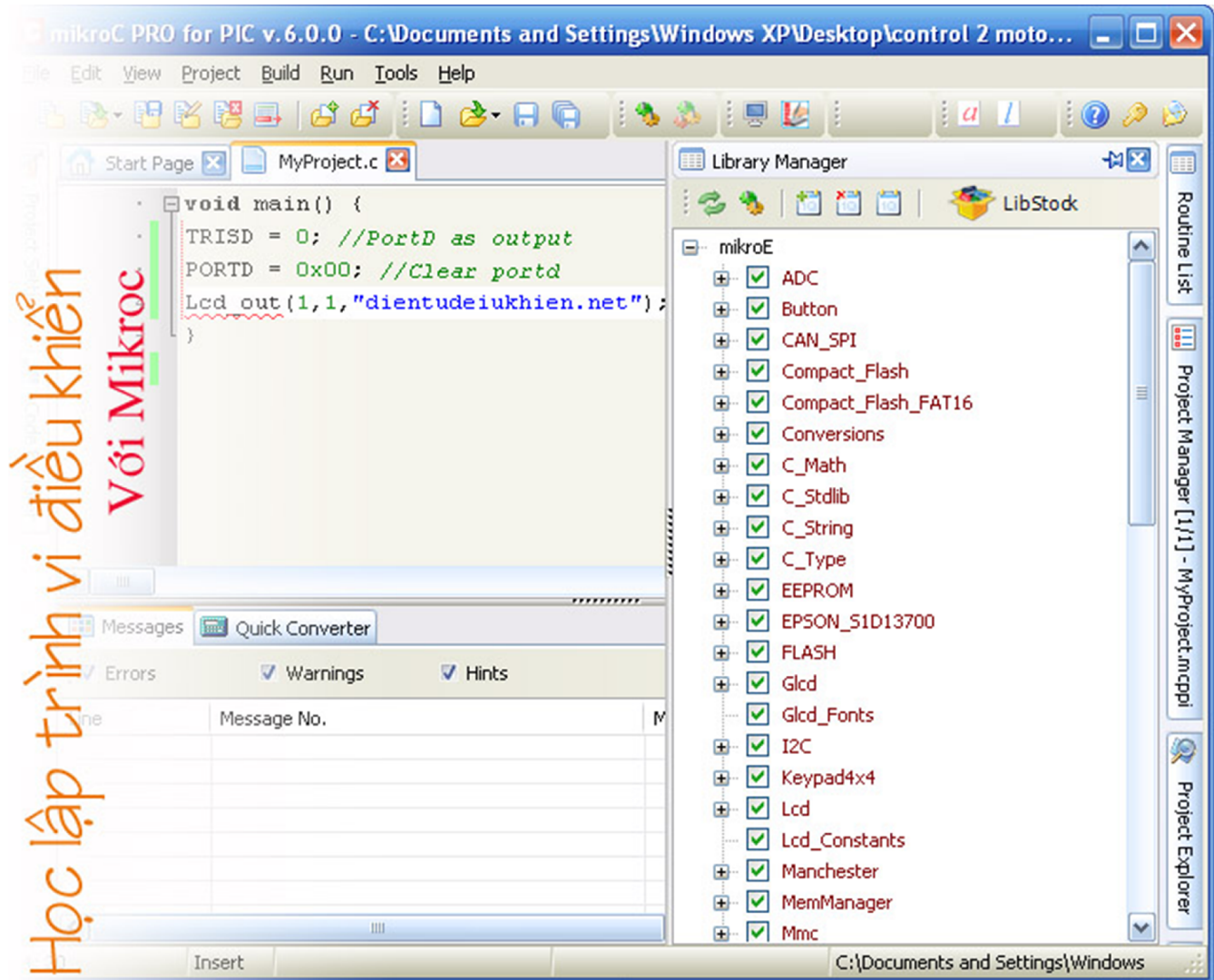


LẬP TRÌNH VI ĐIỀU KHIỂN PIC

Làm quen với MikroC



**Tác giả: Huỳnh Minh Trung*

**Website lập trình vi điều khiển: <http://dientudieukhien.net>*

MỤC LỤC

CHƯƠNG I: Giới thiệu về MikroC	3
CHƯƠNG II: Làm quen với MikroC	5
I- Tải và cài đặt chương trình	5
II- Các thành phần trên giao diện và chức năng	6
III- Tạo một Project như thế nào?.....	14
IV- Các phần cơ bản của một project.....	17
CHƯƠNG III: Thực hành qua các ví dụ đơn giản.	18
I- Ví dụ 1: điều khiển portc của vi điều khiển pic 16f877A:	18
II- Ví dụ 2: Dịch chuyển một bit qua trái <--> phải	19
III- Ví dụ 3: Lập trình ngắt INT Interrupt	21
IV- Ví dụ 4: Cảm biến nhiệt LM35, 7 segments, pic 16f887	23
V- Ví dụ 5: Điều khiển thiết bị quạt và lò sưởi, giao diện GLCD, pic 16f887	26
VI- Xem thêm nhiều project.....	32
TÀI LIỆU THAM KHẢO	33

CHƯƠNG I: GIỚI THIỆU VỀ MIKROC PRO for PIC

Trong hơn một thập kỷ MikroElektronika (Công ty phát triển phần mềm biên dịch MikroC) là một trong số ít các công ty trên thế giới phát triển bộ công cụ cho tất cả các kiến trúc vi điều khiển. Họ đã tạo ra các board mạch ứng dụng cũng như các board mạch dùng trong thí nghiệm và phát triển, trình biên dịch (MikroC for Pic, MikroBasic...), phần mềm nhúng và sách học lập trình cho vi điều khiển.



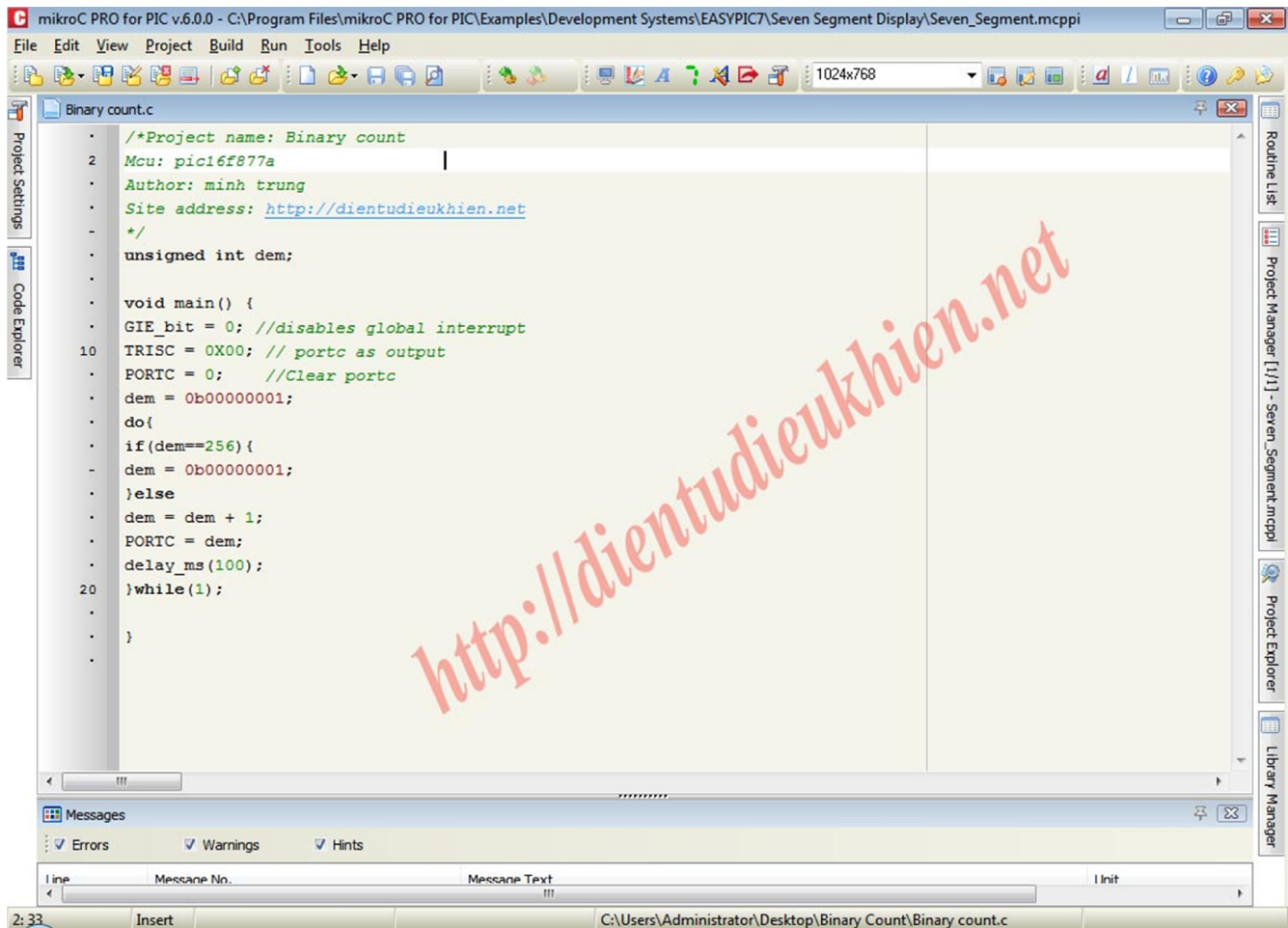
Trụ sở công ty MikroElektronika

MikroE đặt nền tảng trên một ý tưởng đơn giản, nhằm tạo ra các board cũng như các chương trình đơn giản, hỗ trợ cho người sử dụng một cách nhanh chóng và hiệu quả.

Ngày nay các sản phẩm của công ty được rất nhiều nơi cũng như nhiều bộ phận người dùng sử dụng – các kỹ sư, sinh viên, học sinh và các phòng lab trong các trường đại học, cao đẳng, trung học chuyên nghiệp.

Ngoài các sản phẩm về phần cứng và phần mềm nhúng, mikroelektronika còn nổi tiếng về các trình biên dịch như: MikroC for Pic, Dspic, AVR, ARM, 8051; MikroBasic for Pic, AVR, 8051, ARM; MikroPascal for Pic, AVR, ARM, 8051.

