông thường được mà nó chỉ được máy gọi đến khi có xuất hiện cờ tràn timer_0, mà khi đã sử dụng ngắt

timer (tức ngắt định thời) thì bạn không nên dùng delay trong nó (rất dở),vì ngắt timer được tạo ra để thay thế hoàn hảo cho delay, code của bạn mình sửa như sau :

```
#include<18F4450.h>
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=4000000)
int16 count=0;
int8 a=1;

#int_timer0
void ngat(){
++count;
if(count==2000) {count=0;a=a<<1;} // dich trai a 1bit
if(a==256) {a=1;count=0; }}

void main(){
enable_interrupts(global);
enable_interrupts(int_timer0);
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2); //xung timer = xung máy/2
while(true) {
output_b(a);
}}
```

Như vậy ở hàm trên ta sẽ thấy sau mỗi lần tràn ngắt timer_0 (sau hai xung máy) biến đếm count sẽ tăng lên 1 đơn vị và cứ thế cho đến khi đạt giá trị 2000 (đây là giá trị mà bạn cần phải tính để chọn đúng thời gian cần làm làm trễ) khi đó biến a sẽ được dịch bit sang trái,....rồi tiếp tục... chắc khúc này bạn hiểu rồi há.

+Chào các bạn mình mới học pic nên chưa biết nhiều mong được sự giúp đỡ, tui có vài câu hỏi mong được chỉ giáo tui sử dụng ngắt timer nhưng thấy lệnh **set_timer0();** ko có tác dụng nghĩa là đặt số mấy cũng ko thấy thay đổi thậm chí ko có lệnh đó vẫn chạy như thường bạn nào làm ngắt timer rùi chỉ mình với

Code:

```
#include <18F4331.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)

int i=0;
void main()
{
        setup_timer_0(rtcc_div_32);
         set_timer0(10);   ????nếu đặt 1 thì bao lâu mới ngắt
        enable_interrupts(int_rtcc);
        enable_interrupts(global);
        while(true)
        {
        }
}

#int_rtcc
void ngat_timer()
{
        i++;
        output_D(i);

}
```

Cấu hỏi thứ 2 là khi giao tiếp máy tính mình truyền từ PC đến pic: dùng lệnh getc(); thì chỉ thu được 1 kí tự ví dụ truyền số 12 thì thu được 2 số 1 và 2 có bạn nào biết lệnh nào để lấy 1 chuỗi ko?

TL: Về timer 0 và hàm set_timer0(), bạn đọc thêm tài liệu hướng dẫn của CCS C (trang 206, ver. 4, 01/2007) và datasheet của chip.

Về getc(), nếu bạn dùng nó thì tất nhiên chỉ lấy được 1 ký tự là nó đã trở về. Bạn đọc thêm về hàm gets() trong tài liệu hướng dẫn của CCS C (đã nêu trên, trang 148) để đọc 1 chuỗi ký tự.

+**Thắc mắc ngắt timer**

Em viết chương trình ngắt timer nhấp nháy led, nhưng nạp vào pic phải đợi 1 lúc sau nó mới bắt đầu nháy led, bác nào khắc phục jùm em với.

Đây là code:

```
#include <16F877A.h>
#include <DEFS_16F877A.h>
#device 16F877*=16 ADC=10
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD,
NOWRT
#use delay(clock=4000000)
#use fast_io(b)
#use fast_io(c)
#use fast_io(d)

#int_timer0
void interrupt_timer0(){
   int1 a;
   int16 count;
   set_timer0(56);
   ++count;
   if(count == 500)
   {
       count=0;
       a=~a;
       RB0=a;
   }


}
void main()
{
    set_tris_b(0);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);
    enable_interrupts(int_timer0);
    enable_interrupts(global);
    set_timer0(56);

    while(true)
    {
    }
}
```

TL: Trước khi vào vòng while(true) trong main(), bạn nên đặt trước giá trị của count, nếu không thì count có thể mang giá trị bất kỳ, và điều kiện (count == 500) của bạn có thể phải sau khi count được tăng vài chục ngàn giá trị mới thỏa mãn (nếu tình cờ sau khi PIC reset biến count mang giá trị ngẫu nhiên là 501 chẳng hạn).

Tôi thường khởi tạo biến count ngay trước khi vào vòng while(true) trong main(), và dùng điều kiện (count >= 500) thay cho (count == 500) trong phần xử lý ngắt.

Thang8831

http://www.picvietnam.com

6. Chương trình ví dụ sau mô tả cách dùng PWM do CCS cung cấp.

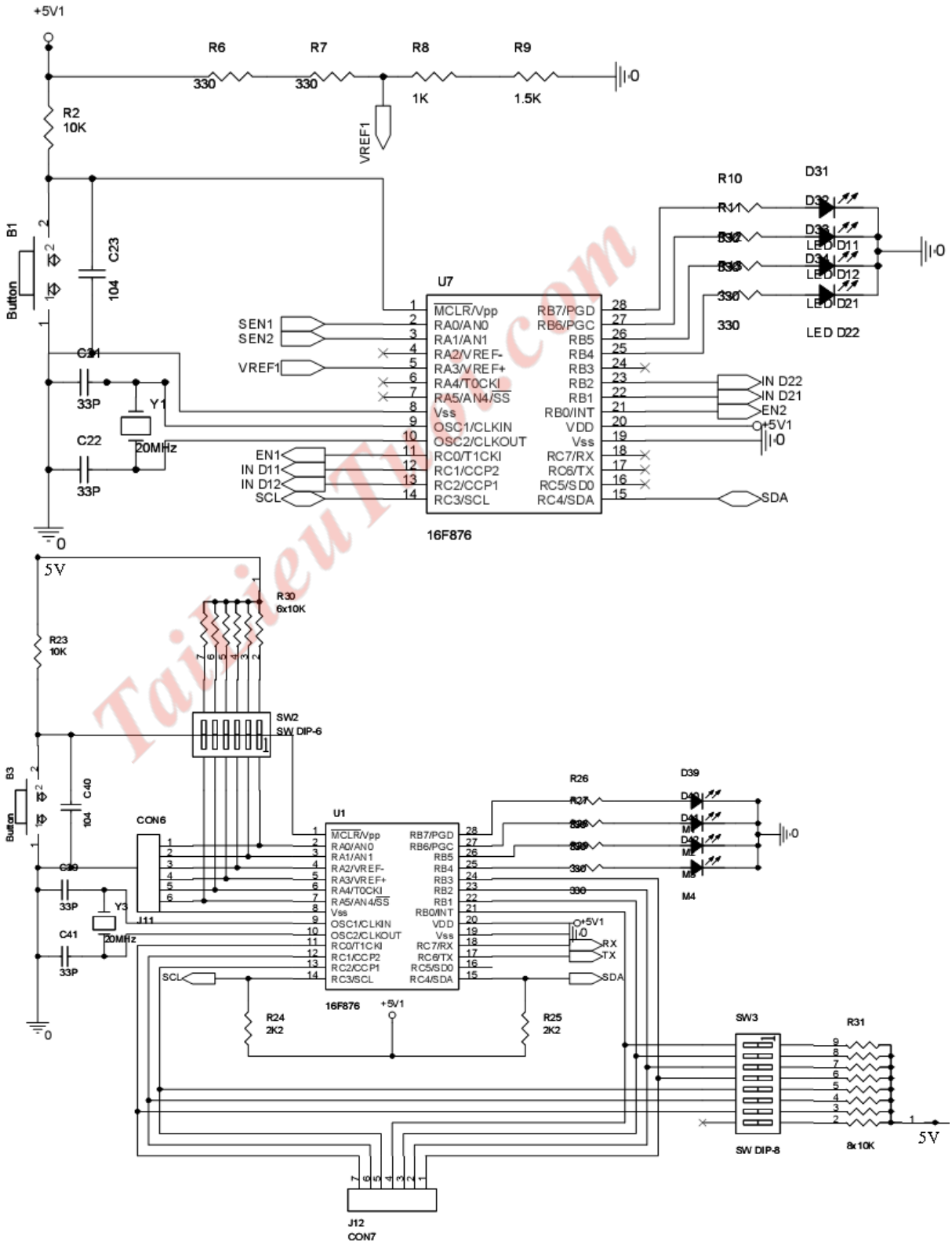
PWM là gì? sử dụng nó vào mục đích gì?
1) PWM là gì? PWM là một bộ điều chế độ rộng xung. Có hai thông số (tạm gọi đơn giản như vậy, và có lẽ cũng chỉ quan tâm đến hai thông số này với PWM) quan trọng của PWM là chu kỳ xung T và thời gian t1 của mức logic 0,
trong ví dụ này thì t1 tương ứng với value. Để "điều chế độ rộng xung" thì chúng ta sẽ giữ nguyên T và thay đổi t1, theo yêu cầu của bài toán cụ thể.
Value trong ví dụ sau lấy được từ đầu vào anlaog, chu kỳ (hay tần số) của xung được chọn lựa từ PC thông qua cổng truyền thông nối tiếp RS232.
2) PWM dùng vào mục đích gì? Có nhiều ứng dụng cho nó, ví dụ truyền thông, điều khiển các van bán dẫn trong các biến tần, làm bộ nguồn chuyển mạch,...ôi nhiều lắm!
Bắt đầu viết nhé:

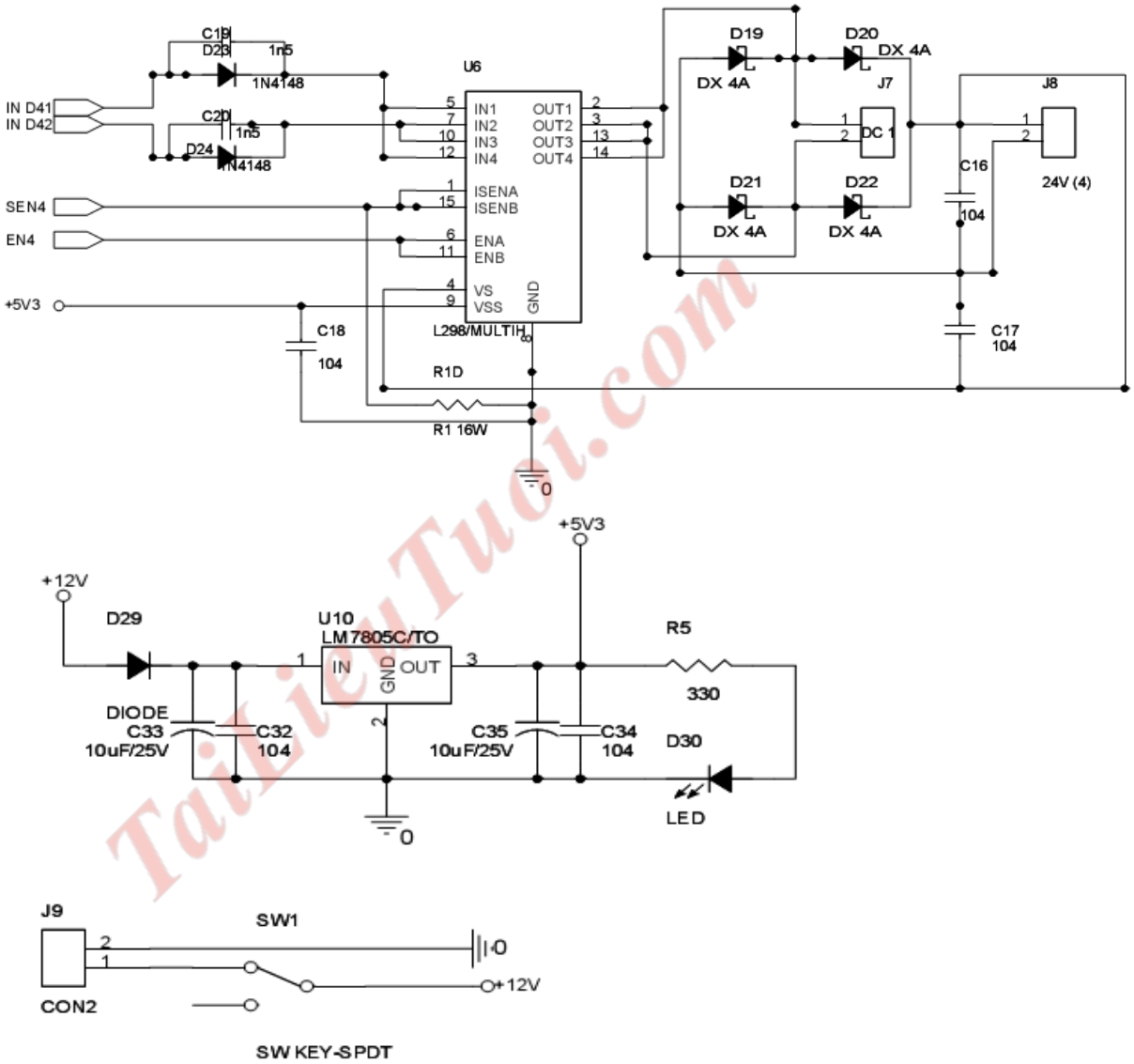
```
#if defined(__PCM__)
#include <16F877.h>

#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, BRGH1OK)

#elif defined(__PCH__)
#include <18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=10000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, BRGH1OK)
#endif

void main() {
char selection=1;
int8 value;

printf("\r\nFrequency:\r\n");
printf(" 1) 7.8 khz\r\n");
printf(" 2) 19.5 khz\r\n");
printf(" 3) 0.48 khz\r\n");
do {
selection=getc();
}while((selection<'1')||(selection>'3'));

setup_ccp1(CCP_PWM); // Configure CCP1 as a PWM

// The cycle time will be (1/clock)*4*t2div*(period+1)
// In this program clock=4000000 and period=127 (below)
// For the three possible selections the cycle time is:
// (1/4000000)*4*1*128 = 12.8 us or 7.8 khz
// (1/4000000)*4*4*128 = 51.2 us or 19.5 khz
// (1/4000000)*4*16*128= 204.8 us or 0.48 khz
switch(selection) {
```

```
case '1' : setup_timer_2(T2_DIV_BY_1, 127, 1);
break;
case '2' : setup_timer_2(T2_DIV_BY_4, 127, 1);
break;
case '3' : setup_timer_2(T2_DIV_BY_16, 127, 1);
break;
}

setup_port_a(ALL_ANALOG);
setup_adc(adc_clock_internal);
set_adc_channel( 0 );
printf("%c\r\n",selection);

while( TRUE ) {
value=read_adc();
//value++;
printf("%2X\r",value);
set_pwm1_duty(value); //value may be an 8 or 16 bit constant or variable
// This sets the time the pulse is
// high each cycle. We use the A/D
// input to make a easy demo.
// the high time will be:
// if value is LONG INT:
// value*(1/clock)*t2div
// if value is INT:
// value*4*(1/clock)*t2div
// for example a value of 30 and t2div
// of 1 the high time is 30us
// WARNING: A value too high or low will
// prevent the output from
// changing.
}
}
+
```

Yêu cầu của e là điều chỉnh độ rộng xung để thay đổi vận tốc của động cơ (trong để tài của e là động cơ của bơm cánh dẫn)
Cảm ơn các trưởng lão namqn! Rất mong nhận được sự giúp đỡ của bạn!

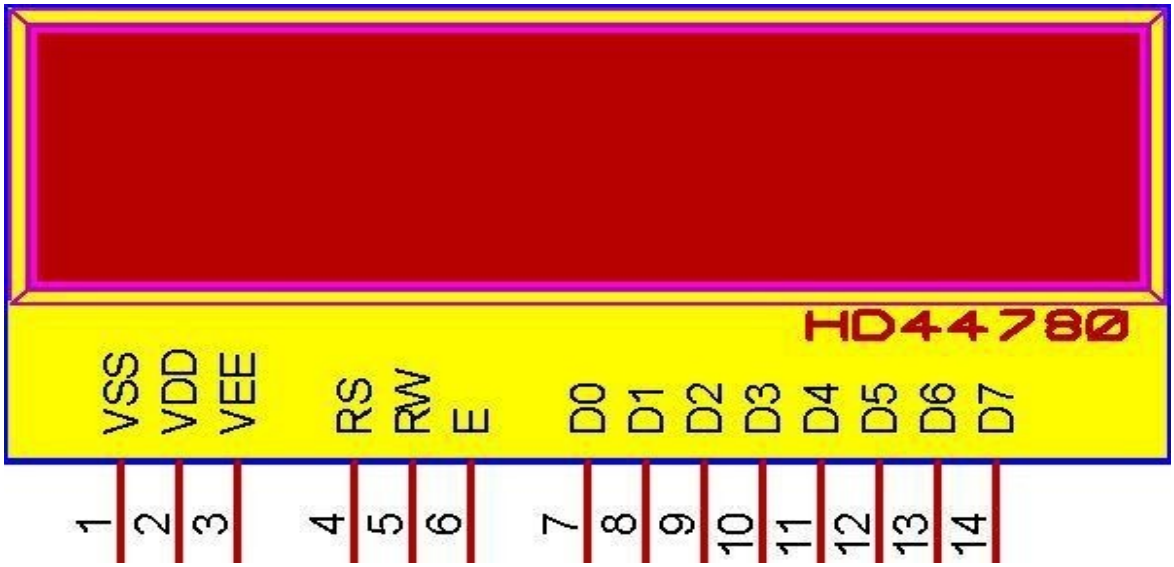
Với động cơ thì tần số điều chế thường quanh quẩn 20 kHz. Bây giờ bạn cho biết bạn dùng loại bộ điều khiển nào để điều chỉnh vận tốc của động cơ, bạn có đo vận tốc chứ? Từ ngõ ra của bộ điều chế độ rộng xung bạn làm thế nào để điều chỉnh vận tốc của động cơ? Động cơ đó có công suất khoảng bao nhiêu?





7. Tìm hiểu về LCD

LCD được tìm hiểu ở đây là HD44780 của hãng Hitachi, gồm 2 dòng, mỗi dòng 16 kí tự.



Thang8831
http://www.picvietnam.com

HD44780 có 14 chân, chức năng của các chân:
1.Các chân **VCC**, **VSS** và **VEE**: Chân VCC_Cấp dương nguồn 5V, chân VCC_Nối đất, chân VEE được dùng để điều khiển độ tương phản của màn hình LCD.
2.Chân chọn thanh ghi **RS** (Register Select):
Có hai thanh ghi rất quan trọng bên trong LCD, chân RS được dùng để chọn các thanh ghi này như sau: Nếu RS = 0 thì thanh ghi mà lệnh được chọn để cho phép người dùng gửi một lệnh chẳng hạn như xoá màn hình, đưa con trỏ về đầu dòng,… Nếu RS = 1 thì thanh ghi dữ liệu được chọn cho phép người dùng gửi dữ liệu cần hiển thị trên LCD.
3.Chân đọc/ghi **R/W**:
Đầu vào đọc/ghi cho phép người dùng ghi thông tin lên LCD khi R/W = 0 hoặc đọc thông tin từ nó khi R/W = 1.
4.Chân cho phép **E** (Enable):
Chân cho phép E được sử dụng bởi LCD để chốt thông tin hiện hữu trên chân dữ liệu của nó. Khi dữ liệu được cấp đến chân dữ liệu thì một xung mức cao xuống thấp phải được áp đến chân này để LCD chốt dữ liệu trên các chân dữ liêu. Xung này phải rộng tối thiểu là 450ns.
5.Các chân **D0 - D7**:
Đây là 8 chân dữ liệu 8 bít, được dùng để gửi thông tin lên LCD hoặc đọc nội dung của các thanh ghi trong LCD.
Để hiển thị các chữ cái và các con số, chúng ta gửi các mã ASCII của các chữ cái từ A đến Z, a đến f và các con số từ 0 - 9,... đến các chân này khi bật RS = 1.
Cũng có các mã lệnh mà có thể được gửi đến LCD để xoá màn hình hoặc đưa con trỏ về đầu dòng hoặc nhấp nháy con trỏ.Dưới đây là bảng liệt kê các mã lệnh:
(Phải qua lần post khác vì số ảnh vượt quá 4.....)
Tìm hiểu về LCD (ct)
Chúng ta cũng sử dụng RS = 0 để kiểm tra bít cờ bận để xem LCD có sẵn sàng nhận thông tin chưa. Cờ bận là D7 và có thể được đọc khi R/W = 1 và RS = 0 như sau:
Nếu R/W = 1, RS = 0 khi D7 = 1 (cờ bận 1) thì LCD bận bởi các công việc bên trong và sẽ không nhận bất kỳ thông tin mới nào. Khi D7 = 0 thì LCD sẵn sàng nhận thông tin mới. Lưu ý chúng ta nên kiểm tra cờ bận trước khi ghi bất kỳ dữ liệu nào lên LCD.
Có thể di chuyển con trỏ đến vị trí bất kì trên màn hình LCD bằng cách nạp vào các giá trị tương ứng như bảng sau và gởi yêu cầu đến LCD:

Mã (Hex)	Lệnh đến thanh ghi của LCD
1	Xoá màn hình hiển thị
2	Trở về đầu dòng
4	Giả con trỏ (dịch con trỏ sang trái)
6	Tăng con trỏ (dịch con trỏ sang phải)
5	Dịch hiển thị sang phải
7	Dịch hiển thị sang trái
8	Tắt con trỏ, tắt hiển thị
A	Tắt hiển thị, bật con trỏ
C	Bật hiển thị, tắt con trỏ
E	Bật hiển thị, nhấp nháy con trỏ
F	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	ép con trỏ Vũ đầu dòng thứ nhất
C0	ép con trỏ Vũ đầu dòng thứ hai
38	Hai dòng và ma trận 5 × 7

Tham khảo thêm về LCD tại đây: http://www.iaehv.nl/users/pouweha/lcd.htm

Có hai cách lập trình cho LCD: dùng 8bit interface (đơn giản) hoặc 4bit interface (phức tạp hơn)

7.1. 8bit interface

Code:

```
/*Để LCD thực thi các lệnh điều khiển:*/

RS = 0;         //chọn thanh ghi lệnh
R/W = 0;        //ghi dữ liệu, R/W = 1;//đọc dữ liệu
E= 1;           //đưa chân E lên mức cao
E= 0;            //tạo sườn xuống để chốt dữ liệu

/*Để LCD thực thi các lệnh hiển thị:*/

RS = 1;         //chọn thanh ghi dữ liệu
R/W = 0;        //ghi dữ liệu
E = 1;          //đưa chân E lên mức cao
E = 0;           //tạo sườn xuống để chốt dữ liệu
```

Sử dụng 8 chân D0 - D7 để truyền thông tin, dữ liệu đến LCD.

- Để điều khiển LCD (Chọn chế độ LCD, bật/tắt hiển thị, bật/tắt/nhấp nháy/di chuyển con trỏ,...): Nhập giá trị tương ứng vào D0-D7 rồi gởi lệnh yêu cầu LCD thực thi lệnh điều khiển, tiếp theo cho LCD thời gian trễ để thực thi (hoặc hỏi cờ bận xem LCD sẵn sàng thực hiện lệnh tiếp theo chưa?)
- Để hiển thị dữ liệu lên LCD:Nhập dữ liệu cần hiển thị vào D0-D7 rồi gởi lệnh yêu cầu LCD thực thi lệnh hiển thị dữ liệu, tiếp theo cho LCD thời gian trễ để thực thi (hoặc hỏi cờ bận xem LCD sẵn sàng thực hiện lệnh tiếp theo chưa?)

Đây là mạch nguyên lý kết nối LCD dùng 8 chân interface với PIC16F877A qua PORTB:

7.2. 4bit interface

Sử dụng 4 chân D4 - D7 (hoặc D0-D3 <- ít dùng) để truyền thông tin, dữ liệu đến LCD.
- Để điều khiển LCD (Chọn chế độ LCD, bật/tắt hiển thị, bật/tắt/nhấp nháy/di chuyển con trỏ,...): Nhập giá trị tương ứng vào D0-D7,lấy giá trị 4bit cao D4-D7 rồi gởi lệnh yêu cầu LCD thực thi lệnh điều khiển, tiếp theo cho LCD thời gian trễ để thực thi (hoặc hỏi cờ bận xem LCD sẵn sàng thực hiện lệnh tiếp theo chưa?).Tiếp tục, gởi 4bit thấp D0-D3 rồi gởi lệnh yêu cầu LCD thực thi lệnh điều khiển, tiếp theo cho LCD thời gian trễ để thực thi (hoặc hỏi cờ bận xem LCD sẵn sàng thực hiện lệnh tiếp theo chưa?)
- Để hiển thị dữ liệu lên LCD:Cũng làm tương tự trên nhưng thay yêu cầu LCD điều khiển bằng yêu cầu LCD hiển thị.

Đây là mạch nguyên lý kết nối LCD dùng 4 chân interface với PIC16F877A qua PORTB:

Nếu trong ứng dụng sử dụng ngắt ngoài thì có thể chuyển sang nối với PORTD hoặc tùy thích.

7.3. LCD_lib_4bit

```
#include <stddef.h>

#define LCD_RS         PIN_D2
//#define LCD_RW       PIN_A1
#define LCD_EN         PIN_D3

#define LCD_D4         PIN_D4
#define LCD_D5         PIN_D5
#define LCD_D6         PIN_D6
#define LCD_D7         PIN_D7

// misc display defines-
#define Line_1         0x80
#define Line_2         0xC0
#define Clear_Scr      0x01

// prototype statements
#separate void LCD_Init ( void );// ham khoi tao LCD
#separate void LCD_SetPosition ( unsigned int cX );//Thiet lap vi tri con tro
#separate void LCD_PutChar ( unsigned int cX );// Ham viet1kitu/1chuoi len LCD
#separate void LCD_PutCmd ( unsigned int cX) ;// Ham gui lenh len LCD
#separate void LCD_PulseEnable ( void );// Xung kich hoat
#separate void LCD_SetData ( unsigned int cX );// Dat du lieu len chan Data
// D/n Cong
#use standard_io (C)
#use standard_io (D)

//khoi tao LCD*********************************************
#separate void LCD_Init ( void )
   {
   LCD_SetData ( 0x00 );
   delay_ms(200);       /* wait enough time after Vdd rise >> 15ms */
   output_low ( LCD_RS );// che do gui lenh
   LCD_SetData ( 0x03 );   /* init with specific nibbles to start 4-bit mode */
   LCD_PulseEnable();
   LCD_PulseEnable();
```

```
   LCD_PulseEnable();
   LCD_SetData ( 0x02 );      /* set 4-bit interface */
   LCD_PulseEnable();         /* send dual nibbles hereafter, MSN first */
   LCD_PutCmd ( 0x2C );    /* function set (all lines, 5x7 characters) */
   LCD_PutCmd ( 0x0C );    /* display ON, cursor off, no blink */
   LCD_PutCmd ( 0x06 );    /* entry mode set, increment & scroll left */
   LCD_PutCmd ( 0x01 );    /* clear display */
   }
#separate void LCD_SetPosition ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
   LCD_SetData ( swap ( cX ) | 0x08 );
   LCD_PulseEnable();
   LCD_SetData ( swap ( cX ) );
   LCD_PulseEnable();
   }
#separate void LCD_PutChar ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
      output_high ( LCD_RS );
      LCD_PutCmd( cX );
      output_low ( LCD_RS );
   }
#separate void LCD_PutCmd ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
   LCD_SetData ( swap ( cX ) );     /* send high nibble */
   LCD_PulseEnable();
   LCD_SetData ( swap ( cX ) );     /* send low nibble */
   LCD_PulseEnable();
   }
#separate void LCD_PulseEnable ( void )
   {
   output_high ( LCD_EN );
   delay_us ( 3 );         // was 10
   output_low ( LCD_EN );
   delay_ms ( 3 );         // was 5
   }
#separate void LCD_SetData ( unsigned int cX )
   {
   output_bit ( LCD_D4, cX & 0x01 );
   output_bit ( LCD_D5, cX & 0x02 );
   output_bit ( LCD_D6, cX & 0x04 );
   output_bit ( LCD_D7, cX & 0x08 );
   }
```

7.4. LCD lib 8bits

```
#define LCD_RS PIN_A0
#define LCD_RW PIN_A1
#define LCD_EN PIN_A2
```

```
#define LCD_data Port_C

// misc display defines-
#define Line_1 0x80
#define Line_2 0xC0
#define Clear_Scr 0x01

// prototype statements
#separate void LCD_Init ( void );// ham khoi tao LCD
#separate void LCD_SetPosition ( unsigned int cX );//Thiet lap vi tri con tro
#separate void LCD_PutChar ( char cX );// Ham viet1kitu/1chuoi len LCD
#separate void LCD_PutCmd (int cX) ;// Ham gui lenh len LCD
#separate void LCD_PulseEnable ( void );
#separate int1 LCD_ready();

// D/n Cong

#use standard_io (C)
#use standard_io (A)
#use standard_io (B)
//khoi tao LCD*********************************************
#separate void LCD_Init ( void )
{
  output_C(0x00); delay_ms(200);
  LCD_Putcmd(0x38);
  LCD_Putcmd(0x0F);
  LCD_Putcmd(0x01);
  LCD_Putcmd(0x06);
}
#separate void LCD_SetPosition (int cX )
{
}
#separate void LCD_PutChar ( char cX )
{
  output_high(LCD_RS);
  output_low(LCD_RW);
  output_C(cX);
  LCD_PulseEnable();
  delay_ms(5);
}
#separate void LCD_PutCmd (int cX )//Gui lenh den LCD
{
  output_low(LCD_RS);
  output_low(LCD_RW);
  output_C(cX);
  LCD_PulseEnable();
  delay_ms(5);
}
#separate int1 LCD_ready ()
{  int1 busy_flag;
```

```
  trisc7 = 1;//set input
  output_low(LCD_RS);
  output_high(LCD_RW);
  LCD_pulseEnable();
  busy_flag = RC7;
  return(busy_flag);
}
#separate void LCD_PulseEnable ( void )
{
output_high ( LCD_EN );
delay_us ( 3 ); // was 10
output_low ( LCD_EN );
delay_ms ( 3 ); // was 5
}
```

7.5. Hiển thị LCD 8bit interface

Chương trình hiển thị dòng chữ "BE YEU" trên hàng 1, bắt đầu tại cột 6, không hỏi cờ bận D7.

Do trong thân hàm comnwrt() và datawrt() đã tạo trễ 1ms cuối thân hàm nên sau khi gọi không cần tạo trễ cho LCD thực thi lệnh.

Code:

```
/*-------------------------------------------------------------------
* Author       : nhh
* Date         : 05/04/06
* Hardware     : PIC16F877A
* Compiler     : CCS C 3.249
* Description   : Hien thi LCD
*===================================================================*/
#include <16F877A.h>
#include <DEFS_16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define RS  RD0
#define RW  RD1
#define E   RD2
#define LCD PORTB

/*Ham yeu cau goi lenh dieu khien LCD*/
void comnwrt(void)
{
 RS = 0;
 RW = 0;
 E  = 1;
 E  = 0;
 delay_ms(1);
}
/*Ham yeu cau goi du lieu hien thi len LCD*/
void datawrt(void)
{
 RS = 1;
 RW = 0;
 E  = 1;
 E  = 0;
 delay_ms(1);
}
/*Ham main*/
```

```
void main(void)
{
 set_tris_B(0);
 set_tris_D(0);
 delay_ms(100);    //   Tao tre 100ms cho LCD khoi dong

 LCD = 0x38;       //   Hai hang, ma tran dot 5*7, 8 bit interface
 comnwrt();
 LCD = 0x0C;       //   Bat hien thi, tat con tro
 comnwrt();

 LCD = 0x85;       //   Vi tri hang 1,cot 6
 comnwrt();

 LCD = 'B';        //   Xuat dong chu "BE YEU" ra LCD
 datawrt();
 LCD = 'E';
 datawrt();
 LCD = ' ';
 datawrt();
 LCD = 'Y';
 datawrt();
 LCD = 'E';
 datawrt();
 LCD = 'U';
 datawrt();
 LCD = '!';
 datawrt();
}
```


//========== **Register Definitions** ==========

//-----Register Files----------

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

#byte PR2      = 0x92

#bit RA4                    = 0x05.4
#bit RA3                    = 0x05.3
#bit RA2                    = 0x05.2
#bit RA1                    = 0x05.1
#bit RA0                    = 0x05.0

#bit RB7                    = 0x06.7
```

```
#bit RB6                = 0x06.6
#bit RB5                = 0x06.5
#bit RB4                = 0x06.4
#bit RB3                = 0x06.3
#bit RB2                = 0x06.2
#bit RB1                = 0x06.1
#bit RB0                = 0x06.0

#bit RC7                = 0x07.7
#bit RC6                = 0x07.6
#bit RC5                = 0x07.5
#bit RC4                = 0x07.4
#bit RC3                = 0x07.3
#bit RC2                = 0x07.2
#bit RC1                = 0x07.1
#bit RC0                = 0x07.0

#bit RD7                = 0x08.7
#bit RD6                = 0x08.6
#bit RD5                = 0x08.5
#bit RD4                = 0x08.4
#bit RD3                = 0x08.3
#bit RD2                = 0x08.2
#bit RD1                = 0x08.1
#bit RD0                = 0x08.0

#bit RE2                = 0x09.2
#bit RE1                = 0x09.1
#bit RE0                = 0x09.0

//----- INTCON ----------------------------------------------------------

#bit    GIE     = 0x0b.7
#bit    PEIE    = 0x0b.6
#bit    TMR0IE  = 0x0b.5
#bit    INTE    = 0x0b.4
#bit    RBIE    = 0x0b.3
#bit    TMR0IF  = 0x0b.2
#bit    INTF    = 0x0b.1
#bit    RBIF    = 0x0b.0

//----- PIR1 ------------------------------------------------------------

#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 --------------------------------------------------------------

#bit CMIF      = 0x0d.6
#bit EEIF      = 0x0d.4
#bit BCLIF     = 0x0d.3
#bit CCP2IF    = 0x0d.0

//----- PIE1 --------------------------------------------------------------

#bit PSPIE     = 0x8c.7
#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 --------------------------------------------------------------

#bit CMIE      = 0x8d.6
#bit EEIE      = 0x8d.4
#bit BCLIE     = 0x8d.3
#bit CCP2IE    = 0x8d.0
```

-Thêm một ví dụ khác, chương trình hiển thị dòng "HELLO PICVIETNAM!".

code

```
/*-------------------------------------------------------------------------
* Author        : nhh
* Date          : 05/04/06
* Hardware      : PIC16F877A
* Compiler      : CCS C 3.249
* Description   : Hien thi LCD
*=========================================================================*/
#include <16F877A.h>
#include <DEFS_16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define RS  RD0
#define RW  RD1
#define E   RD2
#define LCD PORTB

const unsigned char key[]="HELLOPICVIETNAM!";
int i = 0;

/*Ham yeu cau goi lenh dieu khien LCD*/
void comnwrt(void)
{
 RS = 0;
 RW = 0;
 E  = 1;
 E  = 0;
```

```
 delay_ms(1);
}
/*Ham yeu cau goi du lieu hien thi len LCD*/
void datawrt(void)
{
 RS = 1;
 RW = 0;
 E  = 1;
 E  = 0;
 delay_ms(1);
}
/*Ham main*/
void main(void)
{
 set_tris_B(0);
 set_tris_D(0);
 delay_ms(100);   //   Tao tre 100ms cho LCD khoi dong

 LCD = 0x38;      //   Hai hang, ma tran dot 5*7, 8 bit interface
 comnwrt();
 LCD = 0x0C;      //   Bat hien thi, tat con tro
 comnwrt();
 LCD = 0x86;      //   Vi tri hang 1,cot 7
 comnwrt();
 while(true)
   {
       LCD = key[i];
       datawrt();
       delay_ms(100);
       i++;
       if(i==5)               //   Hien thi xong HELLO
         {
            LCD = 0xC3;       //   Vi tri hang 2,cot 4
            comnwrt();
            delay_ms(100);
         }
       if(i==16)              //   Hien thi xong PICVIETNAM!
         {
            delay_ms(1100);
            LCD = 0x01;       //   Xoa man hinh hien thi
            comnwrt();
            delay_ms(500);
            LCD = 0x86;       //   Vi tri hang 1,cot 7
            comnwrt();
            i = 0;
         }
   }
}
```

7.6. Hiển thị LCD 4bit interface

Cái này trong thư viện của CCS C đã có file lcd.c trong thư mục Drivers rất là hay rồi, nên không cần viết lại làm gì.File này rất hay,nhưng chỉ dùng cho LCD 2 line.Các bác tự nghiên cíu nhé!

Chương trình hiển thị chữ "HI!" bắt đầu tại hàng 1, cột 7.Dùng LCD 4bit interface và thư viện lcd.c của CCS C

Code:

```
/*-------------------------------------------------------------------------
* Author       : nhh
* Date         : 05/14/06
* Hardware     : PIC16F877A
* Compiler     : CCS C 3.249
* Description   : Hien thi LCD
*=========================================================================*/
#include <16F877A.h>
#include <DEFS_16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define use_portb_lcd TRUE
#include <lcd.c>

void main(void)
{
    delay_ms(100);          // tao tre 100ms cho LCD khoi dong
    lcd_init();
    lcd_gotoxy(7,1);        // vi tri (x,y)=(7,1)= hang 1, cot 7
    lcd_putc('H');
    lcd_putc('I');
    lcd_putc('!');
}
```

7.7. LCD_8bit interface, có kiểm tra cờ bận.

Bài cuối về LCD. Hoạt động theo 8bit interface, có hỏi cờ bận đảm bảo LCD luôn thực thi đúng lệnh yêu cầu ! Chú ý việc hỏi cờ bận là hết sức cần thiết!

Một điều nữa là Protues mô phỏng cho LCD hơi cà thoạt, nên dùng Picsimulator.Tốt nhất kiếm 1 chú LCD làm cho xom!

Chương trình hiển thị dòng chữ "WONDERFUL PICVIETNAM!",tham khảo source code của CCS C.

Code:

```
/*-------------------------------------------------------------------------
* Author       : nhh
* Date         : 05/04/06
* Hardware     : PIC16F877A
* Compiler     : CCS C 3.249
* Description   : Hien thi LCD
*=========================================================================*/
#include <16F877A.h>
#include <DEFS_16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define E  RD2
#define RS RD0
#define RW RD1
#byte lcd_data = 0x06   // Dia chi PORTB

/* Khai bao nguyen mau cac ham su dung */
byte lcd_read_byte();
void lcd_send_byte( byte address, byte n );
void lcd_init();
void lcd_gotoxy( byte x, byte y);
void lcd_putc( char c);
void lcd_refresh();

/* Doc mot byte tu LCD */
```

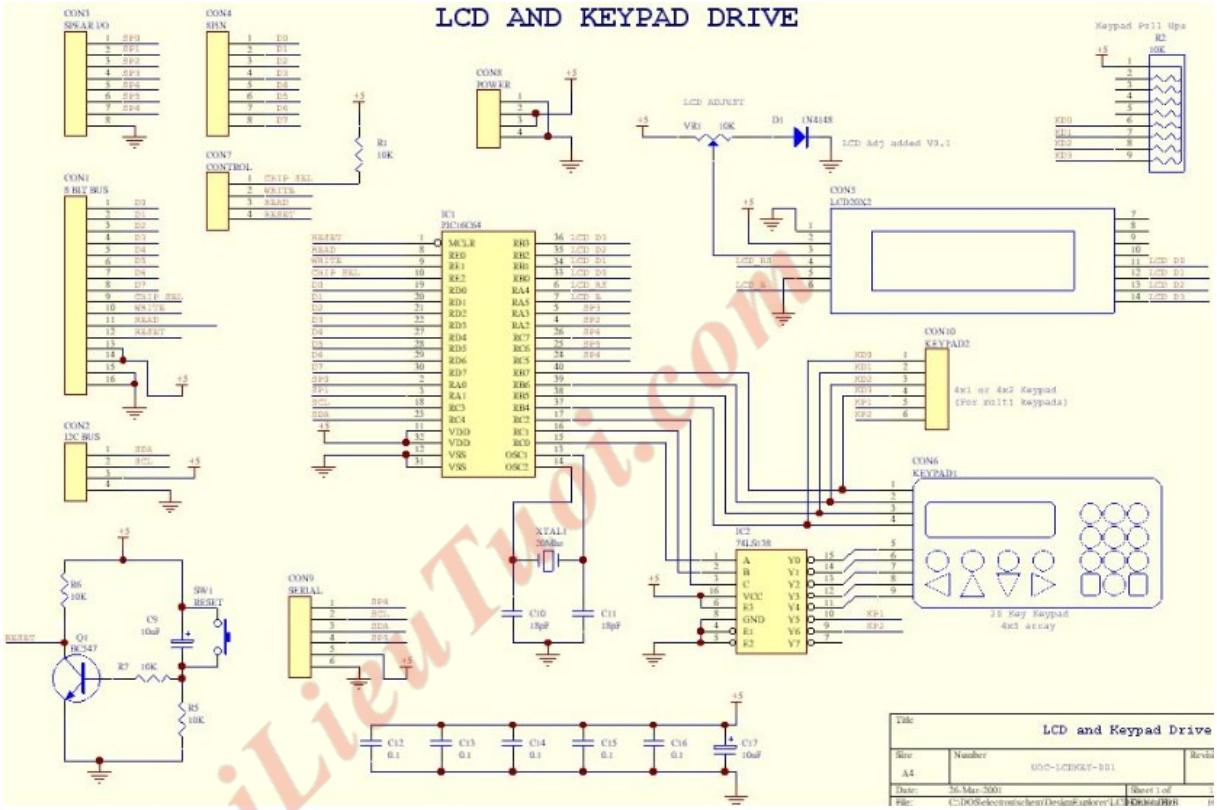
```
byte lcd_read_byte()
{
   byte read_byte;
   set_tris_B(0xFF);        // PORTB = input
   RW = 1;
   delay_cycles(1);
   E  = 1;
   delay_cycles(1);
   read_byte = lcd_data;
   E  = 0;
   set_tris_B(0x00);        // PORTB = output
   return(read_byte);
}

/* Goi 1byte den LCD */
void lcd_send_byte( byte address, byte n )
{
   RS = 0;
   while ( bit_test(lcd_read_byte(),7) ) ;
   RS = address;
   delay_cycles(1);
   RW = 0;
   delay_cycles(1);
   E  = 0;
   lcd_data = n;
   delay_cycles(1);
   E = 1;
   delay_us(2);
   E = 0;
}
/* Khoi tao ban dau cho LCD */
void lcd_init()
{
    byte const lcd_init_string[4] = {0x38, 0x0C, 1 , 6};
    byte i;
    set_tris_B(0x00);
    RS = 0;
    RW = 0;
    E  = 0;
    delay_ms(15);
    for(i=1;i<=3;++i)
       {
         lcd_data = 3;
         delay_cycles(1);
         E = 1;
         delay_us(2);
         E = 0;
         delay_ms(5);
       }
    lcd_data = 2;
    delay_cycles(1);
    E = 1;
    delay_us(2);
    E = 0;
    delay_ms(5);
    for(i=0;i<=3;++i)
      {
         lcd_send_byte(0,lcd_init_string[i]);
      }
}
```