Sau đây là giao diện của chương trình MikroC Pro for Pic 6.0



Giao diện chính chương trình mikroC Pro for Pic ver 6.0

Các bài tập ví dụ trong ebook này được viết và biên dịch bằng MikroC Pro for Pic 6.0, các bạn tải chương trình tại địa chỉ sau:

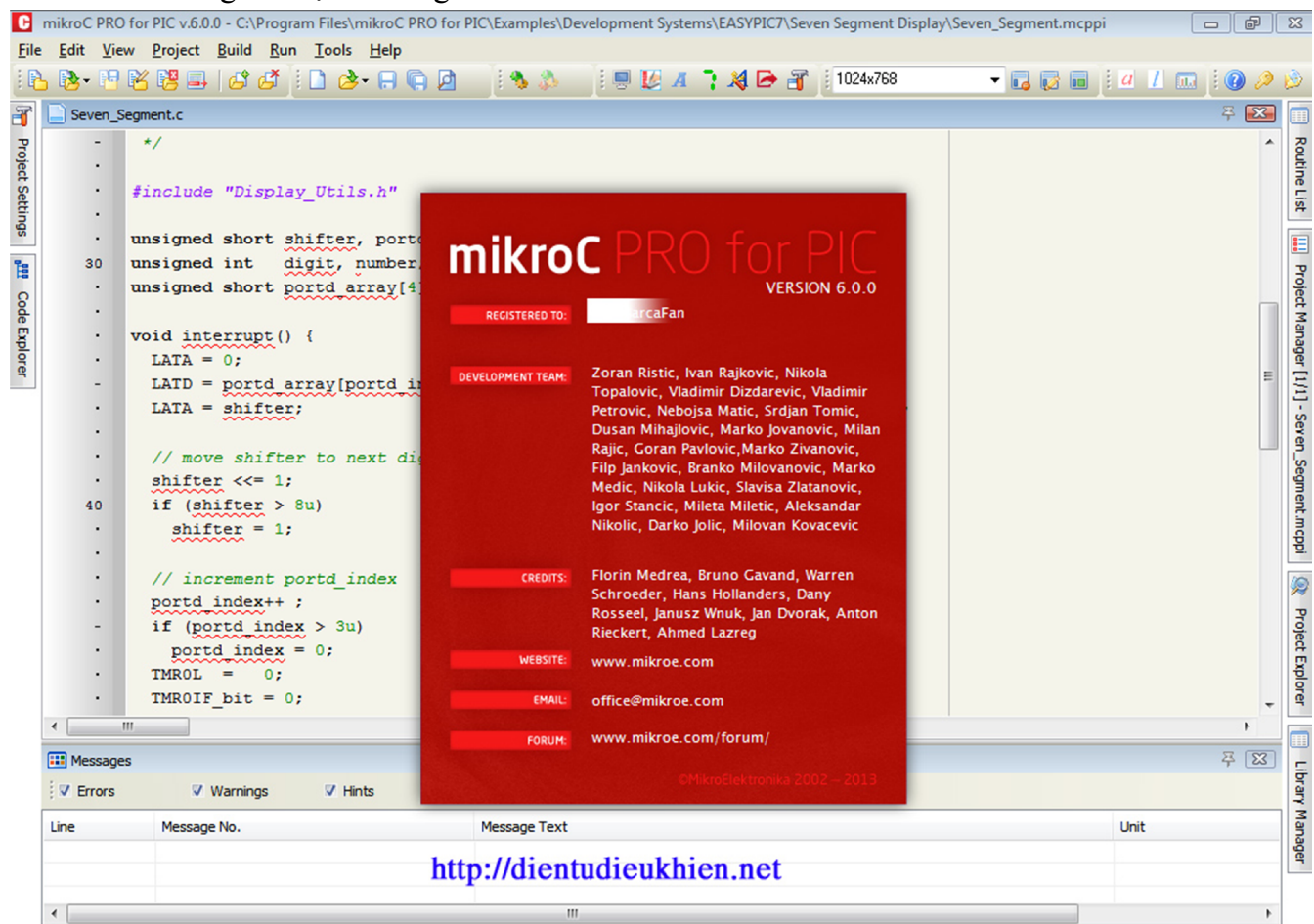
- <http://www.dientudieukhien.net/2014/11/mikrocpro-for-pic-ver-6.html> (full)
- Hoặc tải tại trang chủ mikroC.com (bản free chỉ biên dịch được 2Kb code)
- Các project nâng cao bạn tham khảo tại <http://dientudieukhien.net>

CHƯƠNG II: LÀM QUEN VỚI MIKROC PRO FOR PIC

I- Tải và cài đặt chương trình

Trước tiên bạn phải có trong tay bản cài đặt, các bản cập nhật có tại trang chủ <http://mikroe.com>, bản free ở đây chỉ biên dịch được 2Kb code. Bạn có thể tải bản full tại <http://www.dientudieukhien.net/2014/11/mikropro-for-pic-ver-6.html>, tại đây cũng có bản full 6.4

Sau khi có bản cài đặt bạn tiến hành cài đặt bình thường như các chương trình khác, bạn không nên cài chương trình vào folder mà tên có chứa dấu, nó có thể xảy ra lỗi khi biên dịch sau này. Khi hoàn thành giao diện chương trình như sau:

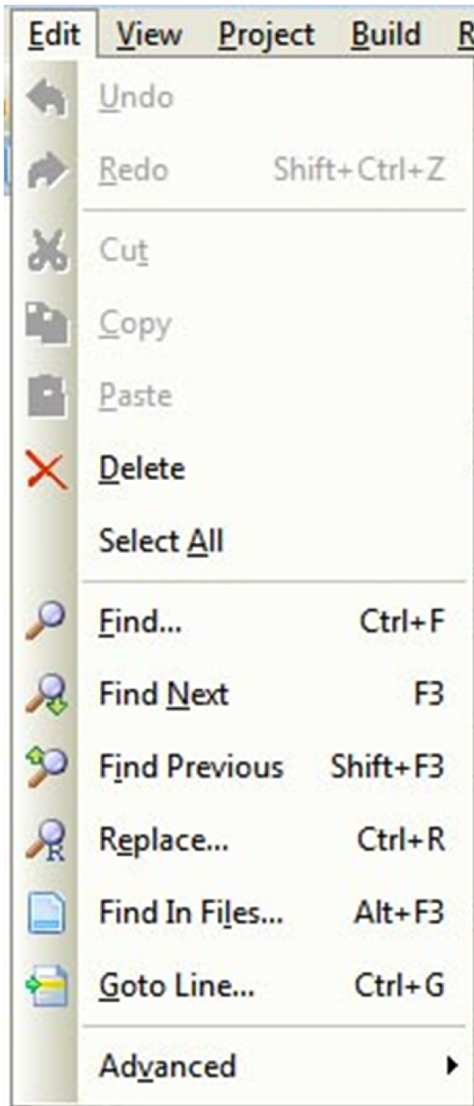


Giao diện chính của mikroC Pro for Pic version 6.0

II- Các thành phần trên giao diện và chức năng

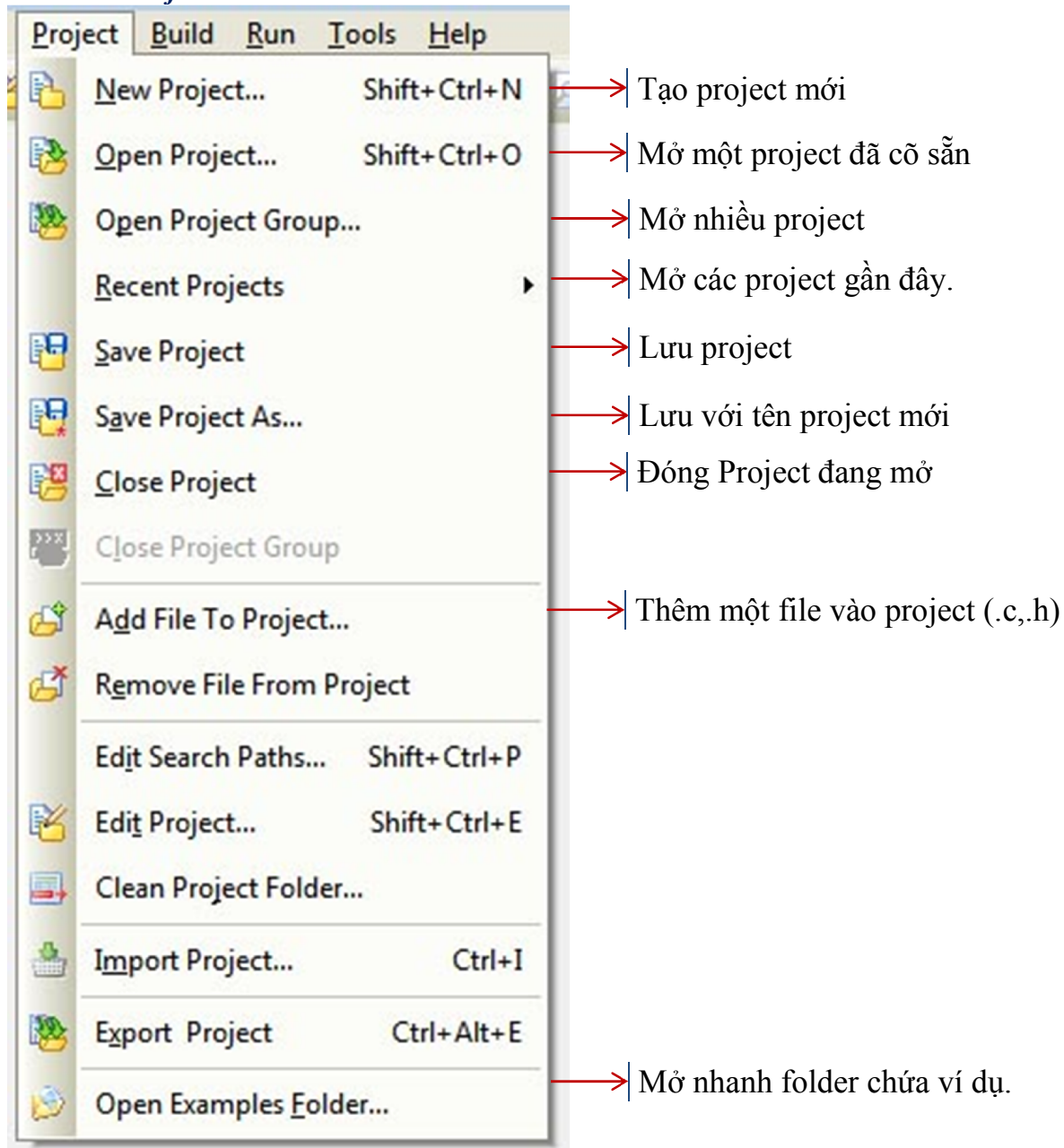
Để trực quan sinh động, mình sẽ trình bày kết hợp giữa hình ảnh và chú thích.

1. Menu Edit

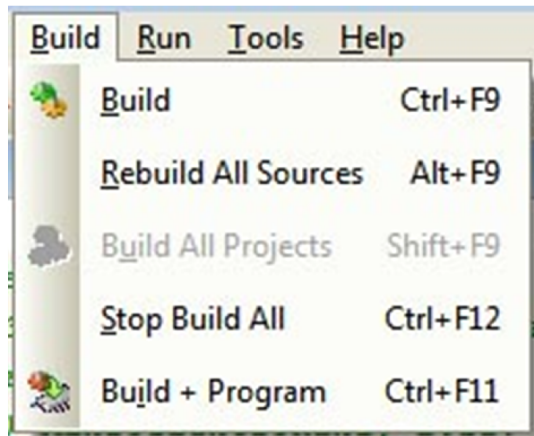


- Ở menu Edit gồm các phần đơn giản giống như trong Microsoft Office thôi, có vài chức năng khác như:
 - Find in Files...: Tìm từ khóa trên file đang mở.
 - Goto Line...: Nhảy con trỏ nhanh đến một line
 - Advanced: Gồm các phần như:
 - Comment: là dùng đánh dấu chú thích, các dòng text được Comment sẽ nằm trong /*text*/.
 - Uncomment: bỏ chọn comment một dòng text.
 - Indent: tăng thụt đầu dòng.
 - Outdent: giảm thụt đầu dòng.
 - Các thuộc tính còn lại đơn giản bạn có thể tìm hiểu thêm.

2. Menu Project



3. Menu Build

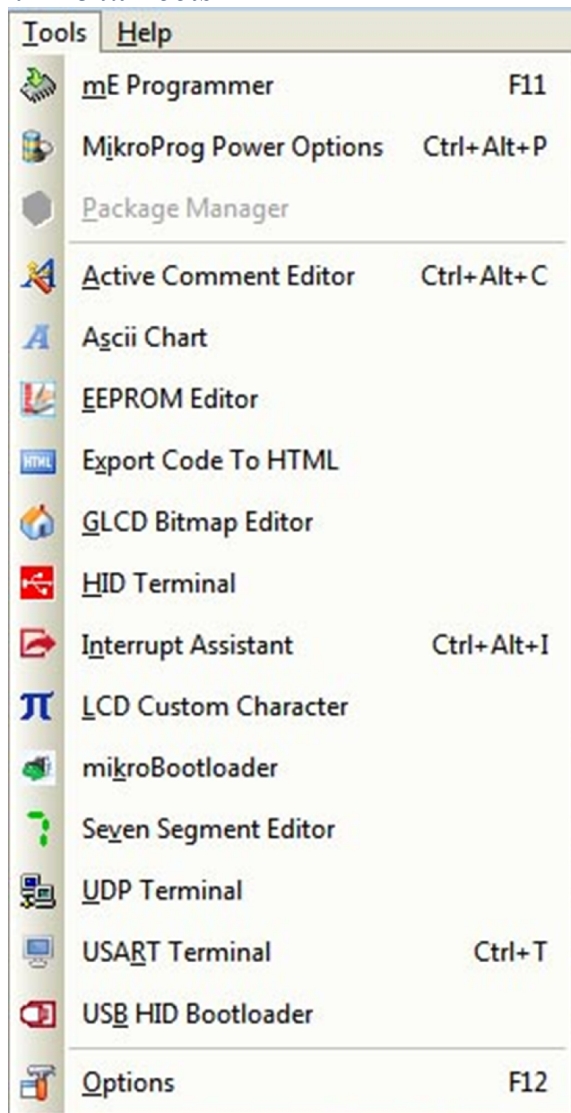


→ Biên dịch sang file hex

→ Biên dịch lại tất cả các source

Biên dịch và nạp chương trình cho vi điều khiển, ở đây phải sử dụng bootloader của mikroc mới được.

4. Menu Tools



→ Nạp chương trình

→ Xem và sửa dữ liệu trên EEPROM

→ Xuất code dạng mã HTML

→ Tool tạo ra ma trận dữ liệu ảnh, để hiển thị trên GLCD

→ Tạo những ký tự đặc biệt trên lcd

→ Chương trình hỗ trợ nạp vi điều khiển

→ Tạo mã hex cho led 7 đoạn

→ Hỗ trợ kết nối USART

→ Hiệu chỉnh màu sắc cho chương trình

5. Nhóm menu nhanh 1



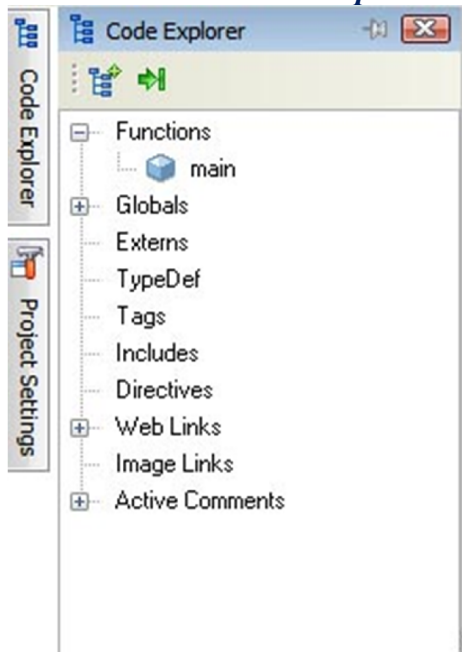
Các biểu tượng này có chức năng tương ứng giống như trong menu Project, nên ở đây mình không giải thích thêm – mất thời gian, đây chẳng qua là shortcut truy cập nhanh thôi.

6. Nhóm menu nhanh 2



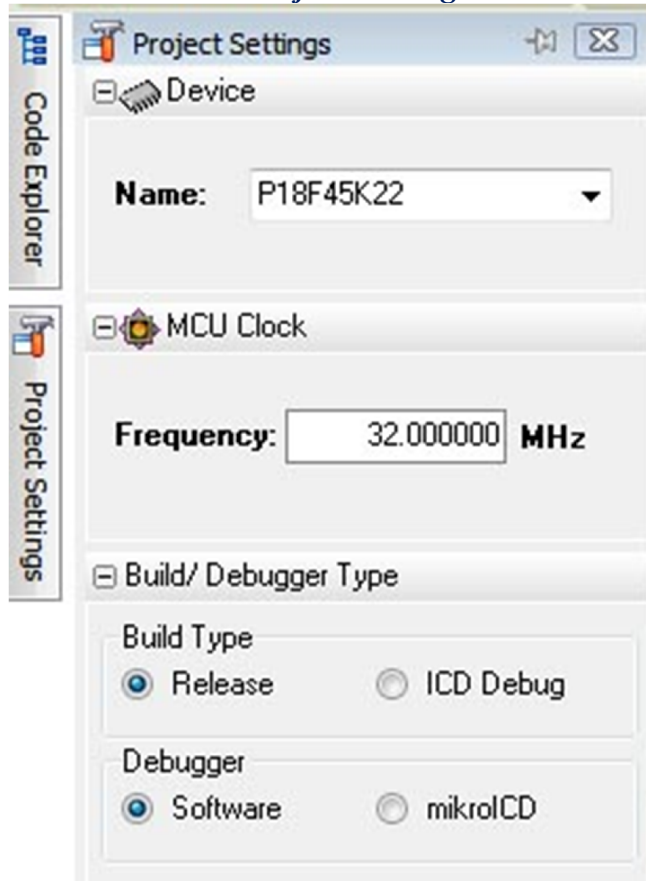
Ở menu truy cập nhanh này tương ứng và giống trong menu **Build** và **Tools** ở trên nên mình cũng không nói lại.

7. Menu nhanh Code explorer



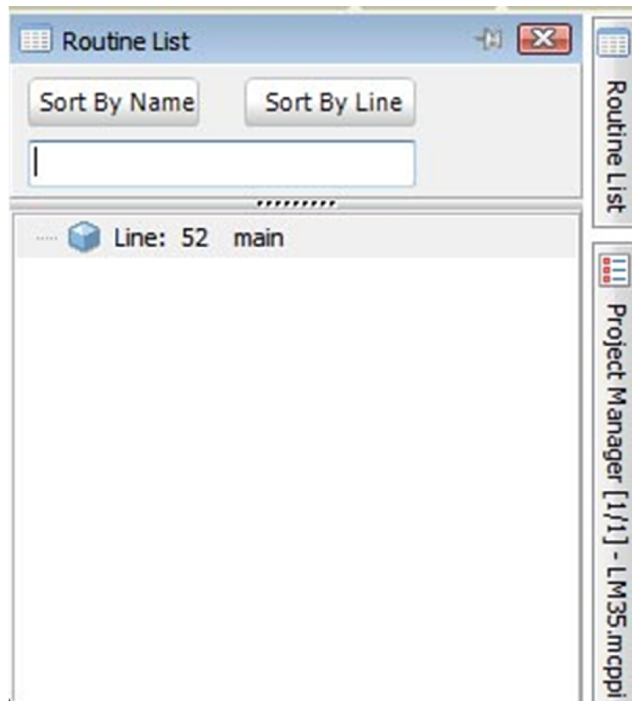
- Menu **Code explorer** hiển thị các thành phần trong source code bao gồm các hàm, ngắt, comment, links....