```
/* Nhay den vi tri (x,y) tren LCD,nhay nham y se bao loi */
void lcd_gotoxy( byte x, byte y)
{
   byte address;
   switch(y)
      {
         case 1:  address=0;
                  address+=x-1;
                  lcd_send_byte(0,0x80|address);
                  break;
         case 2:  address=0x40;
                  address+=x-1;
                  lcd_send_byte(0,0x80|address);
                  break;
         default :lcd_init();
                  lcd_putc("ERROR Y POSITION");
                  while(true); // Dung tai day!
      }
}
/* Hien thi ki tu hoac chuoi ra LCD */
void lcd_putc( char c)
{
   lcd_send_byte(1,c);
}
/* Hien thi ki tu hoac chuoi ra LCD */
void lcd_refresh()
{
   lcd_send_byte(0,1);
   lcd_send_byte(0,6);
}
/* Ham main */
void main (void)
{
   set_tris_B(0);     //PORTB = output
   set_tris_D(0);     //PORTD = output

   lcd_init();

   lcd_gotoxy(5,1);
   lcd_putc("WONDERFUL");

   lcd_gotoxy(4,2);
   lcd_putc("PICVIETNAM!");
}
```


Nên gom các hàm trên thành 1 file lcd_8bit.c chẳng hạn, đến khi sử dụng chỉ việc include nó vào cho khỏe...

Bài viết này em sử dụng Lcd 2 hàng để hiển thị giá trị anlog đưa vào ở kênh A và dưa ra các cổng nối tiếp, thời gian để thay dổi giá trị ADC là 1s. (trong ccs chỉ hổ trợ Lcd 2 hàng nhưng basic lại rất hổ trợ rất nhiều LCD, mình muốn viết điều khiển LCD bằng Basic, nhưng lại muốn nhúng nó vào trong ccs phải làm sao mong các bác chỉ giúp)

7.8. LCD and Keypad drive



CODE:

a. Delay

```
/*
 *      Delay functions
 *      See delay.h for details
 *
 *      Make sure this code is compiled with full optimization!!!
 */
#include        "delay.h"

void
DelayMs(unsigned char cnt)
{
#if     XTAL_FREQ <= 2MHZ
        do {
                DelayUs(996);
        } while(--cnt);
#endif

#if XTAL_FREQ >16MHZ
        unsigned char  i;
        do {
                i = 100;
                do {
                        DelayUs(10);
                } while(--i);
        } while(--cnt);
```

Thang8831
http://www.picvietnam.com

```
#else
 #if   XTAL_FREQ > 2MHZ
        unsigned char  i;
        do {
                i = 4;
                do {
                        DelayUs(250);
                } while(--i);
        } while(--cnt);
 #endif
#endif
}
```

b. lcd

```
/*
 *      LCD interface example
 *      Uses routines from delay.c
 *      This code will interface to a standard LCD controller
 *      like the Hitachi HD44780. It uses it in 4 bit mode, with
 *      the hardware connected as follows (the standard 14 pin
 *      LCD connector is used):
 *
 *      PORTB bits 0-3 are connected to the LCD data bits 4-7 (high nibble)
 *      PORTA bit 4 is connected to the LCD RS input (register select)
 *      PORTA bit 5 is connected to the LCD EN bit (enable)
 *
 *      To use these routines, set up the port I/O (TRISA, TRISB) then
 *      call lcd_init(), then other routines as required.
 *
 */
static bit LCD_RS      @ ((unsigned)&PORTA*8+4);       // Register select
static bit LCD_EN      @ ((unsigned)&PORTA*8+5);       // Enable

#define         LCD_STROBE          ((LCD_EN = 1),(LCD_EN=0))

/* write a byte to the LCD in 4 bit mode */

void
lcd_write(unsigned char c)
{
        PORTB = c >> 4;
        LCD_STROBE;
        PORTB = c;
        LCD_STROBE;
 DelayUs(40);
}
/*
 *      Clear and home the LCD
 */
void
lcd_clear(void)
```

```
{
        LCD_RS = 0;
        lcd_write(0x1);
        DelayMs(2);
}
/* write a string of chars to the LCD */

void
lcd_puts(const char * s)
{
        LCD_RS = 1;  // write characters
        while(*s)
                lcd_write(*s++);
}
/*
 * Go to the specified position
 */
void
lcd_goto(unsigned char pos)
{
        LCD_RS = 0;
        lcd_write(0x80+pos);
}
/* initialise the LCD - put into 4 bit mode */

void
lcd_init(void)
{
        LCD_RS = 0;  // write control bytes
        DelayMs(15);  // power on delay
        PORTB = 0x3;            // attention!
        LCD_STROBE;
        DelayMs(5);
        LCD_STROBE;
 DelayUs(100);
        LCD_STROBE;
        DelayMs(5);
        PORTB = 0x2;            // set 4 bit mode
        LCD_STROBE;
 DelayUs(40);
        lcd_write(0x28);        // 4 bit mode, 1/16 duty, 5x8 font
        lcd_write(0x08);        // display off
        lcd_write(0x0F);        // display on, blink curson on
        lcd_write(0x06);        // entry mode
}
```

c. lcdkey1

```
//*****************************************************************
//                    lcdkey1.C
//                  18 FEBRUARY 1999
//
```

```
// Description:
// A test program for the LCD/Keyboard General purpose PCB
//
// Author: Michael Pearce
//         Chemistry Dept, University of Canterbury
//
// Started: NOVEMBER 1998
//*****************************************************************
#include        <pic.h>

#include <string.h>
#define         XTAL_FREQ 20MHZ
#include        "delay.h"
#include "delay.c"
#include        "lcd.h"
#include "lcd.c"
#include "stdio.h"

#define KeyAddr PORTC
#define KeyInPort PORTB

char DecodeKey(char keycode);
void ScrollMessage(char row,const char Message[]);
unsigned char KeyRead(void);

void putch(char c)
{
        char str[2];
        str[0]=c;
        str[1]=0;
        lcd_puts(str);
}
void main(void)
{
        unsigned int count;
        unsigned char tempc,temps[5]="0";
 OPTION=0x00;
 GIE=0;
 TRISA=0xCF;    //-- Control Pins PA4,PA5 as output
 TRISB=0xF0;    //-- Port B bit 0 to 3 as output
 TRISC=0xF8;    //-- Low 3 bits used for keyboard array
        DelayMs(100);
        //-- initialise LCD --
 lcd_init();

 //-- Clear the display --
 lcd_clear();
 //-- Display opening message --
 lcd_puts("Testing the L.C.D.");
 lcd_goto(40);
```

```
 lcd_puts("This is on Row 2 ?");

 //-- 5 second delay ---
 for(count=0;count<1000;count++)
 {
  DelayMs(3);
 }
 lcd_clear();
 lcd_puts("Testing Scrolling..");
 ScrollMessage(1,"    Software Written By Michael Pearce, Chemistry Department, University
of Canterbury     Firmware Ver 1.00   ");
 //-- 5 second delay ---
 for(count=0;count<1000;count++)
 {
  DelayMs(3);
 }

 lcd_clear();
 lcd_puts("Please press Keys..");
 while(1)
 {
           tempc=KeyRead();
           temps[0]=DecodeKey(tempc);
//         itoa(tempc,temps,10);
  lcd_goto(40);
  lcd_puts("Key Pressed = ");
  lcd_puts(temps);
//  printf("Key Num = %d    ",tempc);
  if(temps[0]=='Z')
  {
           ScrollMessage(1,"      You Wally! You are only ment to press one key at a time!!!!
");
  }
          DelayMs(100);
 }
}
//-------------------------------------------------------
void ScrollMessage(unsigned char row,const char Message[])
{
 char TempS[30];
 unsigned int  MHead=0,Done=0,count;
 if(row >1) row=1;
 row=row*40;
 while(Done==0)
 {
  for(count=0;count<20;count++)
  {
           TempS[count]=Message[MHead+count];
           if(Message[MHead+count+1]==0) Done=1;
          }
```

```
            MHead++;
  lcd_goto(row);
  lcd_puts(TempS);
  DelayMs(200);
 }
}
//-----------------------------------------------------
unsigned char KeyRead(void)
{
        unsigned char line,data,result=0;
        for(line=0;line <8;line++)
        {
                KeyAddr=line;      //-- Set Row To Read
                DelayMs(1);
                data = KeyInPort;  //-- Read in the data
                data = data >> 4;  //-- shift to lower nibble
                data     |= 0xF0;     //-- set upper nibble to 1s
                data ^= 0xFF;      //-- invert everything (XOR)
                if(data !=0)
                {
                        result=line<<4;
                        result+=data;
                        line=10;
                }
        }
        return(result);
}
//-----------------------------------------
char DecodeKey(char keycode)
{
 switch(keycode)
 {
  case 1:
   return('v');
  case 2:
   return('E');
  case 4:
   return('0');
  case 8:
   return('C');
  case 17:
   return(0x7E);
  case 18:
   return('3');
  case 20:
   return('2');
  case 24:
   return('1');
  case 33:
   return(0x7F);
```

```
 case 34:
  return('6');
 case 36:
  return('5');
 case 40:
  return('4');
 case 49:
  return(0x5E);
 case 50:
  return('9');
 case 52:
  return('8');
 case 56:
  return('7');
 case 81:
  return('M');
 case 82:
  return('a');
 case 84:
  return('b');
 case 88:
  return('c');
 case 95:       //-- all function keys hit.
  return('Z');

 default:
  break;
 }
 return(0);
}
```

d. lcdkey2

```
//*****************************************************************
//                    lcdkey2.C
//
//      Firmware Version 2.01 For the LCD & KEYPAD Controler
//
// Driver software for the LCD and Keypad Interface for
// the High Speed Pulse Generator.
// I/O is using Parallel Slave Port interface.
// Data written to port from outside is displayed on LCD
// Data read from the port is the last key pressed.
//
// If all 4 Menu buttons pressed - a game (or something may appear)
//
// Author: Michael Pearce
//        Electronics Workshop, Chemistry Department
//        University of Canterbury
//
// Started: 5 November 1998
//
```

```
//******************** UPDATE INFORMATION ************************
// Version 2.01 12 November 1998
//  Made LCD_L0 and LCD_L1 Clear the line first.
//
//****************************************************************
#include        <pic.h>

#include <string.h>

#define         XTAL_FREQ 8MHZ
#include        "delay.h"
#include "delay.c"
#include        "lcd.h"
#include "lcd.c"
#include "stdio.h"

#define KeyAddr PORTC
#define KeyInPort PORTB

#define SCROLLDELAY 100         //-- ms Delay
#define KEYDELAY    100         //-- us Delay between addressing and reading
//********* LCD CONTROL COMMANDS *********
#define  LCD_CLS   0x10         //-- Clear Screen
#define  LCD_CR    0x11         //-- Carrage Return
#define  LCD_LF    0x12         //-- New Line
#define  LCD_CRLF  0x13         //-- CR and NL
#define  LCD_BEEP  0x14         //-- Makes a beep!!
#define  LCD_L0    0x15         //-- Goes to start of Line 0
#define  LCD_L1    0x16         //-- Goes to start of Line 1
#define  LCD_PUTCH 0x1D         //-- Puts next char direct to LCD bypass Buff
#define  LCD_GOTO  0x1E         //-- Moves Cursor to next byte location
#define  LCD_SHOW  0x1F         //-- Update the display
//****************************************
#define BITNUM(adr, bit)      ((unsigned)(&adr)*8+(bit))
static bit  Beeper   @ BITNUM(PORTC, 3);  //- Beeper Output pin
static bit  ChipSel  @ BITNUM(PORTE, 2);  //- Port Select Pin
static bit  TChipSel @ BITNUM(TRISE, 2);
//------- FLAGS ----------
bit  NewData;
bit  BeepNow;
bit  GotoCommand;
bit  GotoNew;
bit  PutchCommand;
bit  PutchNew;
//------- Global Variables, Buffers etc ----------
unsigned char Buffer[41]="Waiting for data.\0                \0";
unsigned char Head,count,GotoData,PutchData;
unsigned char KeyPressed,LastKeyPressed;
//------- Functions Used in Program --------
void interrupt GlobalInterrupt(void);
```

```
unsigned char KeyRead(void);
char DecodeKey(char keycode);
void DisplayData(void);
void Beep(char time);
//void Beep(void);
//void RunGame(void);
void ScrollMessage(unsigned char row,const char Message[]);
//*********************************************************************
//Main
//*********************************************************************
void main(void)
{
 //-- Setup Variables --
 Beeper=0;
 NewData=0;
 Head=0;
 BeepNow=0;
 GotoCommand=0;
 GotoNew=0;
 GotoData=21;
 //-- Set up Ports --
 TRISA=0xCF;   //-- Control Pins PA4,PA5 as output
 TRISB=0xF0;   //-- Port B bit 0 to 3 as output
 TRISC=0xF0;   //-- bits 0 - 2 used for keyboard array bit 3 for Beep
 TRISD=0xFF;   //-- PSP configured as input
 TRISE=0x07;   //-- PSP Controls as input
 PORTD=0x00;   //-- NULL Output to start with
 //-- Set Chip Sel Pin High to indicate Busy to the master
 ChipSel=1;
 TChipSel=0;
 //-- Set Up Interrupts --
 PSPMODE=0;    //-- Disable the PSP Mode
 PSPIE=0;      //-- Disable PSP Interrupt
 PSPIF=0;      //-- Clear Interrupt Flag
 PEIE=1;       //-- Enable Peripheral Interrupts
 GIE=1;        //-- Enable Global Interrupts
 //-- Initialise the LCD --
 lcd_init();
 lcd_clear();
// ScrollMessage(0,"    LCD & Keypad Driver");
// ScrollMessage(1,"    Firmware Version 2.01      ");
 lcd_puts("LCD & Keypad Driver");
 lcd_goto(40);
 lcd_puts("Version 2.01.1");
 DelayMs(200);
 Beep(100);
 Beep(100);
 Beep(60);
 Beep(60);
 Beep(60);
```

```
 Beep(200);
 DisplayData();
 //-- Signal to the Master controller to tell it that ready to recieve!!
 ChipSel=0;     //- Take Pin Low For 10ms
 DelayMs(20);
 PSPIE=1;       //-- Enable PSP Interrupts
 TChipSel=1;    //- Back into High Impedance State
 PSPMODE=1;     //-- Enable PSP Mode
 //-- Main Program Loop --
 while(1)
 {
  if(NewData==1)
  {
          NewData=0;
   DisplayData();
  }
  if(GotoNew==1)        //-- Move Cursor to selected Position
  {
          GotoNew=0;
          lcd_goto(GotoData);
  }
  if(PutchNew==1)      //-- Put Character Directly to LCD Bypassing Buffer
  {
          PutchNew=0;
          putch(PutchData);
         }
  if(BeepNow==1)
  {
   Beep(80);
//   Beep();
   BeepNow=0;
  }
  KeyPressed=KeyRead();
  if(KeyPressed != 0)
  {
   KeyPressed=DecodeKey(KeyPressed);
          if(KeyPressed != LastKeyPressed)
          {
                  Beep(50);
           LastKeyPressed=KeyPressed;
    PORTD=KeyPressed;
          }
   if(KeyPressed==0xFF)
   {
           Beep(100);
           Beep(50);
           Beep(200);
    // RunGame();  //-- All 4 menu keys were depressed
   }
  }
```

```
 else
 {
        LastKeyPressed=0; //-- Key has been released so allow re pressing
 }
}
}
//****************** END OF Main
//*************************************************************************
//GlobalInterrupt - exactly what it says!!
//*************************************************************************
void interrupt GlobalInterrupt(void)
{
       char Icount,TempD;
if(PSPIF)
{
 if(IBF)          // Input Buffer Full
 {
  TempD=PORTD;
  switch(TempD)
  {
   default:
    if(GotoCommand==1)
    {
          GotoData=TempD;
          GotoCommand=0;
          GotoNew=1;
          break;
    }
    if(PutchCommand==1)
    {
          PutchData=TempD;
          PutchNew=1;
          PutchCommand=0;
          break;
    }
    Buffer[Head++]=TempD;     //-- Add Data to the Buffer and point to next
    if(Head < 40) break;           //-- If not off the end then exit ...
    Head=20;                       //-- ... else CR!
    break;
   //*************************************
   case LCD_CRLF:              //-- Combination of CR and LF
    if(Head < 20)
    {
     Head=0;
    }
    else
    {
     Head=20;
    }
   //*************************************
```

```
 case LCD_LF:            //-- Line Feed
  if(Head >=20)
  {
   for(Icount=0;Icount<20;Icount++)
   {
    Buffer[Icount]=Buffer[Icount+20];
    Buffer[Icount+20]=0; //-- Clear the data out
   }
  }
  else
  {
   Head+=20;
  }
  break;
 //*************************************
 case LCD_CR:            //-- Carrage Return
  if(Head < 20)
  {
   Head=0;
  }
  else
  {
   Head=20;
  }
  break;
 //*************************************
 case LCD_BEEP:          //-- Force A Beep
  BeepNow=1;
  break;
 //*************************************
 case LCD_CLS:           //-- Clear the buffer
  for(Icount=0;Icount<40;Icount++)
  {
   Buffer[Icount]=0;
  }
 //*************************************
 case LCD_L0:            //-- Go to Start of 1st Line
  Head=0;                //-- Also Clear the Line
  for(Icount=0;Icount<20;Icount++) Buffer[Icount]=0;
  break;
 //*************************************
 case LCD_L1:            //-- Go to Start of 2nd Line
  Head=20;               //-- Also Clear the Line
  for(Icount=20;Icount<40;Icount++) Buffer[Icount]=0;
  break;
 //*************************************
 case LCD_SHOW:          //-- Update the Display
  NewData=1;
  break;
 //*************************************
```

```
   case LCD_GOTO:         //-- Put Cursor to location
    GotoCommand=1;        //-- Indicated by next byte
    break;
   //*************************************
   case LCD_PUTCH:        //-- Put next character directly to LCD
    PutchCommand=1;       //-- By Passing the Buffer
    break;
  }
        }
 if(!OBF)        // Output Buffer has been read
 {
        PORTD=0x00;       // Put Null into buffer to indicate no key pressed
//      LastKeyPressed=0; // Allow Beep again on same key
 }
 PSPIF=0;
}
}
//****************** END OF GlobalInterrupt
//*********************************************************************
//KeyRead
//*********************************************************************
unsigned char KeyRead(void)
{
        unsigned char line,data,result=0;
        for(line=0;line <8;line++)
        {
                KeyAddr=line;     //-- Set Row To Read
                DelayUs(KEYDELAY);
                data = KeyInPort;  //-- Read in the data
                data = data >> 4;  //-- shift to lower nibble
                data    |= 0xF0;     //-- set upper nibble to 1s
                data ^= 0xFF;     //-- invert everything (XOR)
                if(data !=0)
                {
                        result=line<<4;   //-- Put line number in upper nibble
                        result+=data;     //-- Put Row Bit pattern in lower nibble
                        line=10;          //-- Terminate the loop
                }
        }
 DelayUs(KEYDELAY);  //-- We Bit More Delay
        return(result);     //-- Return the result
}
//****************** END OF KeyRead
//*********************************************************************
//DecodeKey
//*********************************************************************
char DecodeKey(char keycode)
{
 switch(keycode)
 {
```

```
 case 1:
  return('D');
 case 2:
  return('E');
 case 4:
  return('0');
 case 8:
  return('C');
 case 17:
  return('R');
 case 18:
  return('3');
 case 20:
  return('2');
 case 24:
  return('1');
 case 33:
  return('L');
 case 34:
  return('6');
 case 36:
  return('5');
 case 40:
  return('4');
 case 49:
  return('U');
 case 50:
  return('9');
 case 52:
  return('8');
 case 56:
  return('7');
 case 81:
  return('M');
 case 82:
  return('a');
 case 84:
  return('b');
 case 88:
  return('c');
 case 95:       //-- all function keys hit.
  return(0xFF);

 default:
  break;
 }
 return(0);
}
//******************** END OF DecodeKey
```

```
//*********************************************************************
//DisplayData - Displays the Buffer on the LCD Display
//*********************************************************************
void DisplayData(void)
{
// unsigned char count;
 lcd_clear();                          //-- Clear the LCD
 lcd_goto(0);
 for(count=0;count<20;count++)     //-- Display the first line
 {
  if(Buffer[count]==0)
  {
          count=19;                  //-- Check for end of string character
  }
  else
  {
   putch(Buffer[count]);           //-- Display Character on screen
  }
 }
 lcd_goto(40);                        //-- Move Cursor to second Line
 for(;count<40;count++)             //-- Display the second line
 {
  if(Buffer[count]==0)
  {
          count=40;                  //-- Check for end of string character
  }
  else
  {
   putch(Buffer[count]);           //-- Display Character on screen
  }
 }
// lcd_goto(21);                      //-- Put cursor off the screen
}
//****************** END OF DisplayData
//*********************************************************************
//Beep - Does a small Beep using the SCL Pin
//*********************************************************************
void Beep(char time)
//void Beep(void)
{
// char count;
 for(count=0;count<0xFF;count++)
 {
  Beeper=1;
  DelayUs(time);
//  DelayUs(100);
  Beeper=0;
  DelayUs(time);
//  DelayUs(100);
 }
```

```
}
//****************** END OF Beep
//*************************************************************************
//RunGame - Will possibly run a game of scroll a message or something
//*************************************************************************
void RunGame(void)
{
        ScrollMessage(0,"    Sorry No Game!! ");
        DisplayData();
}
//****************** END OF RunGame
//*************************************************************************
//putch - Prints a single character to the LCD
//*************************************************************************
void putch(char c)
{
        char str[2];
        str[0]=c;
        str[1]=0;
        lcd_puts(str);
}
//****************** END OF putch
//*************************************************************************
//ScrollMessage
//*************************************************************************
void ScrollMessage(unsigned char row,const char Message[])
{
 char TempS[21];
 unsigned int  MHead=0,Done=0;//,count;
 if(row >1) row=1;
 row=row*40;
 while(Done==0)
 {
  for(count=0;count<20;count++)
  {
           TempS[count]=Message[MHead+count];
           if(Message[MHead+count+1]==0) Done=1;
          }
          MHead++;
  lcd_goto(row);
  lcd_puts(TempS);
  DelayMs(SCROLLDELAY);
 }
}
//****************** END OF ScrollMessage
//*************************************************************************
//
//*************************************************************************
//****************** END OF
//*************************************************************************
```

```
//
//*********************************************************************
//****************** END OF
```

/* bai tap su dung chuyen doi ADC the hien len LCD va goi qua cong RS232 sau 1s (su dung ngat int_ad) */

```
#include "16f877a.h"
#use delay(clock=4000000)
#fuses nowdt,protect
#use rs232(baud=9600,parity=n,xmit=pin_c6,rcv=pin_c7)

#include "lcd.c"
long int a;
int x,y,z,t;

#int_ad
isr()
{
x=(a/100)+48; //lay ma Ascii cua gia tri ad
y=((a/10)-(a/100))+48;
z=(a%10)+48;
lcd_putc("\f");//xoa man hinh lcd
printf("Gia tri Digital: %ld \n",a);
lcd_putc("Digital:");
lcd_putc(x );
lcd_putc(y );
lcd_putc(z );
delay_ms(1000);
lcd_putc("\f") ;
}
main()
{
lcd_init();
enable_interrupts(int_ad);
enable_interrupts(global);
setup_port_a(all_analog);
setup_adc(adc_clock_internal);
set_adc_channel(0);
printf("Mach ADC \n");
lcd_putc("khoi tao lcd");
while(1)
{
a=read_adc();
}
}
```

Đây là bài viết em sử dung lcd dể thể hiện giá trị Analog dua vào Kênh 0 và đưa nó qua cổng rs232 sử dung 5 thư viện "lcd.c" (trong basic hổ trợ rất nhiều về diều khiển Lcd, em muốn viết dk Lcd bằng Ba sic rồii nhúng nó vào trong CCS mong các bác giúp đỡ)

Trong thư viện của CCS C không chỉ hỗ trợ LCD 2 hàng thôi đâu bạn, trong thư mục Drivers có 16*2, 20*2 và cả LCD graphic nữa.Bạn tìm kĩ trong đó!

Thang8831
http://www.picvietnam.com

7.9.LM335_F877A_LCD1602

```
#include <stddef.h>

#define LCD_RS          PIN_D2
//#define LCD_RW        PIN_A1
#define LCD_EN          PIN_D3

#define LCD_D4          PIN_D4
#define LCD_D5          PIN_D5
#define LCD_D6          PIN_D6
#define LCD_D7          PIN_D7

// misc display defines-
#define Line_1          0x80
#define Line_2          0xC0
#define Clear_Scr       0x01

// prototype statements
#separate void LCD_Init ( void );// ham khoi tao LCD
#separate void LCD_SetPosition ( unsigned int cX );//Thiet lap vi tri con tro
#separate void LCD_PutChar ( unsigned int cX );// Ham viet1kitu/1chuoi len LCD
#separate void LCD_PutCmd ( unsigned int cX) ;// Ham gui lenh len LCD
#separate void LCD_PulseEnable ( void );// Xung kich hoat
#separate void LCD_SetData ( unsigned int cX );// Dat du lieu len chan Data
// D/n Cong
#use standard_io ( B )
#use standard_io (A)

//khoi tao LCD*********************************************
#separate void LCD_Init ( void )
   {
   LCD_SetData ( 0x00 );
   delay_ms(200);       /* wait enough time after Vdd rise >> 15ms */
   output_low ( LCD_RS );// che do gui lenh
   LCD_SetData ( 0x03 );   /* init with specific nibbles to start 4-bit mode */
   LCD_PulseEnable();
   LCD_PulseEnable();
   LCD_PulseEnable();
   LCD_SetData ( 0x02 );   /* set 4-bit interface */
   LCD_PulseEnable();      /* send dual nibbles hereafter, MSN first */
   LCD_PutCmd ( 0x2C );    /* function set (all lines, 5x7 characters) */
   LCD_PutCmd ( 0b00001100);    /* display ON, cursor off, no blink */
   LCD_PutCmd ( 0x06 );    /* entry mode set, increment & scroll left */
   LCD_PutCmd ( 0x01 );    /* clear display */
   }

#separate void LCD_SetPosition ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
   LCD_SetData ( swap ( cX ) | 0x08 );
```

```
   LCD_PulseEnable();
   LCD_SetData ( swap ( cX ) );
   LCD_PulseEnable();
   }
#separate void LCD_PutChar ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
      output_high ( LCD_RS );
      LCD_PutCmd( cX );
      output_low ( LCD_RS );
   }
#separate void LCD_PutCmd ( unsigned int cX )
   {
   /* this subroutine works specifically for 4-bit Port A */
   LCD_SetData ( swap ( cX ) );     /* send high nibble */
   LCD_PulseEnable();
   LCD_SetData ( swap ( cX ) );     /* send low nibble */
   LCD_PulseEnable();
   }
#separate void LCD_PulseEnable ( void )
   {
   output_high ( LCD_EN );
   delay_us ( 3 );         // was 10
   output_low ( LCD_EN );
   delay_ms ( 3 );         // was 5
   }
#separate void LCD_SetData ( unsigned int cX )
   {
   output_bit ( LCD_D4, cX & 0x01 );
   output_bit ( LCD_D5, cX & 0x02 );
   output_bit ( LCD_D6, cX & 0x04 );
   output_bit ( LCD_D7, cX & 0x08 );
   }
```

7.10. LM35_F877A_LCD1602

```
#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)

#use rs232(baud=115200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)

#include <lcd_lib_4bit.c>
#include <lcd_bargraph.c>

#define INTS_PER_SECOND1 9         // (20000000/(4*4*65536))
#define BUFFER_SIZE      5
#define BG_len           5
```

```
byte buffer_count,int_count1;
byte BGpos;
int8 low,high,min,max,room_temp,tong;
int1 do_F;
int8 buffer[BUFFER_SIZE];

void convert_bcd(int8 x);
void xuat_ra_LCD();
void set_blink();
void bao_dong();
void test();
// Chuong trinh ngat
#INT_EXT
void test()
{
  if (do_F == 1) do_F=0;
  else         do_F=1;
}
#INT_TIMER1                    // This function is called every time
void clock1_isr() {            // timer 1 overflows (65535->0), which is
                               // approximately 19 times per second for
   if(--int_count1==0) {       // this program.
     xuat_ra_lcd();
     int_count1 = INTS_PER_SECOND1;
   }
}
//-------------------------------------------------------------------
void main()
{
 min    =15;   //nhiet do min default
 max    =35;   //nhiet do max default
 do_F   =0 ;
 int_count1=INTS_PER_SECOND1;

  trisa = 0xFF;
  trisb = 0x01;
  trisd = 0x00;
    LCD_init();
    lcd_BGInit(0);
    Printf(LCD_putchar,"Init...");
// Khoi tao cho ngat ngoai
    enable_interrupts (INT_EXT);
    ext_int_edge(H_TO_L);
// Khoi tao cho Timer1
    set_timer1(0);
    setup_timer_1(T1_INTERNAL | T1_DIV_BY_4);
    enable_interrupts(INT_TIMER1);
// Khoi tao che do cho bo ADC
    setup_adc_ports(AN0_VREF_VREF);
    setup_adc(ADC_CLOCK_INTERNAL);
```

```
    delay_ms(500);
// Lay mau nhiet do lan dau tien
    room_temp=read_adc();
    lcd_putcmd(0x01);
    Printf(LCD_putchar,"Init OK.");
    delay_ms(500);
    lcd_putcmd(0x01);
    Printf(LCD_putchar,"DKS Group - DEV1");
    LCD_putcmd(0xC0);
    printf(LCD_putchar," T = ");
    enable_interrupts (GLOBAL);
    while(1){}
}
//end main-----------------------------------------------------------
void xuat_ra_LCD()
{
    buffer[buffer_count] = read_adc();
    tong = tong + buffer[buffer_count];
    buffer_count++;
    if (buffer_count == BUFFER_SIZE)
    {
    room_temp = tong / 5;
    BGpos = room_temp / 10;
    lcd_DrawBG(1,10,BG_len,BGpos);
    tong = 0;
    buffer_count = 0;

    if((room_temp > 40) || (room_temp < 15))
      {
       // lcd_putcmd(0xCA);
       // printf(LCD_putchar,"ALARM !");
        PORTB = 0x41;
      }
    else
     {
      //  lcd_putcmd(0xCA);
      //  printf(LCD_putchar,"      ");
        PORTB = 0x81;
      }
    if (do_F==1) room_temp = 2 * room_temp + 32;

    printf("\n\rNhiet do phong: %u",room_temp);
    convert_bcd(room_temp);
    LCD_putcmd(0xC5);
    LCD_putchar(high); LCD_putchar(low);LCD_putchar(0xDF);
    if (do_F==0) printf(LCD_putchar,"C");
    else printf(LCD_putchar,"F");
    }
}
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;
 }
```

7.11. LM335_F877A_LCD1602

```
/* Mach do nhiet do
- MCU    = PIC16F877A
- Sensor = LM335 (co the thay the bang LM35D)
- MAX232 giao tiep may tinh
- LCD1602A de hien thi gia tri nhiet do
Mo ta phan cung:
- Mach cho sensor mac nhu trong Datasheet cua LM335
  Chan V_out noi qua dien tro 1K voi +5V. Chan nay cung duoc noi voi kenh AN0 cua PIC
  Chan Adj noi voi dien tro 10K de tinh chinh
  Chan GND noi dat
- Mach VDK gom co LCD va max232
  LCD noi voi PORTD cua PIC
    RS -> RD2, RW -> GND, E -> RD3
    D4-D7 -> RD4-RD7
  Max232:
    chan10 -> RC6, chan9 -> RC7
    chan8  -> chan3 DB9, chan7 -> chan2 DB9, chan5 DB9 -> GND
  Kenh AN0 cua PIC noi den chan V_out LM335
  Nut bam noi tai chan RB0 -> nhan ngat ngoai
  Thach anh loai 20MHz, tu 22pF
================================================
- Designer: linhnc308@yahoo.com
- Chuc thanh cong cung VDK PIC
*/
#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>
int8 low,high,key,mode,min,max,mode1,i;
int1 blink,on_off,blink_min,blink_max;
int1 do_F;

void convert_bcd(int8 x);
void set_blink();
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 = 0 ;
 mode   =0 ;
 mode1 = 0 ;
 blink=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);  // Chuyen doi tach so tram, chuc, donvi de hien thi len LED 7
    delay_ms(1000);
    LCD_putcmd(0xC0);
    Printf(LCD_putchar," Init OK");

    while(1)
    {
     if (i==25)
       {
        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);
        LCD_putcmd(0xC0);
        printf(LCD_putchar," T = ");
        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 set_blink()
{
  switch(mode)
  {
    case 1: blink_min=1; break;
    case 2: {blink_max=1; blink_min=0;}  break;
    case 3: {mode=0; blink=0; blink_min=0; blink_max=0;}   break;
  }
}
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(" ");
   }
}
```

7.12. lcd_bargraph

```
#separate void LCD_BGInit (int1 type );//BarGraph type: 1 = Vertical or 0 = Horizontal
#separate void LCD_DrawVBG ( BYTE row, BYTE cloum, BYTE height, BYTE BGpos );//
Hien thi BarGraph
#separate void LCD_DrawBG ( BYTE row, BYTE cloum, BYTE len, BYTE BGpos );// Hien
thi BarGraph

//#separate
typedef struct {
int8 b[8]; /* Data */
}T_font;

const T_font VBG[9]={
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // .....
0x1F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // |||||
0x1F,0x1F,0x00,0x00,0x00,0x00,0x00,0x00,  // |||||
0x1F,0x1F,0x1F,0x00,0x00,0x00,0x00,0x00,  // |||||
0x1F,0x1F,0x1F,0x1F,0x00,0x00,0x00,0x00,  // |||||
0x1F,0x1F,0x1F,0x1F,0x1F,0x00,0x00,0x00,  // |||||
0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x00,0x00,  // |||||
0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x00,  // |||||
0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,  // |||||
};
const T_font BG[6] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // .....
0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,  // |....
0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,  // ||...
0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,  // |||..
0x1E,0x1E,0x1E,0x1E,0x1E,0x1E,0x1E,0x1E,  // ||||.
0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,  // |||||
};
#separate void LCD_BGInit (int1 type) //BarGraph type: 1 = Vertical or 0 = Horizontal
  {
  int8 i,j;
  if (type == 1) // 1 = Vertical
    {
      lcd_putcmd(0x00);
      lcd_putcmd(0x40);    // SET CGRAM Address Counter = 0x00;
      for( j = 0; j < 9; j++ )
        For( i = 0; i <= 7; i++ ) LCD_putchar(VBG[j].b[i]); // Luu cac hinh vao CGRAM
    }
  else          // 0 = Horizontal
    {
      lcd_putcmd(0x00);
      lcd_putcmd(0x40);    // SET CGRAM Address Counter = 0x00;
```

```
      for( j = 0; j < 6; j++ )
        For( i = 0; i <= 7; i++ ) LCD_putchar(BG[j].b[i]); // Luu cac hinh vao CGRAM
    }
  }
#separate void LCD_DrawVBG(BYTE row, BYTE colum, BYTE height, BYTE BGpos)// Hien thi BarGraph
  {
//   row   => Starting row for bargraph 0 to 3
//   colum => Starting Column for bargraph 0 to 39+
//   height=> Length of bargraph in chars 1 to 40+
//   BGois => Position of pointer in segments 5 times Length
BYTE i,j,num_full,VBG_index;
    switch (row)
    {
      case 0 : lcd_putcmd(0x80 + colum);
      case 1 : lcd_putcmd(0xC0 + colum);
      case 2 : lcd_putcmd(0xC0 + colum);
      case 3 : lcd_putcmd(0xC0 + colum);
    }
    num_full = BGpos / 8;
    VBG_index = BGpos % 8;
    for (i=0;i < num_full;i++)
      lcd_putchar(8);
      lcd_putchar(VBG_index);
  }
#separate void LCD_DrawBG(BYTE row, BYTE colum, BYTE len, BYTE BGpos) // Hien thi BarGraph
  {
//   row   => Starting row for bargraph 0 to 3
//   colum => Starting Column for bargraph 0 to 39+
//   height=> Length of bargraph in chars 1 to 40+
//   BGois => Position of pointer in segments 5 times Length
BYTE i,j,num_full,BG_index;
    switch (row)
    {
      case 0 : lcd_putcmd(0x80 + colum);
      case 1 : lcd_putcmd(0xC0 + colum);
      case 2 : lcd_putcmd(0xC0 + colum);
      case 3 : lcd_putcmd(0xC0 + colum);
    }
    num_full = BGpos / 5;
    BG_index = BGpos % 5;
    for (i=0;i < num_full;i++)
      lcd_putchar(5);
      lcd_putchar(BG_index);
  }
```

7.13. Chương trình gửi ký tự ra 2x16 LCD dùng CCS C

```
#include "16F877A.h" // PIC16F877A header file
#use delay(clock=4000000) // for 4Mhz crystal
#fuses XT, NOWDT, NOPROTECT, NOLVP // for debug mode
```

```
#define WRITE_DATA 0
#define WRITE_COMMAND 1
#define NCHAR_PER_LINE 16 // max char numbers per line
#define MS10 10 // 10 milliseconds
#define US400 400 // 400 microseconds
#define LCD_RS PIN_A1
#define LCD_RW PIN_A2
#define LCD_E PIN_A3
//////////////////////////////////////////////////////////
//
/* private */ void lcd_write(byte dat, int1 option) {
delay_us(US400);
if (option == WRITE_DATA)
output_high(LCD_RS);
else // option == WRITE_COMMAND
output_low(LCD_RS);
output_low(LCD_RW);
output_b(dat);

output_high(LCD_E);
delay_us(US400);
output_low(LCD_E);
}
//////////////////////////////////////////////////////////
//
void lcd_init(void) {
output_low(LCD_E); // Let LCD E line low

lcd_write(0x38, WRITE_COMMAND); // Set LCD 16x2, 5x7, 8bits data
delay_ms(15);
lcd_write(0x01, WRITE_COMMAND); // Clear LCD display
delay_ms(MS10);
lcd_write(0x0f, WRITE_COMMAND); // Open display & current
delay_ms(MS10);
lcd_write(0x06, WRITE_COMMAND); // Window fixed (Character Entry Mode?)
delay_ms(MS10);
}
//////////////////////////////////////////////////////////
//
void lcd_display_char(int8 line, int8 pos, char ch) {
line = (line == 0) ? 0 : 1;
pos = (pos > NCHAR_PER_LINE) ? NCHAR_PER_LINE : pos;

lcd_write(0x80 + 0x40 * line + pos, WRITE_COMMAND);
lcd_write(ch, WRITE_DATA);
}

//////////////////////////////////////////////////////////
void lcd_display_str(int8 line, char str[], int8 nchars) {
int8 i;
```

```
for (i = 0; i < nchars; i++)
lcd_display_char(line, i, str[i]);
}
//////////////////////////////////////////////////////
/**
* Display characters to a 2x16 LCD
*
* (1) LCD1 to GND
* (2) LCD2 to VDD 5 volts
* (3) LCD4 (RS) - LCD5 (RW) - LCD6 (E) to A1, A2, A3
* (4) LCD7-LCD14 to B0-B7 (bus data)
*
* Ref: http://pic16.com/bbs/dispbbs.asp?boa...ID=5879&page=1
*/
void main(void) {
int8 i;
char LINE1[] = { "SGN Tech" };
char LINE2[] = { "Xin chao" };

lcd_init();
// use of lcd_display_char()
for (i = 0; i < 8; i++)
lcd_display_char(0, i, LINE1[i]);

// use of lcd_display_str
lcd_display_str(1, LINE2, 8);
}
```

a. CCS C có một ví dụ hay hơn: Chỉ cần dùng 4 bits D4-D7 của LCD:

Nguyên văn bởi **ncv**

Chương trình gửi ký tự ra 2x16 LCD dùng CCS C

```
#include "16F877A.h" // PIC16F877A header file
#use delay(clock=4000000) // for 4Mhz crystal
#fuses XT, NOWDT, NOPROTECT, NOLVP // for debug mode
...
}
```

Các bác cho cháu hỏi cau này về CCS: trong hầu hết các ví dụ của CCS C, họ đều dùng printf (RS232) để xuất dữ liệu. Cháu không hiểu làm như thế để làm gì. Cháu cũng không biết vẽ mạch thế nào để xuất hiện các dòng chữ trong printf.

Hàm printf() xuất một chuỗi ra cổng nối tiếp.
Nếu kết nối PIC với máy tính bằng RS232 và cấu hình thích hợp cho cổng thì máy tính sẽ nhận được chuỗi mà bé đặt trong dấu ().
Vẽ mạch thì bé vẽ theo các mạch giao tiếp máy tính mà các chú đã vẽ trên diễn đàn: Chỉ cần 3 sợi dây Rx (receive), Tx (Transfer) và chân Gnd. Cần có con đệm Max232 nằm ở giữa. Nghĩa là PIC - Max232 - PC.
Nhưng tại sao bé không làm các bài tập đơn giản trước như điều khiển LED chớp tắt, hiển thị số trên LED 7 đoạn hay điều khiển nhiều đèn LED chớp theo nhiều kiểu, .. mà lại làm giao tiếp máy tính cho khó khăn vậy?

+Có hai cách lập trình cho LCD: dùng 8bit interface (đơn giản) hoặc 4bit interface (phức tạp hơn)

Thang8831
http://www.picvietnam.com

1.**8bit interface**
2.**4bit interface**

Sử dụng 4 chân D4 - D7 (hoặc D0-D3 <- ít dùng) để truyền thông tin, dữ liệu đến LCD.
- Để điều khiển LCD (Chọn chế độ LCD, bật/tắt hiển thị, bật/tắt/nhấp nháy/di chuyển con trỏ,...): Nhập giá trị tương ứng vào D0-D7,lấy giá trị 4bit cao D4-D7 rồi gởi lệnh yêu cầu LCD thực thi lệnh điều khiển, tiếp theo cho LCD thời gian trễ để thực thi (hoặc hỏi cờ bận xem LCD sẵn sàng thực hiện lệnh tiếp theo chưa?).Tiếp tục, gởi 4bit thấp D0-D3 rồi gởi lệnh yêu cầu LCD thực thi lệnh điều khiển, tiếp theo cho LCD thời gian trễ để thực thi (hoặc hỏi cờ bận xem LCD sẵn sàng thực hiện lệnh tiếp theo chưa?). Nếu trong ứng dụng sử dụng ngắt ngoài thì có thể chuyển sang nối với PORTD hoặc tùy thích.

Bạn muốn kết nối chân LCD với pic theo kiểu nào cũng được, nhưng ko ai làm như vậy cả, lí do:
- Viết code khó, vì mỗi lần xuất dữ liệu điều khiển hay hiển thị đều có mã lệnh riêng, bạn kết nối lộn xộn dẫn đến khó lập trình và ko cơ động.
- Các chân của pic thường được kéo ra ngoài theo từng port. Nếu nối lộn xộn, ko thẩm mỹ.

Bạn muốn nối theo ý bạn, cứ theo nguyên lí hoạt động của LCD thôi. Bạn xem khi nào xuất lệnh điều khiển, khi nào xuất lệnh hiển thị,...kết hợp với thiết kế của bạn mà viết code.

+Có khả năng bạn cần khởi tạo module LCD trước khi đặt chế độ hiển thị (tôi đã viết 1 lần rồi, nhưng bây giờ tìm bằng chức năng search của diễn đàn thì không ra). Tôi nói lại vậy (quy trình cho các module dùng chip điều khiển tương thích HD44780):
- Làm trễ một khoảng thời gian khi mới bật nguồn cho LCD (40 ms từ thời điểm Vdd = 2.7V, hay 15 ms từ thời điểm Vdd = 4.5V)
- Xuất lệnh 0x33 (0x30 đến 0x3F đều ok)
- Chờ khoảng 4.1 ms trở lên
- Xuất lệnh 0x33 (0x30 đến 0x3F đều ok)
- Chờ khoảng 100 us trở lên
- Xuất lệnh 0x33 (0x30 đến 0x3F đều ok)
- Xuất lệnh đặt chế độ (của bạn là 0x38)
- Xuất lệnh tắt màn hình 0x08
- Xuất lệnh xóa màn hình 0x01
- Xuất lệnh đặt chế độ nhập dữ liệu (tăng hay giảm địa chỉ, có dịch màn hình hay không)

Đó là quy trình khởi tạo cho module LCD chưa từng làm việc lần nào với vi điều khiển của bạn. Thông thường, khi module đã được khởi tạo rồi thì những lần sau bạn có thể dùng thẳng các lệnh đặt chế độ mà không cần thực hiện quy trình khởi tạo như trên.

+Để làm chữ có dấu tối thiểu bạn phải có bộ font chữ có dấu, nhưng với LCD kiểu ký tự dạng như 16x2, 16x4 thì hiển thị chữ có dấu rất xấu. Làm cái này trên LCD graphic tốt hơn nhiều.

TL: Mỗi ký tự hiển thị trên lcd thường có kích thước 7 hàng x 5 cột do đó được xác định bởi 7byte, 3 bit cao nhất mỗi byte ko sử dụng. VD:
0x0E
0x0E
0x04
0x04
0x04
0x04
0x0C
Tạo thành chữ J hoa

Thang8831
http://www.picvietnam.com

Để có bộ font của riêng mình công việc của bạn là tìm ra các byte này ứng với mỗi ký tự. Công đoạn này tốn rất nhiều công sức nếu làm bằng tay, thường người ta dùng phần mềm, nhưng các phần mềm này lại thường ko free.

LCD chủ yếu hiển thị bằng CGROM (Character Generator Read Only Memory), tức là bạn chỉ cần cho biết mã ASCII của ký tự, các pattern thể hiện ký tự (5x7, 5x8 hay 5x10) sẽ được lấy từ ROM. Các bộ điều khiển tương thích HD44780 cũng cho phép người dùng tự định nghĩa tối đa 8 ký tự (5x7 hay 5x8) trong vùng CGRAM (Character Generator Random Access Memory). Vùng nhớ này gồm 64 byte, chứa trực tiếp các pattern để tạo ký tự, và người dùng được phép ghi vào. Tuy nhiên, chỉ có thể vẽ thêm 8 ký tự, nên việc bạn muốn hiển thị tiếng Việt trên LCD ký tự là việc rất khó khăn và hạn chế. Nếu bạn vẫn muốn làm thì hãy tìm đọc datasheet của HD44780 và các tutorial trên mạng. Cách làm khá dài dòng nên không tiện nêu trên diễn đàn.

8. LED ma trận

8.1. font_ascii