8. Menu nhanh Projects settings



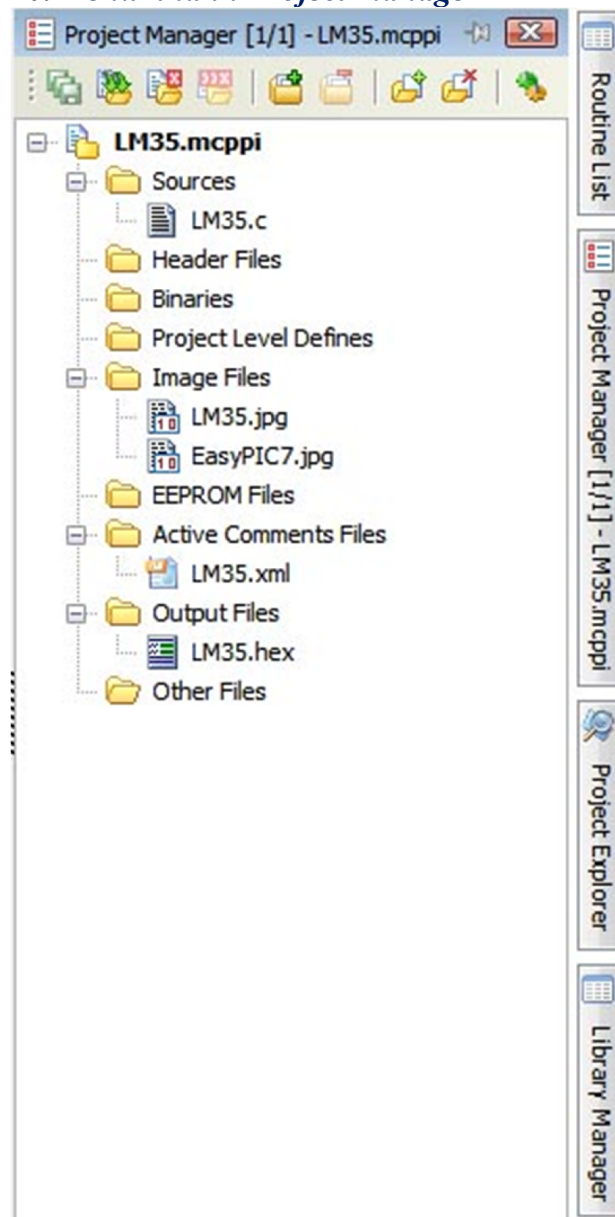
- Khi tạo project mới bạn đã chọn Device và tần số clock, tuy nhiên bạn vẫn có thể thay đổi Device và MCU Clock lại trong Project settings này.

9. Menu nhanh Routine list



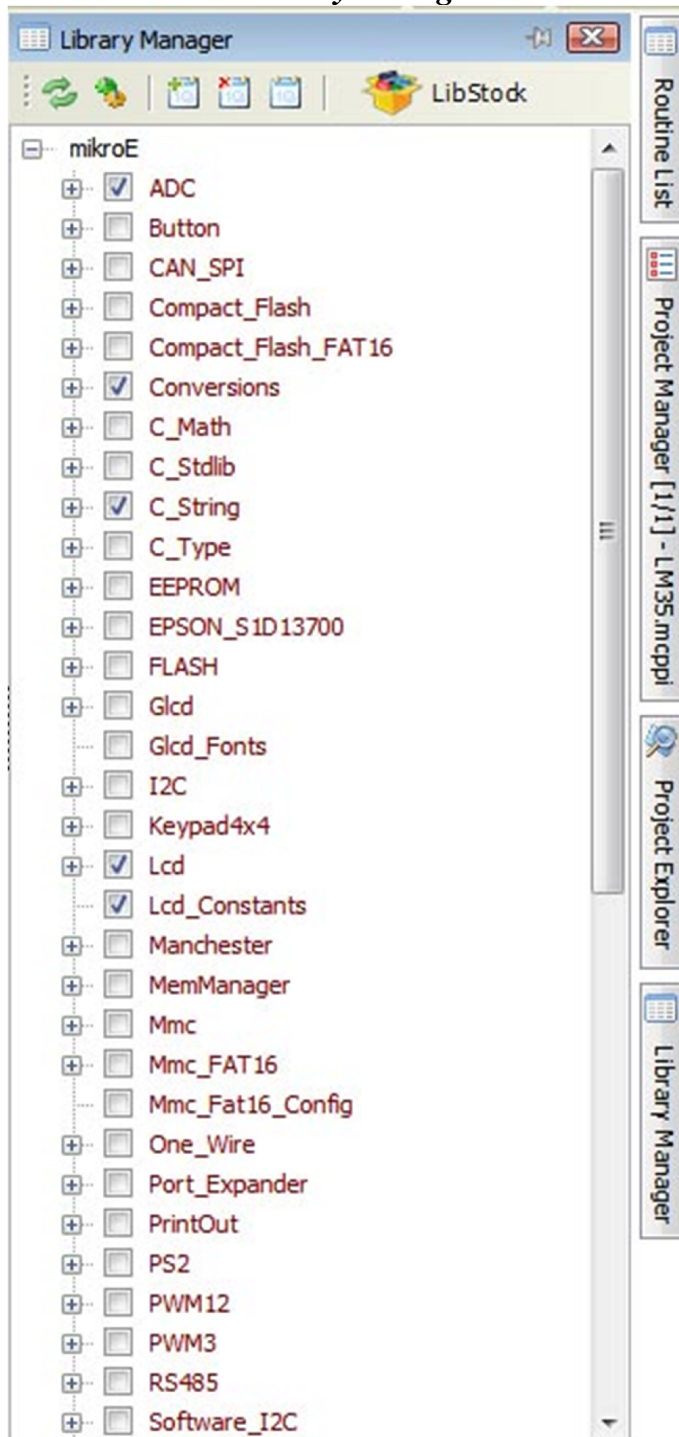
- **Routine List** là nơi quản lý các hàm con có trong chương trình.

10. Menu nhanh Project manager



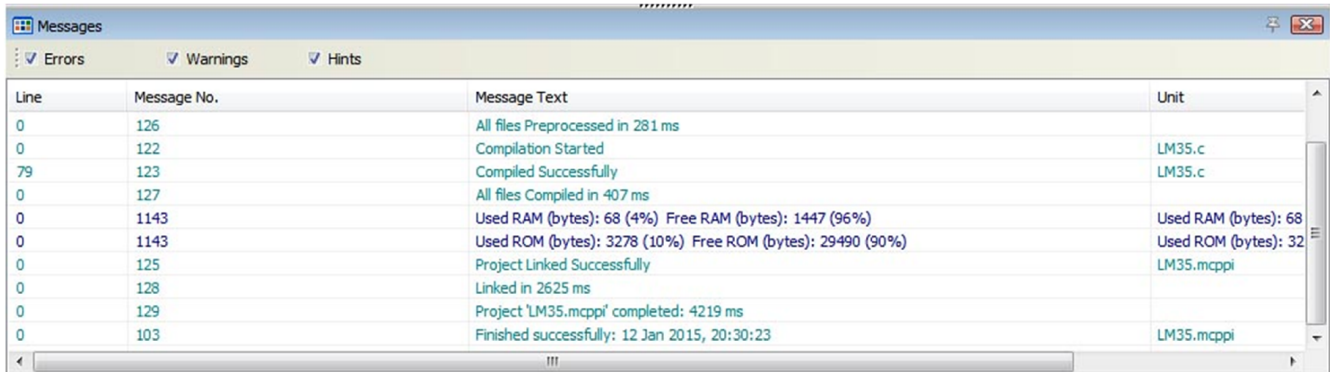
- **Project Manager** nơi bạn có thể dễ dàng quản lý các thành phần trong project như: source code C, các hình ảnh mà bạn đã **add file** vào, eeprom, file xml, file hex và các file khác...

11. Menu nhanh Library manager



- **Library Manager** nơi quản lý các thư viện hàm trong mikroc, nếu trong chương trình bạn có sử dụng thư viện hàm nào thì bạn phải đánh dấu check vào thư viện đó nếu không khi **Build** sẽ bị lỗi.
 - Ví dụ trong chương trình bạn có viết lệnh **Lcd_out(1,1,"dientudieukhien")** thì bạn phải đánh dấu **check** vào **ADC** trong **Library Manager**.

12. Thanh trạng thái message



The screenshot shows the 'Messages' window in MikroC. It has tabs for 'Errors', 'Warnings', and 'Hints'. The 'Errors' tab is selected. The table below represents the data shown in the window.

Line	Message No.	Message Text	Unit
0	126	All files Preprocessed in 281 ms	
0	122	Compilation Started	LM35.c
79	123	Compiled Successfully	LM35.c
0	127	All files Compiled in 407 ms	
0	1143	Used RAM (bytes): 68 (4%) Free RAM (bytes): 1447 (96%)	Used RAM (bytes): 68
0	1143	Used ROM (bytes): 3278 (10%) Free ROM (bytes): 29490 (90%)	Used ROM (bytes): 32
0	125	Project Linked Successfully	LM35.mcppi
0	128	Linked in 2625 ms	
0	129	Project 'LM35.mcppi' completed: 4219 ms	
0	103	Finished successfully: 12 Jan 2015, 20:30:23	LM35.mcppi

Khi bạn **Build (Ctrl+F9)** một chương trình thì tất cả các lỗi (nếu như có) sẽ được hiển thị ở đây. Nếu biên dịch thành công thì sẽ hiển thị các thông tin như: Used RAM, Used ROM và báo **Finished successfully**.

III- Tạo một Project như thế nào.

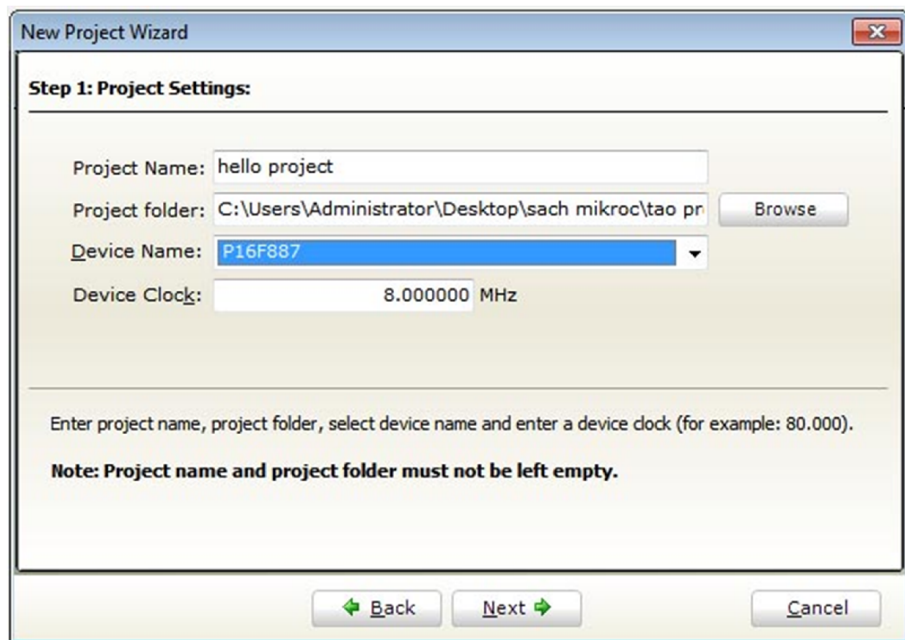
Để tạo một project, bạn chọn “**Project/New Project...**” và lần lượt theo các bước sau:

- **Bước 1:**



Chọn “**Next**” để qua bước 2

- **Bước 2:**



Project Name: đặt tên cho project

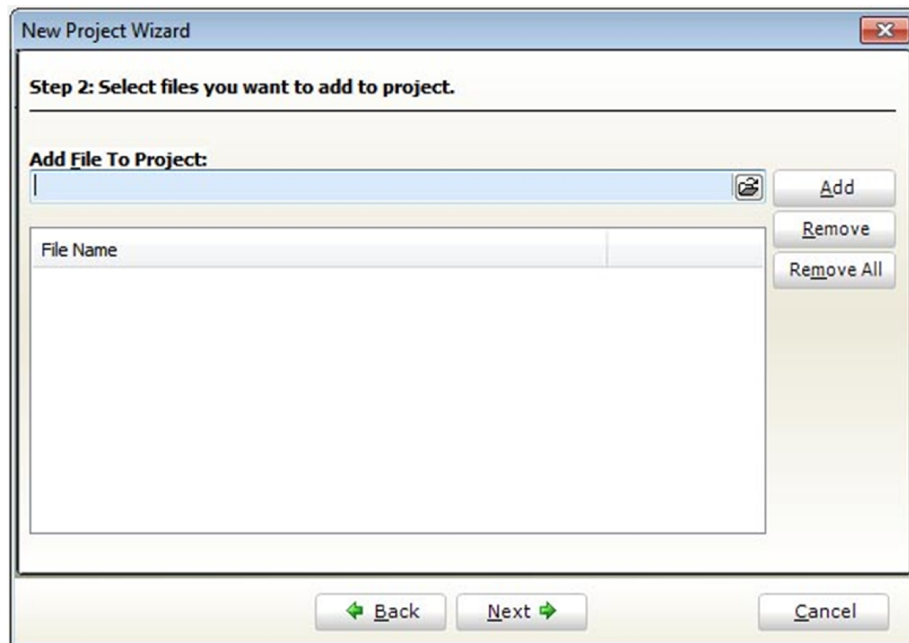
Project folder: Chọn đường dẫn lưu project

Device Name: Chọn loại vi điều khiển

Device clock: Điền tần số dao động

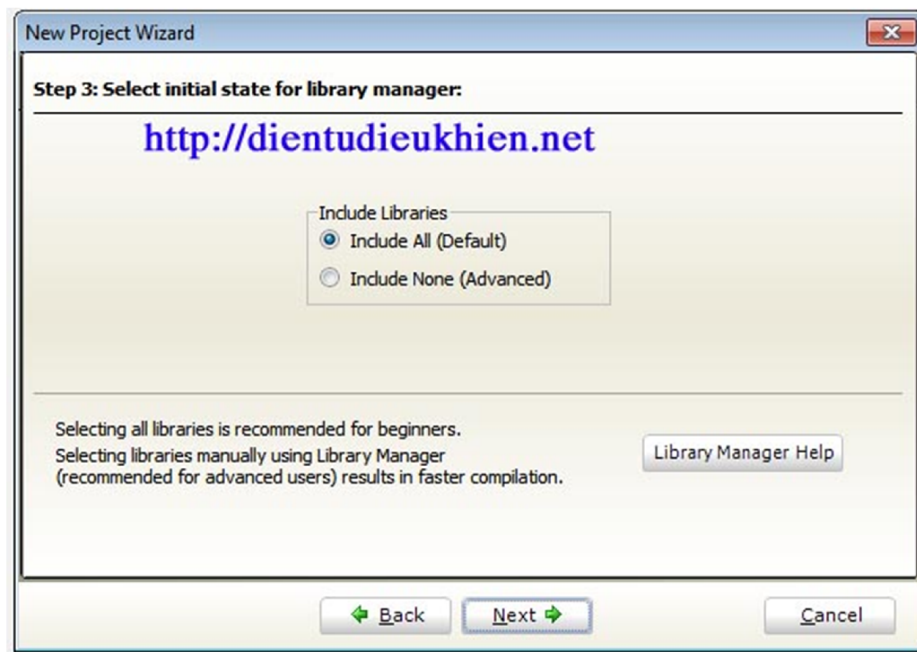
Click “**Next**” để qua bước 3

- **Bước 3:**



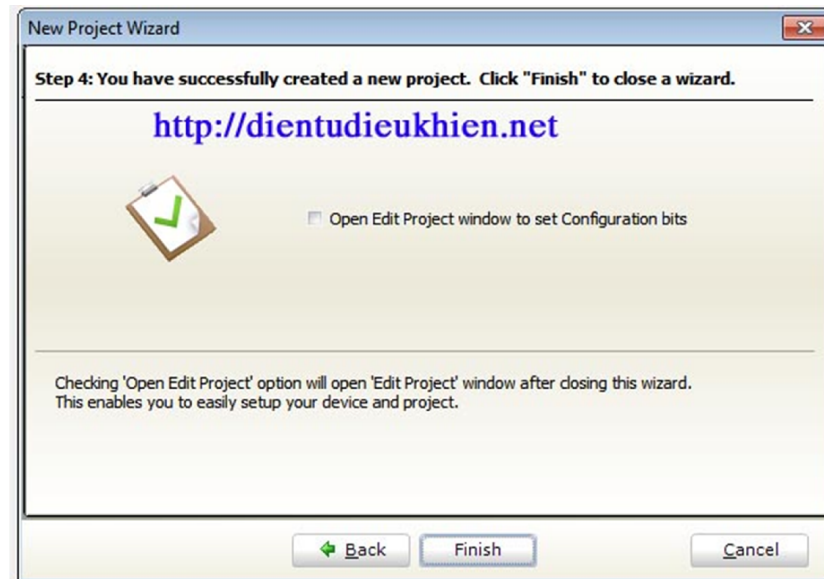
Add File to Project: thêm các file (.C,.h) vào project, nếu không có thì để trống, click “**Next**” để qua bước 4