```
typedef struct {
int8 b[5]; /* Data */
}T_font;

const T_font font[]={
//******************BANG MA ASCII****************
//ascii_code:
0xFF,0xFF,0xFF,0xFF,0xFF,//SPACE    0
0xFF,0xFF,0xA0,0xFF,0xFF,//!        1
0xFF,0xFF,0xF8,0xF4,0xFF,//'    2
0xEB,0x80,0xEB,0x80,0xEB,//#    3
0xDB,0xD5,0x80,0xD5,0xED,//$    4
0xD8,0xEA,0x94,0xAB,0x8D,//%    5
0xC9,0xB6,0xA9,0xDF,0xAF,//&    6
0xFF,0xFF,0xF8,0xF4,0xFF,//'    7
0xFF,0xE3,0xDD,0xBE,0xFF,//(    8
0xFF,0xBE,0xDD,0xE3,0xFF,//)    9
0xD5,0xE3,0x80,0xE3,0xD5,//*    10
0xF7,0xF7,0xC1,0xF7,0xF7,//+    11
0xFF,0xA7,0xC7,0xFF,0xFF,//,    12
0xF7,0xF7,0xF7,0xF7,0xF7,//-    13
0xFF,0x9F,0x9F,0xFF,0xFF,//x    14
0xFF,0xC9,0xC9,0xFF,0xFF,///    15
0xC1,0xAE,0xB6,0xBA,0xC1,//0    16
0xFF,0xBD,0x80,0xBF,0xFF,//1    17
0x8D,0xB6,0xB6,0xB6,0xB9,//2    18
0xDD,0xBE,0xB6,0xB6,0xC9,//3    19
0xE7,0xEB,0xED,0x80,0xEF,//4    20
0xD8,0xBA,0xBA,0xBA,0xC6,//5    21
0xC3,0xB5,0xB6,0xB6,0xCF,//6    22
0xFE,0x8E,0xF6,0xFA,0xFC,//7    23
0xC9,0xB6,0xB6,0xB6,0xC9,//8    24
0xF9,0xB6,0xB6,0xD6,0xE1,//9    25
0xFF,0xC9,0xC9,0xFF,0xFF,//:    26
0xFF,0xA4,0xC4,0xFF,0xFF,////    27
```

```
0xF7,0xEB,0xDD,0xBE,0xFF,//<    28
0xEB,0xEB,0xEB,0xEB,0xEB,//=    29
0xFF,0xBE,0xDD,0xEB,0xF7,//>   30
0xFD,0xFE,0xAE,0xF6,0xF9,//?    31
0xCD,0xB6,0x8E,0xBE,0xC1,//@    32
0x83,0xF5,0xF6,0xF5,0x83,//A    33
0xBE,0x80,0xB6,0xB6,0xC9,//B    34
0xC1,0xBE,0xBE,0xBE,0xDD,//C    35
0xBE,0x80,0xBE,0xBE,0xC1,//D    36
0x80,0xB6,0xB6,0xB6,0xBE,//E    37
0x80,0xF6,0xF6,0xFE,0xFE,//F    38
0xC1,0xBE,0xB6,0xB6,0xC5,//G    39
0x80,0xF7,0xF7,0xF7,0x80,//H         40
0xFF,0xBE,0x80,0xBE,0xFF,//I         41
0xDF,0xBF,0xBE,0xC0,0xFE,//J       42
0x80,0xF7,0xEB,0xDD,0xBE,//K    43
0x80,0xBF,0xBF,0xBF,0xFF,//L    44
0x80,0xFD,0xF3,0xFD,0x80,//M    45
0x80,0xFD,0xFB,0xF7,0x80,//N    46
0xC1,0xBE,0xBE,0xBE,0xC1,//O    47
0x80,0xF6,0xF6,0xF6,0xF9,//P    48
0xC1,0xBE,0xAE,0xDE,0xA1,//Q    49
0x80,0xF6,0xE6,0xD6,0xB9,//R    50
0xD9,0xB6,0xB6,0xB6,0xCD,//S    51
};
//Phan tu hai
const T_font font2[]={
0xFE,0xFE,0x80,0xFE,0xFE,//T    52
0xC0,0xBF,0xBF,0xBF,0xC0,//U    53
0xE0,0xDF,0xBF,0xDF,0xE0,//V    54
0xC0,0xBF,0xCF,0xBF,0xC0,//W    55
0x9C,0xEB,0xF7,0xEB,0x9C,//X   56
0xFC,0xFB,0x87,0xFB,0xFC,//Y89 24    57
0x9E,0xAE,0xB6,0xBA,0xBC,//Z90 25    58
0xFF,0x80,0xBE,0xBE,0xFF,//[    59
0xFD,0xFB,0xF7,0xEF,0xDF,//\    60
0xFF,0xBE,0xBE,0x80,0xFF,//]    61
0xFB,0xE1,0xE0,0xE1,0xFB,//^    62
0x7F,0x7F,0x7F,0x7F,0x7F,//_ 63
0xFF,0xFF,0xF8,0xF4,0xFF,//' 64
0xDF,0xAB,0xAB,0xAB,0xC7,//a    65
0x80,0xC7,0xBB,0xBB,0xC7,//b
0xFF,0xC7,0xBB,0xBB,0xBB,//c
0xC7,0xBB,0xBB,0xC7,0x80,//d
0xC7,0xAB,0xAB,0xAB,0xF7,//e    69
0xF7,0x81,0xF6,0xF6,0xFD,//f
0xF7,0xAB,0xAB,0xAB,0xC3,//g    71
0x80,0xF7,0xFB,0xFB,0x87,//h    72
0xFF,0xBB,0x82,0xBF,0xFF,//i    73
0xDF,0xBF,0xBB,0xC2,0xFF,//j    74
```

```
0xFF,0x80,0xEF,0xD7,0xBB,//k    75
0xFF,0xBE,0x80,0xBF,0xFF,//l    76
0x83,0xFB,0x87,0xFB,0x87,//m    77
0x83,0xF7,0xFB,0xFB,0x87,//n    78
0xC7,0xBB,0xBB,0xBB,0xC7,//o    79
0x83,0xEB,0xEB,0xEB,0xF7,//p    80
0xF7,0xEB,0xEB,0xEB,0x83,//q    81
0x83,0xF7,0xFB,0xFB,0xF7,//r    82
0xB7,0xAB,0xAB,0xAB,0xDB,//s    83
0xFF,0xFB,0xC0,0xBB,0xBB,//t    84
0xC3,0xBF,0xBF,0xDF,0x83,//u    85
0xE3,0xDF,0xBF,0xDF,0xE3,//v    86
0xC3,0xBF,0xCF,0xBF,0xC3,//w    87
0xBB,0xD7,0xEF,0xD7,0xBB,//x    88
0xF3,0xAF,0xAF,0xAF,0xC3,//y    89
0xBB,0x9B,0xAB,0xB3,0xBB,//z    90
0xFB,0xE1,0xE0,0xE1,0xFB,//^    92
0xE3,0xE3,0xC1,0xE3,0xF7,//->93
0xF7,0xE3,0xC1,0xE3,0xE3,//<-94
0xEF,0xC3,0x83,0xC3,0xEF,//95
0xFF,0xFF,0xFF,0xFF,0xFF//BLANK CHAR 96
};
// End of code table
```

8.2. font_ascii2

```
const byte font[51][5]={
//******************BANG MA ASCII****************
//ascii_code:
0xFF,0xFF,0xFF,0xFF,0xFF,//SPACE    0
0xFF,0xFF,0xA0,0xFF,0xFF,//!        1
0xFF,0xFF,0xF8,0xF4,0xFF,//'    2
0xEB,0x80,0xEB,0x80,0xEB,//#    3
0xDB,0xD5,0x80,0xD5,0xED,//$    4
0xD8,0xEA,0x94,0xAB,0x8D,//%    5
0xC9,0xB6,0xA9,0xDF,0xAF,//&    6
0xFF,0xFF,0xF8,0xF4,0xFF,//'    7
0xFF,0xE3,0xDD,0xBE,0xFF,//(    8
0xFF,0xBE,0xDD,0xE3,0xFF,//)    9
0xD5,0xE3,0x80,0xE3,0xD5,//*    10
0xF7,0xF7,0xC1,0xF7,0xF7,//+    11
0xFF,0xA7,0xC7,0xFF,0xFF,//,    12
0xF7,0xF7,0xF7,0xF7,0xF7,//-    13
0xFF,0x9F,0x9F,0xFF,0xFF,//x    14
0xFF,0xC9,0xC9,0xFF,0xFF,///    15
0xC1,0xAE,0xB6,0xBA,0xC1,//0    16
0xFF,0xBD,0x80,0xBF,0xFF,//1    17
0x8D,0xB6,0xB6,0xB6,0xB9,//2    18
0xDD,0xBE,0xB6,0xB6,0xC9,//3    19
0xE7,0xEB,0xED,0x80,0xEF,//4    20
0xD8,0xBA,0xBA,0xBA,0xC6,//5    21
0xC3,0xB5,0xB6,0xB6,0xCF,//6    22
```

```
0xFE,0x8E,0xF6,0xFA,0xFC,//7    23
0xC9,0xB6,0xB6,0xB6,0xC9,//8    24
0xF9,0xB6,0xBE,0xD6,0xE1,//9    25
0xFF,0xC9,0xC9,0xFF,0xFF,//:    26
0xFF,0xA4,0xC4,0xFF,0xFF,////    27
0xF7,0xEB,0xDD,0xBE,0xFF,//<    28
0xEB,0xEB,0xEB,0xEB,0xEB,//=    29
0xFF,0xBE,0xDD,0xEB,0xF7,//>   30
0xFD,0xFE,0xAE,0xF6,0xF9,//?    31
0xCD,0xB6,0x8E,0xBE,0xC1,//@    32
0x83,0xF5,0xF6,0xF5,0x83,//A    33
0xBE,0x80,0xB6,0xB6,0xC9,//B    34
0xC1,0xBE,0xBE,0xBE,0xDD,//C    35
0xBE,0x80,0xBE,0xBE,0xC1,//D    36
0x80,0xB6,0xB6,0xB6,0xBE,//E    37
0x80,0xF6,0xF6,0xFE,0xFE,//F    38
0xC1,0xBE,0xB6,0xB6,0xC5,//G    39
0x80,0xF7,0xF7,0xF7,0x80,//H     40
0xFF,0xBE,0x80,0xBE,0xFF,//I     41
0xDF,0xBF,0xBE,0xC0,0xFE,//J     42
0x80,0xF7,0xEB,0xDD,0xBE,//K    43
0x80,0xBF,0xBF,0xBF,0xFF,//L    44
0x80,0xFD,0xF3,0xFD,0x80,//M    45
0x80,0xFD,0xFB,0xF7,0x80,//N    46
0xC1,0xBE,0xBE,0xBE,0xC1,//O    47
0x80,0xF6,0xF6,0xF6,0xF9,//P    48
0xC1,0xBE,0xAE,0xDE,0xA1,//Q    49
0x80,0xF6,0xE6,0xD6,0xB9//R    50
};
//Phan tu hai
const byte font2[45][5]={
0xD9,0xB6,0xB6,0xB6,0xCD//S    51
0xFE,0xFE,0x80,0xFE,0xFE,//T    52
0xC0,0xBF,0xBF,0xBF,0xC0,//U    53
0xE0,0xDF,0xBF,0xDF,0xE0,//V    54
0xC0,0xBF,0xCF,0xBF,0xC0,//W    55
0x9C,0xEB,0xF7,0xEB,0x9C,//X   56
0xFC,0xFB,0x87,0xFB,0xFC,//Y89 24    57
0x9E,0xAE,0xB6,0xBA,0xBC,//Z90 25    58
0xFF,0x80,0xBE,0xBE,0xFF,//[    59
0xFD,0xFB,0xF7,0xEF,0xDF,//\    60
0xFF,0xBE,0xBE,0x80,0xFF,//]    61
0xFB,0xE1,0xE0,0xE1,0xFB,//^    62
0x7F,0x7F,0x7F,0x7F,0x7F,//_ 63
0xFF,0xFF,0xF8,0xF4,0xFF,//' 64
0xDF,0xAB,0xAB,0xAB,0xC7,//a    65
0x80,0xC7,0xBB,0xBB,0xC7,//b
0xFF,0xC7,0xBB,0xBB,0xBB,//c
0xC7,0xBB,0xBB,0xC7,0x80,//d
0xC7,0xAB,0xAB,0xAB,0xF7,//e    69
```

```
0xF7,0x81,0xF6,0xF6,0xFD,//f
0xF7,0xAB,0xAB,0xAB,0xC3,//g    71
0x80,0xF7,0xFB,0xFB,0x87,//h    72
0xFF,0xBB,0x82,0xBF,0xFF,//i    73
0xDF,0xBF,0xBB,0xC2,0xFF,//j    74
0xFF,0x80,0xEF,0xD7,0xBB,//k    75
0xFF,0xBE,0x80,0xBF,0xFF,//l    76
0x83,0xFB,0x87,0xFB,0x87,//m    77
0x83,0xF7,0xFB,0xFB,0x87,//n    78
0xC7,0xBB,0xBB,0xBB,0xC7,//o    79
0x83,0xEB,0xEB,0xEB,0xF7,//p    80
0xF7,0xEB,0xEB,0xEB,0x83,//q    81
0x83,0xF7,0xFB,0xFB,0xF7,//r    82
0xB7,0xAB,0xAB,0xAB,0xDB,//s    83
0xFF,0xFB,0xC0,0xBB,0xBB,//t    84
0xC3,0xBF,0xBF,0xDF,0x83,//u    85
0xE3,0xDF,0xBF,0xDF,0xE3,//v    86
0xC3,0xBF,0xCF,0xBF,0xC3,//w    87
0xBB,0xD7,0xEF,0xD7,0xBB,//x    88
0xF3,0xAF,0xAF,0xAF,0xC3,//y    89
0xBB,0x9B,0xAB,0xB3,0xBB,//z    90
0xFB,0xE1,0xE0,0xE1,0xFB,//^    92
0xE3,0xE3,0xC1,0xE3,0xF7,//->93
0xF7,0xE3,0xC1,0xE3,0xE3,//<-94
0xEF,0xC3,0x83,0xC3,0xEF,//95
0xFF,0xFF,0xFF,0xFF,0xFF//BLANK CHAR 96
};
// End of code table
```

8.3. led matrix_Ngat ngoai_COM

```
//+===Chuong trinh LED matrix display===========+
//| Thiet ke: Nguyen Chi Linh - DT8K47 - DHBKHN |
//| MCU: PIC16F88 (4K FLASH ROM, 256K EEPROM)   |
//| Cac IC khac:   74154 - demux/decoder 1-of-16 |
//|              74595 - Ghi dich 8bit       |
//+=============================================+
#include <16f88.h>
#include <defs_88.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, MCLR, NOBROWNOUT, NOLVP, NOCPD, NOWRT,
NODEBUG, NOPROTECT, NOFCMEN, NOIESO
#use delay(clock=20000000)
#use      rs232(baud=9600,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)      //Baud_min=4800
Baud_max=115200

#include <input.c>
#include <font_ascii.c> //File chua bo font ma hoa ky tu ASCII

// Dinh nghia cac chan cho ket noi 74595
#bit clk = 0x06.1    //RB0
#bit data = 0x06.3   //RB1
```

```
#bit latch = 0x06.4  //RB3

// Bo nho dem man hinh hien thi
int8 buff_disp[17];     //Bo nho dem cho man hinh LED
int8 max_char=117;       //SO ky tu hien thi toi da
int8 time=5;            //Bien quy dinh toc do chu chay
int1 text_eeprom=0;
int8 chon=0;
int8 address;
int8 choose_text;

//=========KHAI BAO CAC CHUONH TRINH CON================
int8 doc_eeprom(int8 addr);
void send_2_595(int8 temp);
void display();
void copy_2_ram1(int8 index_char);
void copy_2_ram2(int8 index_char);
void update_eeprom();
void convert_bcd(int8 x);
//=============================================
#INT_EXT
EXT_ISR() {
disable_interrupts(GLOBAL);
clear_interrupt(int_ext);
chon++;
if(chon==3) chon = 0;
if (RB7 == 0) RB7=1;
else RB7 = 0;
choose_text = 0;
enable_interrupts(GLOBAL);
}
//===========Chuong trinh chinh================
void main() {
int8 i,j,k;
#bit update_rom = 0x06.6
char const a[119]= " Hello World.LED Matrix PIC16F88 - 74154 - 74595. Bang thong tin dien tu.Nguyen Chi Linh-DT8 DAI HOC BACH KHOA HA NOI ";
char const b[119]= " HAPPY NEW YEAR *2006* - CHUC MUNG NAM MOI - Chuc Mung Nam Moi - Happy new year. linhnc308@yahoo.com 1234567890      ";
char const c[119]= " You like a little flame in my heart. When I see you, the flame is like up. Because I love you. Because I LOVE YOU    ";
char const adc[6] = " ADC=";
//=============================================
  TRISA=0x10;           // Thiet lap chan vao ra
  TRISB=0b00100101;
//==Thiet lap ngat ngoai 0 ==================
  enable_interrupts(INT_EXT);
  ext_int_edge(H_TO_L);
  enable_interrupts(GLOBAL);
//=============================================
```

```
  setup_adc_ports(sAN4);       //Chon kenh AN4 nhung ko hieu sao can them phan khai bao
   setup_adc(ADC_CLOCK_INTERNAL);// ben duoi dechon dung kenh AN4 cho no chay
dung
  //Chon kenh AN4 clear cac bit tai thang ghi ADCON1 (chs0 : 2)
  chs0=0;    //Clear bit 1f.3
  chs1=0;    //Clear bit 1f.4
  chs2=1;    //Clear bit 1f.5
  delay_ms(10);
//=============================================
for(i=0;i<6;++i)
  write_eeprom(0xf0+i,adc[i]);
for(i=0;i<117;++i)
  write_eeprom(i,a[i]);
  write_eeprom(0xff,max_char);     // Luu so ky tu toi da vao ROM
if(update_rom==1)              //Kiem tra cong tac cap nhat du lieu
  update_eeprom();           //Goi chuong trinh con cap nhat(giao tiep qua cong COM)
hien_thi:
for (i=0;i<=16;i++)               // Clear RAM of buff_disp
  buff_disp[i]=0xff;
//Doan chuong trinh nay se hien thi noi dung ban tin luu trong EEPROM
  address = label_address(hien_thi);
  i=0;j=0;
while(1){
 TRISB0 = 1;
 for (i=0;i<=max_char;i++)    // Begin of text
  {
  if(choose_text==0) {choose_text=1; goto hien_thi;}
  if(text_EEPROM==0)
  {
    switch(chon)
    {
     case 0: j=a[i]-32; break;
     case 1: j=b[i]-32; break;
     case 2: j=c[i]-32; break;
     case 3: chon=0;    break;
    }
  }
  else
   j=read_eeprom(i)-32;
  if(j < 51)
   copy_2_ram1(j);
         else
         {
          j=j-51;
          copy_2_ram2(j);
  }
 }
 k=read_adc();
 convert_bcd(k);
  for(i=0;i<8;++i)
```

```
   {
    k = doc_eeprom(0xf0 + i)-32;
    copy_2_ram1(k);
   }
}
}
//========= END MAIN ==============

// === CAC CHUONG TRINH CON =========
//=======Gui du lieu theo duong noi tiep toi 595============
void send_2_595(int8 temp) {
#bit flag_bit = temp.7     // bien temp la du lieu 8-bit can gui
int8 i;
clk=0;
 for(i=0;i<8;i++)
 {
   if(flag_bit)
   data=1;       //bit 1
   else data=0;   //bit 0
   clk=1;
   clk=0;
   temp<<=1; // Dich trai 1 bit
 }
 latch=1;     //Chot du lieu
 latch=0;
}
//===========Chuong trinh con hien thi=========
void display() {
int8 count,column_count;
int8 i;
time = read_adc()/10;      // Viec doc gia tri ADC truoc khi hien thi lam cho viec thay doi
                    // toc do chu chay linh hoat hon, truc tiep tha doi
for (i=0;i<=time;i++)              //Toc do chu chay thay doi boi bien time
{
 column_count=0;                   //Bien dem so cot, xem da quet het 16 cot chua
 for(count=16;count>0;count--)
  {
   send_2_595(buff_disp[count]);
   PORTA=column_count;
   delay_us(500);
   column_count++;
  }
 }
}
//========Copy to Ram1====================
void copy_2_ram1(int8 index_char) {
int8 i,j;

for (j=0;j<=5;j++)
{                                  // Dich RAM
```

```
 for (i=16;i>0;i--)
  buff_disp[i]= buff_disp[i-1];        // Dich RAM sang trai
  buff_disp[0]= font[index_char].b[j];  // Luu ma ascii vao RAM
  display();  // Goi hien thi
}
  buff_disp[0]=0xff;
}
//========Copy to Ram 2==================
void copy_2_ram2(int8 index_char) {

int8 i,j;

for (j=0;j<=5;j++)
{
 for (i=16;i>0;i--)                    // Dich RAM
  buff_disp[i]= buff_disp[i-1];          //Dich RAM sang trai
  buff_disp[0]=font2[index_char].b[j];    //Luu ma ascii vao RAM
  display();                            // Goi hien thi
}
  buff_disp[0]=0xff; // Them mot khoang trang giua hai ky tu
}
//=========Update EEPROM==================
void update_eeprom() {
   byte i,j,addr,max;
   char temp;
   char string[64];
// Hien thi noi dung cua EEPROM
      printf("\r\n256 byte EEPROM of PIC16F88:\r\n");              // Display contents of the first
64
      for(i=0; i<=15; ++i)                                  // bytes of the data EEPROM in hex
      {
        for(j=0; j<=15; ++j)
        printf( "%2x ", doc_eeprom( i*16+j ) );
        printf("\n\r");
      }
// Hien thi noi dung ban tin
      i=0;
      do
      {
       temp = doc_eeprom(i);
       printf( "%C", temp);
       i++;
      } while (temp != 0xff);
//--------Ket thuc      ----------
      printf("\r\nTong so chu: %2u", doc_eeprom(0xff));
      printf("\r\n\nCo thay doi ban tin ko(Y/N)? "); temp=getc();//temp = getc();
      if (temp == 'y' || temp == 'Y')
      {
        printf("\r\nSo chu hien thi moi la: ");
        max_char=gethex();
```

```
        write_eeprom(0xff,max_char);
        printf("\r\nDia chi EEPROM can thay doi: ");
        addr = gethex();
       if (addr >= max_char)
        write_eeprom(0xff,addr);
        printf("\r\nSo ky tu them vao: ");
        max = gethex();                          // Tra ve gia tri Hexa
       if(max >= max_char)
        write_eeprom(0xff,max);   // Cap nhat so ky tu
        printf("\r\nNew: ");
        get_string(string,max+1);
       for (i=0;i<max;i++)   //bat dau qua trinh ghi vao ROM (cap nhat du lieu moi)
        {
          write_eeprom(addr,string[i]);
          addr=addr+1;
        }
        text_eeprom=1;
      }
      else
      {
        printf("Tro ve !");    // Ket thuc viec cap nhat, tro ve hien thi
        text_eeprom = 0;
      }
}
//==============READ EEPROM=====================
int8 doc_eeprom(int8 addr)
{
  EEADR=addr;
  RD=1;
  return(EEDATA);
}
//=======Chuyen gia tri hex ra so ASCII=========
void convert_bcd(int8 x)
{
int8 temp;
int8 a;
  temp=x%10;                    //chia lay phan du, so hang don vi
  write_eeprom(0xf7,temp+0x30);    //Cong them 0x30 de tra ve gia tri SCII
  a=x/10;                       //tach hang tram va hang chuc
  temp=a%10;                    //tach so hang chuc
  write_eeprom(0xf6,temp+0x30);
  temp=x/100;
  write_eeprom(0xf5,temp+0x30);
}
```

8.4. led matrix ket noi RS232

```
//+===Chuong trinh LED matrix display===========+
//| Thiet ke: Nguyen Chi Linh - DT8K47 - DHBKHN |
//| MCU: PIC16F88 (4K FLASH ROM, 256K EEPROM)   |
//| Cac IC khac:   74154 - demux/decoder 1-of-16 |
//|              74595 - Ghi dich 8bit        |
```

```
//+=============================================+

#include <16f88.h>
#include <defs_88.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, MCLR, NOBROWNOUT, NOLVP, NOCPD, NOWRT, NODEBUG, NOPROTECT, NOFCMEN, NOIESO
#use delay(clock=20000000)
#use      rs232(baud=9600,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)      //Baud_min=4800 Baud_max=115200

#include <input.c>
#include <font_ascii.c> //File chua bo font ma hoa ky tu ASCII

// Dinh nghia cac chan cho ket noi 74595
#bit data = 0x06.1   //RB1
#bit clk = 0x06.0    //RB0
#bit latch = 0x06.3  //RB3

// Bo nho dem man hinh hien thi
int8 buff_disp[17];     //Bo nho dem cho man hinh LED
int8 max_char=117;      //SO ky tu hien thi toi da
int8 time=5;            //Bien quy dinh toc do chu chay
//=========KHAI BAO CAC CHUONH TRINH CON================
int8 doc_eeprom(int8 addr);
void send_2_595(int8 temp);
void display();
void copy_2_ram1(int8 index_char);
void copy_2_ram2(int8 index_char);
void update_eeprom();
void convert_bcd(int8 x);
//=============================================
//===========Chuong trinh chinh================
void main() {
int8 i,j;
#bit update_flag = 0x06.4
char const a[119]= " Hello Wolrd.LED Matrix PIC16F88 - 74154 - 74595. Bang thong tin dien tu.Nguyen Chi Linh-DT8 DAI HOC BACH KHOA HA NOI ";// Ky tu NULL duoc them vao cuoi
char const adc[6] = " ADC=";                                  1
//=============================================
  TRISA=0x10;           // Thiet lap chan vao ra
  TRISB=0b00100100;
  setup_adc_ports(sAN4);       //Chon kenh AN4 nhung ko hieu sao can them phan khai bao
   setup_adc(ADC_CLOCK_INTERNAL);// ben duoi dechon dung kenh AN4 cho no chay dung
  //Chon kenh AN4 clear cac bit tai thang ghi ADCON1 (chs0 : 2)
  chs0=0;    //Clear bit 1f.3
  chs1=0;    //Clear bit 1f.4
  chs2=1;    //Clear bit 1f.5
```

```
  delay_ms(10);
//=========================================
for(i=0;i<6;++i)
  write_eeprom(0xf0+i,adc[i]);
for(i=0;i<117;++i)
  write_eeprom(i,a[i]);
  write_eeprom(0xff,max_char);     // Luu so ky tu toi da vao ROM
//=======BEGIN DISPLAY FUNCTION==============
//Doanchuong trinh nay se hien thi noi dung ban tin luu trong EEPROM
while(1){
 for (i=0;i<=16;i++)              // Clear RAM of buff_disp
  buff_disp[i]=0xff;
 if(update_flag==1)              //Kiem tra cong tac cap nhat du lieu
  update_eeprom();           //Goi chuong trinh con cap nhat(giao tiep qua cong COM)
 //  time = read_adc()/10;      //Cap nhat bien quy dinh toc do chu chay tren man hinh
 for (i=0;i<=max_char;i++)    // Begin of text
  {
  j=read_eeprom(i)-32;       //Lay gia tri ASCII cua ky tu
  //if(j==51)   copy_2_ram1(51);
          if(j < 51)
    copy_2_ram1(j);
          else
          {
           j=j-51;
           copy_2_ram2(j);
  }
 }
j=read_adc();
convert_bcd(j);
 for(i=0;i<8;++i)
  {
   j = doc_eeprom(0xf0 + i)-32;
   copy_2_ram1(j);
  }
}
}
//========= END MAIN ==============
// === CAC CHUONG TRINH CON =========
//=======Gui du lieu theo duong noi tiep toi 595============
void send_2_595(int8 temp) {
#bit flag_bit = temp.7     // bien temp la du lieu 8-bit can gui
int8 i;
clk=0;
 for(i=0;i<8;i++)
 {
  if(flag_bit)
  data=1;        //bit 1
  else data=0;   //bit 0
  clk=1;
  clk=0;
```

```
  temp<<=1; // Dich trai 1 bit
 }
 latch=1;      //Chot du lieu
 latch=0;
}
//===========Chuong trinh con hien thi=========
void display() {
int8 count,column_count;
int8 i;

time = read_adc()/10;      // Viec doc gia tri ADC truoc khi hien thi lam cho viec thay doi
                    // toc do chu chay linh hoat hon, truc tiep tha doi
for (i=0;i<=time;i++)              //Toc do chu chay thay doi boi bien time
{
 column_count=0;                   //Bien dem so cot, xem da quet het 16 cot chua
 for(count=16;count>0;count--)
  {
   send_2_595(buff_disp[count]);
   PORTA=column_count;
   delay_ms(1);
   column_count++;
  }
 }
}
//========Copy to Ram1====================
void copy_2_ram1(int8 index_char) {
int8 i,j;

for (j=0;j<=5;j++)
{                                  // Dich RAM
 for (i=16;i>0;i--)
  buff_disp[i]= buff_disp[i-1];        // Dich RAM sang trai
  buff_disp[0]= font[index_char].b[j];  // Luu ma ascii vao RAM
  display();  // Goi hien thi
}
  buff_disp[0]=0xff;
}
//========Copy to Ram 2==================
void copy_2_ram2(int8 index_char) {

int8 i,j;
for (j=0;j<=5;j++)
{
 for (i=16;i>0;i--)                    // Dich RAM
  buff_disp[i]= buff_disp[i-1];          //Dich RAM sang trai
  buff_disp[0]=font2[index_char].b[j];    //Luu ma ascii vao RAM
  display();                           // Goi hien thi
}
  buff_disp[0]=0xff; // Them mot khoang trang giua hai ky tu
}
```

```
//=========Update EEPROM==================
void update_eeprom() {
  byte i,j,addr,max;
  char temp;
  char string[64];
// Hien thi noi dung cua EEPROM
     printf("\r\n256 byte EEPROM of PIC16F88:\r\n");            // Display contents of the first
64
     for(i=0; i<=15; ++i)                                      // bytes of the data EEPROM in hex
     {
       for(j=0; j<=15; ++j)
       printf( "%2x ", doc_eeprom( i*16+j ) );
       printf("\n\r");
     }
// Hien thi noi dung ban tin
     i=0;
     do
     {
      temp = doc_eeprom(i);
      printf( "%C", temp);
      i++;
     } while (temp != 0xff);
//--------Ket thuc     ----------
     printf("\r\nTong so chu: %2u", doc_eeprom(0xff));
     printf("\r\n\nCo thay doi ban tin ko(Y/N)? "); temp=getc();//temp = getc();
     if (temp == 'y' || temp == 'Y')
     {
       printf("\r\nSo chu hien thi moi la: ");
       max_char=gethex();
       write_eeprom(0xff,max_char);
       printf("\r\nDia chi EEPROM can thay doi: ");
       addr = gethex();
      if (addr >= max_char)
       write_eeprom(0xff,addr);
       printf("\r\nSo ky tu them vao: ");
       max = gethex();                          // Tra ve gia tri Hexa
      if(max >= max_char)
       write_eeprom(0xff,max);   // Cap nhat so ky tu
       printf("\r\nNew: ");
       get_string(string,max+1);
      for (i=0;i<max;i++)   //bat dau qua trinh ghi vao ROM (cap nhat du lieu moi)
       {
         write_eeprom(addr,string[i]);
         addr=addr+1;
       }
     }
     else printf("Tro ve !");     // Ket thuc viec cap nhat, tro ve hien thi
}
//=============READ EEPROM====================
int8 doc_eeprom(int8 addr)
```

```
{
  EEADR=addr;
  RD=1;
  return(EEDATA);
}
//=======Chuyen gia tri hex ra so ASCII=========
void convert_bcd(int8 x)
{
int8 temp;
int8 a;
  temp=x%10;                    //chia lay phan du, so hang don vi
  write_eeprom(0xf7,temp+0x30);    //Cong them 0x30 de tra ve gia tri SCII
  a=x/10;                       //tach hang tram va hang chuc
  temp=a%10;                    //tach so hang chuc
  write_eeprom(0xf6,temp+0x30);
  temp=x/100;
  write_eeprom(0xf5,temp+0x30);
}
```

8.5. led matrix (595 va 154) ket noi rs232

```
#include <16f88.h>
#include <defs_88.h>
#device *=16 ADC=8
#FUSES NOWDT, HS, NOPUT, MCLR, NOBROWNOUT, NOLVP, NOCPD, NOWRT,
NODEBUG, NOPROTECT, NOFCMEN, NOIESO
#use delay(clock=10000000)
#use rs232(baud=4800,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)

#include <input.c>
#include <font_ascii.c>

// Dinh nghia cac chan cho ket noi 74595
#bit data = 0x06.1
#bit clk = 0x06.0
#bit latch = 0x06.3

// Bo nho dem man hinh hien thi
int8 buff_disp[17];  //Bo nho dem cho man hinh LED
int8 max_char=24;     //SO ky tu hien thi toi da
int8 time;          //Bien quy dinh toc do chu chay

//=========KHAI BAO CAC CHUONH TRINH CON===============
int8 doc_eeprom(int8 addr);
void send_i2c(int8 temp);
void display();
void copy_2_ram1(int8 index_char);
void copy_2_ram2(int8 index_char);
void update_eeprom();

//=============================================
//===========Chuong trinh chinh================
```

```
void main() {
int8 i,j;
#bit update_flag = 0x06.4
char const a[25]= " Bang thong tin dien tu ";// Ky tu NULL duoc them vao cuoi

TRISA=0x10; // Thiet lap lam dau ra
trisb=0x04;

  setup_adc_ports(sAN4);
  setup_adc(ADC_CLOCK_INTERNAL);

//ra4=1;
//max_char=sizeof(a);
for(i=0;i<25;++i)
  write_eeprom(i,a[i]);

write_eeprom(0xff,max_char);// Luu so ky tu toi da vao ROM
//max_char=doc_eeprom(0xff);   //Doc lay gia tri
do {
 for (i=0;i<=16;i++)        // Clear RAM of buff_disp
  buff_disp[i]=0xff;

 if(update_flag)
  update_eeprom();

 for (i=0;i<=max_char;i++)    // Begin of text
  {
  j=doc_eeprom(i)-32;
        if (j<=51)
    copy_2_ram1(j);
        else
        {
         j=j-52;
         copy_2_ram2(j);
   }
  }
} while(true);
//ra4=1;
}
//========= END MAIN ==============
// === CAC CHUONG TRINH CON =========
//=======Gui du lieu theo duong I2C============
void send_i2c(int8 temp) {
#bit flag_bit = temp.7
int8 i;
 for(i=0;i<8;i++)
 {
  if(flag_bit)
  data=1;     //bit 1
  else data=0;   //bit 0
```

```
  clk=1;
  delay_us(1);
  clk=0;
  temp<<=1; // Dich trai 1 bit
 }
 latch=1;
 latch=0;
}
//===========Chuong trinh con hien thi=========
void display() {
int8 disp_count,column_count;
int8 i;
 delay_us(10);
 time = read_adc();
 time = (time/255)*5;

for (i=0;i<=time;i++) //Toc do chu chay thay doi boi bien time
{
 column_count=0;
 for(disp_count=16;disp_count>0;disp_count--)
  {
   send_i2c(buff_disp[disp_count]);
   PORTA=column_count;
   delay_ms(2);
   //send_i2c(0xff);
   column_count++;
  }
 }
}
//========Copy to Ram1=====================
void copy_2_ram1(int8 index_char) {
int8 i,j;

for (j=0;j<=5;j++)
{  // Dich RAM
 for (i=16;i>0;i--)
 { buff_disp[i]= buff_disp[i-1]; }  //Dich RAM sang trai
  //Luu ma ascii vao RAM
  buff_disp[0]=font[index_char].b[j];
  // Goi hien thi
  display();
 }
  buff_disp[0]=0xff;
}
//========Copy to Ram 2====================
void copy_2_ram2(int8 index_char) {

int8 i,j;
for (j=0;j<=5;j++)
{  // Dich RAM
```

```
 for (i=16;i>0;i--)
 { buff_disp[i]= buff_disp[i-1]; }  //Dich RAM sang trai
  //Luu ma ascii vao RAM
  buff_disp[0]=font2[index_char].b[j];
  // Goi hien thi
  display();
}
  buff_disp[0]=0xff; // Them mot khoang trang giua hai ky tu
}
//=========Update EEPROM==================
void update_eeprom() {
   byte i,j,addr,max;
   char string[64];
   //char answer;
#locate i=0x33
#locate j=0x34
#locate addr=0x35
#locate max=0x36
#locate string = 0x37
      //ra4=0;
       printf("\r\n\n256 byte EEPROM of PIC16F88:\r\n (from 0x00 to 0xFF) \r\n");              //
Display contents of the first 64
      for(i=0; i<=15; ++i)                    // bytes of the data EEPROM in hex
      {
        for(j=0; j<=15; ++j)
        printf( "%2x ", doc_eeprom( i*16+j ) );
        printf("\n\r");
      }
      //ra4=1;
      printf("\r\nTong so chu cua ban tin: %2x", doc_eeprom(0xff));
      printf("\r\nSo chu hien thi moi la: ");
      max_char=gethex();
      write_eeprom(0xff,max_char);
       printf("\r\nDia chi EPROM can thay doi(2 chu so hexa): ");
       addr = gethex();
       if (addr >= max_char)
        write_eeprom(0xff,addr);

       printf("\r\nSo ky tu toi da them vao: ");
       max = gethex();                          // Tra ve gia tri Hexa
       if(max >= max_char)
        write_eeprom(0xff,max);   // Cap nhat so ky tu

       printf("\r\nKy tu moi: ");
       get_string(string,max+1);

       for (i=0;i<max;i++)   //bat dau qua trinh ghi vao ROM (cap nhat du lieu moi)
        {
          write_eeprom(addr,string[i]);
          addr=addr+1;
```

```
        }
    // Ket thuc viec cap nhat, tro ve hien thi
}
//=============READ EEPROM====================
int8 doc_eeprom(byte addr)
{
  EEADR=addr;
  RD=1;
  return(EEDATA);
}
```

8.6. led matrix ver 1.2

```
#include <16f88.h>
#include <defs_88.h>
#fuses NOWDT,HS, NOPUT, NOPROTECT
#use delay(clock=10000000)
#use rs232(baud=4800,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=9)

int8 buff_disp[17];
//char string[50];

// Khai bao cac ham Ngat
//#int_RDA
//RDA_isr()
//{
//}
typedef struct {
int8 b[5]; /* Data */
}T_font;
const T_font font[]={
//******************BANG MA ASCII****************
//ascii_code:
0xFF,0xFF,0xFF,0xFF,0xFF,//SPACE    0
0xFF,0xFF,0xA0,0xFF,0xFF,//!        1
0xFF,0xFF,0xF8,0xF4,0xFF,//'    2
0xEB,0x80,0xEB,0x80,0xEB,//#    3
0xDB,0xD5,0x80,0xD5,0xED,//$    4
0xD8,0xEA,0x94,0xAB,0x8D,//%    5
0xC9,0xB6,0xA9,0xDF,0xAF,//&    6
0xFF,0xFF,0xF8,0xF4,0xFF,//'    7
0xFF,0xE3,0xDD,0xBE,0xFF,//(    8
0xFF,0xBE,0xDD,0xE3,0xFF,//)    9
0xD5,0xE3,0x80,0xE3,0xD5,//*    10
0xF7,0xF7,0xC1,0xF7,0xF7,//+    11
0xFF,0xA7,0xC7,0xFF,0xFF,//,    12
0xF7,0xF7,0xF7,0xF7,0xF7,//-    13
0xFF,0x9F,0x9F,0xFF,0xFF,//x    14
0xFF,0xC9,0xC9,0xFF,0xFF,///    15
0xC1,0xAE,0xB6,0xBA,0xC1,//0    16
0xFF,0xBD,0x80,0xBF,0xFF,//1    17
0x8D,0xB6,0xB6,0xB6,0xB9,//2    18
```