- **Bước 4:**



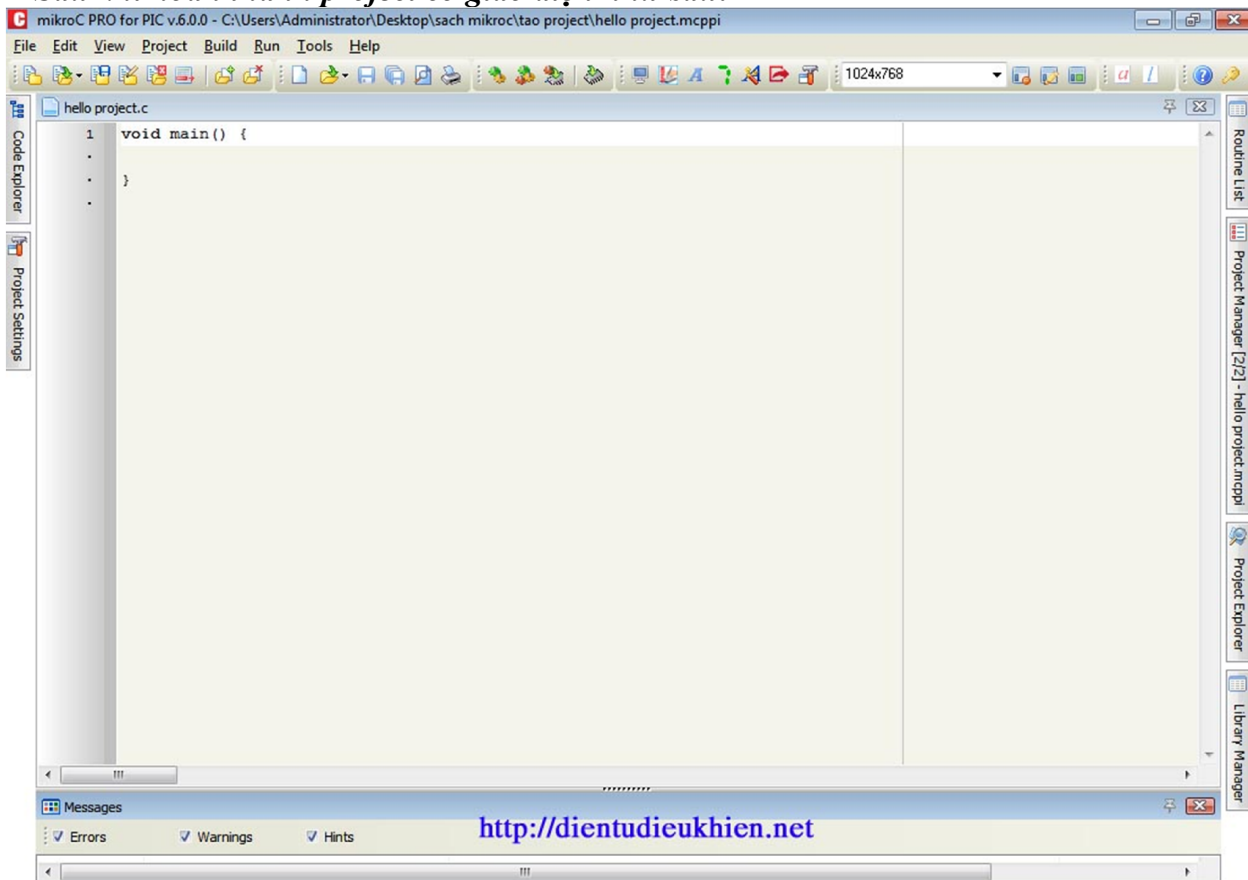
Click “**Next**” để qua bước 5

- **Bước 5:**



Click “Finish” để hoàn thành tạo một project

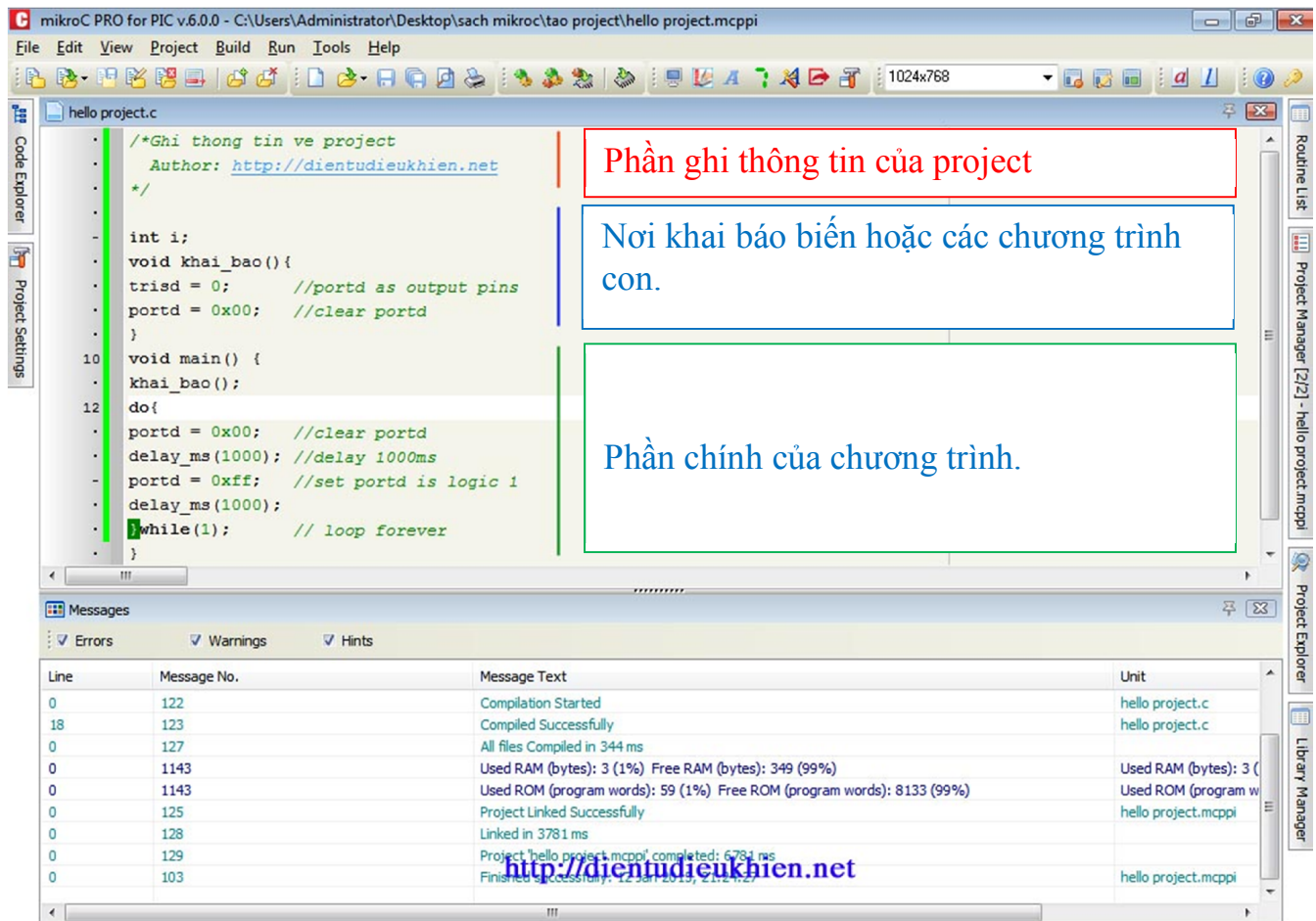
- **Sau khi hoàn thành project có giao diện như sau:**



Giao diện của project mới

IV- Các phần cơ bản của một project

Cấu trúc của một chương trình thông thường gồm có các phần như sau:



Chọn **“Build/Build”** hay **CTRL+F9** để biên dịch chương trình ra file hex, thành công thì trong phần Message báo **“Finish Successful”**

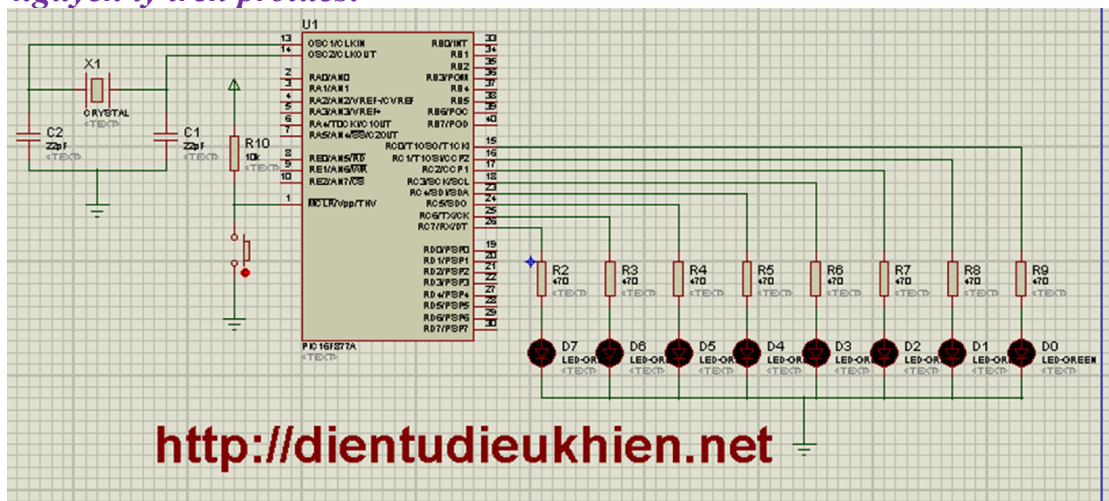
CHƯƠNG III: THỰC HÀNH QUA CÁC VÍ DỤ ĐƠN GIẢN

Các ví dụ được vẽ và mô phỏng trên Protues, bạn có thể tải và cài đặt chương trình theo link: <http://www.dientudieukhien.net/2012/06/chuong-trinh-mo-phong-protues-78-sp2.html>

I- Ví dụ 1: điều khiển portc của vi điều khiển pic 16f877A

Yêu cầu: Viết chương trình on/off các led ở portc của vi điều khiển pic 16f877A

Sơ đồ nguyên lý trên protues:



<http://dientudieukhien.net>

On/Off portd

Code:

/*On-Off led tren portd

Author: <http://dientudieukhien.net>

*/

void khai_bao(){

trisc = 0; //portd as output pins

portc = 0x00; //clear portd

}

void main() {

khai_bao();

do{

portc = 0x00; //clear portd

delay_ms(1000); //delay 1000ms

portc = 0xff; //set portd is logic 1

delay_ms(1000);

}while(1); // loop forever

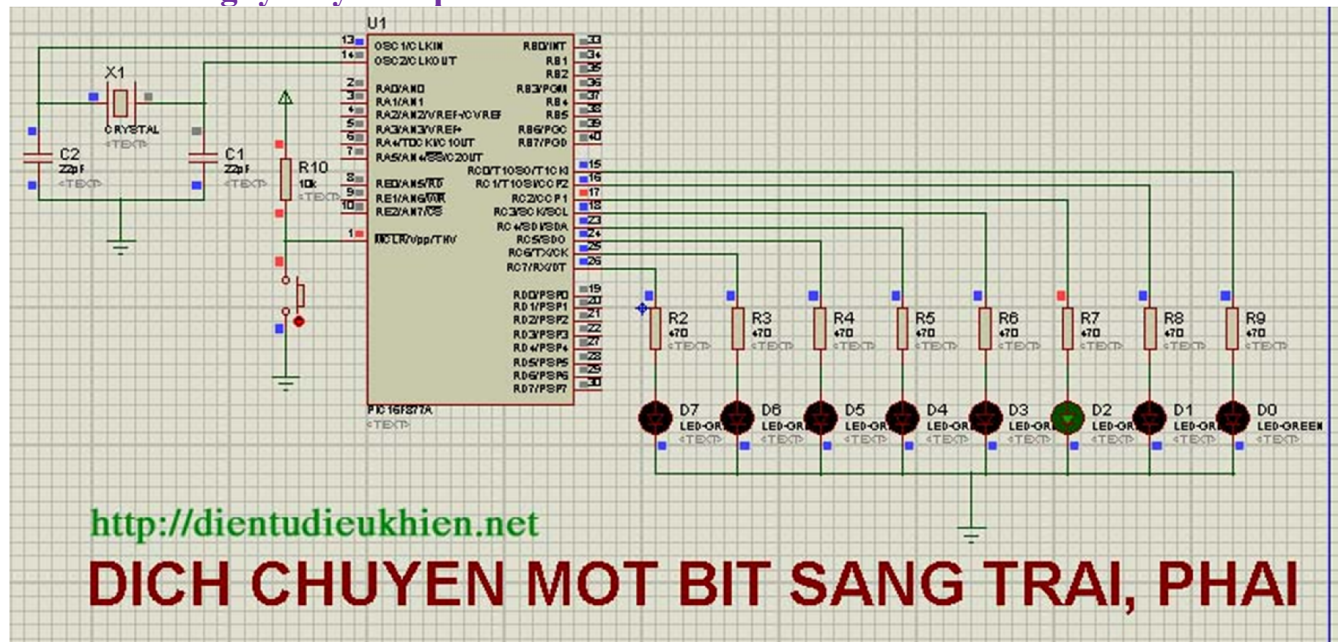
}

Link project: <http://www.dientudieukhien.net/2015/01/dieu-khien-on-off-led-tren-pic-16f877a.html>

II- Ví dụ 2: Dịch chuyển một bit qua trái <--> phải

Yêu cầu: có 8 led đơn kết nối vào portc của vi điều khiển pic 16f877A, 1 led bật sáng và dịch chuyển qua trái, sau đó dịch qua phải.

Sơ đồ nguyên lý trên protues:



Code:

```

/*Dịch chuyển một bit sang trái sau đó sang phải
CMU: 16f877A
Clock: 8Mhz
Author: Minh Trung
Site: http://dientudieukhien.net
*/
int i;                //Khai báo biến i
void movetoleft(){    //Hàm dịch chuyển bit sang trái
portc = portc<<1;
}
void movetoright(){   //Hàm dịch chuyển bit sang phải
portc = portc>>1;
}

void main(){          //Hàm chính
trisc = 0;            //portc as output
portc = 0x00;         //Clear portc
portc = 0b00000001;

```

```
delay_ms(500);  
do{                               //Loop forever  
for(i=0;i<7;i++){  
movetoleft();  
delay_ms(500);  
}  
for(i=0;i<7;i++){  
movetoright();  
delay_ms(500);  
}  
}while(1);                        // End loop  
}
```

Link project: <http://www.dientudieukhien.net/2015/01/dich-chuyen-mot-bit-qua-trai-phai-16f877A.html>

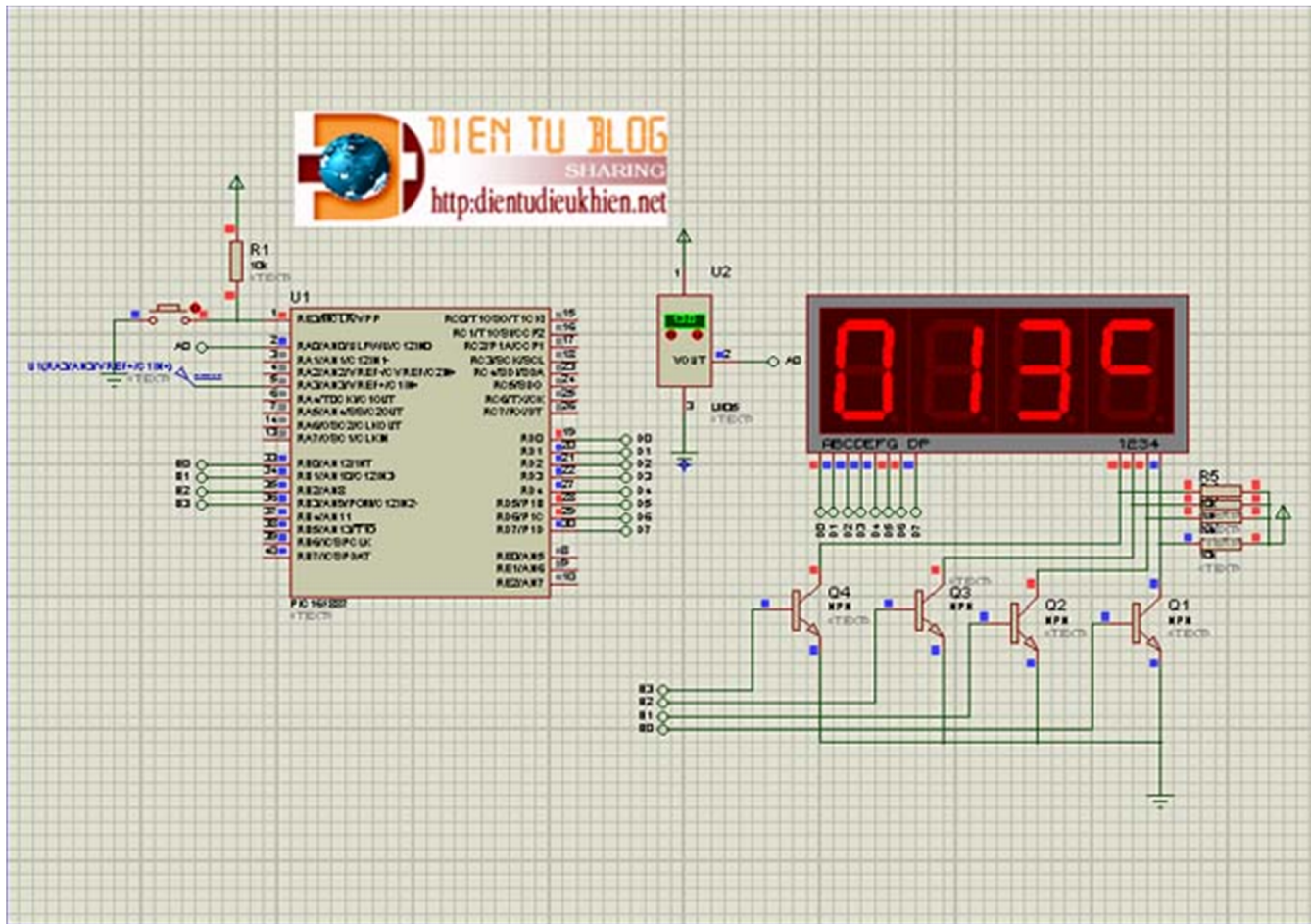

```
}  
  
void main() {  
    TRISC=0x00; //PortC at output.  
    GIE_bit = 1;    //Enable Global interrupt  
    INTEDG_bit = 1; //Interrupt on rising edge of INT pin  
    INTE_bit = 1;   // Enable External interrupt  
    INTF_bit = 0;   //Clear External interrupt flag  
    while(1){  
        PORTC=0x00;    // clear portc  
    }  
}
```

Link project: <http://www.dientudieukhien.net/2014/11/how-to-use-int-external-interrupt.html>

IV- Ví dụ 4: Cảm biến nhiệt LM35, 7 segments, pic 16f887

Yêu cầu: Sử dụng vi điều khiển Pic 16f887 xử lý tín hiệu từ cảm biến nhiệt lm35, hiển thị ra các led 7 đoạn.

Sơ đồ nguyên lý vẽ trên protues:



Code:

/*Project covert adc from sensor LM35, processing and display value on lcd.

Author: Minh Trung

Site: <http://dientudieukhien.net>

*/

//----- Returns mask for common cathode 7-seg. display

unsigned short mask(unsigned short num) {

switch (num) {

case 0 : return 0x3F;

case 1 : return 0x06;

case 2 : return 0x5B;

```
case 3 : return 0x4F;
case 4 : return 0x66;
case 5 : return 0x6D;
case 6 : return 0x7D;
case 7 : return 0x07;
case 8 : return 0x7F;
case 9 : return 0x6F;
case 10 : return 0x61;
} //case end
}

unsigned int value_adc=0;
unsigned int number,tempc;
unsigned short digit,shifter,portd_index;
unsigned short portd_array[4];

void interrupt(){
    portb = 0;
    portd = portd_array[portd_index];
    portb = shifter;
    shifter <<= 1;
    if(shifter>8u){
        shifter = 1;
    }
    portd_index++;
    if(portd_index > 3u){
        portd_index = 0;
    }
    TMR0 = 0;
    T0IF_bit = 0;
    T0IE_bit = 1;
}

void main() {
    Adc_Init();
    ANSEL=0b00001001; //Config porta.b0, porta.b3 as analog
    ANSELH=0x00;
    ADCON1=0x30;      //Config porta.B3 as Vref+
    TRISA.b0=1;       //porta.b0, porta.b3 as input
    TRISA.b3=1;
```

```
TRISB=0X00;
TRISD=0X00;
C1ON_bit=0;    //disable compare
C2ON_bit=0;
OPTION_REG = 0X04; //Ftimer0=Fosc/4, assign prescaler 1:32 to TMR0
GIE_bit = 1;
T0IE_bit = 1;
digit = 0;
portd_index = 0;
shifter = 1;
number = 123;
do{
value_adc=Adc_read(0); //Get Adc value
delay_ms(100);
tempc=value_adc*0.14648; //Vref is 1.5V
digit = tempc/100u;
portd_array[3] = mask(digit);
digit = (tempc/10u)%10u;
portd_array[2] = mask(digit);
digit = tempc%10u;
portd_array[1] = mask(digit);
digit = 10;
portd_array[0] = mask(digit);
delay_ms(1000);
number++;
if(number>999u){
    number = 0;
}

}while(1) ;
}
```