```
0xDD,0xBE,0xB6,0xB6,0xC9,//3    19
0xE7,0xEB,0xED,0x80,0xEF,//4    20
0xD8,0xBA,0xBA,0xBA,0xC6,//5    21
0xC3,0xB5,0xB6,0xB6,0xCF,//6    22
0xFE,0x8E,0xF6,0xFA,0xFC,//7    23
0xC9,0xB6,0xB6,0xB6,0xC9,//8    24
0xF9,0xB6,0xBE,0xD6,0xE1,//9    25
0xFF,0xC9,0xC9,0xFF,0xFF,//:    26
0xFF,0xA4,0xC4,0xFF,0xFF,////    27
0xF7,0xEB,0xDD,0xBE,0xFF,//<    28
0xEB,0xEB,0xEB,0xEB,0xEB,//=    29
0xFF,0xBE,0xDD,0xEB,0xF7,//>   30
0xFD,0xFE,0xAE,0xF6,0xF9,//?    31
0xCD,0xB6,0x8E,0xBE,0xC1,//@    32
0x83,0xF5,0xF6,0xF5,0x83,//A    33
0xBE,0x80,0xB6,0xB6,0xC9,//B    34
0xC1,0xBE,0xBE,0xBE,0xDD,//C    35
0xBE,0x80,0xBE,0xBE,0xC1,//D    36
0x80,0xB6,0xB6,0xB6,0xBE,//E    37
0x80,0xF6,0xF6,0xFE,0xFE,//F    38
0xC1,0xBE,0xB6,0xB6,0xC5,//G    39
0x80,0xF7,0xF7,0xF7,0x80,//H          40
0xFF,0xBE,0x80,0xBE,0xFF,//I          41
0xDF,0xBF,0xBE,0xC0,0xFE,//J       42
0x80,0xF7,0xEB,0xDD,0xBE,//K    43
0x80,0xBF,0xBF,0xBF,0xFF,//L    44
0x80,0xFD,0xF3,0xFD,0x80,//M    45
0x80,0xFD,0xFB,0xF7,0x80,//N    46
0xC1,0xBE,0xBE,0xBE,0xC1,//O    47
0x80,0xF6,0xF6,0xF6,0xF9,//P    48
0xC1,0xBE,0xAE,0xDE,0xA1,//Q    49
0x80,0xF6,0xE6,0xD6,0xB9,//R    50
0xD9,0xB6,0xB6,0xB6,0xCD,//S    51
};
//Phan tu hai
const T_font font2[]={
0xFE,0xFE,0x80,0xFE,0xFE,//T    52
0xC0,0xBF,0xBF,0xBF,0xC0,//U    53
0xE0,0xDF,0xBF,0xDF,0xE0,//V    54
0xC0,0xBF,0xCF,0xBF,0xC0,//W    55
0x9C,0xEB,0xF7,0xEB,0x9C,//X   56
0xFC,0xFB,0x87,0xFB,0xFC,//Y89 24   57
0x9E,0xAE,0xB6,0xBA,0xBC,//Z90 25   58
0xFF,0x80,0xBE,0xBE,0xFF,//[    59
0xFD,0xFB,0xF7,0xEF,0xDF,//\    60
0xFF,0xBE,0xBE,0x80,0xFF,//]    61
0xFB,0xE1,0xE0,0xE1,0xFB,//^    62
0x7F,0x7F,0x7F,0x7F,0x7F,//_ 63
0xFF,0xFF,0xF8,0xF4,0xFF,//' 64
0xDF,0xAB,0xAB,0xAB,0xC7,//a    65
```

```
0x80,0xC7,0xBB,0xBB,0xC7,//b
0xFF,0xC7,0xBB,0xBB,0xBB,//c
0xC7,0xBB,0xBB,0xC7,0x80,//d
0xC7,0xAB,0xAB,0xAB,0xF7,//e    69
0xF7,0x81,0xF6,0xF6,0xFD,//f
0xF7,0xAB,0xAB,0xAB,0xC3,//g    71
0x80,0xF7,0xFB,0xFB,0x87,//h    72
0xFF,0xBB,0x82,0xBF,0xFF,//i    73
0xDF,0xBF,0xBB,0xC2,0xFF,//j    74
0xFF,0x80,0xEF,0xD7,0xBB,//k    75
0xFF,0xBE,0x80,0xBF,0xFF,//l    76
0x83,0xFB,0x87,0xFB,0x87,//m    77
0x83,0xF7,0xFB,0xFB,0x87,//n    78
0xC7,0xBB,0xBB,0xBB,0xC7,//o    79
0x83,0xEB,0xEB,0xEB,0xF7,//p    80
0xF7,0xEB,0xEB,0xEB,0x83,//q    81
0x83,0xF7,0xFB,0xFB,0xF7,//r    82
0xB7,0xAB,0xAB,0xAB,0xDB,//s    83
0xFF,0xFB,0xC0,0xBB,0xBB,//t    84
0xC3,0xBF,0xBF,0xDF,0x83,//u    85
0xE3,0xDF,0xBF,0xDF,0xE3,//v    86
0xC3,0xBF,0xCF,0xBF,0xC3,//w    87
0xBB,0xD7,0xEF,0xD7,0xBB,//x    88
0xF3,0xAF,0xAF,0xAF,0xC3,//y    89
0xBB,0x9B,0xAB,0xB3,0xBB,//z    90
0xFB,0xE1,0xE0,0xE1,0xFB,//^     92
0xE3,0xE3,0xC1,0xE3,0xF7,//->93
0xF7,0xE3,0xC1,0xE3,0xE3,//<-94
0xEF,0xC3,0x83,0xC3,0xEF,//95
0xFF,0xFF,0xFF,0xFF,0xFF//BLANK CHAR 96
};
// End of code table
//==========CAC CHUONH TRINH CON================
byte doc_eeprom(byte addr);
void viet_eeprom(byte addr,byte data);
//============Chuong trinh con hien thi=========
void display()
{
int8 disp_count,column_count;
int8 time;
for (time=0;time<6;time++)
{
 column_count=0;
 for(disp_count=16;disp_count>0;disp_count--)
  {
   PORTB=buff_disp[disp_count];
   PORTA=column_count;
   delay_ms(2);
   column_count++;
  }
```

```
 }
}

//========Copy to Ram1====================
void copy_2_ram1(int8 index_char)
{
int8 i,j;

for (j=0;j<=5;j++)
{  // Dich RAM
 for (i=16;i>0;i--)
 { buff_disp[i]= buff_disp[i-1]; }  //Dich RAM sang trai
  //Luu ma ascii vao RAM
  buff_disp[0]=font[index_char].b[j];
  // Goi hien thi
  display();
}
  buff_disp[0]=0xff;
}
//========Copy to Ram 2==================
void copy_2_ram2(int8 index_char)
{
int8 i,j;

for (j=0;j<=5;j++)
{  // Dich RAM
 for (i=16;i>0;i--)
 { buff_disp[i]= buff_disp[i-1]; }  //Dich RAM sang trai
  //Luu ma ascii vao RAM
  buff_disp[0]=font2[index_char].b[j];
  // Goi hien thi
  display();
}
  buff_disp[0]=0xff;
}
//============Chuong trinh chinh================
void main()
{
int8 count,i;
int8 const max_char=56;
// Khai bao chuoi ky tu hien thi tai day
char const a[57]= {"Truong Dai Hoc Bach Khoa Ha noi. Khoa Dien tu-Vien Thong"};// Ky tu
NULL duco them
// Khoi tao chuong trinh chinh

// Khoi tao ngat
   //enable_interrupts(INT_RDA);
   //enable_interrupts(GLOBAL);
//=================
// Ghi du lieu vao EEPROM
```

```
for(i=0;i<57;i++)
  viet_eeprom(i,a[i]);

  printf("Ban tin hien thi: \n\r");
for(i=0;i<57;i++)
  printf("%c",a[i]);   //dinh dang %c de bieu thi gui ky tu, \n\r la xuong dong
  //neu dinh dang la %u thi la so nguyen ko dau

while(1)
{
// Clear RAM of buff_disp
 for (i=0;i<=16;i++)
    buff_disp[i]=0xff;
 TRISA=0x00; // Thiet lap lam dau ra
 TRISB=0x00;
// Begin of text
 for (i=0;i<=max_char;i++)
  {
  count=a[i]-32;
        if (count<=51)
    copy_2_ram1(count);
        else
        {
         count=count-52;
         copy_2_ram2(count);
   }
  }
}
}
//=====Viet vao EEPROM=====================
void viet_eeprom(byte addr,byte data)
{
GIE=0; // disable all interrups
//EEIE=1;
eeadrh=0;
eedath=0;
EEADR=addr; // dia chi
EEDATA=data;// du lieu
EEPGD=0;
WREN=1; // cho phep viet vao EEPROM
EECON2=0x55;
EECON2=0xAA;
WR=1; // bat dau viet vao EEPROM
while(EEIF==0);// doi viet vao EEPROM ket thuc
GIE=1;
wren=0;
EEIF=0; // Bit EEIF phai ve 0 de viet nhung lan tiep theo
}
//===========Doc du lieu tu EEPROM============
```

```
byte doc_eeprom(byte addr)
{
  EEADR=addr;
  RD=1;
  return(EEDATA);
}
```

8.7. 16f877a_8x16_2mau

```
//+===Chuong trinh LED matrix display==========+
//| Thiet ke: Nguyen Chi Linh - DT8K47 - DHBKHN |
//| MCU: PIC16F877a (16K FLASH ROM, 256B EEPROM)  |
//| Cac IC khac:  74154 - demux/decoder 1-of-16 |
//|              74595 - Ghi dich 8bit       |
//+============================================+
#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)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)

// Dinh nghia cac chan cho ket noi 74595
#define clk    RD1    //RD1
#define data   RD0    //RD0
#define latch  RD2    //RD2
#define data_h PORTB    // Du lieu hang
#define decode_c PORTD  // Giai ma cot

// Bo nho dem man hinh hien thi
int8 buff_disp[64];     //Bo nho dem cho man hinh LED
int8 max_char=118;       //SO ky tu hien thi toi da
int8 time=5;            //Bien quy dinh toc do chu chay
int1 text_on_rom=0,stop=0;
int8 chon=3;
int8 address;
int8 choose_text = 0;
int1 Buffer_Screen_Full;
int8 Num_Byte_Buffer=0;
#define BUFFER_SIZE 32
BYTE sbuffer[BUFFER_SIZE];
BYTE next_in = 0;
BYTE next_out = 0;
//=========KHAI BAO CAC CHUONH TRINH CON===============
int8 doc_eeprom(int8 addr);
void send_64(int32 High_Byte,int32 LowByte);
void display();
void copy_2_ram1(int8 index_char);
void copy_2_ram2(int8 index_char);
void update_eeprom();
void convert_bcd(int8 x);
```

```
#include <font_ascii.c> //File chua bo font ma hoa ky tu ASCII
#include <input.c>
//============Chuong trinh phuc vu ngat====================
#int_rda
void serial_isr() {
  int t;

  sbuffer[next_in]=getc();
  t=next_in;
  next_in=(next_in+1) % BUFFER_SIZE;
  if(next_in==next_out)
   next_in=t;          // Buffer full !!
}
//===========Chuong trinh chinh=================
void main()
{
int8 i,num_char,ascii_code,k;
#bit update_rom = 0x05.5

char const a[119]= " Hello World.LED Matrix PIC16F877A - ULN2803 - 74HC595. Bang thong tin dien tu. ^-^Nguyen Chi Linh-DT8 DAI HOC BKHN^-^";
char const b[119]= " HAPPY NEW YEAR *2006* - CHUC MUNG NAM MOI - Chuc Mung Nam Moi - Happy new year. linhnc308@yahoo.com 1234567890      ";
char const c[119]= " You like a little flame in my heart. When I see  ou, the flame is like up. Because I love you. Because I LOVE YOU    ";
char const d[119]= " Do an mon: Thiet Ke Mach Logic - GVHD: Nguyen Nam Quan. Nhom sinh vien: Nguyen Chi Linh - Tek Song Leng. Lop Dien Tu8";
char const e[119]= " ABCDEFGHIJKLMNOPQRSTUVWXYZ - abcdefghijklmnopqrstuvwxyz - 1234567890 - ~!@#$%^&*()_+*/><:?., linhnc308@yahoo.com  ";
char const adc[6] = " ADC=";
//=============================================
  TRISA = 1;
  TRISB = 0;          // Thiet lap chan vao ra
  TRISD = 0;
  TRISE = 0;
  PORT_B_PULLUPS(TRUE);
  PORTE = 0xFF;
//==Thiet lap ngat ngoai 0 ==================
//  enable_interrupts(global);
//  enable_interrupts(int_rda);
//=============================================
  setup_adc_ports(AN0);       //Chon kenh AN4 nhung ko hieu sao can them phan khai bao
   setup_adc(ADC_CLOCK_INTERNAL);// ben duoi dechon dung kenh AN4 cho no chay dung
  delay_ms(10);
//======= HIEN THI TRAI TIM =======
//======= HIEN THI BAN TIN ========
for(i=0;i<6;++i)
```

```
  write_eeprom(0xf0+i,adc[i]);
for(i=0;i<118;++i)
  write_eeprom(i,a[i]);
  write_eeprom(0xff,max_char);     // Luu so ky tu toi da vao ROM
//if(update_rom==1)              //Kiem tra cong tac cap nhat du lieu
//   update_eeprom();           //Goi chuong trinh con cap nhat(giao tiep qua cong COM)
hien_thi:
for (i=0;i<=64;i++)              // Clear RAM of buff_disp
  buff_disp[i]=0xff;
//Doanchuong trinh nay se hien thi noi dung ban tin luu trong EEPROM
  address = label_address(hien_thi);
  num_char=0;ascii_code=0;

while(1) {

 for (num_char=0;num_char < max_char;++num_char)         // Hien thi chu tren man hinh
  {
  /* if(choose_text==0) {choose_text=1; goto hien_thi;}  // Xem xet viec cap nhat hien thi
  if(text_on_rom==0)
  {
    switch(chon)
     {
     case 0: j=a[i]-32; break;
     case 1: j=b[i]-32; break;
     case 2: j=c[i]-32; break;
     case 3: j=d[i]-32; break;
     case 4: j=e[i]-32; break;
     case 5: chon=0;    break;
    }
  }
  else
    j=read_eeprom(i)-32;
 */
  ascii_code = a[num_char] - 32;
  if(ascii_code < 51)
    copy_2_ram1(ascii_code);
          else
          {
           ascii_code = ascii_code - 51;
           copy_2_ram2(ascii_code);
  }
 }
//  Ket thuc hient hi TEXT
/* k=read_adc();
 convert_bcd(k);
  for(i=0;i<8;++i)      // Hien thi gia tri doc dc tu ADC
   {
    k = doc_eeprom(0xf0 + i)-32;
    copy_2_ram1(k);
   }
```

```
*/
}
}
//========= END MAIN ==============
// === CAC CHUONG TRINH CON =========
/*
void send_8(int8 byte_shift) {
int8 i,temp;
#bit MSB = temp.7;
temp = byte_shift;
for(i = 0;i < 8; ++i)
 {
   data = MSB;
   clk=1;
   shift_left(&temp,1,0);
   clk=0;
 }
// latch=1;      //Chot du lieu
// latch=0;
}
// == Send data to 64 Colum ==
void send_64(int32 ByteH,int32 ByteL) {
int8 i;
int8 Bytesend[8];

Bytesend[7] = Make8(ByteH,3);
Bytesend[6] = Make8(ByteH,2);
Bytesend[5] = Make8(ByteH,1);
Bytesend[4] = Make8(ByteH,0);
Bytesend[3] = Make8(ByteL,3);
Bytesend[2] = Make8(ByteL,2);
Bytesend[1] = Make8(ByteL,1);
Bytesend[0] = Make8(ByteL,0);

clk = latch = 0;
for(i = 7;i >=0 ; --i)
 {
   send_8(Bytesend[i]);
 }

 latch=1;      //Chot du lieu
 latch=0;
}

*/
void send_64(int32 ByteH,int32 ByteL) {
int8 i;
#bit LSB_bit = ByteH.0;

for(i = 0;i < 64 ; ++i)
```

```
 {
  data = bit_test(ByteH,31);    // 1 chu ky xung Clock la ~3.2uS
  clk=1;
  shift_left(&ByteL,4,0);
  shift_left(&ByteH,4,0);
  LSB_bit = bit_test(ByteL,31);
  clk=0;
 }
 latch=1;      //Chot du lieu
 latch=0;

}
//===========Chuong trinh con hien thi=========
void display() {
int8 count;
int32 Colum_L=0,Colum_H=0;
int8 i;
#bit bit0_Of_ColumH = Colum_H.0;

time = read_adc()/10;     // Viec doc gia tri ADC truoc khi hien thi lam cho viec thay doi
                    // toc do chu chay linh hoat hon, truc tiep tha doi
for (i=0;i<=time;i++)          //Toc do chu chay thay doi boi bien time
{
 //Colum_L = 1;                //Bien dem so cot, xem da quet het 16 cot chua
 //Colum_H = 0;
 bit_set(Colum_L,0);
 for(count=64;count>0;count--)
  {
   send_64(Colum_L,Colum_H);
   PORTB = buff_disp[count];
   shift_left(&Colum_L,4,0);
   shift_left(&Colum_H,4,0);
   bit0_Of_ColumH = bit_test(Colum_L,31);
   delay_us(150);
   PORTB = 0xFF;
  }
 }
}
//========Copy to Ram1====================
void copy_2_ram1(int8 index_char) {
int8 i,j;

for (j=0;j<=5;j++)
{
  if ((stop==1) && (Buffer_Screen_Full==1))
  {
    display();     // Call display() function, no shift text on screen
  }
  else
  {
```

```
    for (i=64;i>0;i--)
       buff_disp[i]= buff_disp[i-1];       // Dich RAM sang trai >> Tao hieu ung chu chay

     buff_disp[0]= font[index_char].b[j];   // Luu ma ascii vao RAM man hinh
     Num_Byte_Buffer++;                     // Count Number of byte  are loaded to buffer
     if ((stop == 1) && (Num_Byte_Buffer == 63))
     {
       Num_Byte_Buffer = 0;
       Buffer_Screen_Full = 1;
     }
     else
       Buffer_Screen_Full = 0;

     display();  // Goi hien thi
   }
}
  buff_disp[0]=0xff;
}
//========Copy to Ram 2==================
void copy_2_ram2(int8 index_char) {
int8 i,j;

for (j=0;j<=5;j++)
{
  if ((stop==1) && (Buffer_Screen_Full==1))
  {
    display();     // Call display() function, no shift text on screen
  }
  else
  {
    for (i=64;i>0;i--)
       buff_disp[i]= buff_disp[i-1];       // Dich RAM sang trai >> Tao hieu ung chu chay

     buff_disp[0]= font2[index_char].b[j];   // Luu ma ascii vao RAM man hinh
     Num_Byte_Buffer++;                     // Count Number of byte  are loaded to buffer
     if ((stop == 1) && (Num_Byte_Buffer == 63))
     {
       Num_Byte_Buffer = 0;
       Buffer_Screen_Full = 1;
     }
     else
       Buffer_Screen_Full = 0;

    display();  // Goi hien thi
  }
}
  buff_disp[0]=0xff; // Them mot khoang trang giua hai ky tu
}
//==========Update EEPROM====================
void update_eeprom() {
```

```
  byte i,j,addr,max;
  char temp;
  char string[64];
// Hien thi noi dung cua EEPROM
     printf("\r\n256 byte EEPROM of PIC16F88:\r\n");            // Display contents of the first
64
     for(i=0; i<=15; ++i)                                      // bytes of the data EEPROM in hex
     {
       for(j=0; j<=15; ++j)
       printf( "%2x ", doc_eeprom( i*16+j ) );
       printf("\n\r");
     }
// Hien thi noi dung ban tin
     i=0;
     do
     {
      temp = doc_eeprom(i);
      printf( "%C", temp);
      i++;
     } while (temp != 0xff);
//--------Ket thuc      ----------
     printf("\r\nTong so chu: %2u", doc_eeprom(0xff));
     printf("\r\n\nCo thay doi ban tin ko(Y/N)? "); temp=getc();//temp = getc();
     if (temp == 'y' || temp == 'Y')
     {
       printf("\r\nSo chu hien thi moi la: ");
       max_char=gethex();
       write_eeprom(0xff,max_char);
       printf("\r\nDia chi EEPROM can thay doi: ");
       addr = gethex();
      if (addr >= max_char)
       write_eeprom(0xff,addr);
       printf("\r\nSo ky tu them vao: ");
       max = gethex();                          // Tra ve gia tri Hexa
      if(max >= max_char)
       write_eeprom(0xff,max);   // Cap nhat so ky tu
       printf("\r\nNew: ");
       get_string(string,max+1);
      for (i=0;i<max;i++)   //bat dau qua trinh ghi vao ROM (cap nhat du lieu moi)
       {
         write_eeprom(addr,string[i]);
         addr=addr+1;
       }
       text_on_rom=1;
     }
     else
     {
      printf("Tro ve !");     // Ket thuc viec cap nhat, tro ve hien thi
      text_on_rom = 0;
     }
```

```
}
//=============READ EEPROM====================
int8 doc_eeprom(int8 addr)
{
  EEADR=addr;
  RD=1;
  return(EEDATA);
}
//=======Chuyen gia tri hex ra so ASCII=========
void convert_bcd(int8 x)
{
int8 temp;
int8 a;
  temp=x%10;                    //chia lay phan du, so hang don vi
  write_eeprom(0xf7,temp+0x30);    //Cong them 0x30 de tra ve gia tri SCII
  a=x/10;                       //tach hang tram va hang chuc
  temp=a%10;                    //tach so hang chuc
  write_eeprom(0xf6,temp+0x30);
  temp=x/100;
  write_eeprom(0xf5,temp+0x30);
}
```

9. Động cơ

Mình đang thử ctr điều khiển động cơ: Khi có tín hiệu từ chân RB0 thì động cỏ quay, nếu đang quay mà có tín hiệu từ RB1 thì động cơ quay ngược lại bằng thời gian no đã quay xuôi, bạn nào có thể viết cho mình một ví dụ như vậy bằng CCS được ko, cảm ơn nhiều.

Bạn có thể dùng ngắt ngoài để điều khiển cái động cơ của bạn:

9.1. DC Motor

9.1.1. code

```
#include <16F877A.h>
#device *=16
#device adc=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, BROWNOUT, NOLVP,
NOCPD, NOWRT
#use delay(clock=20000000)
```

--code.C

```
 #include <16f877a.h>
#include <def_877a.h>
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP,
NOCPD, NOWRT
#use delay(clock=20000000)
#include <lcd_lib_4bit.c>

#define PWMEn1 TRISC2
#define PWMEn2 TRISC1
#define INTS_PER_SECOND1 19

#define Set_value 35
int8 int_count1;
```

```
int16 so_vong,count;
void DispLine1(int8 pos,int16 POSCNT);
void DispLine2(int8 pos,int16 POSCNT);

#int_rtcc                       // Ngat Timer 0
void Timer0_isr()               // Dem so vong quay dong co
{
  count++;
}
#INT_TIMER1                     // Chuong trinh ngat Timer 1
void Timer11_isr() {            // Ham duoc goi khi TImer1 tran (65535->0)
                          // Xap xi 19 lan / giay
   if(--int_count1==0)
   {
    int_count1 = INTS_PER_SECOND1;
    so_vong = (count*255 + get_timer0())/14;
    count = 0;
    set_timer0(0);
   }
}
void main() {
  int16 value,current;
  int1 OutOfRange = 0;

  TRISD = 0;
  TRISB = 0xF0;

  lcd_init();
  printf(lcd_putchar,"DC Motor Control");
 setup_ccp1(CCP_PWM);   // Configure CCP1 as a PWM
           //   The cycle time will be (1/clock)*4*t2div*(period+1)
              //   In this program clock=20000000 and period=127 (below)
        //   For the three possible selections the cycle time is:
                    //     (1/20000000)*4*2*128 =  51.2 us or 19.5 khz
                    //     (1/10000000)*4*1*128 =  51.2 us or 19.5 khz
                    //     (1/10000000)*4*4*128 = 204.8 us or 4.9 khz
                    //     (1/10000000)*4*16*128= 819.2 us or 1.2 khz
 // set frequency
 //setup_timer_2(T2_DIV_BY_1, 31, 1);  //78.12KHz
 //setup_timer_2(T2_DIV_BY_1, 46, 1);  //208.3KHz
 //setup_timer_2(T2_DIV_BY_1, 255, 1);  //19.53KHz value = 0..1023
 //setup_timer_2(T2_DIV_BY_4, 255, 1);  //4.5KHz
  setup_timer_2(T2_DIV_BY_16, 255, 16); //1.2KHz
//=====================================
 setup_timer_0 (RTCC_DIV_1|RTCC_EXT_H_TO_L);
  set_timer0(0);
  set_timer1(0);
  setup_timer_1(T1_INTERNAL | T1_DIV_BY_4);
  enable_interrupts(INT_RTCC);
  enable_interrupts(INT_TIMER1);
```

```
  enable_interrupts(GLOBAL);

  count = 0;
  int_count1 = INTS_PER_SECOND1;

  setup_adc(adc_clock_internal);
  set_adc_channel( 0 );delay_ms(10);

  PWMEn1 = 0;
  TRISA = 0xFF;
  TRISC = 0;
  RD2 = 1;

  value=1023;
  set_pwm1_duty(value);
  current = value;
  lcd_putcmd(1);
  lcd_putcmd(0x80);
  Printf(lcd_putchar,"D=");

 while( TRUE ) {
  DispLine1(2,value);
  DispLine2(0,so_vong);
  if (so_vong > Set_value)          // Error Point 1 - Down
  {
   value = value - (so_vong-Set_Value);
   OutOfRange = 0;
   lcd_putcmd(0x87);
   lcd_putchar(" ");
  }
  if (!OutOfRange)
  {
   if (so_vong < Set_value)       // Error Point 2 - Up
    value = value + (Set_Value-so_vong);
    if (value > 1024)
    {
     value = 1023;
     lcd_putcmd(0x87);
     lcd_putchar("E");
     OutOfRange = 1;
    }
   }
   set_pwm1_duty(value);
 }
}
void DispLine1(int8 pos,int16 POSCNT)
{
int8 chuc,donvi;
int32  temp;
  temp = ((int32)POSCNT*100)/1023;
```

```
  chuc = temp / 10;
  donvi = temp % 10;
  lcd_putcmd(0x80 + pos);
  lcd_putchar(chuc + 0x30);
  lcd_putchar(donvi + 0x30);
  lcd_putchar("%");
}
void DispLine2(int8 pos,int16 POSCNT)
{
int8 chucnghin,nghin,tram,chuc,donvi;
int16  temp;
  temp = POSCNT;
//   chucnghin = temp / 10000;
//   temp = temp % 10000;
  nghin = temp / 1000;
  temp = temp % 1000;
  tram = temp / 100;
  temp = temp % 100;
  chuc = temp / 10;
  donvi = temp % 10;
  lcd_putcmd(0xC0 + pos);
  //lcd_putchar(chucnghin + 0x30);
  lcd_putchar(nghin + 0x30);
  lcd_putchar(tram + 0x30);
  lcd_putchar(chuc + 0x30);
  lcd_putchar(donvi + 0x30);
  Printf(lcd_putchar,"V/S");
}
```

9.1.2. Position_Control

```
//
// The following code uses the C-18 compiler and a PIC18F4331 at 40MHz (Phase lock
looped 10MHz oscillator (PLL))
// It demonstrates TOGGLING PIN 7 OF PORT B AT 25Hz (200Hz WITH 1:8 PRESCALE)
using timer0
// on the PIC18F4331.
//
// The following files should be included in the MPLAB project:
//
//                                                  -- Main source code file
//              p18f4331.lkr   -- Linker script file
//
//              The following project files are included by the linker script:
//
//              c018i.o                 -- C startup code
//              clib.lib          -- Math and function libraries
//              p18f4331.lib   -- Processor library
//
//-------------------------------------------------------------------
#include      <p18f4431.h>                              // Register definitions
//#include      <p18f452.h>                               // Register definitions
```

```
#include    <stdlib.h>
#include    <math.h>
#include    <string.h>
#include    <i2c.h>                                    // I2C library functions
#include    <pwm.h>                                    // PWM library functions
#include    <adc.h>                                    // ADC library functions
#include    <portb.h>                                  // PORTB library function
#include    <timers.h>                                 // Timer library functions
#include    <usart.h>                                  // USART library functions
#include    <delays.h>                          /* for 'Delay10KTCYx'     */

//-----------------------------------------------------------------
// Configuration bits
//-----------------------------------------------------------------
//
//// DO NOT PUT COMMENTS INSIDE ANY PRAGMA
//// Programs the pic configuration registers
#pragma romdata CONFIG
_CONFIG_DECL(_CONFIG1H_DEFAULT & _OSC_HSPLL_1H,
                    _CONFIG2L_DEFAULT & _BOREN_ON_2L &
_PWRTEN_ON_2L,
                    _CONFIG2H_DEFAULT & _WDTEN_OFF_2H,
                    _CONFIG3L_DEFAULT,
                    _CONFIG3H_DEFAULT & _MCLRE_ON_3H ,
                _CONFIG4L_DEFAULT & _LVP_OFF_4L & _STVREN_OFF_4L,
                    _CONFIG5L_DEFAULT,
                    _CONFIG5H_DEFAULT,
                    _CONFIG6L_DEFAULT,
                _CONFIG6H_DEFAULT,
                    _CONFIG7L_DEFAULT,
                    _CONFIG7H_DEFAULT);
#pragma romdata

//& _BKBUG_OFF_4L

//-----------------------------------------------------------------
//Constant Definitions
//-----------------------------------------------------------------
//#define     TIME  3                                  // Array index for segment
delay time
//#define     INDEX       PORTBbits.RB0           // Input for encoder index pulse
#define     TRUE  1
//#define     INPUT1
#define     OUTPUT      0

//ENCODER INITIALIZATION POSITION
#define     ENCDR_START_H   0x7F    //b'01111111'  ; THIS IS 0X7F
#define     ENCDR_START_L   0xFF

#define     ASCII_NUM_OFFSET        48
```

```
#define BAUD_9600  64                // for 40MHz (PLL x 10.00 MHz crystal)
                        // and set USART_BRGH_LOW
#define BAUD_19200          129              // and set USART_BRGH_HIGH
#define BAUD_57600          42               // and set USART_BRGH_HIGH
                                                    // USART_BRGH_HIGH and 64, for
9600 baud @ 10MHz  (0.16% ERROR IN RATE)
                                                                  // USART_BRGH_LOW
and 64, for 9600 baud @ 40MHz   (0.16% ERROR IN RATE)
                                                                  // USART_BRGH_HIGH
and 129, for 19200 baud @ 40MHz
                                                          //**      // BRG16_HIGH,
USART_BRGH_HIGH and 172, for 57,600 baud @ 40MHz  (.35% ERROR IN RATE)
                                                                  // USART_BRGH_HIGH
and 42, for 57,600 baud @ 40MHz  (.94% ERROR IN RATE)

// DEFINE TIMER INTERRUPT RATE FOR SERVO LOOP, ETC... (e.g. WRITING
ACCELEROMETER DATA TO RS-232)
#define TMR0_HIGH_BYTE      0xCF //;0xEC  ;Load TMR0 with the value for 250 Hz
THIS WAS FOR TIMER 1
#define          TMR0_LOW_BYTE  0x2C //;0x78
//                              ;set the timer for 0xF63C = 63,036 = 2^16 - 2,500 steps, since
2,500 x 1/2.5^6 = 1 milisecond
//                              ;0xFB1E FOR 2000Hz (FOR 10MHz CRYSTAL /4)
//                              ;0xF63C FOR 1000Hz (FOR 10MHz CRYSTAL /4)
//                              ;0xEC78 FOR 500Hz (FOR 10MHz CRYSTAL /4)
//                              ;0xCF2C FOR 200Hz? (FOR 10MHz CRYSTAL /4)
//                              ;0x9E58 FOR 100Hz (FOR 10MHz CRYSTAL /4)
//                              ;0x3CB0 FOR 50Hz (FOR 10MHz CRYSTAL /4)  200hz
//                              ;0x0BDC        for 40 Hz (FOR 10MHz CRYSTAL /4)
//                              ;FORMULA IS PERIOD/ TICK_TIME = STEPS, THEN 2^16
- STEPS
//                              ;WHERE TICK TIME IS 1/(CRYSTAL SPEED/4)
//        2^16 = 65536
                                //0x3CB0 = 15536 FOR 200Hz (FOR 40MHz Clock)
                                //0xB1E0 = 45536 FOR 500Hz (FOR 40MHz Clock)
                                //0xD8F0 = 55536 FOR 1,000Hz (FOR 40MHz Clock)
                                //0xF830 = 63536 FOR 5,000Hz (FOR 40MHz Clock)
                                //0xFC18 = 64536 FOR 10,000Hz (FOR 40MHz Clock)
                                //0xFE70 = 65136 FOR 10,000Hz (FOR 40MHz Clock)
//--------------------------
// Defines by Li Jiang
//------------------------

#define STEP_90 3396;  //  3396 is 90 degree.
#define STEP_80 3019;
#define STEP_70 2641;
#define STEP_60 2264;
#define STEP_50 1887;
#define STEP_40 1509;
```

```
#define STEP_30 1132;
#define STEP_20 755;
#define STEP_10 377;

#define STEP_COUNTS 200;

static int STEP;
// Variable defined by Li

static unsigned int  x, x_old;
static long int  error, integError_old,integError, Kp, Ki, Kd; //these should be signed

static long double       vel, vel_old, control;          //these should also be signed
static int xDes;
int Portb_Data;
int Step_Flag = 0;
int Step_Data[80];

int old_control=0;
int Step_Count = 1;
int Torque_Flag = 0;
int Torque_Control=0;

int STEP_Mount = 0;

int Torque_Step = 0;

int STEPS = 0;
//---------------------------------------------------------------------
// Variable declarations
//---------------------------------------------------------------------
          unsigned char str[7], charNumber;
          unsigned int    result;
//        int       hundreds, tens, tens, ones, remainder;

          unsigned int number, number1, number2;
          int readBuffer = 0;
//unsigned char          TRUE = 1;
union
{
 struct
 {
  unsigned Timeout:1;        //flag to indicate a TMR0 timeout
  unsigned None:7;
 } Bit;
 unsigned char Byte;
} Flags;

//---------------------------------------------------------------------
// Function Prototypes
```

```
//-------------------------------------------------------------------
void Setup(void);                                   // Configures peripherals and variables
// Writes a string from ROM to the USART
void putrsUSARTWill(const rom char *data);
void WriteUSARTWill(char data);
void SendRS232Int(unsigned int data);
void SendRS232Char(char data);
unsigned int simpleAnalog(void);
void PWM14out(unsigned int result);

void InterruptHandlerHigh (void);
void PID_Init(void);
void PID_Control(void);
void SendRS232NUM(long int);
unsigned int simpleAnalog1(void);
//-------------------------------------------------------------------
//        Setup() initializes program variables and peripheral registers
//-------------------------------------------------------------------
void Setup(void)
{
EnablePullups();         // Enable PORTB pullups

//SETUP port A for analog input and QUADRATURE ENCODER READ CAPABILITY
TRISA = 0b11111011;                  //0xF8 This is b'11111000'    0=output, 1=input

TRISB = OUTPUT; //0;
PORTB = 0b11111110;

TRISC = 0b10000000; //;                                  // 0=output, 1=input
//                  SETUP SO THAT THE RX PIN IS AN INPUT
// ------------------------------------------------------------
// FROM INTERRUPT CODE FROM MICROCHIP  (timer 0 and interrupt settings)
  Flags.Byte = 0;
  INTCON = 0x20;    //= 00100000          //disable global and enable TMR0 interrupt
  INTCON2 = 0x84;   //= 10000100           //TMR0 high priority
  RCONbits.IPEN = 1;          //enable priority levels

  TMR0L = TMR0_LOW_BYTE;       //INITIALIZE timer 0
  TMR0H = TMR0_HIGH_BYTE;      //INITIALIZE timer 0
 // T0CON = 0x82;  // =0b10000010 //set up timer0 - prescaler 1:8
 // T0CON = 0b10000001;  // =0b10000010 //set up timer0 - prescaler 1:4
  T0CON = 0b10001000;                        //set up timer0 - with no prescaler
  INTCONbits.GIEH = 1;         //enable interrupts
//for timer 1
//        ;FOR CALCULATING INTERGRAL TERM AND VELOCITY TERM (ADD IN
SOMETHING TO READ THE ENCODER TOO
//        bsf     IPR1,TMR1IP ;set Timer1 as a high priority interrupt source
//        bcf     PIR1,TMR1IF ;clear the Timer1 interrupt flag
//        bsf     PIE1,TMR1IE ;enable Timer1 interrupts
```

```
//      MOVLW     b'00110000'    ;PRESCALER = 1:8
//  T1CON = 0b10001000;                  //set up timer0 - with no prescaler
//      movlw timer1Hi                                  ;reload T1 registers with constant
time count (user defined)
//      movwf TMR1H
//      movlw timer1Lo
//      movwf TMR1L
//      bsf     T1CON,TMR1ON     ;turn on Timer1

//configure USART (CLOCK DEPENDENT)
//      baudUSART(BAUD_16_BIT_RATE);  THIS DOESN'T CURRENTLY WORK
                                                // 0b00001000);
      //USE THIS IF WE NEED TO SET BRG16_HIGH FOR BAUD RATE
GENERATOR (SEE p. 224)
                                                //BAUDCTL = BAUDCTL |
                  //THIS DOESN'T WORK EITHER
                  //_asm
                  //      MOVLW     B'01001010'          ; SETTING Baud rate
WITH 16 BIT BRGH and 10MHz crystal
                  //      MOVWF     BAUDCTL                ; SEE P. 171 IN
DATASHEET
                  //_endasm
      OpenUSART( USART_TX_INT_OFF &
            USART_RX_INT_OFF &
            USART_ASYNCH_MODE &
            USART_EIGHT_BIT &
            USART_CONT_RX &
            USART_BRGH_HIGH,
            BAUD_19200 );                                   //
USART_BRGH_HIGH and 64, for 9600 baud @ 10MHz  (0.16% ERROR IN RATE)
                                                // USART_BRGH_LOW
and 64, for 9600 baud @ 40MHz   (0.16% ERROR IN RATE)
                                                // USART_BRGH_HIGH
and 129, for 19200 baud @ 40MHz
                                          //**      // BRG16_HIGH,
USART_BRGH_HIGH and 172, for 57,600 baud @ 40MHz  (.35% ERROR IN RATE)
                                                // USART_BRGH_HIGH
and 42, for 57,600 baud @ 40MHz  (.94% ERROR IN RATE)

      RCSTA =     0b10010000;           //SETTING RECEIVE AND CONTROL
REG (SEE P. 167 IN DATASHEET)

WriteUSART(13);         //carriage return       //0x41=A,  0x48=H
putrsUSARTWill("PIC18F4331 DC Servomotor");

//putrsUSART(ready);
//SETUP QUADRATURE ENCODER READ CAPABILITY
      QEICON = 0x98;                //  = B'10011000'       (see p 171)
//      MOVLW     B'00000000'         // DIGITAL FILTER CONTROL REGISTER
(see p 178)
```

```
        DFLTCON     = 0x00;                         //FILTER INPUTS A & B
//INITIALIZE ENCODER COUNT to center of 16-bit range
        POSCNTH     = ENCDR_START_H;                // High byte of 16-bit encoder
count
        POSCNTL = ENCDR_START_L;            // low byte
// A/D SETUP (ADCON2 CLOCK DEPENDENT)
        // can also use OpenADC( ADC_FOSC_32 & ADC_RIGHT_JUST &
ADC_8ANA_0REF, ADC_CH0 & ADC_INT_OFF)  //(WON'T WORK FOR PICF4331)
// ADCON0: A/D CONTROL REGISTER 0  (USE TO SAMPLE SINGLY,
SEQUENTIALLY, OR SIMULTANEOUSLY
        ADCON0      = 1;    // = B'00000001'               ; set up, but don't start conversion
on analog input A0 (see p 181 in datasheet)
                                    //to turn off A/D make LSB low      ; SET FOR Fosc/8
conversion time
// ADCON1: A/D CONTROL REGISTER 1 (SET Vref AND A/D CONVERSION BUFFER
OPTIONS (WILL STORE UP TO 4 A/D READINGS IN BUFFER)
        ADCON1      = 0;    // = B'00000000' (see p 245 in datasheet)      // SET FOR
Fosc/32 conversion time ;right justified
                                    //      use     B'00010000'   ; for 4 buffer
LOCATIONS ; (see p 245 in datasheet)
// ADCON2 – A/D CONTROL REGISTER 2 (JUSTIFICATION AND ACQUISITION &
CONVERSION TIMING
        ADCON2 = 0b10001010; //40MHz    right justified, (2 * T_AD) aquisition time,
t_osc*32 acquisition time (see p 246 in datasheet)
                                    // = B'10000001'       //10MHz       ; right justified, 0
T_AD aquisition time, t_osc/8 acquisition time (see p 246 in datasheet)
                                    // USE B'10000011'   //8MHZ        ; right justified, 0
T_AD aquisition time, t_osc/4 acquisition time (see p 246 in datasheet)
                                    // = B'10000001'       //10MHz       ; right justified, 0
T_AD aquisition time, t_osc/8 acquisition time (see p 246 in datasheet)
// ADCON3: A/D CONTROL REGISTER 3 (INTERRUPTS AND TRIGGER SOURCES)
        ADCON3 = 0xC0;      // = B'11000000'               //no interrupts (see p 247 in
datasheet)
// ADCHS: A/D CHANNEL SELECT REGISTER (FOR PINS WHERE THERE ARE 2
ALTERNATIVES FOR A PARTICULAR ANALOG INPUT PIN)
        ADCHS = 0;                          //SET GROUP PINS FOR ACQUISITION
SEQUENCES(see p 248)
// ANSEL0: ANALOG SELECT REGISTER 0 (SET A PARTICULAR PIN AS ANALOG
INPUT BY MAKING PIN# = 1, ELSE DIG. IN)
        ANSEL0 = 0b00000011;        // = B'00000001';       // SET PIN 0 AS ANALOG, ALL
OTHERS DIGITAL, // SELECT WHICH PINS ARE ANALOG VS. DIGITAL (see p 249)
                //      can also use    SetChanADC(ADC_CH_0);  //to change which channel
is sampled (WON'T WORK FOR PICF4331)
//ANSEL1: ANALOG SELECT REGISTER 1(IF YOU WANT TO USE PIN AN8 FOR
ANALOG IN)
        ANSEL1 = 0; // = B'00000000'
//************************************************************/
// PWM SETUP FOR POWER PWM PINS (PORTS B & D)  (LOOSELY CLOCK
DEPENDENT)
// PWM Timer Control register 0 (PTCON0)
```

```
        PTCON0 = 0;
// PWM Timer Control register 1 (PTCON1)
        PTCON1 = 0x80;      // = 0b10000000
// PWM Control register 0 (PWMCON0)
        PWMCON0 = 0x6F;  // = B'01101111'  CONFIGURED FOR PWM PINS 1 AND 3
TO OUTPUT
                                                    // B'01011111' to CONFIGURE FOR
ALL PWM PINS TO OUTPUT
// PWM Control register 1 (PWMCON1)
        PWMCON1 = 0;
// PWM Period Registers (PTPERH and PTPERL)
        PTPERH = 0x00;      //0x03 FOR 12 BIT DUTY CYCLE RESOLUTION, 0x0F FOR
14 BIT
        PTPERL = 0xFF;
//ENABLE WRITE TO PWM REGISTER
//      BSF    pidStat2, pwm_enable
//      BSF    PORTB, 6      ; INDICATE THAT PWM IS ENABLED BY LIGHTING LED
B6
//DISABLE WRITE TO PWM REGISTER
//      BCF    pidStat2, pwm_enable
//      BCF    PORTB, 6      ; INDICATE THAT PWM IS disABLED BY LIGHTING LED
B6

}                                                   //end of setup

//-------------------------------------------------------------------
// main()
//-------------------------------------------------------------------

void main(void)
{
//      unsigned char str[7];
//int readBuffer = 0;

Setup();                                                    // Setup
peripherals and software

PID_Init();

while(TRUE)
{
//TOGGLE PIN B7 TO MONITOR LOOP TIME SEE JUST HOW FAST THIS LOOP CAN
COMPLETE
//      PORTBbits.RB7 = ~PORTBbits.RB7;

//      ClrWdt();                                         // Clear the WDT

//--- READING FROM SERIAL PORT
//Is data available in the read buffer?
//while (!DataRdyUSART());
```

```
//
//Read a fixed-length string of characters from the specified USART.
//char inputstr[10];
//getsUSART( inputstr, 5 );
//
//Read a byte (one character) out of the USART receive buffer, including the 9th bit if
enabled.
//int result;
//result = ReadUSART();
//result |= (unsigned int)

//WriteUSARTWill(55);

        if(DataRdyUSART())
                {
                readBuffer = ReadUSART();                   //read ASCII value of key pressed
on keyboard
                PORTBbits.RB5 = ~PORTBbits.RB5;             //toggle pin 5 if receive
register being read
                WriteUSARTWill(readBuffer);                 //echo value to RS-232
                        //*************** STEP ***********************
//          if (readBuffer =='z')
//            {
//                        Torque_Control += 10;
//                }
//
//          if (readBuffer =='x')
//            {
//                        Torque_Control -= 10;
//                }
//          if (readBuffer =='c')
//            {
//                        Torque_Control += 1;
//                }
//
//          if (readBuffer =='v')
//            {
//                        Torque_Control -= 1;
//                }
//
//                        if (readBuffer =='t')
//                        {
//
//                                putrsUSARTWill("Torque Step");
//                                WriteUSARTWill(13);
//                                Torque_Step = 1;
//                        }
//
//                        if (readBuffer =='y')
//                        {
```

```
//                              Torque_Step = 0;
//                      }

            if (readBuffer =='k')
               {
                              Torque_Flag = 1;
                      }
                  if (readBuffer =='l')
                      {

                              Torque_Flag = 0;
            xDes =  x;
                      }

                      if (readBuffer =='1')
                      {
                              STEP_Mount = STEP_10;  STEPS =10;
                      }

                      if (readBuffer =='2')
                      {
                              STEP_Mount = STEP_20;  STEPS =20;
                      }

                      if (readBuffer =='3')
                      {
                              STEP_Mount = STEP_30;  STEPS =30;
                      }

                      if (readBuffer =='4')
                      {
                              STEP_Mount = STEP_40;  STEPS =40;
                      }
                      if (readBuffer =='5')
                      {
                              STEP_Mount = STEP_50;  STEPS =50;
                      }
                      if (readBuffer =='6')
                      {
                              STEP_Mount = STEP_60;  STEPS =60;
                      }
                      if (readBuffer =='7')
                      {
                              STEP_Mount = STEP_70;  STEPS =70;
                      }
                      if (readBuffer =='8')
                      {
                              STEP_Mount = STEP_80;  STEPS =80;
                      }
                      if (readBuffer =='9')
```

```
{
        STEP_Mount = STEP_90;  STEPS =90;
}
if (readBuffer =='0')
{
        STEP_Mount = 0;  STEPS =0;
}

if (readBuffer =='q')
{
        STEP_Mount += 38;  STEPS +=1;
}

if (readBuffer =='w')
{
        STEP_Mount += 75;  STEPS +=2;
}

if (readBuffer =='e')
{
        STEP_Mount += 113;  STEPS +=3;
}

if (readBuffer =='r')
{
        STEP_Mount += 151;  STEPS +=4;
}
if (readBuffer =='t')
{
        STEP_Mount += 189 ;  STEPS +=5;
}
if (readBuffer =='y')
{
        STEP_Mount += 226;  STEPS +=6;
}
if (readBuffer =='u')
{
        STEP_Mount += 264;  STEPS +=7;
}
if (readBuffer =='i')
{
        STEP_Mount += 302;  STEPS +=8;
}
if (readBuffer =='o')
{
        STEP_Mount += 339; STEPS +=9;
}

if (readBuffer =='s')
{
```

```
                        putrsUSARTWill("Step");
                        WriteUSARTWill(13);
                        xDes =  x + STEP_Mount;
                        STEP = STEP_Mount;
                        Step_Flag = 1;
                }

        if (readBuffer =='b')
                {

                        putrsUSARTWill("Step_back");
                        WriteUSARTWill(13);
                        xDes =  x - STEP;
                        STEP = 0;
                        Step_Flag = 1;
                }

//              if (readBuffer =='1')
//              {
//
//                      putrsUSARTWill("Step");
//                      WriteUSARTWill(13);
//                      xDes =  x + STEP_10;
//                      STEP = STEP_10;
//                      Step_Flag = 1;
//              }
//              if (readBuffer =='2')
//              {
//
//                      putrsUSARTWill("Step");
//                      WriteUSARTWill(13);
//                      xDes =  x + STEP_20;
//                      STEP = STEP_20;
//                      Step_Flag = 1;
//              }
//              if (readBuffer =='3')
//              {
//
//                      putrsUSARTWill("Step");
//                      WriteUSARTWill(13);
//                      xDes =  x + STEP_30;
//      `               STEP = STEP_30;
//                      Step_Flag = 1;
//              }
//              if (readBuffer =='4')
//              {
//
//                      putrsUSARTWill("Step");
//                      WriteUSARTWill(13);
```

```
//                              xDes =  x + STEP_40;
//                              STEP = STEP_40;
//                              Step_Flag = 1;
//                      }
//                      if (readBuffer =='5')
//                      {
//
//                              putrsUSARTWill("Step");
//                              WriteUSARTWill(13);
//                              xDes =  x + STEP_50;
//                              STEP = STEP_50;
//                              Step_Flag = 1;
//                      }
//                      if (readBuffer =='6')
//                      {
//
//                              putrsUSARTWill("Step");
//                              WriteUSARTWill(13);
//                              xDes =  x + STEP_60;
//                              STEP = STEP_60;
//                              Step_Flag = 1;
//                      }
//                      if (readBuffer =='7')
//                      {
//
//                              putrsUSARTWill("Step");
//                              WriteUSARTWill(13);
//                              xDes =  x + STEP_70;
//                              STEP = STEP_70;
//                              Step_Flag = 1;
//                      }
//                      if (readBuffer =='8')
//                      {
//
//                              putrsUSARTWill("Step");
//                              WriteUSARTWill(13);
//                              xDes =  x + STEP_80;
//                              STEP = STEP_80;
//                              Step_Flag = 1;
//                      }
//                      if (readBuffer =='9')
//                      {
//
//                              putrsUSARTWill("Step");
//                              WriteUSARTWill(13);
//                              xDes =  x + STEP_90;
//                              STEP = STEP_90;
//                              Step_Flag = 1;
//                      }
```

```
                    // ************************D TERM************************
//                  if (readBuffer =='a')
//                  {
//                          putrsUSARTWill("D term:  ");
//                          Kd +=500;
//                          SendRS232NUM(Kd);
//                  }
//                  if (readBuffer =='d')
//                  {
//                          putrsUSARTWill("D term:  ");
//                          Kd -=500;
//                          SendRS232NUM(Kd);
//                  }
//                  //******************  P TERM ***********************
//                  if (readBuffer =='q')
//                  {
//                          putrsUSARTWill("P term:  ");
//                          Kp +=10;
//                          SendRS232NUM(Kp);
//                  }
//                  if (readBuffer =='e')
//                  {
//                          putrsUSARTWill("P term:  ");
//                          Kp -=10;
//                          SendRS232NUM(Kp);
//                  }
//
//                  if (readBuffer =='o')
//                  {
//                          putrsUSARTWill("I term:  ");
//                          Kp =0;
//                          SendRS232NUM(Ki);
//                  }
//                  if (readBuffer =='p')
//                  {
//                          putrsUSARTWill("I term:  ");
//                          Kp =700;
//                          SendRS232NUM(Ki);
//                  }
//                  //******************  I TERM ***********************
//                  if (readBuffer =='z')
//                  {
//                          putrsUSARTWill("I term:  ");
//                          Ki +=3;
//                          SendRS232NUM(Ki);
//                  }
//                  if (readBuffer =='c')
//                  {
//                          putrsUSARTWill("I term:  ");
//                          Ki -=3;
```

```
//                                  SendRS232NUM(Ki);
//                              }
                        }
//      if (readBuffer == '6')   {
//                  PORTBbits.RB6 = ~PORTBbits.RB6;
//                  WriteUSARTWill(13);                 //carriage return         //or
WriteUSART(0x0D); //0x41=A  //48 = 0
//                  WriteUSARTWill(54);                 //send ASCII 54 = '6' to RS-232
//      }

WriteUSARTWill(13);                 //carriage return       //or WriteUSART(0x0D);
        //0x41=A  //48 = 0
//SendRS232Int(x);
//WriteUSARTWill(9);
//SendRS232NUM(vel);
//
//WriteUSARTWill(9);
//SendRS232NUM((int)integError_old);
//
//WriteUSARTWill(9);
//SendRS232NUM(error);
//
//WriteUSARTWill(9);
//SendRS232NUM(Torque_Control);
//
//WriteUSARTWill(9);
//SendRS232NUM((unsigned int)simpleAnalog());
//
//WriteUSARTWill(9);
//SendRS232NUM((unsigned int)simpleAnalog1());

WriteUSARTWill(9);
SendRS232NUM((unsigned int)STEPS);

WriteUSARTWill(9);
SendRS232NUM((unsigned int)STEP_Mount);

////      putrsUSART(ready);          // put prompt to USART
//
////this works for getting the 16-bit encoder register value
//      number = POSCNTH;
////      charNumber = POSCNTL;
//      result = POSCNTL + (number  << 8);                  // High byte of 16-bit encoder
count
//
//      WriteUSARTWill(13);                 //carriage return       //or WriteUSART(0x0D);
        //0x41=A  //48 = 0
//      SendRS232Int(result);       // decimal "decode" and send encoder value to RS-232
//
//
```

```
////read and printout analog voltage
//        result = simpleAnalog();
//        WriteUSARTWill(9);          //tab
//        SendRS232Int(result);       // decimal "decode" and send encoder value to RS-232
//
//        result = result*16;         //to turn 10-bit A/D into 14-bit PWM
//
//
//// WRITE OUT TO 14-BIT PWM PIN
//        PWM14out(result);
////      PDC1L = (result & 0xFF);                  //these are used by Torque control
////      PDC1H = ( (result >> 8) & 0x3F );    // bit shift and mask off to most significant bits
//
////try to print out binary form number  (THIS WORKS!!!)
////      WriteUSARTWill(9);          //tab
////      SendRS232Int(0b11111111); // decimal "decode" and send encoder value to RS-232
//
//
//
////      ultoa(result, str);         //convert number "result" to string
////      putrsUSARTWill(str);
//
//   Delay10KTCYx (100);   // pause for a moment ( (  ) * 10,000 cycles)
//                                                  // AT 40MHz, 10,000 cycles = 1ms
//
////      PORTB = 0xFF;
////     Delay10KTCYx (255);   /* pause for a moment (255 * 10,000 cycles) */
//
//
////      PORTB = 0x00;
    Delay10KTCYx (255);   /* pause for a moment (255 * 10,000 cycles) */
  }     //end main while loop

CloseUSART();

}       //end of Main()
//---------------------------------------------------------------------
// putrsUSARTWill()
// Writes a string of characters in program memory to the USART
//---------------------------------------------------------------------
//special version of putrsUSART that doesn't also output the null byte at the end of the string
void putrsUSARTWill(const rom char *data)
{
 do
 {  // Transmit a byte
   while(BusyUSART());
        if(*data != 0) putcUSART(*data);     //avoid sending null bit at end of string
 } while( *data++  ); //&& !0
}
void WriteUSARTWill(char data)
```

```
{
//  if(TXSTAbits.TX9)  // 9-bit mode?
//  {
//    TXSTAbits.TX9D = 0;       // Set the TX9D bit according to the
//    if(USART_Status.TX_NINE)  // USART Tx 9th bit in status reg
//      TXSTAbits.TX9D = 1;
//  }
        while( BusyUSART() );
        TXREG = data;      // Write the data byte to the USART
}
void SendRS232NUM(long int data)
{
        char     index;
        unsigned char  outputChar;
        int       divisor[6] = {10000, 1000, 100, 10, 1, 1};

        if (data <0)
        {
                while(BusyUSART());
                TXREG = '-';
                data = -data;
        }
for (index = 0; index < 5; index++)
        {
                outputChar = data/divisor[index] + ASCII_NUM_OFFSET;

                        while( BusyUSART() );
                        TXREG = outputChar;      // Write the data byte to the USART
                                                                        //        could have used
WriteUSARTWill( outputChar );
                data = data%divisor[index];
        }
}
void SendRS232Int(unsigned int data)
{
        char     index;
        unsigned char  outputChar;
//        #define          DECIMAL_DIGITS   5
//        int       tenThousands, thousands, hundreds, tens, tens, ones, remainder, divisor;
//        divisor = 10000;
        unsigned int    divisor[6] = {10000, 1000, 100, 10, 1, 1};
//        int       divisor[4] = {100, 10, 1, 1};
//  if(TXSTAbits.TX9)  // 9-bit mode?
//  {
//    TXSTAbits.TX9D = 0;       // Set the TX9D bit according to the
//    if(USART_Status.TX_NINE)  // USART Tx 9th bit in status reg
//      TXSTAbits.TX9D = 1;
//  }
        for (index = 0; index < 5; index++) {
                outputChar = data/divisor[index] + ASCII_NUM_OFFSET;
```

```
//                  if( (outputChar != ASCII_NUM_OFFSET) ) {
                        while( BusyUSART() );
                        TXREG = outputChar;     // Write the data byte to the USART
                                                //        could have used
WriteUSARTWill( outputChar );
//                  }
                    data = data%divisor[index];
            }
}  //end of SendRS232int

void SendRS232Char(char data)
{
            char    index;
            unsigned char  outputChar;
//          #define         DECIMAL_DIGITS  3
            unsigned char  divisor[4] = {100, 10, 1, 1};

//  if(TXSTAbits.TX9)  // 9-bit mode?
//  {
//    TXSTAbits.TX9D = 0;     // Set the TX9D bit according to the
//    if(USART_Status.TX_NINE)  // USART Tx 9th bit in status reg
//      TXSTAbits.TX9D = 1;
//  }
            for (index = 0; index < 3; index++) {
                    outputChar = data/divisor[index] + ASCII_NUM_OFFSET;
//                  if( (outputChar != ASCII_NUM_OFFSET) ) {
                        while( BusyUSART() );
                        TXREG = outputChar;     // Write the data byte to the USART
                                                //        could have used
WriteUSARTWill( outputChar );
//                  }
                    data = data%divisor[index];
            }
} //end of SendRS232Char
// SIMPLE analog DOES AN ANALOG CONVERSION AND PUTS VALUES IN
"BUFFER"
unsigned int simpleAnalog(void)
{
            unsigned int    result;             // 10-bit A/D value
    ADCON0     = 0b00000001;
 ///        SetChanADC(ADC_CH_0);  //to change which channel is sampled
   // SetChanADC(1);
            ConvertADC();                           // start conversion
 // ADCON0 = 1;
    while( BusyADC() );         // LOOKS FOR END OF CONVERSION
            result = ReadADC();             // RESULT OF ANALOG CONVERSION

            ////this works for getting the 16-bit encoder register value
            //        number = POSCNTH;
            ////       charNumber = POSCNTL;
```

```
        //      result = POSCNTL + (number  << 8);              // High byte of 16-bit
encoder count
        //
        //      result = (result & 0x03FF);             // only 10 LSBs
        return(result);
        //-----------
}//--------END OF SIMPLE_ANLG ROUTINE ------

unsigned int simpleAnalog1(void)
{
        unsigned int    result;         // 10-bit A/D value
   ADCON0     = 0b00000101;
 ///       SetChanADC(ADC_CH_0);  //to change which channel is sampled
  // SetChanADC(1);
        ConvertADC();                           // start conversion
 // ADCON0 = 1;
   while( BusyADC() );          // LOOKS FOR END OF CONVERSION
        result = ReadADC();             // RESULT OF ANALOG CONVERSION
        return(result);
        //-----------
}//--------END OF SIMPLE_ANLG ROUTINE ------

void PWM14out(unsigned int result)
{
// WRITE OUT TO 14-BIT PWM PIN
        PDC1L = (result & 0xFF);                        // masking off the MSByte
        PDC1H = ( (result >> 8) & 0x03);//0x3F );   // bit shift to right (MSByte to the
LSByte)
                                                                        // and mask off to
most significant bits
}//--------END OF PWM14out ROUTINE ------
//---------------------------------------------------------------------------
// High priority interrupt vector

#pragma code InterruptVectorHigh = 0x08
void
InterruptVectorHigh (void)
{
  _asm
    goto InterruptHandlerHigh //jump to interrupt routine
  _endasm
}
//---------------------------------------------------------------------------
// High priority interrupt routine
#pragma code
#pragma interrupt InterruptHandlerHigh

void
InterruptHandlerHigh ()
{
```

```
        PORTB = (Portb_Data|0b00000001);
 if (INTCONbits.TMR0IF)
   {                            //check for TMR0 overflow
    INTCONbits.TMR0IF = 0;          //clear interrupt flag
    Flags.Bit.Timeout = 1;          //indicate timeout
        //reset timer
          TMR0L = TMR0_LOW_BYTE;       //INITIALIZE timer 0
          TMR0H = TMR0_HIGH_BYTE;      //INITIALIZE timer 0
    LATBbits.LATB7 = !LATBbits.LATB7; //toggle LED on RB0
          PID_Control();
   }
        PORTB = (Portb_Data&0b11111110);
 // Put the controller here:
 //PUT OTHER IF STATMENTS AND INTERRUPTS HERE
}  //END OF HIGH PRIORITY INTERRUPT ROUTINE
//_asm
//        MOVLW       D'49'
//        MOVWF       TXREG
//_endasm
//lessons learned about serial com (Rs-232)
//        putrsUSART( "Hello, the answer is 0." );      //for some reason, the putrsUSART
command puts a <0> symbol after the string in quotes
//        this is because it prints the NULL at the end of the string
//        WriteUSART(46);     //46=. //THIS WORKS          //0x41=A,  0x48=H
//        WriteUSART("");  //doesn't work     //46=.  //0x41=A,  0x48=H
//        putcUSART( "." );  //doesn't work
////      TXREG = 13;           // add 48 for 0 (numbers), 65 for upper case "A", 97 for lower
case "a"
//THIS WORKS
//WriteUSARTWill(13);               //carriage return         //or WriteUSART(0x0D);
        //0x41=A
//number = 'a';
//        number = number + 2;
//        TXREG = number;             // add 48 for 0 (numbers), 65 for upper case "A", 97 for
lower case "a"
//THIS WORKS
//        WriteUSARTWill(13);                 //carriage return         //or WriteUSART(0x0D);
        //0x41=A  //48 = 0
//        SendRS232Int(result);         // "decode" and send encoder value to RS-232
//        SendRS232Char(number);     // "decode" and send CHAR to RS-232
// THIS WORKS!!!!!!!!
// try to print out binary form number  THIS WORKS!!!!!!!!
//        WriteUSARTWill(9);          //tab
//        SendRS232Int(0b11111111); // decimal "decode" and send encoder value to RS-232
// SYNTAX FOR DELAYS
//   Delay10KTCYx (100);   // pause for a moment ( (  ) * 10,000 cycles)
                                                        // AT 40MHz, 10,000 cycles = 1ms
//   Delay1KTCYx (255);   /* pause for a moment (255 * 1,000 cycles) */
//        Delay100TCYx(255);  /* pause for a moment (255 * 100 cycles) */
//        Delay10TCYx(255);  /* pause for a moment (255 * 10 cycles) */
```

```
//static long int x, x_old, error, integError_old, integError, Kp, Ki, Kd;       //these should be
signed
//static long int          vel, vel_old, control;              //these should also be signed
//static int xDes;
// // controller code
// add/declare variable names  and initialize in "setup" if necessary

int counter = 0;

void PID_Init(void)
{
int number;
number = POSCNTH;
x = POSCNTL + (number  << 8);
x_old = x;
xDes =  x ;
error = xDes - x;//0;
vel = 0;
vel_old = 0;
integError = 0;
integError_old = 0;
control = 0;
Kp = 500;
Ki = 3;
Kd = -4000;
}
//after initializing encoder to initial position
void PID_Control(void)
{
int number;
//if (Step_Flag ==1)
//{
//
//
//                  Step_Data[counter] = x;
//                  counter++;
//
//
//         if (counter ==80)
//         {
//                  int i;
//                  INTCON =0;
//                  for (i = 0; i<80; i++)
//                  {
//                           WriteUSARTWill(13);
//                           SendRS232Int(Step_Data[i]);
//                  }
//                  INTCON  = 0x20;
//         }
//}
```

```
// In servo loop
////this works for getting the 16-bit encoder register value

//number = POSCNTH;
//WriteUSARTWill(13);                 //carriage return        //or WriteUSART(0x0D);
        //0x41=A  //48 = 0
//SendRS232Int(number);
//number =  POSCNTL;
//WriteUSARTWill(9);                  //carriage return        //or WriteUSART(0x0D);
        //0x41=A  //48 = 0
//SendRS232Int(number);
        number = POSCNTH;
        x = POSCNTL + (number  << 8);           // High byte of 16-bit encoder count
if(xDes >= x)
{
        error = xDes - x;
}
else
{
        error = x - xDes;
        error = -error;
}
//if (error==0)
//{
//        integError_old=0;
//}
if (Step_Flag ==1)
{
        error = (Step_Count/1000.0)*error;
        Step_Count++;

        if (Step_Count == 1000)
        {
                Step_Flag =0;
                Step_Count =1;
        }
}
//vel = x - x_old;
vel = 0;
if(x>x_old)
{
        vel = x - x_old;
}
else if (x < x_old)
{
        vel = x_old - x;
        vel = -vel;
}
//WriteUSARTWill(13);                 //carriage return        //or WriteUSART(0x0D);
        //0x41=A  //48 = 0
```

```
//WriteUSARTWill(9);
//SendRS232Int(x);
//
//WriteUSARTWill(9);
//SendRS232Int(xDes);
//
//WriteUSARTWill(9);
////
//
//SendRS232NUM((int)(vel));

vel = 0.80 * vel_old + .20 * vel;

if (integError_old+error>99999)
{
        integError = 99999;
}
else if (integError_old+error<-99999)
{
        integError = -99999;
}
else
{
        integError = integError_old + error;
}

/*
if (error<3&&error>-3)
{
        integError = 0;
}
*/
//
//WriteUSARTWill(9);
//SendRS232NUM(error);
control = Kp * error + Kd * vel + Ki * integError;
//control = 0;

//if(control==0)
//control =10;
//
//WriteUSARTWill(9);
//SendRS232NUM((int)(control));

// scale this to 14 bit value
if (control<0)
{
        control = -control;
        PORTB = 0b11111011;
        Portb_Data = 0b11111011;
```

```
}
else
{
Portb_Data = 0b11111111;
PORTB = 0b11111111;
}
//if (control==0)
//{
//control =1;
//}
control = (int)((control/1000000.0)*1023.0);
if (control>600)
{
        control = 600;
}
//WriteUSARTWill(9);
//SendRS232NUM(control);

if (Torque_Flag ==1)
{
        control = 0;
        Portb_Data = 0b11111111;
        PORTB = 0b11111111;
}
if (Torque_Flag ==1 &&      Torque_Step ==1)
{
        control = Torque_Control;
        Portb_Data = 0b11111111;
        PORTB = 0b11111111;
}
PWM14out(control);
//old_control= control;
//PWM_out = control ;//* something; // to get 14-bit value
x_old = x;
vel_old = vel;
integError_old = integError;
}
```

9.1.3. check_encoder

```
#include <16f876a.h>
#include <def_876a.h>
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP,
NOCPD, NOWRT
#use delay(clock=20000000)

#include <lcd_lib_4bit.c>

#define PWMEn1 TRISC2
#define PWMEn2 TRISC1
#define INTS_PER_SECOND1 19
#define Set_Speed 45
```

```
#define Set_duty  1023
#define Direction RC0

int8 int_count1;
int16 current_speed,count;
int16 new_speed,old_speed;
int16 value;

BYTE next_in, buffer[4];
BYTE state;

#int_rtcc                    // Ngat Timer 0
void Timer0_isr()                // Dem so vong quay dong co
{
  count++;
}
//////////////////////////////////////////////////////
void DispLine1(int8 pos,int16 POSCNT);
void DispLine2(int8 pos,int16 POSCNT);
Void Init();

void main() {

  Init();
//  Printf("\n\r Nhap gia tri: ");
  delay_ms(5000);
  count = 0;
  while( TRUE )
  {
    new_speed = count*255 + get_timer0();
    DispLine2(0,new_speed);
  }
}
void init()
  {
   TRISB = 0x00;
   TRISA = 0xFF;
   TRISC = 0b10000000;
   lcd_init();
   lcd_putcmd(0x80);
   Printf(lcd_putchar,"Encoder..");
   setup_timer_0 (RTCC_DIV_1|RTCC_EXT_L_TO_H);  // Timer0 is Counter
   set_timer0(0);
   enable_interrupts(INT_RTCC);
   enable_interrupts(GLOBAL);

   delay_ms(500);
   setup_ccp1(CCP_PWM);   // Configure CCP1 as a PWM
   setup_timer_2(T2_DIV_BY_16, 255, 1); //1.2KHz
   value=100;
```

```
  set_pwm1_duty(value);
  }

void DispLine1(int8 pos,int16 POSCNT)
{
int8 tram,chuc,donvi;
int32  temp;
   temp = ((int32)POSCNT*100)/1023;

   tram = temp / 100;
   temp = temp % 100;
   chuc = temp / 10;
   donvi = temp % 10;

   lcd_putcmd(0x80 + pos);
   if (tram == 0) lcd_putchar(" ");
   else   lcd_putchar(tram + 0x30);
   lcd_putchar(chuc + 0x30);
   lcd_putchar(donvi + 0x30);
   lcd_putchar("%");
}
void DispLine2(int8 pos,int16 POSCNT)
{
int8 tram,chuc,donvi;
int16  temp;
   temp = POSCNT;
//   chucnghin = temp / 10000;
//   temp = temp % 10000;
//   nghin = temp / 1000;
//   temp = temp % 1000;
   tram = temp / 100;
   temp = temp % 100;
   chuc = temp / 10;
   donvi = temp % 10;

   lcd_putcmd(0xC0 + pos);
   //lcd_putchar(chucnghin + 0x30);
   lcd_putchar(tram + 0x30);
   lcd_putchar(chuc + 0x30);
   lcd_putchar(donvi + 0x30);
}
  // The cycle time will be (1/clock)*4*t2div*(PR2+1)
  // In this program clock=20000000 and period=127 (below)
  // For the three possible selections the cycle time is:
  // (1/20000000)*4*1*256 =  51.2 us or 19.5 khz   // 20MHz
  // (1/10000000)*4*1*128 =  51.2 us or 19.5 khz   // 10MHz
  // (1/10000000)*4*4*128 = 204.8 us or 4.9 khz    // 10Mhz
  // (1/10000000)*4*16*128= 819.2 us or 1.2 khz    // 10MHz
  // set frequency
  // setup_timer_2(T2_DIV_BY_1, 31, 1);  //78.12KHz
```

```
// setup_timer_2(T2_DIV_BY_1, 46, 1);  //208.3KHz
// setup_timer_2(T2_DIV_BY_1, 255, 1);  //19.53KHz value = 0..1023
// setup_timer_2(T2_DIV_BY_4, 255, 1);  //4.5KHz
// setup_timer_2(T2_DIV_BY_X, PR2, POSTSCALES);
```

9.2. DK Step Motor

9.2.1. Code

DKMOTOR.h chỉ khai báo config cho 16F876A thôi.đoạn cod cho file nầy

```
#include <16F876A.h>
#device adc=8
#FUSES NOWDT //No Watch Dog Timer
#FUSES PUT //Power Up Timer
#FUSES NOPROTECT //Code not protected from reading
#FUSES NODEBUG //No Debug mode for ICD
#FUSES BROWNOUT //Reset when brownout detected
#FUSES NOLVP //Low Voltage Programming on B3(PIC16) or B5(PIC18)
#FUSES NOCPD //No EE protection
#FUSES NOWRT //Program memory not write protected
#use delay(clock=20000000)
```

9.2.2. Step_motor_F877A

```
#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)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
//#use i2c(Master,Fast,sda=PIN_C3,scl=PIN_C4)

#include <input.c>
int1 mode;
int8 duty_cycle,period;

//-----Ngat Ngoai------
#INT_EXT
void ngatngoai()
{
if (mode== 0) mode = 1;
else mode = 0;
}
//--------Ngat Noi Tiep------
#INT_RDA
void ngatrs232()
{
char c;
c = getc();
putc(c);

if (c=='h')  duty_cycle++;
else if (c=='l') duty_cycle--;
```

```
if (duty_cycle > 20) duty_cycle=20;
if (duty_cycle < 1 ) duty_cycle=1;
}

void main()
{
// Khoi tao cho ngat ngoai + RS232 nhan byte
     enable_interrupts (INT_EXT);
      enable_interrupts(INT_RDA);
     ext_int_edge(H_TO_L);
     enable_interrupts (GLOBAL);
//     ----------------------------
TRISD = 0;
duty_cycle = 2;
Printf("Chuong trinh dieu khien dong co buoc: ");
Printf(" Nhap chu ky:");
period = 3;
while (1)
{
 //Dieu khien dong co buoc: DK nua buoc (ULN2803)
  if (mode)
  {
    PortD = 0x60;
    delay_ms(duty_cycle);
    PortD = 0x70;
    delay_ms(duty_cycle);
    PortD = 0x30;
    delay_ms(duty_cycle);
    PortD = 0xB0;
    delay_ms(duty_cycle);
    PortD = 0x90;
    delay_ms(duty_cycle);
    PortD = 0xD0;
    delay_ms(duty_cycle);
    PortD = 0xC0;
    delay_ms(duty_cycle);
    PortD = 0xE0;
    delay_ms(duty_cycle);
  }
  else
  {
    PortD = 0xE0;
    delay_ms(duty_cycle);
    PortD = 0xC0;
    delay_ms(duty_cycle);
    PortD = 0xD0;
    delay_ms(duty_cycle);
    PortD = 0x90;
    delay_ms(duty_cycle);
    PortD = 0xB0;
```

```
      delay_ms(duty_cycle);
      PortD = 0x30;
      delay_ms(duty_cycle);
      PortD = 0x70;
      delay_ms(duty_cycle);
      PortD = 0x60;
      delay_ms(duty_cycle);
    }
}
}
/*
// Dieu khien dong co buoc: DK 1 pha
   if (mode)
   {
      PortD = 0x70;
      delay_ms(20-duty_cycle);
      PortD = 0xB0;
      delay_ms(20-duty_cycle);
      PortD = 0xD0;
      delay_ms(20-duty_cycle);
      PortD = 0xE0;
      delay_ms(20-duty_cycle);
   }
   else
   {
      PortD = 0xE0;
      delay_ms(20-duty_cycle);
      PortD = 0xD0;
      delay_ms(20-duty_cycle);
      PortD = 0xB0;
      delay_ms(20-duty_cycle);
      PortD = 0x70;
      delay_ms(20-duty_cycle);
   }
 //Dieu khien dong co buoc: DK 2 pha (ULN2803)
   if (mode)
   {
      PortD = 0x60;
      delay_ms(period-duty_cycle);
      PortD = 0x30;
      delay_ms(period-duty_cycle);
      PortD = 0x90;
      delay_ms(period-duty_cycle);
      PortD = 0xC0;
      delay_ms(period-duty_cycle);
   }
   else
   {
      PortD = 0xC0;
      delay_ms(period-duty_cycle);
```

```
    PortD = 0x90;
    delay_ms(period-duty_cycle);
    PortD = 0x60;
    delay_ms(period-duty_cycle);
    PortD = 0x30;
    delay_ms(period-duty_cycle);
  }
//Dieu khien dong co buoc: DK nua buoc (ULN2803)
  if (mode)
  {
    PortD = 0x60;
    delay_ms(period-duty_cycle);
    PortD = 0x70;
    delay_ms(period-duty_cycle);
    PortD = 0x30;
    delay_ms(period-duty_cycle);
    PortD = 0xB0;
    delay_ms(period-duty_cycle);
    PortD = 0x90;
    delay_ms(period-duty_cycle);
    PortD = 0xD0;
    delay_ms(period-duty_cycle);
    PortD = 0xC0;
    delay_ms(period-duty_cycle);
    PortD = 0xE0;
    delay_ms(period-duty_cycle);
  }
  else
  {
    PortD = 0xE0;
    delay_ms(period-duty_cycle);
    PortD = 0xC0;
    delay_ms(period-duty_cycle);
    PortD = 0xD0;
    delay_ms(period-duty_cycle);
    PortD = 0x90;
    delay_ms(period-duty_cycle);
    PortD = 0xB0;
    delay_ms(period-duty_cycle);
    PortD = 0x30;
    delay_ms(period-duty_cycle);
    PortD = 0x70;
    delay_ms(period-duty_cycle);
    PortD = 0x60;
    delay_ms(period-duty_cycle);
  }
*/
```

9.2.3. Chương trình điều khiển động cơ bước .

Bàn phím có 5 phím:quay thuận,quay ngược,quay thuận bước nhỏ,quay ngược bước nhỏ,stop. Trình dịch CCS.Mong các bạn cho ý kiến để cải tiến

Thang8831
http://www.picvietnam.com

```
#include "C:\Comport\ccs\DKMOTOR.h"
#define stop PIN_C0
#define dkth PIN_C1
#define dkng PIN_C2
#define hpb1 PIN_C3
#define hpb2 PIN_C4
#BYTE trisb =0x86
#BYTE OSCCON=0x8F
#BYTE trisc = 0x87
char a,b ;
char PeekKey() ;
char GetKey();
void buoc1();
void buoc2();
void stp();
void dkthuan();
void dknghich();
void main()
{ //su dung loai 2cuon co chan giua noi B+
setup_adc_ports(NO_ANALOGS);
setup_adc(ADC_OFF);
setup_spi(FALSE);
setup_timer_1(T1_DISABLED);
setup_timer_2(T2_DISABLED,0,1);
setup_comparator(NC_NC_NC_NC);
trisc=7F;
b=2;
while(1)
{
a=PeekKey();
switch(a)
{
case 0 : stp();
break;
case 1 : dkthuan();
break;
case 2 : dknghich();
break;
case 3 : buoc1();
break;
case 4 : buoc2();
break;
case 5 : stp();
break;
}
}
}
//***************
char PeekKey(void)
```

```
{ if(input(stop)==0) return(0);
if(input(dkth)==0) return(1);
if(input(dkng)==0) return(2);
if(input(hpb1)==0) return(3);
if(input(hpb2)==0) return(4);
else return (a); }
// **********
char GetKey(void)
{ char nKey;
nKey=PeekKey();
// wait for key release
while ((input(stop)==0)||(input(dkth)==0) )
(input(dkng)==0 )||(input(hpb1)==0 )||(input(hpb2)==0) ;
delay_ms(10);
return(nKey); }
// **********
void dkthuan()
{ output_b(8);
delay_ms(b);
output_b(2);
delay_ms(b);
output_b(1);
delay_ms(b);
output_b(4);
delay_ms(b);
output_b(0);
}
//*******
void dknghich()
{ output_b(4);
delay_ms(b);
output_b(1);
delay_ms(b);
output_b(8);
delay_ms(b);
output_b(2);
delay_ms(b);
output_b(0);
}
// *********
void stp()
{ output_b(0);
delay_ms(2);
}
//*******
void buoc1()
{ char i;
for (i=1;i<=15;++i)
dkthuan();
a=0;
```

```
}
//**********
void buoc2()
{ char i;
for (i=1;i<=15;++i)
dknghich();
a=0;
}
// Thanhphuc email thuyphuc81@yahoo.com
// Dieu khien moto buoc loai 2 cuon day co chan giua
```

9.2.4. Điều khiển động cơ bước

```
//================================================ =======
// Ten chuong trinh     : Thuc hien vao ra
// Nguoi thuc hien      : linhnc308
// Ngay thuc hien         : 1/09/2006
// Phien ban     : 1.0
// Mo ta phan cung      : Dung PIC16F877A - thach anh 20MHz
//================================================ =======
#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)
#CASE
// Dinh nghia ten cac cong ra
#define Relay1  RD0
#define Relay2  RD1
#define Relay3  RD2
#define Relay4  RD3
#define Relay5  RD4
#define Relay6  RD5
#define Relay7  RD6
#define Relay8  RD7
#define Relay9  RC4
#define Relay10 RC5
#define Relay11 RC6
#define Relay12 RC7

#define IN1     RA0
#define IN2     RA1
#define IN3     RA2
#define IN4     RA3

#define AllRelay1 PORTD    // PIN D0 : D7
#define AllRelay2 PORTC    // PIN C4 : C7
#define Step      PORTD
#Define AllInput  PORTA
```

```
#define OFF 0
#define ON  1

#define OutEnable1 TRISD   // Relay Output
#define OutEnable2 TRISC   // Relay Output
#define InEnable   TRISA   // Input
#define StepEnable TRISD   // Step Motor

#define PWM_Enable TRISC2  // PWM, PIN_C2

void main()
{
int8 duty_cycle;

  delay_ms(250);
// Khoi tao che do vao ra
  OutEnable1 = 0x00;
  OutEnable2 = 0x00;
  InEnable   = 0xFF;
  StepEnable = 0x00;
  PWM_Enable = 1;   // Khong cho phep xuat PWM
//=======================================
// Test Mode
  duty_cycle = 1;
while (1)
{
 //Dieu khien dong co buoc: DK nua buoc (ULN2803)
  if (IN1 == 0)
  {
    Step = 0x1;
    delay_ms(duty_cycle);
    Step = 0x3;
    delay_ms(duty_cycle);
    Step = 0x2;
    delay_ms(duty_cycle);
    Step = 0x6;
    delay_ms(duty_cycle);
    Step = 0x4;
    delay_ms(duty_cycle);
    Step = 0xc;
    delay_ms(duty_cycle);
    Step = 0x8;
    delay_ms(duty_cycle);
    Step = 0x9;
    delay_ms(duty_cycle);
  }
  else if (IN2 == 0)
  {
    Step = 0x9;
    delay_ms(duty_cycle);
```

```
      Step = 0x8;
      delay_ms(duty_cycle);
      Step = 0xc;
      delay_ms(duty_cycle);
      Step = 0x4;
      delay_ms(duty_cycle);
      Step = 0x6;
      delay_ms(duty_cycle);
      Step = 0x2;
      delay_ms(duty_cycle);
      Step = 0x3;
      delay_ms(duty_cycle);
      Step = 0x1;
      delay_ms(duty_cycle);
    }
    else Step = 0x00;
}
}
/*
// Dieu khien dong co buoc: DK 1 pha
    if (mode)
    {
      Step = 0x70;
      delay_ms(20-duty_cycle);
      Step = 0xB0;
      delay_ms(20-duty_cycle);
      Step = 0xD0;
      delay_ms(20-duty_cycle);
      Step = 0xE0;
      delay_ms(20-duty_cycle);
    }
    else
    {
      Step = 0xE0;
      delay_ms(20-duty_cycle);
      Step = 0xD0;
      delay_ms(20-duty_cycle);
      Step = 0xB0;
      delay_ms(20-duty_cycle);
      Step = 0x70;
      delay_ms(20-duty_cycle);
    }
 //Dieu khien dong co buoc: DK 2 pha (ULN2803)
    if (mode)
    {
      Step = 0x60;
      delay_ms(period-duty_cycle);
      Step = 0x30;
      delay_ms(period-duty_cycle);
      Step = 0x90;
```

```
    delay_ms(period-duty_cycle);
    Step = 0xC0;
    delay_ms(period-duty_cycle);
  }
  else
  {
    Step = 0xC0;
    delay_ms(period-duty_cycle);
    Step = 0x90;
    delay_ms(period-duty_cycle);
    Step = 0x60;
    delay_ms(period-duty_cycle);
    Step = 0x30;
    delay_ms(period-duty_cycle);
  }
//Dieu khien dong co buoc: DK nua buoc (ULN2803)
  if (mode)
  {
    Step = 0x60;
    delay_ms(period-duty_cycle);
    Step = 0x70;
    delay_ms(period-duty_cycle);
    Step = 0x30;
    delay_ms(period-duty_cycle);
    Step = 0xB0;
    delay_ms(period-duty_cycle);
    Step = 0x90;
    delay_ms(period-duty_cycle);
    Step = 0xD0;
    delay_ms(period-duty_cycle);
    Step = 0xC0;
    delay_ms(period-duty_cycle);
    Step = 0xE0;
    delay_ms(period-duty_cycle);
  }
  else
  {
    Step = 0xE0;
    delay_ms(period-duty_cycle);
    Step = 0xC0;
    delay_ms(period-duty_cycle);
    Step = 0xD0;
    delay_ms(period-duty_cycle);
    Step = 0x90;
    delay_ms(period-duty_cycle);
    Step = 0xB0;
    delay_ms(period-duty_cycle);
    Step = 0x30;
    delay_ms(period-duty_cycle);
    Step = 0x70;
```

```
    delay_ms(period-duty_cycle);
    Step = 0x60;
    delay_ms(period-duty_cycle);
  }
*/
```

10. Capture

10.1. Code cho CCS

```
//////////////////////////////////////////////////////////////////////
//// EX_CCPMP.C ////
//// ////
//// This program will show how to use the built in CCP to ////
//// measure a pulse width. ////
//// ////
//// Configure the CCS prototype card as follows: ////
//// Connect a pulse generator to pin 3 (C2) and pin 2 (C1) ////
//// See additional connections below. ////
//// ////
//// This example will work with the PCM and PCH compilers. The ////
//// following conditional compilation lines are used to include a ////
//// valid device for each compiler. Change the device, clock and ////
//// RS232 pins for your hardware if needed. ////
//////////////////////////////////////////////////////////////////////
//// (C) Copyright 1996,2003 Custom Computer Services ////
//// This source code may only be used by licensed users of the CCS ////
//// C compiler. This source code may only be distributed to other ////
//// licensed users of the CCS C compiler. No other use, ////
//// reproduction or distribution is permitted without written ////
//// permission. Derivative programs created using this software ////
//// in object code form are not restricted in any way. ////
//////////////////////////////////////////////////////////////////////

#if defined(__PCM__)
#include <16F88.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7) // Jumpers: 8 to 11, 7 to 12

#elif defined(__PCH__)
#include &lt;18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7) // Jumpers: 8 to 11, 7 to 12
#endif
long rise,fall,pulse_width;

#int_ccp2
void isr()
{
rise = CCP_1;
```

```
fall = CCP_2;

pulse_width = fall - rise; // CCP_1 is the time the pulse went high
} // CCP_2 is the time the pulse went low
// pulse_width/(clock/4) is the time

// In order for this to work the ISR
// overhead must be less than the
// low time. For this program the
// overhead is 45 instructions. The
// low time must then be at least
// 9 us.

void main()
{
printf("\n\rHigh time (sampled every second):\n\r");
setup_ccp1(CCP_CAPTURE_RE); // Configure CCP1 to capture rise
setup_ccp2(CCP_CAPTURE_FE); // Configure CCP2 to capture fall
setup_timer_1(T1_INTERNAL); // Start timer 1

enable_interrupts(INT_CCP2); // Setup interrupt on falling edge
enable_interrupts(GLOBAL);

while(TRUE) {
delay_ms(1000);
printf("\r%lu us ", pulse_width/5 );
}
}
```

10.2. Sử dụng capture newcode

```
#include <16F877A.h>
#use delay(clock=4000000)
#fuses HS,NOWDT, NOPROTECT
#use rs232(baud=4800,xmit=PIN_C6,rcv=PIN_C7)

#bit TMR1IF 0x0C.0

int16 CCP1Value;
int16 CCP1OldValue;
BOOLEAN CCP1Captured;

#int_CCP1
CCP1_isr()
{
if(TMR1IF)
{
CCP1Value = CCP_1 +(65535-CCP1OldValue);
CCP1OldValue = CCP_1;
TMR1IF=0;
}
else
```

```
{
CCP1Value = CCP_1 - CCP1OldValue;
CCP1OldValue = CCP_1;
}
CCP1Captured = TRUE;
}
//---------------------------------------------------------------------
void Init_ccp(void)
{
setup_ccp1(CCP_CAPTURE_RE);
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
CCP1Value = 0;
CCP1OldValue = 0;
CCP1Captured = TRUE;
enable_interrupts(INT_CCP1);
enable_interrupts(GLOBAL);
}
//---------------------------------------------------------------------
void main()
{
float Freq;
Init_ccp();
printf("Frequence test:\r\n");
while (TRUE) {

if (CCP1Captured) {

// F = 1/T
// Timer1 prescaler DIV_BY_8
// Pic16F877A 4MHz -> 0.000001 * 8 = 1uS * 8
// PIC16f877a 20MHz -> 200nS * 8

Freq = 1.0/((float)CCP1Value*8e-6);
printf("Freq:%f\r\n",Freq);
CCP1Captured = FALSE;
}
}
}
```

10.3. Capture_LCD_5MHz

```
#include <16F877A.h>
#include <def_877a.h>
#use delay(clock=20000000)
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP,
NOCPD, NOWRT
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)

#bit TMR1IF = 0x0C.0

#include <LCD_lib_4bit.c>
```

```
int16 CCP1Value;  // Gia tri CCP hien tai
int16 CCP1OldValue;  // Gia tri CCP truoc do
BOOLEAN CCP1Captured;
float Freq;
int8 char1,char2,char3,char4,char5,char6,mode;
int8 count,count1,count2,count3;
#int_CCP1
CCP1_isr()
{
if(TMR1IF)
{
CCP1Value = CCP_1 +(65535-CCP1OldValue);
CCP1OldValue = CCP_1;
TMR1IF=0;
}
else
{
CCP1Value = CCP_1 - CCP1OldValue;
CCP1OldValue = CCP_1;
}
CCP1Captured = TRUE;
}
//-------------------------------------------------------------------------
#INT_TIMER0
Timer0_isr()
{
}
//---------------------------------------
void Init_ccp(void)
{
setup_ccp1(CCP_CAPTURE_RE);
setup_timer_0(RTCC_EXT_L_TO_H|RTCC_DIV_64);
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
set_timer0(0);
CCP1Value = 0;
CCP1OldValue = 0;
CCP1Captured = TRUE;
enable_interrupts(INT_CCP1);
//enable_interrupts(INT_TIMER1);
enable_interrupts(GLOBAL);
}
//-------------------------------------------------------------------------
void Convert_CCP1()
{
int32 temp;
//Freq = (1.0/((float)CCP1Value*2e-7));   // For Time_Div_1
Freq = (1.0/((float)CCP1Value*16e-7));     //For_Time_Div_8
if (Freq >= 1000 )
{
      mode = 1;
```

```
        temp =  freq;
        char1 = (temp / 100000) + 0x30;
        temp =   temp % 100000;
        char2 = (temp / 10000) + 0x30;
        temp = (temp % 10000);
        char3 = (temp / 1000) + 0x30;
        temp = (temp % 1000);
        char4 = (temp / 100) + 0x30;
        temp = temp % 100;
        char5= (temp/10 ) + 0x30;
        char6 = (temp % 10) + 0x30;
        goto exit;
}
else
if (Freq >= 1000000 )
{
        mode = 2;
        temp =  freq;
        char1 = (temp / 100000) + 0x30;
        temp =   temp % 100000;
        char2 = (temp / 10000) + 0x30;
        temp = (temp % 10000);
        char3 = (temp / 1000) + 0x30;
        temp = (temp % 1000);
        char4 = (temp / 100) + 0x30;
        temp = temp % 100;
        char5= (temp/10 ) + 0x30;
        char6 = (temp % 10) + 0x30;
        goto exit;
}
else
     {
        mode = 0;
        temp =  (int32)(freq * 1000);
        char1 = (temp / 100000) + 0x30;
        temp =   temp % 100000;
        char2 = (temp / 10000) + 0x30;
        temp = (temp % 10000);
        char3 = (temp / 1000) + 0x30;
        temp = (temp % 1000);
        char4 = (temp / 100) + 0x30;
        temp = temp % 100;
        char5= (temp/10 ) + 0x30;
        char6 = (temp % 10) + 0x30;
        goto exit;
     }
exit:
   temp = 0;
}
//-----------------------------------------
```

```
void convert_timer0_value()
{
     count = get_timer0();
     count1 = count/100 + 0x30;
     count = count%100;
     count2 = count/10 + 0x30;
     count3 = count%10 + 0x30;
}
//-------------------------------------------------------------
void main()
{
  TRISB = 0;                          // F = 1/T
  Init_ccp();                         // Timer1 prescaler DIV_BY_1
  LCD_init();                         // PIC16f877a 20MHz -> 0.0000002 = 200nS
  //printf("Frequence test:\r\n");    // Pic16F877A 4MHz -> 0.000001 = 1uS
  LCD_putcmd(0xC0);
  Printf(LCD_putchar,"Counter = ");
  LCD_putcmd(0x80);
  Printf(LCD_putchar,"Freq = ");

  while (TRUE)
  {
    if (CCP1Captured)
    {
      Convert_CCP1();
      LCD_putcmd(0x87);
      if (char1 != 0x30)    LCD_putchar(char1);
      if (((char1 != 0x30) && (char2 == 0x30)) || (char2 != 0))
      LCD_putchar(char2);
      LCD_putchar(char3);
      LCD_putchar(".");
      LCD_putchar(char4);
      LCD_putchar(char5);
      switch (mode)
      {
        case 0: {LCD_putchar(char6);lcd_putcmd(0x8F);Printf(LCD_putchar," Hz");} break;
        case 1: {LCD_putchar(char6);lcd_putcmd(0x8F);Printf(LCD_putchar,"KHz");} break;
        case 2: {LCD_putchar(char6);lcd_putcmd(0x8F);Printf(LCD_putchar,"MHz");}
break;
      }
      printf("Freq:%f\r\n",Freq);
      CCP1Captured = FALSE;
    }
   convert_timer0_value();
   LCD_putcmd(0xCa);
   LCD_putchar(count1);
   LCD_putchar(count2);
   LCD_putchar(count3);
  }
}
```

Thang8831
http://www.picvietnam.com

10.4. Sử dụng capture_LCD

```
#include <16F877A.h>
#include <def_877a.h>
#use delay(clock=20000000)
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP,
NOCPD, NOWRT
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)

#bit TMR1IF = 0x0C.0

#include <LCD_lib_4bit.c>

int16 CCP1Value;  // Gia tri CCP hien tai
int16 CCP1OldValue;  // Gia tri CCP truoc do
BOOLEAN CCP1Captured;
float Freq;
int8 char1,char2,char3,char4,char5,char6;

#int_CCP1
CCP1_isr()
{
if(TMR1IF)
{
CCP1Value = CCP_1 +(65535-CCP1OldValue);
CCP1OldValue = CCP_1;
TMR1IF=0;
}
else
{
CCP1Value = CCP_1 - CCP1OldValue;
CCP1OldValue = CCP_1;
}
CCP1Captured = TRUE;
}
//-------------------------------------------------------------------------
#INT_TIMER1
Timer1_isr()
{
IF (RB0==1) RB0=0;
Else RB0=1;
}
//---------------------------------------
void Init_ccp(void)
{
setup_ccp1(CCP_CAPTURE_RE);
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
CCP1Value = 0;
CCP1OldValue = 0;
CCP1Captured = TRUE;
enable_interrupts(INT_CCP1);
```

```
//enable_interrupts(INT_TIMER1);
enable_interrupts(GLOBAL);
}
//---------------------------------------------------------------------
void Convert_CCP1()
{
long temp;
int8 temp2;
Freq = 1.0/((float)CCP1Value*1.6e-6);
temp = (long)freq;
char1 = ((temp / 100) + 0x30);
temp2 = (temp % 100);
char2 = ((temp2 / 10) + 0x30);
char3 = ((temp2 % 10) + 0x30);
}
//-------------------------------------------------------------
void main()
{
TRISB = 0;
Init_ccp();
LCD_init();
printf("Frequence test:\r\n");
while (TRUE) {
if (CCP1Captured) {
// F = 1/T
// Timer1 prescaler DIV_BY_8
// Pic16F877A 4MHz -> 0.000001 * 8 = 1uS * 8
// PIC16f877a 20MHz -> 0.0000002 * 8
Convert_CCP1();
LCD_putcmd(0x80);
Printf(LCD_putchar,"Freq = ");
LCD_putchar(char1);
LCD_putchar(char2);
LCD_putchar(char3);
Printf(LCD_putchar," Hz");

printf("Freq:%f\r\n",Freq);
CCP1Captured = FALSE;
}
}
}
```

10.5. Sử dụng capture

```
#include <16F877A.h>
#use delay(clock=4000000)
#fuses HS,NOWDT, NOPROTECT
//#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9,errors)

int16 CCP1Value;
int16 CCP1OldValue;
```

```
BOOLEAN CCP1Captured;

#int_CCP1   // Ngat do CCCP1 xay ra, thuc hien lenh...
CCP1_isr()
{
CCP1Value = CCP_1 - CCP1OldValue;
CCP1OldValue = CCP_1;
CCP1Captured = TRUE;
}
//-----------------------------------------------------------------------
void Init_ccp(void)  //Khoi tao chuc nang CCCP
{
setup_ccp1(CCP_CAPTURE_RE);
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
CCP1Value = 0;
CCP1OldValue = 0;
CCP1Captured = TRUE;
enable_interrupts(INT_CCP1);
enable_interrupts(GLOBAL);
}
//-----------------------------------------------------------------------
void main()
{
float Freq;
Init_ccp();
printf("Frequence test:\r\n");
while (TRUE) {
if (CCP1Captured) {
// F = 1/T
// Timer1 prescaler DIV_BY_8
// Pic16F876 4Mz -> 0.000001 * 8
Freq = 1.0/((float)CCP1Value*8e-6);
printf("Freq:%f\r\n",Freq);
CCP1Captured = FALSE;
}
}
}
```

11. SPI

```
//   SPI MICROWIRE INTERFACE HANDLER
//   COPYRIGHT PROPERTY OF ALPHADATA DESIGNS LIMITED (c) 1999
//   published by permission of Alphadata designs on
//   Hi-Tech C website, http://www.workingtex.com/htpic. Thanks!
//   P4/0 : OUTPUT : DATA OUT
//   P4/1 : OUTPUT : CLOCK
//   P4/2 : OUTPUT : CHIP SELECT ATOD
//   P4/3 : OUTPUT : CHIP SELECT DTOA
//   P4/7 : INPUT  : DATA IN
//
//-----------------------------------------------------
// Version History
```

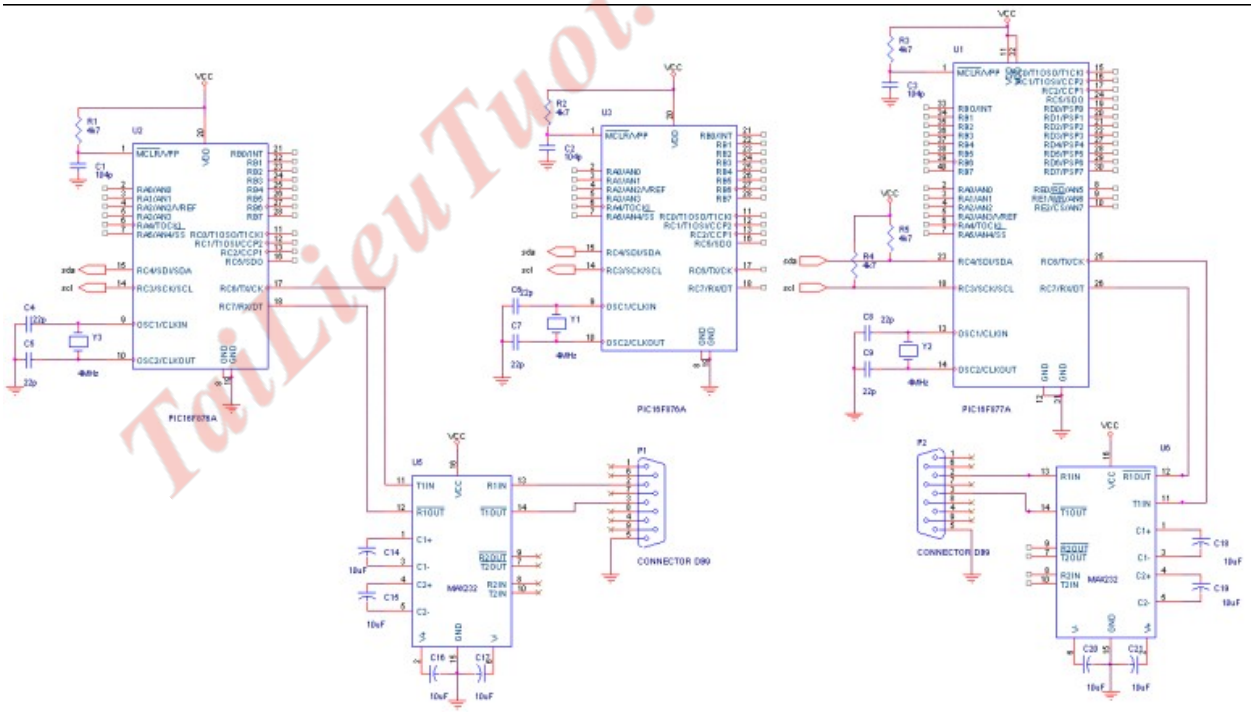
```
//-------------------------------------------------------
// Issue 1.0 : 21/12/1999 : First Officially Released
//-------------------------------------------------------

#include "h8genlib.h"
#include "ioh8314.h"

extern byte p4dr;

// Select device
void spi_dac_select(void)
{
  p4dr = p4dr & 0b11110111;
  P4DR = p4dr;
}
// Deselect device
void spi_dac_deselect(void)
{
        p4dr = p4dr | 0b00001000;
        P4DR = p4dr;
}
// Select device
void spi_adc_select(void)
{
  p4dr = p4dr & 0b11111011;
  P4DR = p4dr;
}
// Deselect device
void spi_adc_deselect(void)
{
        p4dr = p4dr | 0b00000100;
        P4DR = p4dr;
}
// Set clock high
void spi_hiclock(void)
{
        p4dr = p4dr | 0b00000010;
  P4DR = p4dr;
}
// Set clock low
void spi_loclock(void)
{
  p4dr = p4dr & 0b11111101;
  P4DR = p4dr;
}
// Set Data high
void spi_hidata(void)
{
  p4dr = p4dr | 0b00000001;
  P4DR = p4dr;
```

```
}
// Set data low
void spi_lodata(void)
{
        p4dr = p4dr & 0b11111110;
        P4DR = p4dr;
}
// end
```

12. Các chuẩn giao tiếp

12.1. Chuẩn giao tiếp I2C

-Sơ đồ:



-Mình dùng lệnh I2C trong CCS có hiệu quả ko ?!

Vì mình đang viết thử I2C. Thấy chương trình mẫu(EX_Slave) trong CCS C mô phỏng bộ nhớ ngoài 24xx chuẩn, nên nạp chạy thử thấy: viết từ Master (Ctr Mater lấy ví dụ: EX_EXTEE) ít nhất là 4 lần mới được, còn đọc thì lại ko được. Xem tài liệu thì thấy có rất nhiều bit bẫy tình huống I2C, nhưng các lệnh về I2C trong CCS thì đơn giản và ít. Vậy, có vấn đề gì không.

Code:

```
#include <16F876A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)  // Jumpers: 8 to 11, 7 to
12

#use i2c(SLAVE, SDA=PIN_C4, SCL=PIN_C3, address=0xa0)

typedef enum {NOTHING, CONTROL_READ,
              ADDRESS_READ, READ_COMMAND_READ} I2C_STATE;
I2C_STATE fState;
BYTE address, buffer[0x10];
```

```
#INT_SSP
void ssp_interupt ()
{
   BYTE incoming;
   if (i2c_poll() == FALSE) {
      if (fState == ADDRESS_READ) {  //i2c_poll() returns false on the
         i2c_write (buffer[address]);//interupt receiving the second
         fState = NOTHING;           //command byte for random read
operation
      }
   }
   else {
      incoming = i2c_read();
      if (fState == NOTHING){
         fState = CONTROL_READ;
      }
      else if (fState == CONTROL_READ) {
         address = incoming;
         fState = ADDRESS_READ;
      }
      else if (fState == ADDRESS_READ) {
         buffer[address] = incoming;
         fState = NOTHING;
      }
   }
}
void main ()
{
   int i;

   fState = NOTHING;
   address = 0x00;
   for (i=0;i<0x10;i++)
      buffer[i] = 0x00;
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_SSP);

   while (TRUE) {}
}
```


Hà vừa thử chtr I2c: Master 16F877A, Slave 16F876A, cùng giao tiếp PC để kiểm tra dữ liệu đọc và viết của Master và Slave> Thấy rất tốt ! I2C hay thiệt! Nhiệm vụ bây giờ mình chỉ tạo giao thức dữ liệu, còn chtr service plug&play slave thì thấy hơi khó! Nhưng mình cố gắn viết thử. Cảm ơn Hoanf, mọi người rất nhiều !!
Code:

```
*******************************************
Chtr Master, mình thay đổi chút ít từ chtr mẫu EX_EXTEE.
          1.chỉ liên kết 2401.c (thay vì địa chỉ mặc đinh là 0xa0,
mình thay là 0x10).
          2. Vẫn truyền lệnh từ PC
*******************************************
#include <16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
```

```
#include <input.c>
#include <2401.c>

void main() {

   BYTE value, cmd;
   EEPROM_ADDRESS address;

   init_ext_eeprom();

   do {
      do {
         printf("\r\nRead or Write: ");
         cmd=getc();
         cmd=toupper(cmd);
         putc(cmd);
      } while ( (cmd!='R') && (cmd!='W') );

      printf("\n\rLocation: ");

      address = gethex1();
      if(cmd=='R')
         printf("\r\nValue: %X\r\n",READ_EXT_EEPROM( address ) );
      if(cmd=='W') {
         printf("\r\nNew value: ");
         value = gethex();
         printf("\n\r");
         WRITE_EXT_EEPROM( address, value );
      }
   } while (TRUE);
}
```


Code:

```
*********************************************
Chtr Slave, thì có thay đổi chút ít từ chtr mẫu EX_Slave:
                  1. giám sát 3 ô nhớ trên PC
                  2. khai báo địa chỉ la 0x10.
*********************************************
#include <16F876A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#use i2c(SLAVE, SDA=PIN_C4, SCL=PIN_C3, address=0x10)

BYTE address, buffer[0x10];

#INT_SSP
void ssp_interupt ()
{
   BYTE incoming, state;

        state = i2c_isr_state();

        if(state < 0x80)                     //Master is sending data
        {
                incoming = i2c_read();
                if(state == 1)               //First received byte is
address
```

```
                        address = incoming;
                if(state == 2)                //Second received byte is data

                        buffer[address] = incoming;
        }
        if(state == 0x80)                      //Master is requesting data
        {
                i2c_write(buffer[address]);
        }
}
void main ()
{
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_SSP);

   while (TRUE) {
          printf("\r\nValue0: %X\r\n",buffer[0]);
          delay_ms(1000);
          printf("\r\nValue1: %X\r\n",buffer[1]);
          delay_ms(1000);
          printf("\r\nValue2: %X\r\n",buffer[2]);
          delay_ms(1000);
          printf("\r\nValue3: %X\r\n",buffer[3]);
          delay_ms(1000);
}
}
```

Mình mới viết được chương trình giao tiếp I2C đơn giản dùng các hàm của CCS.
Mời các bạn xem và cho nhận xét
Master: truyền dữ liệu cho Slave. Mỗi lần truyền 1 byte.
Code:

```
#include <16F877A.H>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use Delay(Clock=4000000)

#define SLAVE_ADDRESS 0x10
#use i2c(master, sda=PIN_C4, scl=PIN_C3)

void write_I2C(int8 a)
{
   i2c_start();
   i2c_write(SLAVE_ADDRESS);
   i2c_write(a);
   i2c_stop();
}
void main()
{
   int8 value;

   value = 0;
   while(1){
      write_I2C(value);
      value++;
      delay_ms(100);
   }
}
```

Slave thì chỉ tiến hành kiểm tra có phải Master truyền hay kô. Nếu truyền thì nhận byte dữ liệu và hiển thị lên port_B:

Code:

```
#include <16F877A.H>
#fuses XT,NOWDT,NOPROTECT,NOLVP

#use delay(Clock=4000000)
#use i2c(SLAVE, SDA=PIN_C4, SCL=PIN_C3, address=0x10)

int8 value;
#INT_SSP
void i2c_isr()
{
   int8 state;
   int8 address;
   state = i2c_isr_state();
   if(state == 0)
      address = i2c_read();
   else if(state < 0x80)
         value = i2c_read();
}
void main()
{
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_SSP);

   set_tris_b(0x00);
   while(1){
      output_b(value);
   }
}
```


Mình gặp trục trặc khi đọc dữ liệu từ Slave về.
Đây là đoạn code sử dụng cho 2 con PIC 18F877A.
Master: Yêu cầu Slave nhận dữ liệu liên tục (cách nhau 500ms) và nó sẽ hiển thị giá trị nhận được lên Port_b. Dùng leds để quan sát.

Code:

```
#include <16F877A.H>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use Delay(Clock=4000000)

#define SLAVE_ADDRESS 0x10
#use i2c(master, sda=PIN_C4, scl=PIN_C3)

int8 read_I2C()
{
   int8 value;

   i2c_start();
   i2c_write(SLAVE_ADDRESS + 1);
   value = i2c_read();
   i2c_stop();
   return value;
}
void main()
{
   int8 value;
```

```
   set_tris_b(0x00);

   while(1){
      value = read_I2C();
      output_b(value);
      delay_ms(500);
   }
}
```

Slave: Truyền dữ liệu cho master và mỗi lần truyền thì giá trị cần truyền tăng lên 1 đơn vị.

Code:

```
#include <16F877A.H>
#fuses XT,NOWDT,NOPROTECT,NOLVP

#use delay(Clock=4000000)
#use i2c(SLAVE, SDA=PIN_C4, SCL=PIN_C3, address=0x10)

int8 value = 0x01;

#INT_SSP
void i2c_isr()
{
   int8 state;
   int8 address;
   state = i2c_isr_state();
   if(state >= 0x80){
      i2c_write(value);
      value++;
   }
}
void main()
{
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_SSP);

   set_tris_b(0x00);
   while(1){
      output_b(value);
   }
}
```

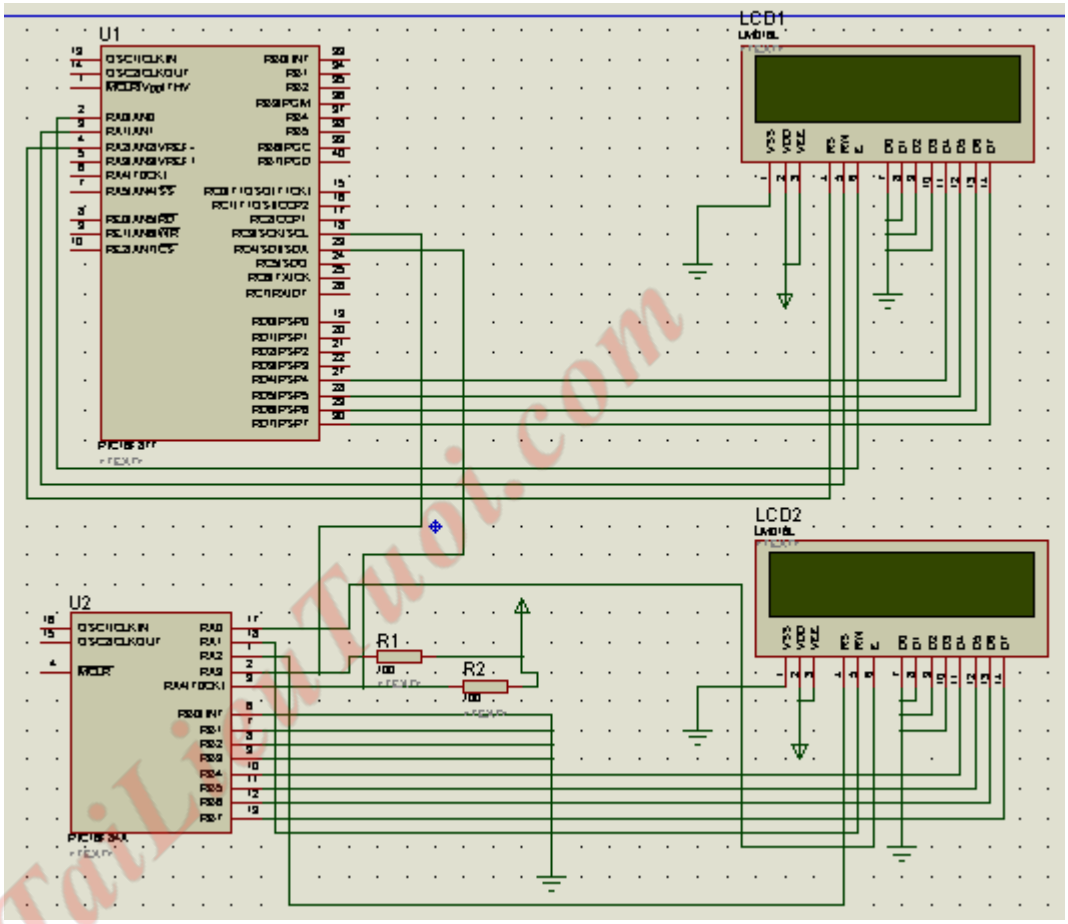
Theo Hà nghĩ, ở Master, khi mình đọc cũng phải chờ xung ACK từ Slave thì chtr ko bị rối. Bạn thử thêm một chtr con chờ Bus trong <2401.c>:

Code:

```
//**************code thêm vào*********************
BOOLEAN ext_eeprom_ready()
{
   int1 ack;
   i2c_start();
   ack = i2c_write(SLAVE_ADDRESS);
   i2c_stop();
   return !ack;
}
//*************************************************
int8 read_I2C()
{
    int8 value;
```

```
    while(!ext_eeprom_ready());                          // code thêm vào
    i2c_start();
    i2c_write(SLAVE_ADDRESS + 1);
    value = i2c_read();
    i2c_stop();
    return (value);
}
void main()
{
    int8 value;

    value = 0x00;
    set_tris_b(0x00);

    while(1){
    value = read_I2C();
    output_b(value);
    delay_ms(500);
    }
}
```


12.1.1. Master_Slave



12.1.1.1. I2Cmaster

```
//*************************************************************************
// Day la chuong trinh vi du su dung truyen thong I2C
// This program is a sample of I2C comunication.
//This is master mode
//This include next funtions:
//                      1. Send ASCII code periodically to slave
//                      2. Display send data on LCD display
// Hanoi: 12/5/2007
//Writer: Nguyen Khac Hieu
//user: all of my friends and students etc...
//*************************************************************************
#include <16F84A.h>
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=4000000)
#byte PORTA=5
#byte PORTB=6
// Khai bao LCD
#define rs  PIN_A2
#define rw  PIN_A1
#define stb PIN_A0
#byte   db = 6 // khai bao c?ng D cho LCD
#define Amode 0x0
#define Bmode 0x0F
#include <lcd1_lib.c>
```

```
#use I2C(master, sda=PIN_A4, scl=PIN_A3)
// Chuong trinh chinh
void khoitao();
void main(void)
{
        int asci,i;
        set_tris_b(Bmode);
        set_tris_a(Amode);
        lcd_init();
   lcd_clear();
   lcd_data("Start I2C test");
        delay_ms(3000);
                while(1)
                {
                        asci=0x30;
                        lcd_clear();
                        do{
                                i2c_start();
                                i2c_write(0xa0);
                                for(i=1;i<=8;++i){
                                        i2c_write(asci);
                                        lcd_data(asci);
                                        asci=asci+1;
                                        delay_us(100);
                                }
                        i2c_stop();
                        delay_ms(1000);

                        }while(asci<0xFE);
                }
}
```

12.1.1.2. I2Cslave

```
//*************************************************************************
// This program is a sample of I2C comunication.
//This is slave mode
//This include next funtions:
//                      1. Receive data from I2C master CPU
//                      2. Display received data on LCD display
// Hanoi: 12/5/2007
//Writer: Nguyen Khac Hieu
//user: all of my friends, my students etc...
//*************************************************************************
#include <16F877A.h>
#fuses NOWDT,PUT,HS,NOPROTECT
#use delay(clock=4000000)
// Khai bao LCD
#define rs  PIN_A2
#define rw  PIN_A1
#define stb PIN_A0
#byte   db = 0x08
```

```
#define Amode 0x00
#define Dmode 0x0F
#include <lcd2_lib.c>
#use I2C(slave, sda=PIN_C4, scl=PIN_C3, address=0xa0, NOFORCE_SW)
// Chuong trinh chinh
void khoitao();
void main(void)
{
        int indata;
        set_tris_d(Dmode);
        set_tris_a(Amode);
//      set_tris_c(0x99);
        lcd_init();
   lcd_clear();
   lcd_data("Start I2C test");
        delay_ms(2000);
        lcd_clear();
                while(1)
                {
                        if(i2c_poll()){
                                indata=i2c_read();
                                lcd_data(indata);
                                }
                }
}
```

12.1.2. lcd1_lib

```
//////////////////////////////////////////////
//  LCD control Library
//  functions are below
//    lcd_init()-------- initialize
//    lcd_ready()------- busy check
//    lcd_cmd(cmd)------ send command
//    lcd_data(string)-- display string
//    lcd_clear() ------ clear display
//////////////////////////////////////////////
////////// lcd ready check function
int lcd_ready(){
   int high,low;
   set_tris_b(Bmode | 0xF0);   //upper is input
   output_low(rs);
   output_high(rw);           //read mode
   output_high(stb);
   high= PORTB& 0xF0;            //input upper
   output_low(stb);
   output_high(stb);
   low=PORTB & 0xF0;             //input lower
   output_low(stb);
   set_tris_b(Bmode);
```

```
   return(high | (low>>4));    //end check
}
////////// lcd display data function
void lcd_data(int asci){
   PORTB = asci;                //set upper data
   output_low(rw);             //set write
   output_high(rs);            //set rs high
   output_high(stb);           //strobe
   output_low(stb);
   asci=asci<<4;
   PORTB = asci;                //set lower data
   output_high(stb);           //strobe
   output_low(stb);
   while(bit_test(lcd_ready(),7));
}
////////// lcd command out function
void cmdout(int cmd){
   PORTB = cmd;                 //set upper data
   output_low(rw);             //set write
   output_low(rs);             //set rs low
   output_high(stb);           //strobe
   output_low(stb);
   cmd=cmd<<4;
   PORTB = cmd;                 //set lower data
   output_high(stb);           //strobe
   output_low(stb);
}
void lcd_cmd(int cmd){
   cmdout(cmd);
   while(bit_test(lcd_ready(),7)); //end check
}
//////////  lcd display clear function
void lcd_clear(){
   lcd_cmd(1);                 //initialize command
}
///////// lcd initialize function
void lcd_incmd(int cmd){
   PORTB = cmd;                 //mode command
   output_low(rw);             //set write
   output_low(rs);             //set rs low
   output_high(stb);           //strobe
   output_low(stb);
   delay_us(100);
}
void lcd_init(){
   delay_ms(15);
```

```
   lcd_incmd(0x30);          //8bit mode set
   delay_ms(5);
   lcd_incmd(0x30);          //8bit mode set
   delay_ms(1);
   lcd_incmd(0x30);          //8bit mode set
   lcd_incmd(0x20);          //4bit mode set
   lcd_cmd(0x2E);            //DL=0 4bit mode
   lcd_cmd(0x08);            //disolay off C=D=B=0
   lcd_cmd(0x0D);            //display on C=D=1 B=0
   lcd_cmd(0x06);            //entry I/D=1 S=0
}
```

12.1.3. lcd2_lib

```
////////////////////////////////////////////////
//  LCD control Library
//  functions are below
//    lcd_init()-------- initialize
//    lcd_ready()------- busy check
//    lcd_cmd(cmd)------ send command
//    lcd_data(string)-- display string
//    lcd_clear() ------ clear display
////////////////////////////////////////////////
////////// lcd ready check function
int lcd_ready(){
   int high,low;
   set_tris_d(Dmode | 0xF0);   //upper is input ------------
   output_low(rs);
   output_high(rw);            //read mode
   output_high(stb);
   high=db & 0xF0;             //input upper
   output_low(stb);
   output_high(stb);
   low=db & 0xF0;              //input lower
   output_low(stb);
   set_tris_d(Dmode);//-------------------------------------
   return(high | (low>>4));    //end check
}
////////// lcd display data function
void lcd_data(int asci){
   db = asci;                  //set upper data
   output_low(rw);             //set write
   output_high(rs);            //set rs high
   output_high(stb);           //strobe
   output_low(stb);
   asci=asci<<4;
   db = asci;                  //set lower data
   output_high(stb);           //strobe
   output_low(stb);
   while(bit_test(lcd_ready(),7));
}
```

```
////////// lcd command out function
void cmdout(int cmd){
   db = cmd;                //set upper data
   output_low(rw);           //set write
   output_low(rs);           //set rs low
   output_high(stb);         //strobe
   output_low(stb);
   cmd=cmd<<4;
   db = cmd;                //set lower data
   output_high(stb);         //strobe
   output_low(stb);
}
void lcd_cmd(int cmd){
   cmdout(cmd);
   while(bit_test(lcd_ready(),7)); //end check
}
//////////  lcd display clear function
void lcd_clear(){
   lcd_cmd(1);              //initialize command
}
///////// lcd initialize function
void lcd_incmd(int cmd){
   db = cmd;                //mode command
   output_low(rw);           //set write
   output_low(rs);           //set rs low
   output_high(stb);         //strobe
   output_low(stb);
   delay_us(100);
}
void lcd_init(){
   delay_ms(15);
   lcd_incmd(0x30);          //8bit mode set
   delay_ms(5);
   lcd_incmd(0x30);          //8bit mode set
   delay_ms(1);
   lcd_incmd(0x30);          //8bit mode set
   lcd_incmd(0x20);          //4bit mode set
   lcd_cmd(0x2E);            //DL=0 4bit mode
   lcd_cmd(0x08);            //disolay off C=D=B=0
   lcd_cmd(0x0D);            //display on C=D=1 B=0
   lcd_cmd(0x06);            //entry I/D=1 S=0
}
```

12.2. Giao tiếp RS232

Serial Port - lập trình giao tiếp nối tiếp

Tôi xin copy tất cả các bài của tôi vào box này để cho tổng quát hơn giới thiệu về giao tiếp nối tiếp qua Serial Port như RS-232, RS-485. Hi vọng là cho các bạn có kiến thức đầy đủ về lập trình nối tiếp nói chung và qua RS232, RS-485 nói riêng.

Giới thiệu giao tiếp cổng com - chapter1

Tôi xin giới thiệu cho các bạn chi tiết các thông tin, cách lập trình truyền thông nối tiếp dùng chuẩn RS-232 và RS-485.

Có thể có nhiều bạn đã biết rồi hoặc là chưa biết nhưng tôi mong rằng sẽ bổ sung cho các bạn một số điều cơ bản.

Tất cả mọi vấn đề tôi dịch từ cuốn Serial_complete của Jan Axelson và thực tế lập trình truyền thông giao tiếp với vdk AT89C51(một loại vi điều khiển đơn giản). Hi vọng các vấn đề tôi đưa ra sẽ làm các bạn môt phần nào hiểu biết thêm về cổng thông dụng này cổng Comm(communicactions).

Do thời gian không có nhiều vừa làm vừa ôn thi nên mỗi ngày tôi sẽ up lên từng vấn đề một. Mong các bạn góp ý thêm. Tôi làm cái này không có ý qua mặt các cao thủ, mong các cao thủ bỏ quá cho. Các bạn có thể có thêm nhiều ví dụ khi vào trang web http://www.lvr.com

1. Thứ nhất tôi xin giới thiệu một cách khái quát về các cổng và đặc biệt là cổng RS-232 và RS-485.

Trước hết tôi xin định nghĩa liên kết: là dùng dây dẫn hoặc trung gian khác như công nghệ không dây,.. để kết nối các máy tính hoặc là kết nối các thiết bị kết nối qua PC. Khoảng cách có thể rất ngắn vài cm đến hàng ngàn km, thời gian có thể 1 giây có khi đến hàng 1 tuần. Các điểm kết nối vào mạng có thể là 2 hoặc nhiều điểm.

Đặc điểm so sánh giữa các cổng nay quên không mang bảng so sánh, mai mang sau)

- Chuẩn RS-232 chỉ có thể kết nối nhiều nhất 2 thiết bị, với khoảng cách dài nhất là 50-100 feet(12,7->25,4 m), tốc độ 20k bit/s
- Chuẩn RS-485 có thể kết nối tối đa là 32 thiết bị, khoảng cách dài hơn tối đa là 4000feet(1016 m-> hơn 1km) gấp 40 lần RS-232. Tốc độ cao 10 mega bit/s. Cả hai chuẩn này có thẻ có sẵn trong mainboard khi mua hoặc là có thể lắp thêm rất dễ dàng, giá mua rất rẻ so với các giao diện khác.Chuẩn RS-232 dùng rộng rãi, mua dễ dàng, đơn giản khi lắp thêm với nhiều cách thiết lập. Còn RS-485 dùng với khoảng cách lớn hơn, tốc độ cao hơn, nhiều đầu nối hơn.
- Chuẩn IrDA(Inared Data Asociation) dùng các UART giống nhau và định dạng dữ liệu giống như RS-232 nhưng có thêm bộ giải mã. Dữ liệu truyền từ nguồn phát hồng ngoại đến các thiết bị không giây. Giao diện này rất là có ích cho các liên kết ngắt, giữa các thiết bị mà không thể có cáp nối ở giữa(có thể là cho đẹp).
- Chuẩn MIDI(Musical instrument digital interface): giao diện số hóa các dụng cụ âm nhạc. Chuẩn này dùng dòng 5mA, tốc độ 31,5k bit/s
- Microwire, I2C,SPI là các chuẩn nối tiếp đồng bộ, dùng trong các liên kết ngắn. Nhiều vdk có 1 hoặc nhiều chuẩn này.
- USB(Universial Sẻial Bus) và Fireware(chuẩn IEEE-1384) là chuẩn mới, tốc độ cao, thông minh kết nối với PC và các PC khác, thiết bị ngoại vi. USB ra đời đã dần thay thế chuẩn RS-232 và chuẩn máy in Centronic như là một lựa chọn mới hiện đại cho các TB ngoại vi. Fireware tốc độc truyền dữ liệu nhanh hơn nhiều và được dùng để truyền nhạc tiếng, nhạc hình, hoặc các block dung lượng lớn.
- Ethernet là các chuẩn mạng gần gũi thường dùng trong nhiều mạng. Tốc độ cao nhưng yêu cầu phần cứng và phần mềm khá phức tạp, đắt hơn nhiều so với các chuẩn khác.
- Đồng hành với chuẩn nối tiếp là chuẩn song song, có nhiều đường dữ liệu. Ví song song nên chuyển nhiều bit cùng một lúc, rất nhanh. Thường có một loạt các đường dữ

liệu -> dữ liệu truyền đi theo một chiều ở một thời điểm. Nếu dùng để nối khoảng cách xa thì tiền mua quả là đắt đỏ cho việc thực hành của sinh viên cũng như các ứng dụng trong công nghiệp, ..

- Chuẩn máy in Cenntronics(IEEE-1284). Mọi pc đều có chuẩn này. Tốc độ truyền cao qua cáp. Được ứng dụng với các máy quét, các thiết bị lưu trữ mở rộng như đĩa cứng,.. và nhiều thiết bị ngoại vi đặc biệt khác. ChuẩnIEEE-488 là chuẩn song song dùng trong các ứng dụng điều khiển và trong âm nhạc.

Nói thêm về cổng RS-232 và RS-485:

Cổng nối tiếp là một phần của PC ngay khi nó mới ra đời. Với mỗi cổng Com hoặc Comm(communications) trong PC là một cổng nối tiếp không đồng bộ được điều khiển bởi cácUART. Mỗi cổng Com có thể có giao diện RS-232, RS – 485 hoặc cổng có thể để dành cho một modem trong hoặc thiết bị khác. Mỗi PC có thể có các dạng khác nhau của các cổng nối tiếp nhưUSB, Firewire, và I2C nhưng chúng dùng các giao thức khác nhau và yêu cầu các thành phần khác nhau.

Giao diện nối tiếp mới nhất là USB và Firewire với các tính năng: nhanh, và nhiều lợi ích khác nữa. Thực tế thì nên dùng USB trong việc thay thế cổng RS-232 ở bất kì nơi nào có thể được trong các thiết kế mới, ứng dụng mới. Và với nhều thiết bị ngoại vi, giao diện mới này lại rất là phù hợp.

Nhưng RS – 232 và các giao diện giống nó vẫn sẽ tiếp tục phổ biến trong các ứng dụng như là hệ thống điều khiển, điều hành. Những giao diện này không đắt đỏ, dễ dàng lập trình, cho phép cáp dài,và dễ dùng kết hợp với các thiết bị vi điều khiển rẻ tiền, các máy tính cũ. Như cổngUSB đã được sử dụng rộng rãi, các bộ chuyển đổi sẵn sàng để chuyểnUSB thành cổng RS-232 hoặc RS – 285. Bộ chuyển đổi sẽ kết nối với cổng USB của PC và chuyển đổi giữa USB và các giao diện khác. Thiết lập rất đơn giản để thêm một cổng RS-232 hoặc RS – 485 vào bất kỳ hệ thống nào.

Uart

Tiếp theo tôi xin giới thiệu cho các bạn UART:

Trong những máy tính IBM – PC đầu tiên, UART điều khiển cổng nối tiếp là 8250, tốc độ lớn nhất là 57.600 bit /giây. Các UART từ thời gian này đã được tiếp tục cải thiện thêm nhiều đặc điểm mới như thêm bộ đệm, tốc độ tăng lên. Ngày nay, UART trong PC là một phần không thể thiếu của chíp đa chức năng bao gồm một hay nhiều UART hỗ trọ cổng song song, thiết bị lưu trữ và các bộ phận hệ thống khác.

UART chuyển đổi giữa dữ liệu nối tiếp và song song. Một chiều, UART chuyển đổi dữ liệu song song bus hệ thống ra dữ liệu nối tiếp để truyền đi. Một chiều khác, UART chuyển đổi dữ liệu nhận được dạng dữ liệu nối tiếp thành dạng dữ liệu song song cho CPU có thể đọc vào bus hệ thống.

UART của PC hỗ trợ cả hai kiểu giao tiếp là giao tiếp đồng thời và không giao tiếp đồng thời. Giao tiếp đồng thời tức là UART có thể gửi và nhận dữ liệu vào cùng một thời điểm. Còn giao tiếp không đồng thời(không kép) là chỉ có một thiết bị có thể chuyển dữ liệu vào một thời điểm, với tín hiệu điều khiển hoặc mã sẽ quyết định bên nào có thể truyền dữ liệu. Giao tiếp không đồng thời được thực hiện khi mà cả 2 chiều chia sẻ một đường dẫn hoặc nếu có 2 đường nhưng cả 2 thiết bị chỉ giao tiếp qua một đường ở cùng một thời điểm.
Thêm vào đường dữ liệu, UART hỗ trợ bắt tay chuẩn RS232 và tín hiêu điều khiển như RTS, CTS, DTR, DCR, RT và CD.

Để thuận tiện, các chương trình gửi và nhận dữ liệu trong định dạng không đồng bộ đơn giản hơn những gì bạn tưởng. PC và nhiều vi xủ lí khác có một bộ phận gọi là UART(universal asynchronous receiver/transmitter: truyền /nhận không đồng bộ chung) vì thế có thể vận dụng phần lớn những chi tiết truyền và nhận dữ liệu.

Thang8831
http://www.picvietnam.com

Trong PC, hệ điều hành và ngôn ngữ lập trình hỗ trợ cho lập trình liên kết nối tiếp mà không cần phải hiểu rõ chi tiết cấu trúc UART . Để mở liên kết, ứng dụng lựa chọn một tần số dữ liệu hoặc là thiết lập khác hoặc cho phép truyền thông tại các cổng. Để gửi 1 byte, ứng dụng ghi byte này vào bộ đệm truyền của cổng được lựa chọn, và UART gửi dữ liệu này, từng bít một, trong định dạng yêu cầu, thêm bít Start, bít Stop, bít chẵn lẻ khi cần. Trong một cách đơn giản, byte nhận được tự động được lưu trữ trong bộ đệm. UART cso thể dùng nhanh một ngắt để báo choCPU và các ứng dụng biết dữ liệu đang nhận được và các sự kiện khác.

Một vài vi điều khiển không bao gồm UART, và thỉnh thoảng bạn cần nhiều hơn các UART mà vi xử lí có. Trong trường hợp này, có 2 lựa chọn: thêm UART ngoài, hoặc mô phỏng UART trong mã chương trình. Basic Stamp của Parallax là một ví dụ của chip với một UART bổ sung trong mã chuơng trình. UART là một thiết bị đơn giản hỗ trợ tốt cả hai kiểu truyền thông đồng bộ và không đồng bộ.

Định dạng và giao thức đồng bộ và không đồng bộ

--Định dạng và giao thức Format và Protocol):

Máy tính trong một mắt xích nối tiếp có thể là nhiều dạng khác nhau, nhưng tất cả chúng phải cùng đồng ý một thoả thuận và qui tắc để cho dữ liệu có thể trao đổi giữa chúng. Sự thoả thuận này phải chắc chắn rằng mọi sự chuyển dữ liệu phải tìm thấy được nơi nó cần đến, và mỗi máy tính phải hiểu thông điệp gửi tới nó.

Dưới đây giới thiệu về cách định dạng dữ liệu và giao thức dùng trong truyền thông nối tiếp. Chủ yếu là định dạng không đồng bộ dùng 2 chuẩn thông dụng là RS-232 và RS-485.

+Gửi dữ liệu nối tiếp

Trong một liên kết nối tiếp, nơi gửi dữ liệu sẽ gửi từng bit một ở mỗi thời điểm nối tiếp nhau. Một liên kết nối tiếp chỉ có 2 thiết bị thì phải có đường dẫn dành cho mỗi chiều truyền hoặc là nó chỉ có 1 đường dẫn được chia sẻ bởi cả 2 thiết bị với thoả thuận của 2 thiết bị này. Khi mà có 3 hoặc nhiều thiết bị, tất cả các thiết bị này thường dùng chung một đường dẫn, và giao thức mạng quyết định xem thiết bị nào có quyền truyền nhận dữ liệu.

Một tín hiệu đòi hỏi bởi tất cả mọi liên kết nối tiếp là tín hiệu xung đồng hồ, hoặc là có sự tham khảo về thời gian để điều khiển đường truyền dữ liệu. Nơi truyền và nơi nhận dùng xung đồng hồ để quyết định khi nào gửi và khi nào đọc mỗi bít. Có hai dạng định dạng dữ liệu là đồng bộ và không đồng bộ, và mỗi định dạng này dùng các dạng xung đồng hồ khác nhau.

+Định dạng đồng bộ:

Trong truyền đồng bộ, mọi thiết bị dùng một xung đồng hồ được phát ra bởi một thiết bị hoặc từ một nguồn xung ngoài. Xung đồng hồ có thể có một tần số cố định hoặc cóc thể chốt tại những khoảng thời gian không đều. Mọi bít truyền đi được đồng bộ với đồng hồ. Nói cách khác, mỗi bít được truyền đi là dựa vào sự chuyển đổi của xung(như tăng hoặc giảm của sường xung). Nơi nhận dùng sự chuyển đổi xung để quyết định khi nào đọc mỗi bít truyền tới. Từ hình vẽ các bạn cũng có thể thấy là nơi truyền sẽ truyền các bit khi mà nhận thấy sự chuyển sườn xung từ cao xuống thấp, và nơi nhận thì ngược lại phát hiện khi nào có sự chuyển sườn xung từ thấp lên cao thì đọc các bit.Chi tiết chính xác của giao thức này có thể biến đổi khác đi. Ví dụ, nơi nhận có thể chốt dữ liệu nhận trong sườn xung tăng hoặc giảm, hoặc là phát hiện mức logic ở mức cao hoặc thấp. Định dạng đồng bộ dùng các cách khác nhau để bắt đầu và kết thúc việc truyền dữ liệu, bao gồm bít Start và bít Stop và tín hiệu lựa chọn chíp.

+Định dạng không đồng bộ:

Trong truyền không đồng bộ, liên kết không bao gồm đường xung đồng hồ, bởi vì mỗi điểm đầu cuối của liên kết đã có xung đồng hồ cho riêng từng cái. Mỗi điểm sẽ cần phải đồng ý cùng một tần số của đồng hồ và mọi đồng hồ chỉ khác nhau một vài %. Mỗi byte truyền đi bao gồm bít Start để đồng bộ đồng hồ và một hoặc nhiều bít Stop cho tín hiệu kết thúc việc truyền trong mỗi một từ được truyền đi. Cổng RS-232 trong PC dùng định dạng khoong đồng

bộ để giao tiếp với modems(thiết bị mã hoá, giải mã dữ liệu) và các thiết bị khác. Dù RS-232 có thể truyền dữ liệu đồng bộ nhưng liên kết không đồng bộ vẫn được dùng phổ biến hơn. Phần lớn liên kết RS-485 dùng giao tiếp không đồng bộ.

+**Truyền không đồng bộ**: có thể dùng một trong vài cách định dạng phổ biến. Phổ biến nhất là kiểu 8-N-1, nơi truyền sẽ truyền mỗi byte dữ liệu một bít Start, tiếp theo là 8 bít dữ liệu bắt đầu với bít 0(bít có trọng số nhỏ nhất Least Sìgnificant Bit) và kết thúc với 1 bít Stop.

Các bạn có thể xem hình vẽ để hiểu thêm về cách định dạng này.

Chữ N trong định dạng 8-N-1 chỉ rằng truyền dữ liệu không dùng bít chẵn lẻ. Một dạng định dạng khác là bao gồm một bít chẵn lẻ giống như dạng đơn giản của kiểm soát lỗi.

Khi số các bit 1 trong byte là chẵn thì bít Odd Parity Bit = 1 và bít lẻ = 0,.. Một số dạng khác không phổ biến là dùng một số khác nhau của số bít dữ liệu. Rất nhiều cổng nối tiếp hỗ trợ mọi nơi từ 5 ->8 bít dữ liệu, cộng với bít chẵn lẻ.

Tốc độ số bít là số bít một giây được truyền đi hoặc là nhận về trong một đơn vị thời gian. Tốc độ bus là số các sự kiện hình xảy ra hoặc truyền dữ liệu trên giây. Hai giá trị này thường đồng nhất với nhau trong nhiều liên kết. Trong đường dây điện thoại, môdem tốc độ cao mã hoá nhiều bít trong mỗi chu kì dữ liệu vì thế tốc độ bus thực tế nhỏ hơn tốc độ bit(bit rate).

Mọi bít cần thiết cho truyền một giá trị từ bít Start đến bít Stop gọi là một Word. Mỗi bít trong dạng Word gọi là một Character. Trong vài liên kết, các bít là kí tự văn bản(dạng chữ hoặc số), trong khi các dạng kí tự khác lại là giá trị nhị phân. Thời gian truyền các các kí tự trong một giây bằng với tổng thời gian truyền từng bít trong word cộng lại. Thêm bít start và bít Stop làm tăng thời gian truyền mỗi byte lên 25% (vì có 10 bít cần truyền trong khi chỉ dùng có 8 bít). Với định dạng 8-N-1, một byte truyền với thời gian bằng 1/10 tần số bus: do đó 9600 bít/s truyền 960 byte/s. Nếu nơi nhận đòi hỏi phải có một thời gian kiểm tra dữ liệu nhận được, nơi truyền sẽ kéo dài độ rộng của bít Stop ra 1,5 hoặc 2 bít.

Cơ chế chống mất dữ liệu

Tôi xin giới thiệu cho các bạn cơ chế chống mất dữ liệu trong khi truyền nhận dữ liệu giữa các nút trong mạng lưới:

Phần lớn các máy tính trong mạng nối tiếp có nhiều việc phải làm bên cạnh việc chờ nhận dữ liệu. Ví dụ, mỗi đơn vị dữ liệu có thể thu thập theo chu kì và lưu trữ dữ liệu tới khi một mắt xích khác trong mạng yêu cầu dữ liệu này. Hoặc một điều khiển có thể đáp ứng các điều kiện điều khiển và điều hành, thỉnh thoảng lại nhận thông tin hoặc nhận các yêu cầu từ trong mạng.

Một máy tính muốn truyền dữ liệu trong khi một máy nhận khác đang bận với các công việc khác. Việc thiết kế mạng phải đòi hỏi rằng mỗi nơi nhận có thể biết được dữ liệu nào chuyển đến nó và tất cả mọi dữ liệu đến máy nhận phải không có lỗi.

Có nhiều cách làm để thực hiện điều đó, bao gồm bắt tay(handshaking), bộ đệm(buffering), dùng dò(polling) và ngắt(interrupts) để phát hiện dữ liệu đã đến, kiểm soát lỗi(error checking), và thừa nhận dữ liệu đã tới(acknowledging). Mỗi liên kết có thể dùng một hoặc nhiều cách trong số những cách này.

+**Bắt tay(handshaking):**

Với tín hiệu bắt tay, máy phát có thể xác định khi nào máy tính này phải truyền dữ liệu và máy nhận có thể biết khi nào nó sẵn sàng nhận dữ liệu. Tín hiệu có thể biến đổi qua RS-232 hoặc RS-485 theo giao thức chuẩn hoặc giao thức qui ước.

Một trong những dạng bắt tay về phần cứng, nơi nhận đưa ra dòng mức cao khi sẵn sàng nhận dữ liệu, và nơi truyền chờ tín hiệu này trước khi truyền dữ liệu. Nơi nhận có thể đưa ra dòng mức thấp trong mọi thời điểm, thậm chí cả trong quá trình chờ dòng phản hồi cao trước khi kết thúc quá trình truyền nhận. Một số dạng liên kết khác hoạt động giống nguyên tắc ở trên nhưng với bắt tay bằng phần mềm, bằng cách nơi nhận gửi một mã để báo nó sẵn sàng nhận dữ liệu, và một mã khác để báo báo cho nơi truyền dừng quá trình gửi dữ liệu.

+Bộ đệm(Buffer):

Bộ đệm là một dạng khác để nơi nhận có thể chắc chắn là không mất một dữ liệu nào gửi đến chúng. Bộ đệm có thể có ích cho phía truyền, nơi cho phép ứng dụng làm việc có hiệu quả bằng cách lưu trữ dữ liệu để gửi khi liên kết sẵn sàng để truyền nhận dữ liệu.

Bộ đệm có thể là bộ đệm phần cứng, phần mềm hoặc cả hai. Cổng nối tiếp dùng tất cả các dạng này nhưng máy tính cổ nhất có 16 byte bộ đệm phần cứng được tích hợp trong những UART. Trong chiều nhận, điều đó có nghĩa rằng UART có thể lưu trữ 16 byte trước khi phần mềm cần đọc chúng. Trong chiều nhận, UART có thể lưu trữ 16 byte và UART sẽ cẩn thận truyền mỗi byte theo từng bít từng bít theo giao thức lựa chọn.

Khi bộ đệm phần cứng không đủ rộng, một máy tính cá nhân có thể dùng bộ đệm phần mềm, bộ đệm này có thể lập trình được kích thước và kích thước tối đa cho phép bởi bộ nhớ hệ thống. Các thiết bị phần mềm của cổng truyền nhận dữ liệu giữa bộ đệm phần cứng và phần mềm.

Trong các vi điều khiển, bộ đệm có xu hướng trở nên nhỏ hơn, và một số chip không có bộ đệm phần cứng. Việc làm hẹp bộ nhớ đệm điều quan trọng hơn ở đây là các chíp này dùng các công nghệ khác để chắc chắn là không dữ liệu nào bị mất.

+Thăm dò và ngắt:

Sự kiện gây ra ở cổng nối tiếp bao gồm khi truyền và nhận dữ liệu, thay đổi tín hiệu bắt tay, và gửi , nhận thông điệp lỗi. Có hai cách cho ứng dụng phát hiện và gây ra những sự kiện này.

Các thứ nhất là có chương trình tự động nhảy tới các chuỗi sự kiện được sắp xếp trước(như bảng vector ngắt) khi một sự kiện xảy ra. Ứng dụng phản ứng nhanh và tự động hoạt động ở cổng mà không lãng phí thời gian kiểm tra, chỉ cần biết như không có hoạt động nào xảy ra.

Dạng lập trình này gọi là chạy đua sự kiện(event-driven) bởi vì một sự kiện bên ngoài có thể xảy ra trong bất kì thời điểm nào và chương trình chạy tới một bảng đặc biệt.

Trong VB, sự kiện OnComm của MSComm(Microsoft Communication Control 6.0 - Điều khiển ActiveX) làm công việc này. OnComm chạy đáp ứng lại ngắt phần cứng hoặc bộ đếm của bộ đệm phần mềm đạt tới giá trị xảy ra sự kiện. Nhiều bộ vi điều khiển có ngắt phần cứng dùng với mục đích này.

Cách thứ hai là thăm dò bằng cách đọc theo từng chu kì hoặc phát ra tín hiệu tìm kiếm khi nào một sự kiện xảy ra. Dạng lập trình này gọi là lập trình thủ tục, và không dùng ngắt phần cứng. Ứng dụng phải chắc chắn thăm dò cổng một cách đầy đủ để không mất bất kì một dữ liệu nào hoặc sự kiện nào. Tần số thăm dò phụ thuộc vào kích thước bộ đệm và tổng dữ liệu cần lấy(cần cho phản ứng nhanh). Ví dụ, nếu một thiết bị có 16 byte bộ đệm và dò cổng 1 lần/1 giây, thiết bị này chỉ có thể nhận không thể lớn hơn 16 byte/ 1 giây hoặc là bộ đệm sẽ bị tràn hoặc là dữ liệu sẽ bị mất.

Phương pháp thăm dò thường áp dụng cho truyền dữ liệu ngắn, đột ngột hoặc khi máy tính gửi dữ liệu và chớ đợi tín hiệu phản hồi nhanh. Một giao diện thăm dò không yêu cầu ngắt phần cứng, và bạn có thể chạy dạng lập trình này ở trên cổng mà không có đường ngắt. Nhiều giao diện thăm dò dùng ngắt timer của hệ thống để có kế hoạch đọc cổng sau một khoảng thời gian cố định.

+Thừa nhận(Actknowledgments):

Một vài liên kết có các nút chấp nhận mệnh lệnh mà không có một phản ứng nào, nhưng bình thường nó có ích cho nút nhận để cho bên truyền biết rằng một thông điệp đã truyền qua, thậm chí nếu bên nhận không có một thông tin nào phản hồi. Sự thừa nhận này đặc biệt có ích trong mạng lưới, khi có nhiều nút chia sẻ cùng đường truyền thông và nơi truyền đang chuyển đổi tại thời gian không đúng có thể cản trở một thông điệp của nơi truyền khác.

Actknowledgments có thể là một byte đã được định nghĩa sẵn, như là một giá trị mà đã

đồng hoá với bên nhận, hoặc là nút truyền có thể cho rằng có một nút nhận được thông điệp của nó khi nó nhận được yêu cầu dữ liệu đáp lại. Nếu nút truyền không nhận được phản ứng phản ứng mà nó yêu cầu, nút này sẽ cho rằng có một lỗi và truyền lại hoặc là làm các công việc khác.

Khi truyền tới một nút mà không có bộ đệm nhận hoặc có bộ đệm kích thước nhỏ, bên truyền có thể dùng Actknowlegments để chắc chắn rằng sẽ có sự tham gia của nút nào đó vào quá trình truyền nhận trước khi gửi một gói dữ liệu. Nút truyền bắt đầu bằng cách gửi một byte tới tín hiệu mà nó muốn gửi dữ liệu. Khi nút nhìn thấy byte, nút này gửi một Actknowlegment và sau đó tập trung vào việc xem xét ở đầu vào nối tiếp của nó. Khi nơi nhận nhận được Actknowgment , nó biết rằng đã an toàn và yên tâm cho việc gửi dữ liệu.

+Kiểm tra lỗi(Error Checking):

Bên nhận có thể dùng Error- Checking để kiểm tra rằng mọi dữ liệu đến đúng đích. Cách để kiểm tra thông điệp lỗi bao gồm gửi dữ liệu bản sao và byte kiểm tra lỗi.

Một dạng đơn giản kiểm tra lỗi đơn giản là dùng dữ liệu bản sao. Bên truyền gửi mỗi thông điệp 2 lần và bên nhận kiểm tra để xác định rằng 2 thông điệp này đều giống nhau trong cả 2 lần. Tất nhiên, điều đó có nghĩa rằng mỗi thông điệp sẽ mất gấp đôi thời gian truyền. Nó quả thật là hữu ích khi gửi một dữ liệu ngắn quan trong thỉnh thoảng, bất chợt. Nhiều điều khiển hồng ngoại dùng dạng thức này.

Mộtcách thức khác của eror-checking là gửi một byte kiểm tra lỗi cùng với dữ liệu. Checksum tính toán bằng cách thực hiện một vài phép tính toán số học hoặc logic trên byte đó trong thông điệp. Vì thế sẽ thêm một byte kiểm tra vào trong mỗi byte của thông điệp và dùng byte sau cùng để làm kết quả của sự kiểm tra.

Nơi nhận lặp đi lặp lại quá trình tính toán này, và nếu nó nhận được các kết quả khác nhau thì có nghĩa rằng nó không nhận được đúng dữ liệu đã gửi.

Một dạng khác của byte kiểm tra là CRC(cyclic redundancy code) dùng nhiều tính toán phức tạp và thực tế hơn nhiều so với checksum. Một vài giao thức phổ biến dùng trong file truyền đi là Kermit, Xmodem, Ymodem, và Zmodem.

Khi một nút phát hiện lỗi hoặc nhận được thông điệp mà nó không hiểu, nó sẽ có gắng thông báo cho nút gửi dữ liệu để báo rằng nút này có thể truyền thử lại hoặc truyền dữ liệu khác để trả lời trong hoàn cảnh này. Sau một số lần cố gằng gửi, nếu nút truyền đủ biết để giữ nút để hiển thị một thông điệp lỗi, một âm thanh báo hiệu, hoặc làm điều gì đó để cho con người biết hoạt động của lỗi và sau đó tiếp tục với trách nhiệm tốt nhất mà nó có thể làm được.

Nút nhận nên biết phải làm gì với thông điệp ngắn hơn so với mong đợi. Dù đợi mãi cho thông điệp kế thúc, nó phải có sự kiện time out và để cho nơi điều hành biết là có một lỗi. Nơi điều hành có thể gửi lại sau đó hoặc là tiếp tục. Nếu không thì mạng có thể bị treo trong sự chờ đợi kết thúc.

Giới thiệu điều khiển ActiveX MSComm

MSComm trong VB dùng điều khiển truyền thông nối tiếp. Điều khiển này có trong bản VB Proessional và Enterprise editions, nhưng không có trong phiên bản Learning editions(giá rẻ nhất). Điều khiển này dễ dàng trong lập trình và hoạt động tốt hơn các dạng truy suất cổng khác. Nếu là bản đầy đủ các bạn dễ dàng tìm được điều khiển này trong Project-> Component(Ctr-T) Chọn Microsoft Comm Control 6.0

Các đặc tính của MSComm:

-Những tính chất của MSComm liên quan đến thiết lập cổng, truyền nhận dữ liệu, dùng tín hiệu bắt tay, hoặc đồng nhất các điều khiển. Các tính chất của MSComm được sắp xếp theo chức năng:

Thiết lập:

- CommID: trả lại handles đồng nhất tới thiết bị truyền thông có kiểu Long. Tính chất này không có lúc thiết kế mà chỉ có khi thi hành, thuộc tính này là ReadOnly.
- CommPort: dạng object.CommPort = value. Value là chỉ số của cổng Com có giá trị từ 1 -> 16 và mặc định có giá trị =1. Các bạn cần phải thiết lập thông số này trước khi mở cổng. Sẽ có lỗi error 68 (Device unavailable) nếu như không mở được cổng này.
- InBuferSize: thiết lập hoặc trả lại kích thước của bộ đệm nhận, tính =byte. Mặc định là 1024 byte. Các bạn không được nhầm lẫn với đặc tính InBufferCount là số byte đang chờ trong bộ đệm.
- InputLen : object.InputLen [= value] thiết lập hoặc trả lại số byte mỗi lần thuộc tính Input đọc được. Mặc định giá trị Value=0 tức là thuộc tính Input sẽ đọc hết nội dung của bộ đệm khi nó được dùng. Nếu số kí tự trong bộ đệm nhận không = InputLen thì thuộc tính Input sẽ trả lại kí tự rỗng "". Ví thế bạn cần phải chọn cách kiểm tra InBufferCount để chắc chắn số kí tự yêu cầu đã có đủ trước khi dùng lệnh .Input. Tính chất này rất là có ích khi đọc dữ liệu một máy mà dữ liệu ra được định dạng bằng các khối có chiều dài cố định.
- InputMode: object.InputMode [= value] . Value = 0 hay = comInputModeText dữ liệu nhận được dạng văn bản kiểu kí tự theo chuẩn ANSI. Dữ liệu nhận được sẽ là một sâu.
- Value=1 hay = comInputModeBinary dùng nhận mọi kiểu dữ liệu như kí tự điều khiển nhúng, kí tự NULL,.. Giá trị nhận được từ Input sẽ là một mảng kiểu Byte.
- NullDiscard: object.NullDiscard [= value] tính chất này quyết định kí tự trống có được truyền từ cổng đến bộ đệm nhận hay không. Nếu value= True kí tự này không được truyền. value = false kí tự trống sẽ được truyền. Kí tự trống được định nghĩa theo chuẩn ASCII là kí tự 0 – chr$(0).
- OutBuferSize: giống như InBuferSize, mặc định là 512.
- ParityReplace: thiết lập và trả lại kí tự thay thế kí tự không đúng trong lỗi giống nhau.
- PortOpen: thiết lập và trả lại tính trạng của cổng(đóng hoặc mở). object.PortOpen [= value] value = true cổng mở. =false cổng đóng và xóa toàn bộ dữ liệu trong bộ đệm nhận và truyền. Cần phải thiết lập thuộc tính CommPort đúng với tên của cổng trước khi mở cổng giao tiếp. Thêm vào đó cổng giao tiếp của thiết bị của bạn phải hỗ trợ giá trị trong thuộc tính Setting thì thiết bị của bạn mới hoạt động đúng, còn không thì nó sẽ hoạt động rất dở hơi nếu không nói là nó chạy không tốt. Đường DTR và RTS luôn giữ lại trạng thái của cổng.
- RthresHold: object.Rthreshold [= value] value kiểu số nguyên. Thiết lập số kí tự nhận được trước khi gây lên sự kiện comEvReceive. Mặc định =0 tức là không có sự kiện OnComm khi nhận được dữ liệu. Thiết lập = 1 tức là sự kiện OnComm xảy ra khi bất kì kí tự nào bị thay thế trong bộ đệm nhận.
- Settings: object.Settings [= value] thiết lập hoặc trả lại các thông số tần số baud, bít dữ liệu, bít chẵn lẻ, bít stop. Nếu Value không có giá trị khi mở sẽ gây ra lỗi 380 (Invalid property value).

  Value có dạng "BBBB,P,D,S". Trong đó, BBBB là tần số bus, P : thiết lập bít đồng bộ, D: số bít dữ liệu, S: số bít stop.
  Mặc định của nó là : "9600,N,8,1"

  Sau đây là một số tần số bus 110,300,600,1200,2400,4800,9600(mặc định), 1400,19200,28800,38400,56000,115200,128000,256000.
  Các giá trị của P: E(even), M: mark, N: none(mặc định), O: old, S: Space.
  D : có giá trị từ 4-> 8(mặc định).
  S: số bít stop có giá trị 1, 1.5, 2;
- SThreshold: thiết lập và và trả lại số kí tự nhỏ nhất được cho phép trong bộ đệm gửi để xảy ra sự kiện OnComm = comEvSend . Theo mặc định giá trị này = 0 tức là khi

truyền sẽ không gây ra sự kiện OnComm. Nếu thiết lập thông số này =1 thì sự kiện OnComm xảy ra khi bộ đệm truyền rỗng. Sự kiện OnComm = comEvSend chỉ xảy ra khi mà số kí tự trong bộ đệm truyền nhỏ hơn hoặc = Sthreshold. Nếu số kí tự trong bộ đệm này luôn lớn hơn Sthreshold thì sự kiện này không thể xảy ra.

Truyền nhận dữ liệu:
+ CommEvent: trả lại phần lớn sự kiện giao tiếp hoặc có lỗi. CommEvent xảy ra khi có lỗi hoặc khi xảy ra sự kiện nào đó. Sau đây là một số hằng số lỗi:
comEventBreak 1001 A Break signal was received.
comEventFrame 1004 Framing Error. The hardware detected a framing error.
comEventOverrun 1006 Port Overrun. A character was not read from the hardware before the next character arrived and was lost.
comEventRxOver 1008 Receive Buffer Overflow. There is no room in the receive buffer.
comEventRxParity 1009 Parity Error. The hardware detected a parity error.
comEventTxFull 1010 Transmit Buffer Full. The transmit buffer was full while trying to queue a character.
comEventDCB 1011 Unexpected error retrieving Device Control Block (DCB) for the port.
Một số sự kiện :
Constant Value Description
comEvSend 1 There are fewer than Sthreshold number of characters in the transmit buffer.
comEvReceive 2 Received Rthreshold number of characters. This event is generated continuously until you use the Input property to remove the data from the receive buffer.
comEvCTS 3 Change in Clear To Send line.
comEvDSR 4 Change in Data Set Ready line. This event is only fired when DSR changes from 1 to 0.
comEvCD 5 Change in Carrier Detect line.
comEvRing 6 Ring detected. Some UARTs (universal asynchronous receiver-transmitters) may not support this event.
comEvEOF 7 End Of File (ASCII character 26) character received.
+ EOFEnable: object.EOFEnable [= value] quyết định các hành động nếu MSComm tìm thấy kí tự kết thúc file. Nếu value=true khi tìm thấy kí tự kết thúc file thì sẽ gây lên sự kiện comEvEOF trong OnCommEvent. Nếu value= false thì sẽ không gây lên sự kiện này.
+ InBufferCout: trả lại số kí tự đang có trong bộ đệm nhận Bạn có thể xoá bộ đệm nhận bằng cách đặt thuộc tính này =0 . Không nhầm với thuộc tính InBufferSize là tổng kích thước của bộ đệm nhận.
+ Input: nhận và xoá dữ liệu trong bộ đệm nhận.Nếu InputMode là comInputModeText thì giá trị trả về sẽ là một xâu tức có kiểu String , dữ liệu dạng text trong một biến kiểu Variant. Nếu InputMode = comInputModeBinary thì thuộc tính này sẽ trả lại dữ liệu dạng nhị phân dưới dạng một mảng kiểu byte trong một biến Variant.
+ OutBufferCount: trả lại số kí tự trong bộ đệm truyền.
+ Output: ghi dữ liệu vào bộ đệm truyền. có thể truyền kiểu text hoặc kiểu nhị phân. Nếu truyền bằng kiểu text thì cho một biến Variant = kiểu String, nếu truyền kiểu nhị phân thì cho cho Output= variant = một mảng kiểu Byte.
Bắt tay(handshaking):
+ Break : thiết lập hoặc xoá tín hiệu. object.Break [= value] value = true hoặc false. Khi set value= true thì thông số Break này sẽ gửi một tín hiệu break. Tín hiệu break trì hoàn việc truyền dữ liệu và đưa đường truyền vào trạng thái break tới khi mà value = false.
+ CDHolding: quết định xem sự truyền này đến đâu bằng cách truy vấn đường

CD(Carrier Detect). Carrier Detect là tín hiệu gửi từ modem tới máy tính kết nối với nó thống báo rằng nó đang online. Nếu giá trị = true thì nó đường CD đang ở mức cao, nếu = false thì đường dây này đang ở mức thấp. Tính chất này không có trong lúc thiết kế chỉ có trong khi chạy chương trình.
Carrier Detect được biết như là Receive Line Signal Detect (RLSD).
+ CTSHolding: quết định khi nào bạn gửi dữ liệu bằng cách truy vấn trạng thái đường Clear To Send (CTS). Thông thường tín hiệu CTS được gửi từ modem tới máy tính kết nối với nó để báo rằng đang quá trình truyền dữ liệu. Thuộc tính Readonly chỉ xuất hiện khi chạy chương trình. Đường Clear To Send dùng trong RTS/CTS (Request To Send/Clear To Send) bắt tay phần cứng. CTSHolding cho bạn một cách để tự tay dò đường Clear To Send nếu bạn cần biết trạng thái của nó.
+ DSRHolding: biết trạng thái của đường Data Set Ready (DSR). Tín hiệu Data Set Ready truyền từ modem tới máy tính nối với nó để thông báo rằng modem đã sẵn sàng hoạt động. Tính chất này dùng khi viết Data Set Ready/Data Terminal Ready handshaking routine cho máy Data Terminal Equipment (DTE)- máy trang bị đầu cuối dữ liệu.
+ DTREnable: tính chất này quyết định khi nào cho phép đường Data Terminal Ready (DTR) trong truyền thông. Tín hiệu DTR gửi từ máy tính tới modem đẻ báo rằng máy tính sẵn sàng là nơi nhận dữ liệu. Khi DTREnable = true thì đường Data Terminal Ready set lên cao khi cổng mở, và thấp khi cổng đóng. Nếu DTREnable = false thì đường đó luôn mức thấp. Trong phần lớn trường hợp set đường Data Terminal Ready thành thấp để hang up telephone.
+ Handshaking: thiết lập và trả lại giao thức bắt tay phần cứng. object.Handshaking [= value].
Các giá trị của value:
Setting Value Description
comNone 0 (Default) No handshaking.
comXOnXOff 1 XON/XOFF handshaking.
comRTS 2 RTS/CTS (Request To Send/Clear To Send) handshaking.
comRTSXOnXOff 3 Both Request To Send and XON/XOFF handshaking.

Handshaking chỉ là giao thức truyền thông nội tại quyết định bởi dữ liệu nào được truyền từ cổng phần cứng tới bộ đệm nhận. Khi kí tự của dữ liệu tới cổng nối tiếp, thiết bị truyền thông sẽ chuyển nó vào trong bộ đệm nhận và chương trình của bạn có thể đọc chúng. Nếu không có bộ đệm dữ liệu hoặc chương trình của bạn cần đọc kí tự trực tiếp từ phần cứng , bạn có thể mất dữ liệu bởi vì kí tự từ phần cứng đến rất nhanh. Giao thức Handshaking đảm bảo dữ liệu không bị mất, khi dữ liệu đến cổng quá nhanh thì thiết bị truyền thông sẽ chuyển dữ liệu vào trong bộ đệm nhận.
+ RTSEnable: quết định khi nào cho phép đường Request To Send (RTS), Tín hiệu RTS từ máy tính tới modem để yêu cầu được tryền dữ liệu. Khi RTSEnable = true thì đường RTS mức cao khi cổng mở, tích mức thấp khi cổng đóng. Và hiển nhiên khi RTSEnable thì đường RTS luôn mức thấp.RTS dùng trong RTS/CTS hardware handshaking. RTSEnable cho phép bạn dò đường RTS khi cần biết tình trạng của đường này.

Các tính chất trên không có lúc thiết kế giao diện mà chỉ có lúc chạy chương trình (dùng trong viết code).

Ngoài ra còn có các thuộc tính khác như với các loại điều khiển khác:
Index: thiết lập và trả về một số xác định thứ tự nếu form bạn có nhiều điều khiển như thế này.
, Name: tên điều khiển, Object, Parent: trả về form hoặc đối tượng mà điều khiển này nằm trên đó,
Tag: thiết lập và trả về một biểu thức. Người dùng định nghĩa

Truyền dữ liệu kiểu text và nhị phân trong VB

Chào các bạn hôm nay tôi lại tiếp tục giới thiệu cho các bạn hiểu về cách truyền, nhận dữ liệu dạng text và binary trong VB.

Trước hết, các bạn cần biết biến kiểu Variant(tra trong từ điển có nghĩa là tương tự, gần giống nhau). Do vậy mà một biến Variant có thể gán = bất kì kiểu gì cũng được.Sau đây là chi tiết về 2 vấn đề chính:

VB cho phép bạn truyền dữ liệu dạng Text hay là dạng Binary. Thuộc tính InputMode quyết định điều khiển MSComm dùng dạng nào.

1.Kiểu văn bản(Text):

Với thuộc tính InputMode = comInputModeText thì MSComm sẽ gửi và nhận dữ liệu dạng xâu theo chuẩn ANSI (không phải chuẩn ASCII). Để gửi một xâu ra port, bạn cần phải cho thuộc tính Output của MSComm = 1 xâu. Ví dụ:

```
Code: Dim SampleText as String
        'ví dụ bạn muốn truyền một xâu "ABC"
        SampleText = "ABC"
        ' gửi kí tự này ra cổng
        MSComm1.Output = SampleText
```

MSComm gửi một mã ANSI 8 bít cho mỗi kí tự trong xâu mà bạn gửi. Để đọc một xâu từ cổng, cần đặt một xâu = thuộc tính Input của MSComm. Ví dụ bạn muốn đọc dữ liệu từ cổng và ghi vào một biến SampleText có kiểu String:

Code:

```
Dim SampleText as String
SampleText = MSComm1.Input ' khi đó SampleText sẽ là dữ liệu đọc được
MSComm lưu trữ mỗi mã ANSI 8 bít như một kí tự văn bản.
```

Thực tế như các bạn đã biết thì giá trị truyền cho MSComm1.Output phải là kiểu Variant. Ở đây thuộc tính Output chấp nhận kiểu một biến Variant chứa một xâu kí tự và MSComm sẽ đọc xâu kí tự và gán tự động vào một biến Variant vì Variant chính là kiểu của Output. Nói cách khác ở đây có sự chuyển kiểu ngầm định giữa kiểu String sang kiểu Variant.

Ngay ở bên trong bản thân chương trình VB lại lưu trữ xâu dưới dạng mã Unicode 16 bít nhưng sự chuyển đổi giữa kiểu Unicode và kiểu xâu kí tự ANSI 8 bít của MSComm diễn ra một cách tự động.

Sự chuyển kiểu của số ASCII Hex:

Số ASCII Hex là số hexa bình thường mà ta vẫn dùng như 0xA5(trong C,C++) hoặc 0A5h(trong ASM,..) đại diện cho số 165 trong hệ Decimal(165 = 16*10+ 5).

Với các ứng dụng dùng định dạng ASCII Hex, VB có một hàm chuyển đổi giữa kiểu xâu ASCII Hex và giá trị mà nó đại diện. Toán tử Hex$ chuyển đổi một số sang dạng kí tự ASCII Hex:

Ví dụ, để kiểm tra bạn có thể dùng hàm rất đơn giản xem nó in ra thế nào :

Code:

```
debug.print Hex$(165)
```

thì kết quả sẽ hiện trên một Dialog là : A5

Toán tử Val chuyển đổi từ kiểu ASCII Hex sang kiểu giá trị của xâu đó:

Ví dụ: ta thấy 0xA5 = 165 để thử xem có đúng không dùng lệnh;

Code:

```
  debug.print Val("&h" & "A5")
```

Kết quả là 165.

Xâu đầu tiên "&h" được thêm vào để báo cho VB biết để đối xử với giá trị đưa ra sau đó như là một số hexadecimal.'

2. Kiểu nhị phân(Binary Mode):

Để truyền dữ liệu dưới dạng nhị phân, cần thiết lập thuộc tính InputMode của MSComm thành comInputModeBinary.

VB cung cấp một kiểu dữ liệu kiểu Byte để lưu trữ dữ liệu nhị phân. Các byte được ghi và đọc từ cổng nối tiếp được lưu trữ trong một biến Variant(nội dung của nó chứa một mảng mà các phần tử của mảng có kiểu Byte). Thậm chí nếu chỉ đọc, ghi duy nhất có 1 byte thì dữ liệu này cũng phải đặt trong một mảng byte, chứ không được lưu trữ trong một biến kiểu byte thông thường. Để ghi một mảng kiểu byte ra cổng nối tiếp gồm 2 bước.

+ Bước đầu: lưu trữ mảng kiểu byte vào một biến variant

+ Bước 2: gửi dữ liệu đi bằng cách thiết lập thông số Output của MSComm bằng biến Variant đó.

Code:

```
Dim BytesToSend(0 to 1) as Byte ' khai báo một mảng 2 phần tử
        Dim Buffer as Variant
        ' lưu trữ dữ liệu vào mảng kiểu byte ở trên
        BytesToSend(0) = &H4A
        BytesToSend(1) = &H23
        ' cho vào một biến Variant
        Buffer = BytesToSend()
        'ghi vào cổng nối tiếp
        MSComm1.Output = Buffer
```

Để đọc các byte tại cổng nối tiếp, bạn cũng làm tương tự như trên, đọc vào một biến Variant sau đó cho một mảng = biến đó.

Code:

```
        Dim BytesReceived()  as Byte ' khai báo một mảng động
        Dim Buffer as Variant ' khai báo biến variant
        'đọc dữ liệu từ cổng nối tiếp
        Buffer = MSComm1.Input
        ' ghi dữ liệu đọc được vào mảng động
        BytesReceived() = Buffer
```

Các ban lưu ý là phải khai báo một mảng byte động. Có 2 cách để chuyển đổi giữ mảng bytes và kiểu Variant. Bạn có thể cho một biến = một biến có số chiều đã được biết và VB làm công việc chuyển đổi này tự động:

Code:

```
Dim DimensionedArray(15) as Byte ' mảng đã khai báo số chiều =15
Dim DynamicBytesArray()  as Byte
Dim Buffer As Variant
Dim Count As Integer
' lưu trữ một mảng mảng vào một biến variant. Mảng này đã được biết số phần
tử
Buffer = DimensionedArray()
```

```
'để sao chép nội dung của một biến variant vào một mảng thì mảng này phải
khai báo là một mảng động( chưa biết số phần tử)
DynamicBytesArray() = Buffer
```

Đối với VB 6.0 bạn hoàn toàn có thể gán 2 mảng với nhau vì nó sẽ tự sao chép nội dung từ mảng nguồn sang mảng đích mà không cần phải làm bằng cách sao chép từng phần tử của 2 mảng cho nhau(như trong C thì bạn phải làm điều này rồi vì gán 2 tên mảng thực chất là bạn chỉ là cho con trỏ mảng đích trỏ vào địa chỉ của phần tử của mảng nguồn thôi, đây là sự sao chép bề mặt). Tuy nhiên bạn vẫn có thể làm điều này trong VB:

Code:

```
'lưu trữ một mảng kiểu byte trong một biến variant
Buffer = CVar(DynamicByteArray())
' CVar   -> Convert to Variant  Chuyển thành kiểu variant


'lưu nội dung của biến variant này trong một mảng kiểu byte
For Count = 0 to (LenB(Buffer)-1)
        DimmensionedArray(Count) = CByte(Buffer(count))
Next Count
' CByte -> Convert to Byte chuyển kiểu thành kiểu Byte
```

Gửi nhận dữ liệu bằng phương pháp dò

Tiếp theo tôi xin giới thiệu cho các bạn phương pháp lấy dữ liệu bằng phương pháp thăm dò(polling).

Giao tiếp tại cổng bằng phương pháp dò tức là bạn chỉ đọc hoặc ghi ra cổng khi nào cần bằng cách dùng thuộc tính Input hoặc Output của MSComm.

1. Gửi dữ liệu:

Thuộc tính Output dùng để ghi dữ liệu ra cổng. Biến dùng ở bên phải cú pháp là một biến kiểu Variant. Đây là cú pháp để ghi dữ liệu:

Code:

```
        Dim DataToWrite As Variant
        MSComm1.Output = DataToWrite
```

Khi gửi một khối nhỏ dữ liệu, cần phải thiết lập thuộc tính OutBuferSize phải lớn hơn hoặc bằng số lượng lớn nhất các byte mà các bạn cần chuyển trong một lần.

Đối với việc truyền dữ liệu có tính lâu dài về thời gian dùng OutBufferCount để chắc chắn rằng bộ đệm không bị tràn. Khi bạn có nhiều dữ liệu cần gửi để tránh cho tràn bộ đệm , bạn nên đọc giá trị của OutBufferCount và so sánh với giá trị của OutBufferCount để kiểm tra xem bộ đệm còn bao nhiêu sau khi gửi dữ liệu đầu tiên. Sau đó làm đầy bộ đệm bằng cách ghi bằng đó các byte hoặc nhở hơn dữ liệu vào bộ đệm thì bộ đệm sẽ không bị tràn. Hoặc bạn có thể gửi dữ liệu đã đóng gói với kích thước xác định và chỉ gửi các gói này được OutBufferCount chỉ rằng có đủ chỗ trống trong bộ đệm cho gói dữ liệu này. Ví dụ, OutBufferSize = 1024 và kích thước 1 gói là 512, bạn chỉ có thể gửi được gói dữ liệu này khi mà OutBuferCount <= 512.

2. Nhận dữ liệu:

Để đọc dữ liệu được truyền đến, ứng dụng đọc dữ liệu từ InBuferCount theo từng chu kì. Khi bộ đếm chỉ rằng một số các kí tự mà ứng dụng cần đã đến(như muốn lấy 5 byte chẳng hạn) thì ứn dụng sẽ đọc dữ liệu với thuộc tính Input của MSComm:

Code:

```
Dim BytesToRead As Integer
```

```
Dim DataIn As Variant

'thiết lập số byte cần đọc
NumberOfBytesToRead = 512
MSComm1.InputLen = NumberOfBytesToRead

' dò bộ đệm nhận đến khi bộ đệm có đầy đủ số byte cần đọc
Do
        DoEvents
Loop Until MSComm1.InBufferCount> NumberOfBytesToRead
' khi tổng số byte đã tới thì đọc lưu vào DataIn
DataIn = MSComm1.Input
```

Thuộc tính InBuferSize phải đủ độ rộng để cho lượng lớn nhất dữ liệu có thể tới mà không bị mất trước khi MSComm có thể đọc chúng. Nếu dữ liệu đến bằng các block với kích thước cố định thì cần thiết lập thuộc tính InBufferSize bằng bội số của kích thước 1 block.

Nếu tổng dữ liệu đến không biết kích thước thế nào, ứng dụng nên đọc bộ đệm nhận ngay khi bộ đệm chỉ nhận được 1 byte để tránh việc không kiểm soát đượcbộ đệm gây ra tràn dữ liệu. Chờ đợi nhiều byte để đọc là một việc làm không có hiệu quả bởi vì không có cách nào biết được byte nào sẽ đến cuối cùng. Nếu chờ nhiều hơn 1 byte rồi mới đọc, chương trình nên bao gồm có một "time out" chính là tổng thời gian từ lúc chờ mà tổng số byte vẫn không đến(như bạn chờ 6 byte mà mãi không đến chẳng lẽ ứng dụng chờ mấy giờ à, thế thì bạn cần phải qui định là sao bao nhiêu lâu thì đọc thôi chẳng cần chờ nữa).

Bạn có thể kết hợp phương pháp lập trình theo thủ tục và theo sự kiện bằng cách sử dụng timer để biết khi nào thì đọc cổng. Ví dụ, dùng một sự kiện Timer gây ra ở cổng đeer đọc cổng một lần / một giây.

Trên đây là cách đọc, ghi dữ liệu bằng phương pháp dò. Ngày mai tôi xin giới thiệu cho các bạn cách dùng ngắt(Interrupt), tức là dùng sự kiện OnComm của MSComm.

12.2.1. Giao tiep COM_LCD

Ngoài ra, để dùng giao tiếp rs232 rất đơn giản:

#uses rs232
printf (xuất)
getc(nhập)

Trong CCs tiếng việt trên , tôi có trình bày cách set po rt , VD po rt B

1/ thiết lập cơ chế truy xuất nhanh po rtB :

khai báo #U SE FAST_IO(po rtB) khởi tạo 1 biến danh định chỉ tới địa chỉ PO rt B :Vd po rtB ở địa chỉ 0x3F thì set:

#LOCATE portB = 0x3F

portB chỉ là tên , có thể dùng tên khác tùy ý.

2/ sau đó trong chương trình , set chân vào ra :

VD : set_tris_B(0x11110000b) ; // B0->B3 : ngõ ra , B4->B7 : ngõ vào
VD : set_tris_B(255); // tất cả là ngõ vào
VD : set_tris_B(0) ; // tất cả là ngõ ra

khi đó chỉ cần:

Vd : tất cả là ngõ vào thì có thể khiểm tra 1 cái gì đó :

if (portB==0x0011b) { . . .} // nếu B0 và B1 có tín hiệu vào thì . . .

nếu set tất cả là ngõ ra thì :

Vd : portB = 127 ; // =0x01111111 : xuất tín hiệu ra trên B0->B6 ,B7 không có .

+

Chẳng lẽ để giao tiếp rs232 chỉ cần có 2 lệnh đó thôi sao? *thế còn việc kiểm tra xem lúc nào*

bộ đệm tràn? như ở bên máy tính em làm cả khâu kiểm tra này, còn ở PIC thì sao? xử lý khi báo tràn như thế nào? EM có đọc datasheet của PIC 16F877 nhưng chưa thông lắm! Mong bác chỉ giáo. Ngoài ra em còn muốn xử lý data xuất/nhập này thì làm như thế nào? có phải ghi từ thanh ghi đệm vào bộ nhớ PIC rồi lấy data từ đó xử lý? Ví dụ như là em muốn cho nó xuất led chẳng hạn!
Mong bác thông cảm, có lẽ em hỏi hơi bị ngây ngô! Em mới chuyển từ 8051 sang PIC mấy tuần nay! CHưa biết gì cả!

Tôi thấy như vậy đã là đủ rồi, kho các hàm của CCS rất nhiệu Về RS232 cho pic thì bạn xem lại trong datasheet, PIC chi có 8 bit cho truyền và 8 bit cho nhân. Tài liệu đầy đủ nhất là help cua nó, kết hợp các hàm lại với nhau sẽ giúp bạn giải quyết nhiều bài toán. Còn về lập trình giao tiếp RS232 tôi đã có một bài mẫu. Chương trình nhận ký tự từ bàn phím và hiên thị ra LCD, rồi xuất trả lại máy tính ký tự đó.
Code:

```
#include <16f877a.h>
#include <def_877a.h>
#use delay(clock=20000000)
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD,
NOWRT
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=9)
#include <lcd_lib_4bit.c>
//#include <input.c>
int8 count=0;
char string_in[16];

#INT_RDA
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();
```

Thang8831
http://www.picvietnam.com

```
lcd_putcmd(0x01);
lcd_putcmd(line_1);
printf("Enter a String.");
printf("Or anything you want!");
while (1) {}
}
```

12.2.2. USART-RS232

```
//PROGRAMM FOR rs232 communication
//
// PROCESSOR : PIC16F877A
// CLOCK        : 20MHz, EXTERNAL
// SPEED     : 9600 bps(1bit=104us)
#include         <pic.h>
__CONFIG(WDTDIS & LVPDIS & BORDIS & HS & PWRTEN) ;
unsigned char ch;
void InitUsart(void) {
        // TX Pin - output
        TRISC6 = 0;

        // RX Pin - input
        TRISC7 = 1;

        // RX Setting, 8bit, enable receive,
        RCSTA = 0x90;

        // TX Setting, 8bit, Asinchronius mode, High speed
        TXSTA = 0x24;

        // Set Baudrade - 9600 (from datasheet baudrade table)
        SPBRG = 129;
}
void  WriteByte(unsigned char byte) {
        // wait until register is empty
        while(!TXIF);
        // transmite byte
        TXREG = byte;
}
unsigned char ReadByte(void) {
        // wait to receive character
        while(!RCIF);
        // return received character
        return RCREG;
}
// main function
void main( void ) {
        // Init Interface
        InitUsart();
        // loop forever - echo
        while(1) {
                ch = ReadByte();
```

```
                WriteByte(ch);
                WriteByte('*');
        }
}
```

12.2.3. RS232TUT.H

```
// RS232TUT.h : main header file for the RS232TUT application
//
#if !
defined(AFX_RS232TUT_H__B483DFBF_2BC7_4F5C_A17A_182A0133B7B9__INCLUD
ED_)
#define
AFX_RS232TUT_H__B483DFBF_2BC7_4F5C_A17A_182A0133B7B9__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h"           // main symbols
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTApp:
// See RS232TUT.cpp for the implementation of this class
//

class CRS232TUTApp : public CWinApp
{
public:
        CRS232TUTApp();
// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CRS232TUTApp)
        public:
        virtual BOOL InitInstance();
        //}}AFX_VIRTUAL
// Implementation
        //{{AFX_MSG(CRS232TUTApp)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.
#endif // !
defined(AFX_RS232TUT_H__B483DFBF_2BC7_4F5C_A17A_182A0133B7B9__INCLUD
ED_)
```

12.2.4. RS232TUT

```
// RS232TUT.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "RS232TUT.h"
#include "RS232TUTDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTApp
BEGIN_MESSAGE_MAP(CRS232TUTApp, CWinApp)
	//{{AFX_MSG_MAP(CRS232TUTApp)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTApp construction

CRS232TUTApp::CRS232TUTApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}
/////////////////////////////////////////////////////////////////////////////
// The one and only CRS232TUTApp object
CRS232TUTApp theApp;
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTApp initialization
BOOL CRS232TUTApp::InitInstance()
{
	AfxEnableControlContainer();
	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.
#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif
	CRS232TUTDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
```

```
		// TODO: Place code here to handle when the dialog is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with Cancel
	}
	// Since the dialog has been closed, return FALSE so that we exit the
	//  application, rather than start the application's message pump.
	return FALSE;
}
```

12.2.5. RS232TUTDlg

```
// RS232TUTDlg.h : header file
//
//{{AFX_INCLUDES()
#include "mscomm.h"
//}}AFX_INCLUDES
#if !
defined(AFX_RS232TUTDLG_H__4C1AE02B_9689_41FF_8F79_A9F74E508A84__INCL
UDED_)
#define
AFX_RS232TUTDLG_H__4C1AE02B_9689_41FF_8F79_A9F74E508A84__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTDlg dialog
class CRS232TUTDlg : public CDialog
{
// Construction
public:
	CString getCurStrInCombobox(const CComboBox & a);
	void InitComboBox();
	void Settings();
	CRS232TUTDlg(CWnd* pParent = NULL);// standard constructor
// Dialog Data
	//{{AFX_DATA(CRS232TUTDlg)
	enum { IDD = IDD_RS232TUT_DIALOG };
	CComboBox   m_cboStopBit;
	CComboBox   m_cboParityBit;
	CComboBox   m_cboHandshaking;
	CComboBox   m_cboDataBit;
	CComboBox   m_cboComPort;
	CComboBox   m_cboBitRate;
	CMSComm     m_mscomm;
	CString     m_strReceive;
	CString     m_strTransfer;
	//}}AFX_DATA
```

```
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CRS232TUTDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV support
        //}}AFX_VIRTUAL
// Implementation
protected:
        HICON m_hIcon;

        // Generated message map functions
        //{{AFX_MSG(CRS232TUTDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        afx_msg void OnButtonExit();
        afx_msg void OnEditchangeComboComport();
        afx_msg void OnButtonSend();
        afx_msg void OnOnCommMscomm1();
        afx_msg void OnButtonClear();
        afx_msg void OnButtonConnect();
        DECLARE_EVENTSINK_MAP()
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !
defined(AFX_RS232TUTDLG_H__4C1AE02B_9689_41FF_8F79_A9F74E508A84__INCL
UDED_)
```

12.2.6. RS232TUTDlg.CPP

```
// RS232TUTDlg.cpp : implementation file
//
#include "stdafx.h"
#include "RS232TUT.h"
#include "RS232TUTDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();
```

```
// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTDlg dialog
CRS232TUTDlg::CRS232TUTDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CRS232TUTDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CRS232TUTDlg)
        m_strReceive = _T("");
        m_strTransfer = _T("");
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
void CRS232TUTDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CRS232TUTDlg)
        DDX_Control(pDX, IDC_COMBO_STOPBIT, m_cboStopBit);
```

```
	DDX_Control(pDX, IDC_COMBO_PARITYBIT, m_cboParityBit);
	DDX_Control(pDX, IDC_COMBO_HANDSHAKING, m_cboHandshaking);
	DDX_Control(pDX, IDC_COMBO_DATABIT, m_cboDataBit);
	DDX_Control(pDX, IDC_COMBO_COMPORT, m_cboComPort);
	DDX_Control(pDX, IDC_COMBO_BITRATE, m_cboBitRate);
	DDX_Control(pDX, IDC_MSCOMM1, m_mscomm);
	DDX_Text(pDX, IDC_EDIT_RECEIVE, m_strReceive);
	DDX_Text(pDX, IDC_EDIT_TRANSFER, m_strTransfer);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CRS232TUTDlg, CDialog)
	//{{AFX_MSG_MAP(CRS232TUTDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON_EXIT, OnButtonExit)
	ON_CBN_EDITCHANGE(IDC_COMBO_COMPORT,
OnEditchangeComboComport)
	ON_BN_CLICKED(IDC_BUTTON_SEND, OnButtonSend)
	ON_BN_CLICKED(IDC_BUTTON_CLEAR, OnButtonClear)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CRS232TUTDlg message handlers

BOOL CRS232TUTDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}
	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
```

```
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	InitComboBox();

	return TRUE;  // return TRUE  unless you set the focus to a control
}
void CRS232TUTDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
void CRS232TUTDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}
// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
```

```
HCURSOR CRS232TUTDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}
void CRS232TUTDlg::Settings()
{
		// if port is already opened then close port.
		if( m_mscomm.GetPortOpen())
				m_mscomm.SetPortOpen(false);
		// Setting comport
		m_mscomm.SetCommPort(m_cboComPort.GetCurSel()+ 1);
		// Setting Handshaking
		m_mscomm.SetHandshaking(m_cboHandshaking.GetCurSel());
		//
		CString strBitRate = getCurStrInCombobox(m_cboBitRate);
		CString strParity =  getCurStrInCombobox(m_cboParityBit);
		CString strDataBit     =  getCurStrInCombobox(m_cboDataBit);
		CString strStopBit     =   getCurStrInCombobox(m_cboStopBit);

		CString strSetting;
	strSetting.Format("%s,%c,%s,%s",strBitRate,strParity[1],strDataBit,strStopBit);
		m_mscomm.SetSettings(strSetting);//"9600,N,8,1");
		m_mscomm.SetRThreshold(1);  //
		// set for input direction
		m_mscomm.SetInputLen(1);
		m_mscomm.SetInBufferSize(1024);
		m_mscomm.SetInputMode(0);
		m_mscomm.SetOutBufferSize(1024);
		m_mscomm.SetPortOpen(true);
}
void CRS232TUTDlg::InitComboBox()
{
		// ComboBox ComPort
		m_cboComPort.ResetContent();
		m_cboComPort.AddString("COM1");
		m_cboComPort.AddString("COM2");
		m_cboComPort.SetCurSel(0);
		//m_cboComPort
		// ComboBox BitRate
		m_cboBitRate.ResetContent();
		m_cboBitRate.InsertString(0,"600");
		m_cboBitRate.InsertString(1,"1200");
		m_cboBitRate.InsertString(2,"2400");
		m_cboBitRate.InsertString(3,"4800");
		m_cboBitRate.InsertString(4,"9600");
		m_cboBitRate.InsertString(5,"14400");
		m_cboBitRate.InsertString(6,"19200");
		m_cboBitRate.InsertString(7,"28800");
		m_cboBitRate.InsertString(8,"38400");
		m_cboBitRate.InsertString(9,"56000");
```

```
		m_cboBitRate.InsertString(10,"57600");
		m_cboBitRate.InsertString(11,"115200");
		m_cboBitRate.InsertString(12,"128000");
		m_cboBitRate.InsertString(13,"256000");

		m_cboBitRate.SetCurSel(4);

		// ComboBox Data Bit
		m_cboDataBit.ResetContent();
		m_cboDataBit.AddString("5");
		m_cboDataBit.AddString("6");
		m_cboDataBit.AddString("7");
		m_cboDataBit.AddString("8");

		m_cboDataBit.SetCurSel(3);

		// ComboBox Stop Bit
		m_cboStopBit.ResetContent();
		m_cboStopBit.AddString("1");
		m_cboStopBit.AddString("1.5");
		m_cboStopBit.AddString("2");

		m_cboStopBit.SetCurSel(0);

		// ComboBox Parity Bit
		m_cboParityBit.ResetContent();
		m_cboParityBit.InsertString(0,"None");
		m_cboParityBit.InsertString(1,"Old");
		m_cboParityBit.InsertString(2,"Even");
		m_cboParityBit.InsertString(3,"Mark");
		m_cboParityBit.InsertString(4,"Space");

		m_cboParityBit.SetCurSel(0);

		// ComboBox Handshaking
		m_cboHandshaking.ResetContent();
		m_cboHandshaking.InsertString(0,"None");
		m_cboHandshaking.InsertString(1,"XON/XOFF");
		m_cboHandshaking.InsertString(2,"RTS");
		m_cboHandshaking.InsertString(3,"RTS/CTS + XON/XOFF");

		m_cboHandshaking.SetCurSel(0);
}
void CRS232TUTDlg::OnButtonExit()
{
	// TODO: Add your control notification handler code here
	if	( m_mscomm.GetPortOpen() )
			m_mscomm.SetPortOpen(false);
	OnOK();
}
```

```
void CRS232TUTDlg::OnEditchangeComboComport()
{
	// TODO: Add your control notification handler code here
}
void CRS232TUTDlg::OnButtonSend()
{
	// TODO: Add your control notification handler code here
	UpdateData();
	Settings();
	// send data
	m_mscomm.SetOutput((COleVariant)m_strTransfer);
	UpdateData(false);
}
BEGIN_EVENTSINK_MAP(CRS232TUTDlg, CDialog)
   //{{AFX_EVENTSINK_MAP(CRS232TUTDlg)
	ON_EVENT(CRS232TUTDlg, IDC_MSCOMM1, 1 /* OnComm */,
OnOnCommMscomm1, VTS_NONE)
	//}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

void CRS232TUTDlg::OnOnCommMscomm1()
{
	// TODO: Add your control notification handler code here

	UpdateData();
	switch( m_mscomm.GetCommEvent()){
	case 1: // comEvSend
		break;
	case 2://  comEvReceive
		VARIANT  data = m_mscomm.GetInput();
		m_strReceive += (CString)data.bstrVal;
		break;
	case 3:// comEvCTS
		break;
	case 4://comEvDSR
		break;
	case 5: //comEvCD
		break;
	case 6://comEvRing
		break;
	case 7: //comEvEOF
		break;
	};
	UpdateData(false);
}
void CRS232TUTDlg::OnButtonClear()
{
	// TODO: Add your control notification handler code here
	UpdateData();
	m_strReceive = "";
```

```
	UpdateData(false);
}
CString CRS232TUTDlg::getCurStrInCombobox(const CComboBox &a)
{
		CString str;
		a.GetLBText(a.GetCurSel(),str);
		return str;
}
```

12.2.7. StdAfx.H

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//
#if !
defined(AFX_STDAFX_H__A63EF4CE_6555_4F4F_B8F9_88951D96BF49__INCLUDED
_)
#define
AFX_STDAFX_H__A63EF4CE_6555_4F4F_B8F9_88951D96BF49__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN		// Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>		// MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>			// MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !
defined(AFX_STDAFX_H__A63EF4CE_6555_4F4F_B8F9_88951D96BF49__INCLUDED
_)
```

12.2.8. mscomm.H

```
#if !
defined(AFX_MSCOMM_H__E6B47B70_15D5_4522_B55C_51522629ECEA__INCLUDE
D_)
#define
AFX_MSCOMM_H__E6B47B70_15D5_4522_B55C_51522629ECEA__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++

// NOTE: Do not modify the contents of this file.  If this class is regenerated by
//  Microsoft Visual C++, your modifications will be overwritten.
/////////////////////////////////////////////////////////////////////////////
// CMSComm wrapper class

class CMSComm : public CWnd
{
protected:
        DECLARE_DYNCREATE(CMSComm)
public:
        CLSID const& GetClsid()
        {
                static CLSID const clsid
                        = { 0x648a5600, 0x2c6e, 0x101b, { 0x82, 0xb6, 0x0, 0x0, 0x0, 0x0,
0x0, 0x14 } };
                return clsid;
        }
        virtual BOOL Create(LPCTSTR lpszClassName,
                LPCTSTR lpszWindowName, DWORD dwStyle,
                const RECT& rect,
                CWnd* pParentWnd, UINT nID,
                CCreateContext* pContext = NULL)
        { return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd,
nID); }

   BOOL Create(LPCTSTR lpszWindowName, DWORD dwStyle,
                const RECT& rect, CWnd* pParentWnd, UINT nID,
                CFile* pPersist = NULL, BOOL bStorage = FALSE,
                BSTR bstrLicKey = NULL)
        { return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd,
nID,
                pPersist, bStorage, bstrLicKey); }
// Attributes
public:

// Operations
public:
        void SetCDHolding(BOOL bNewValue);
        BOOL GetCDHolding();
        void SetCommID(long nNewValue);
        long GetCommID();
        void SetCommPort(short nNewValue);
        short GetCommPort();
        void SetCTSHolding(BOOL bNewValue);
        BOOL GetCTSHolding();
        void SetDSRHolding(BOOL bNewValue);
        BOOL GetDSRHolding();
        void SetDTREnable(BOOL bNewValue);
```

```
        BOOL GetDTREnable();
        void SetHandshaking(long nNewValue);
        long GetHandshaking();
        void SetInBufferSize(short nNewValue);
        short GetInBufferSize();
        void SetInBufferCount(short nNewValue);
        short GetInBufferCount();
        void SetBreak(BOOL bNewValue);
        BOOL GetBreak();
        void SetInputLen(short nNewValue);
        short GetInputLen();
        void SetNullDiscard(BOOL bNewValue);
        BOOL GetNullDiscard();
        void SetOutBufferSize(short nNewValue);
        short GetOutBufferSize();
        void SetOutBufferCount(short nNewValue);
        short GetOutBufferCount();
        void SetParityReplace(LPCTSTR lpszNewValue);
        CString GetParityReplace();
        void SetPortOpen(BOOL bNewValue);
        BOOL GetPortOpen();
        void SetRThreshold(short nNewValue);
        short GetRThreshold();
        void SetRTSEnable(BOOL bNewValue);
        BOOL GetRTSEnable();
        void SetSettings(LPCTSTR lpszNewValue);
        CString GetSettings();
        void SetSThreshold(short nNewValue);
        short GetSThreshold();
        void SetOutput(const VARIANT& newValue);
        VARIANT GetOutput();
        void SetInput(const VARIANT& newValue);
        VARIANT GetInput();
        void SetCommEvent(short nNewValue);
        short GetCommEvent();
        void SetEOFEnable(BOOL bNewValue);
        BOOL GetEOFEnable();
        void SetInputMode(long nNewValue);
        long GetInputMode();
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !
defined(AFX_MSCOMM_H__E6B47B70_15D5_4522_B55C_51522629ECEA__INCLUDE
D_)
```

12.2.9. mscomm.CPP

```
// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++
// NOTE: Do not modify the contents of this file.  If this class is regenerated by
```

```
//  Microsoft Visual C++, your modifications will be overwritten.

#include "stdafx.h"
#include "mscomm.h"
/////////////////////////////////////////////////////////////////////////////
// CMSComm
IMPLEMENT_DYNCREATE(CMSComm, CWnd)
/////////////////////////////////////////////////////////////////////////////
// CMSComm properties
/////////////////////////////////////////////////////////////////////////////
// CMSComm operations
void CMSComm::SetCDHolding(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x1, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                 bNewValue);
}
BOOL CMSComm::GetCDHolding()
{
        BOOL result;
        InvokeHelper(0x1, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetCommID(long nNewValue)
{
        static BYTE parms[] =
                VTS_I4;
        InvokeHelper(0x3, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                 nNewValue);
}
long CMSComm::GetCommID()
{
        long result;
        InvokeHelper(0x3, DISPATCH_PROPERTYGET, VT_I4, (void*)&result, NULL);
        return result;
}
void CMSComm::SetCommPort(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x4, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                 nNewValue);
}
short CMSComm::GetCommPort()
{
        short result;
        InvokeHelper(0x4, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
```

```
}
void CMSComm::SetCTSHolding(BOOL bNewValue)
{
	static BYTE parms[] =
		VTS_BOOL;
	InvokeHelper(0x5, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		bNewValue);
}
BOOL CMSComm::GetCTSHolding()
{
	BOOL result;
	InvokeHelper(0x5, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
	return result;
}
void CMSComm::SetDSRHolding(BOOL bNewValue)
{
	static BYTE parms[] =
		VTS_BOOL;
	InvokeHelper(0x7, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		bNewValue);
}
BOOL CMSComm::GetDSRHolding()
{
	BOOL result;
	InvokeHelper(0x7, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
	return result;
}
void CMSComm::SetDTREnable(BOOL bNewValue)
{
	static BYTE parms[] =
		VTS_BOOL;
	InvokeHelper(0x9, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		bNewValue);
}
BOOL CMSComm::GetDTREnable()
{
	BOOL result;
	InvokeHelper(0x9, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
	return result;
}
void CMSComm::SetHandshaking(long nNewValue)
{
	static BYTE parms[] =
		VTS_I4;
	InvokeHelper(0xa, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		nNewValue);
}
```

```
long CMSComm::GetHandshaking()
{
        long result;
        InvokeHelper(0xa, DISPATCH_PROPERTYGET, VT_I4, (void*)&result, NULL);
        return result;
}
void CMSComm::SetInBufferSize(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0xb, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                 nNewValue);
}
short CMSComm::GetInBufferSize()
{
        short result;
        InvokeHelper(0xb, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetInBufferCount(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0xc, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                 nNewValue);
}
short CMSComm::GetInBufferCount()
{
        short result;
        InvokeHelper(0xc, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetBreak(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0xd, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                 bNewValue);
}
BOOL CMSComm::GetBreak()
{
        BOOL result;
        InvokeHelper(0xd, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetInputLen(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
```

```
        InvokeHelper(0xe, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
short CMSComm::GetInputLen()
{
        short result;
        InvokeHelper(0xe, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetNullDiscard(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x10, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                bNewValue);
}
BOOL CMSComm::GetNullDiscard()
{
        BOOL result;
        InvokeHelper(0x10, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetOutBufferSize(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x11, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
short CMSComm::GetOutBufferSize()
{
        short result;
        InvokeHelper(0x11, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetOutBufferCount(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x12, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
short CMSComm::GetOutBufferCount()
{
        short result;
        InvokeHelper(0x12, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetParityReplace(LPCTSTR lpszNewValue)
```

```
{
        static BYTE parms[] =
                VTS_BSTR;
        InvokeHelper(0x13, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                lpszNewValue);
}
CString CMSComm::GetParityReplace()
{
        CString result;
        InvokeHelper(0x13, DISPATCH_PROPERTYGET, VT_BSTR, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetPortOpen(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x14, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                bNewValue);
}
BOOL CMSComm::GetPortOpen()
{
        BOOL result;
        InvokeHelper(0x14, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetRThreshold(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x15, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
short CMSComm::GetRThreshold()
{
        short result;
        InvokeHelper(0x15, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetRTSEnable(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x16, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                bNewValue);
}
BOOL CMSComm::GetRTSEnable()
{
        BOOL result;
```

```
        InvokeHelper(0x16, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetSettings(LPCTSTR lpszNewValue)
{
        static BYTE parms[] =
                VTS_BSTR;
        InvokeHelper(0x17, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                lpszNewValue);
}
CString CMSComm::GetSettings()
{
        CString result;
        InvokeHelper(0x17, DISPATCH_PROPERTYGET, VT_BSTR, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetSThreshold(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x18, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
short CMSComm::GetSThreshold()
{
        short result;
        InvokeHelper(0x18, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetOutput(const VARIANT& newValue)
{
        static BYTE parms[] =
                VTS_VARIANT;
        InvokeHelper(0x19, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                &newValue);
}
VARIANT CMSComm::GetOutput()
{
        VARIANT result;
        InvokeHelper(0x19, DISPATCH_PROPERTYGET, VT_VARIANT, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetInput(const VARIANT& newValue)
{
        static BYTE parms[] =
                VTS_VARIANT;
        InvokeHelper(0x1a, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
```

```
                &newValue);
}
VARIANT CMSComm::GetInput()
{
        VARIANT result;
        InvokeHelper(0x1a, DISPATCH_PROPERTYGET, VT_VARIANT, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetCommEvent(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x1b, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
short CMSComm::GetCommEvent()
{
        short result;
        InvokeHelper(0x1b, DISPATCH_PROPERTYGET, VT_I2, (void*)&result, NULL);
        return result;
}
void CMSComm::SetEOFEnable(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x1c, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                bNewValue);
}
BOOL CMSComm::GetEOFEnable()
{
        BOOL result;
        InvokeHelper(0x1c, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}
void CMSComm::SetInputMode(long nNewValue)
{
        static BYTE parms[] =
                VTS_I4;
        InvokeHelper(0x1d, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}
long CMSComm::GetInputMode()
{
        long result;
        InvokeHelper(0x1d, DISPATCH_PROPERTYGET, VT_I4, (void*)&result, NULL);
        return result;
}
```

12.2.10. Giao tiep pc va pic6f877 qua cong rs232

Thang8831
http://www.picvietnam.com

chào các bạn mình đang viết chương trình giao tiếp giữa pic6f877 và pc
chương trình trên pic viết bằng ccsc lệnh :

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#device 16F877*=16 ADC=8
#use delay(clock=1000000)
#use rs232(baud=4800, xmit=PIN_C6, rcv=PIN_C7, PARITY=N,BITS =7,STOP=2)
#include <input.c>
#include <STDLIB.h>
void main()
{
//int status;
char value;
lcd_init();
lcd_putc("begin");

value=getc();
putc(value);}
```

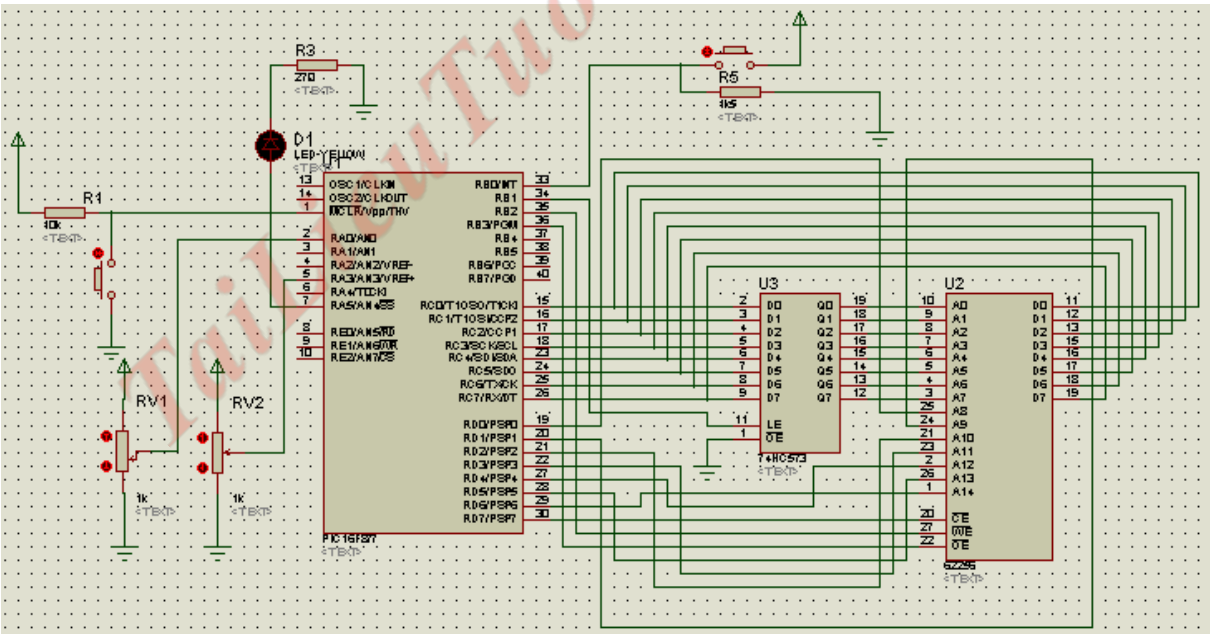
trên pc dùng chương trình giao tiếp viết bằng matlab(trong diễn đàn)
nhưng sao mình kô thấy nó nhận dc gì cả
có ai làm cái này rồi thì có thể giúp mình dc kô
Ban tham khao nhe

```
#include <16F876A.h>
#device adc=8
#use delay(clock=20000000)
#fuses NOWDT,HS
#use rs232(baud=2400,parity=N,xmit=PIN_C6,rcv=PIN_C7,bi ts=8)

char c;

#INT_RDA
Receive_isr()
{
c=getc(); // nhan ky tu.
}
void main(void)
{
set_tris_b(0x00);
output_b(0x00);

enable_interrupts(INT_RDA);
enable_interrupts (GLOBAL);

while(1)
{
output_b(c);
}
}
```

Thang8831
http://www.picvietnam.com

13. Ghi đọc RAM ngoài

13.1. Sơ đồ



13.2. Code

```
////////////////////////////////////////////////////////////////////////////
//Chuong trinh ghi va doc RAM ngoai
//Su dung PIC16F877, chot bang 74HCS573, RAM 62256 32Kx8
//
//Cong viec can thuc hien:
//Ghi du lieu vào RAM sau moi lan bam nut,
//sau 10 lan ghi, sang LED va doc lan luot 10 gia tri do.
//
////////////////////////////////////////////////////////////////////////////
#include <16F877.h>
#device *=16
#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES RC                       //Resistor/Capacitor Osc with CLKOUT
#FUSES NOPROTECT                //Code not protected from reading
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES LVP                      //Low Voltage Programming on B3(PIC16) or
B5(PIC18)
#FUSES NOWRT                    //Program memory not write protected
#FUSES NODEBUG                  //No Debug mode for ICD
```

```
#use delay(clock=20000000)
//khai bao bien
int8 adc;
int ghi;                                    //ghi=1:dang ghi du lieu vao RAM
int16 diachi;

#int_ext
void ngat_RB0()
{
int16 diachi;                               //bien dem so lan ghi vao RAM
   if( diachi < 10 ) {
      adc=read_adc();                       //doc gia tri ADC
      output_high( PIN_D7 );                 //khoa RAM
//thiet lap dia chi cho RAM:
      output_high( PIN_B1 );                //LE=1, cho phep xac lap 8bit
thap dia chi RAM
      output_c( diachi );                   //xac lap 8bit thap dia chi RAM
      output_low( PIN_B1 );                 //LE=0, chot 8bit thap dia chi
RAM
      output_d( diachi>>8 );                //xac lap 8bit cao dia chi RAM,
                                            //dong thoi mo RAM (RD7=0)
//ghi gia tri vao RAM:
      output_low( PIN_B2 );                 //chuyen sang che do ghi, WEbu=0
      output_c( adc );                      //ghi gia tri vao RAM
      output_high( PIN_D7 );                //khoa RAM
      diachi++;
   }
   if( diachi == 10 ) {
      output_high( PIN_A5 );                //sang LED
      diachi = 0;
      ghi = 0;                              //da ghi xong, cho phep xu li
tiep
   }
}
// Chuong trinh chinh
main()
{
setup_adc_ports( AN0_AN1_VSS_VREF );        //A0,A1 la ADC, VRef+ la A3
setup_adc( ADC_CLOCK_INTERNAL );
set_adc_channel(0);                         //chon AN0

enable_interrupts( global );
enable_interrupts( int_ext );               //chon ngat ngoai
ext_int_edge( L_TO_H );                     //ngat dua vao canh len

while(true) {
   ghi = 1;                                       //dang ghi du lieu vao RAM
   if( ghi == 0 ) {                               //du lieu da ghi xong
      output_high( PIN_B2 );                      //khong cho phep che do
ghi
      output_low( PIN_B3 );                       //chuyen sang che do doc,
OEbu=0
            for( diachi=0;diachi<=10;diachi++ ) {
            //thiet lap dia chi cho RAM:
            output_high( PIN_B1 );                //LE=1, cho phep xac lap
8bit thap dia chi RAM
            output_c( diachi );                   //xac lap 8bit thap dia
chi RAM
            output_low( PIN_B1 );                 //LE=0, chot 8bit thap dia
chi RAM
```

```
            output_d( diachi>>8 );              //xac lap 8bit cao dia chi
RAM,
                                                //dong thoi mo RAM (RD7=0)
            //doc gia tri tu RAM:
            input_c();              //doc gia tri tu RAM
            delay_ms(700);
            output_high( PIN_D7 );              //khoa RAM
      }
   }
}
}
```

Project 1: Kết nối PIC 16F877A với EEPROM 25AA640.

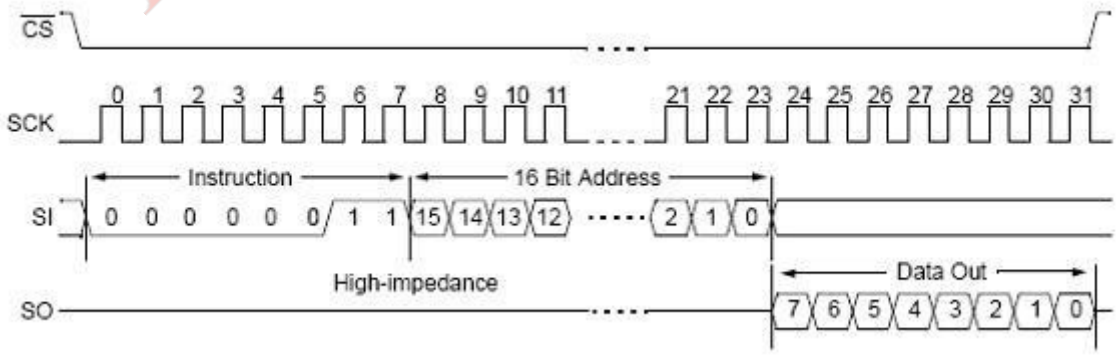
Sorry mọi người là tối hôm nay em tìm mỏi cả mắt mà không thấy bất cứ một thằng EEPROM nào có chuẩn giao tiếp SPI, cho nên ở Project này em chỉ xin được làm chay thôi, ai có điều kiện mạch thật hoặc có trình giả lập tốt thì xin test + đưa ra ý kiến cho em phát.

SPI là một chuẩn dữ liệu giao tiếp đơn giản nhất có tốc độ lớn nhất, tuy nhiên có độ an toàn không cao khi mà dây clock bị ảnh hưởng => dẫn đến ảnh hưởng đến toàn hệ thống. Với PIC16F877A thì có 3 chân cho chế độ SPI đó là: RC3(clock), RC4 (SDI), RC5 (SD0) , còn chân select chíp thì lấy bất cứ một chân I/O thông thường.

Cơ chế SPI là quá trình dịch bít qua lại giữa Slave và Master qua 2 đường đây SDI, SDO. Ứng với mỗi IC khác nhau lại cho một chuẩn truyền tiếp riêng để điều khiển quá trình truyền. Với EEPROM 25AA640 cơ chế đó là:

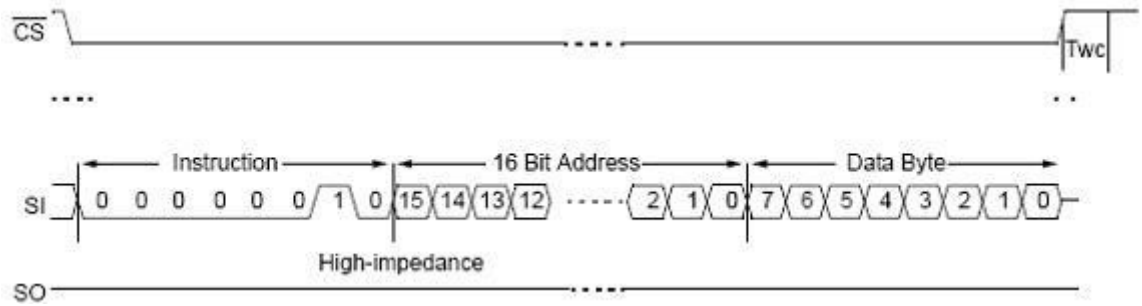
Đọc byte:

Truyền lệnh 0000011 tiếp đến là truyền địa chỉ 16 byte, và đọc dữ liệu . Khi chân CS lên 1 => cũng là lúc báo hiệu kết thúc đường truyền.



Write byte

Viết lệnh command: 00000010, sau đó truyền địa chỉ 16 bit, rồi bắt đầu truyền dữ liệu. Quá trình truyền kết thức khi CS = 1



PHP Code:

```
void main()
{
// init ban dau
   OUTPUT_LOW(PIN_C2);
   setup_spi(SPI_MASTER|SPI_L_TO_H|SPI_CLK_DIV_4);
   OUTPUT_HIGH(PIN_C2);
   delay(5);
// truyen du lieu co gia tri 0x55 xuong eeprom tai dia chi 0x0004
   OUTPUT_LOW(PIN_C2);
   spi_write(0x02);         // command = 0x02 -> ghi du lieu
   spi_write(0x00);
   spi_write(0x04);
   wpi_write(0x55);
   OUTPUT_HIGH(PIN_C2);
   delay(5);
// Doc du leu
   OUTPUT_LOW(PIN_C2);
   spi_write(0x03);        // command -= 0x03 -> doc du lieu
   spi_write(0x00);
   spi_write(0x04);
   wpi_read(buff);
   OUTPUT_HIGH(PORTD);
   delay(5);

   while(1);
}
```

cho em hỏi về vòng lặp while

các bác có thể cho mình biết cách sử dụng vòng lặp while dc ko?
chương trình mình viết như sau nhưng vòng lặp while ko thực hiện đc
Code:

```
int8 a,b;
main()
{
while(a==8){
a++;
portb=00;
delay_ms(100);
portb=0xFF;
delay_ms(100);
}
```

TL: Bên ngoài vòng while nên khởi tạo giá trị cho biến a. Điều kiện lặp là a==8 do đó nếu giá trị a ban đầu không phải là 8 thì vòng lặp không chạy. Nếu vòng lặp có chạy thì chỉ chạy 1 lần, vì bên trong vòng lặp a bị thay đổi.

Code của bạn chỉ cần sửa lại thành

Code:

```
int8 a,b;
main()
{
a = 0;
while(a<8){
a++;
portb=00;
delay_ms(100);
portb=0xFF;
```

Thang8831
http://www.picvietnam.com

```
delay_ms(100);
}
```