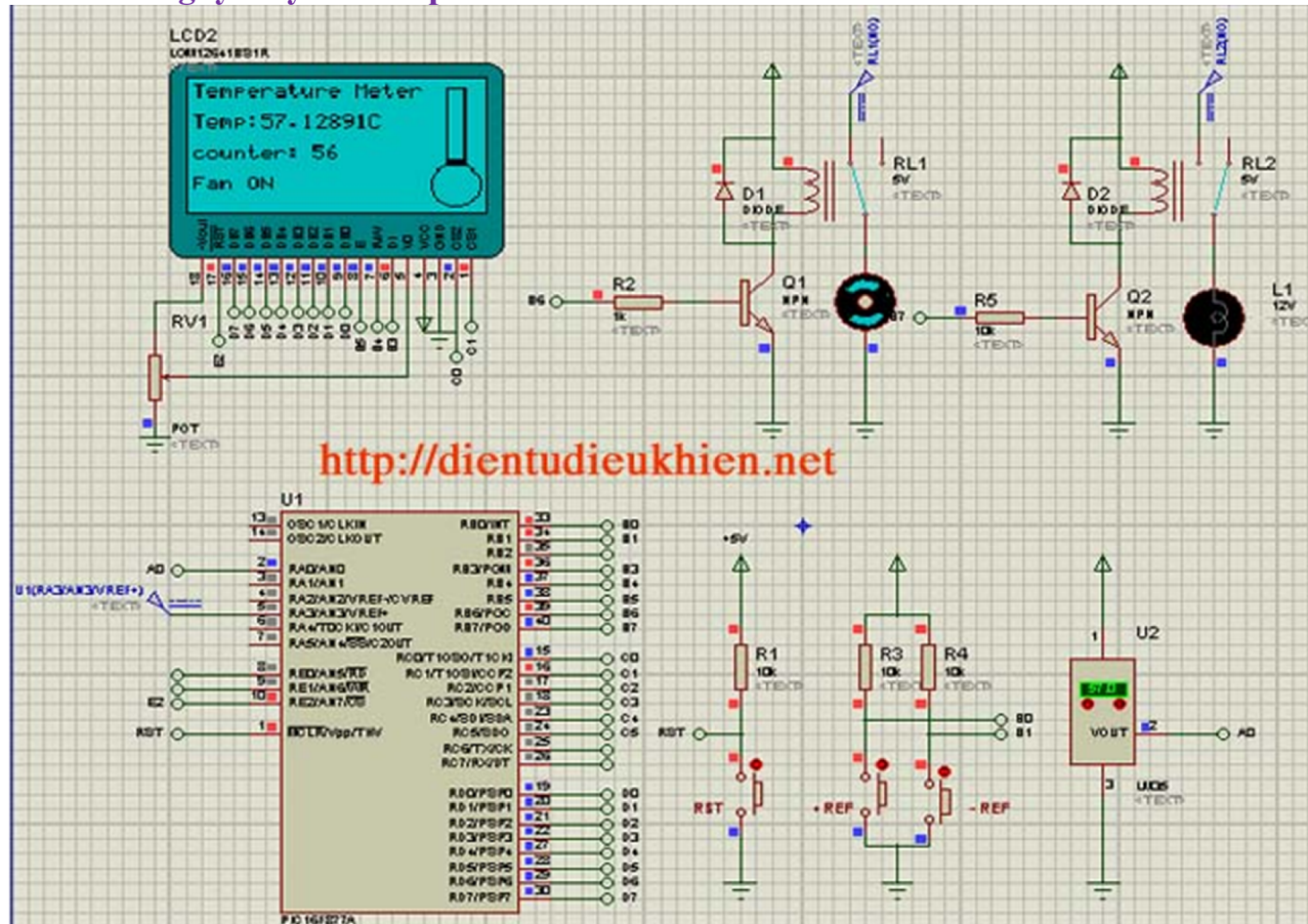
***PS:** Như vậy trong ví dụ trên bạn biết cách điều khiển led 7 đoạn, cách lấy tín hiệu và xử lý tín hiệu từ cảm biến nhiệt LM35.

Link project: <http://www.dientudieukhien.net/2014/11/using-pic-16f887-to-program-and-display-temperature-from-lm35-on-7segment.html>

V- Ví dụ 5: Điều khiển thiết bị quạt và lò sưởi, giao diện GLCD, pic 16f887

Yêu cầu: ta có 2 thiết bị là quạt gió (motor) và lò sưởi (bóng đèn). Khi nhiệt độ của phòng trên mức nhiệt đặt trước (Temp Ref) thì quạt gió hoạt động, lò sưởi tắt. Ngược lại thì quạt tắt và lò sưởi được mở. Giao diện người dùng được hiển thị trên GLCD. Mức nhiệt độ Ref được điều chỉnh qua 2 button được kết nối qua RB0, RB1, LM35 kết nối RA0, quạt kết nối RB6, lò sưởi RB7, GLCD kết nối với PORTD.

Sơ đồ nguyên lý vẽ trên protues:

**Code:**

/*

This project uses LM35 sensor that senses the temperature and then turns on Fan if temperature is high and turns on Heater if temperature is low.

The Graphic LCD displays temperature value in text and graphically.

User can select reference number using buttons. This reference

number is compared with the temperature to control Fan or Heater.

The reference number is stored in EEPROM.

Temperature sensor is connected to AN0. And Vref 1.5V connected to AN3

Author: Sameer Semmoor

Shared by <http://dientudieukhien.net>

*/

// Glcd module connections

char GLCD_DataPort at PORTD;

sbit GLCD_CS1 at RC0_bit;

sbit GLCD_CS2 at RC1_bit;

sbit GLCD_RS at RB3_bit;

sbit GLCD_RW at RB4_bit;

sbit GLCD_EN at RB5_bit;

sbit GLCD_RST at RE2_bit;

sbit GLCD_CS1_Direction at TRISC0_bit;

sbit GLCD_CS2_Direction at TRISC1_bit;

sbit GLCD_RS_Direction at TRISB3_bit;

sbit GLCD_RW_Direction at TRISB4_bit;

sbit GLCD_EN_Direction at TRISB5_bit;

sbit GLCD_RST_Direction at TRISE2_bit;

// End Glcd module connections

unsigned int adc_value; //stores analog values converted to digital

float TempC; //stores temperature value

float _TempC=0; //copy temperature value

unsigned short comp_value; //contains temperature value without a decimal point

char TempC_text[15];

unsigned short cnt; //stores the counter value

char cnt_txt[4];

bit temp_changed_flag; //will be set if temperature value changed

bit increment_switch; //will be set if increment switch was pressed

bit decrement_switch; //will be set if decrement switch was pressed

//configuration function-----

void _init()

{

```

ADC_Init(); // Initialize ADC module with default settings
ADCON1 = 0b00000101;

TRISA.B0 = 1; // PORT A0 as input

//counter switches as input
TRISB.B0 = 1; //increment
TRISB.B1 = 1; //decrement

TRISB.B6 = 0; PORTB.B6 = 0; //FAN as output
TRISB.B7 = 0; PORTB.B7 = 0; //HEATER as output

Glcd_Init(); // Initialize GLCD
Glcd_Fill(0x00); // Clear GLCD

Glcd_Write_Text("Temperature Meter", 0, 0, 1);
}
//end configuration function-----

//draw circle and rectangle-----
void draw()
{
    Glcd_Circle(116, 52, 11, 1);
    Glcd_Rectangle(112, 42, 120, 2, 1);
}
//end draw circle and rectangle-----

//get temperature-----
void get_adc()
{
    adc_value = ADC_Read(0); //read analog from AN0

    //calculate temperature
    TempC = adc_value * 1.5;

    TempC = TempC / 1024;

    TempC = TempC * 100;

    if(TempC != _TempC) //display temperature value if temperature changed

```

```

{
    temp_changed_flag = 1; //set this flag because temperature value changed
    _TempC = TempC; //copyt the new temperature value
    comp_value = TempC; //copy tmeperature value without decimal point

    floattostr(TempC,TempC_text); //convert temperature to string

    Glcd_Write_Text("Temp: ", 0, 2, 1);
    Glcd_Write_Text(TempC_text, 30, 2, 1); //display temeperature
    if(TempC < 1){ Glcd_Write_Text(" ", 82, 2, 1);} //if temperature value is below 1
clear -1 position
    Glcd_Write_Text("C", 78, 2, 1); //display temperature measurement unit
}
}
//end get temperature-----

//draw the progressbar-----
void progress_bar()
{
    //update progress bar circle
    if(tempC > 0 && tempC < 11){Glcd_Box(113, 3, 119, 41, 0);Glcd_Circle_Fill(116, 52,
10, 0);Glcd_Circle_Fill(116, 52, 2, 1);}
    if(tempC > 10 && tempC < 21){Glcd_Box(113, 3, 119, 41, 0);Glcd_Circle_Fill(116, 52,
10, 0);Glcd_Circle_Fill(116, 52, 4, 1);}
    if(tempC > 20 && tempC < 31){Glcd_Box(113, 3, 119, 41, 0);Glcd_Circle_Fill(116, 52,
10, 0);Glcd_Circle_Fill(116, 52, 6, 1);}
    if(tempC > 30 && tempC < 41){Glcd_Box(113, 3, 119, 41, 0);Glcd_Circle_Fill(116, 52,
10, 0);Glcd_Circle_Fill(116, 52, 8, 1);}
    if(tempC > 40 && tempC < 51){Glcd_Box(113, 3, 119, 41, 0);Glcd_Circle_Fill(116, 52,
10, 0);Glcd_Circle_Fill(116, 52, 10, 1);}

    //update progress bar rectangle
    if(tempC > 50 && tempC < 61){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 39, 119,
41, 1);}
    if(tempC > 60 && tempC < 71){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 35, 119,
41, 1);}
    if(tempC > 70 && tempC < 81){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 30, 119,
41, 1);}
    if(tempC > 80 && tempC < 91){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 25, 119,
41, 1);}
}

```

```

    if(tempC > 90 && tempC < 101){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 20, 119,
41, 1);}
    if(tempC > 100 && tempC < 111){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 15,
119, 41, 1);}
    if(tempC > 110 && tempC < 121){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 10,
119, 41, 1);}
    if(tempC > 120 && tempC < 131){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 5, 119,
41, 1);}
    if(tempC > 130 && tempC < 141){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 4, 119,
41, 1);}
    if(tempC > 140 && tempC < 151){Glcd_Box(113, 3, 119, 41, 0);Glcd_Box(113, 3, 119,
41, 1);}
}
//end draw the progressbar-----

//get counter value-----
void counter()
{
    if(increment_switch == 1) //if increment switch was pressed
    {
        cnt++; //increment counter
        if(cnt > 148){cnt = 150;} //counter should not exceed max
        eeprom_write(0,cnt); //store counter in eeprom
        delay_ms(300);
    }
    if(decrement_switch == 1) //if decrement switch was pressed
    {
        cnt--; //decrement counter
        if(cnt < 1){cnt = 1;} //counter should not go below min
        eeprom_write(0,cnt); //store counter in eeprom
        delay_ms(300);
    }

    bytetostr(cnt,cnt_txt); //convert counter value to string
    Glcd_Write_Text("counter: ", 0, 4, 1);
    Glcd_Write_Text(cnt_txt, 47, 4, 1); //display counter
}
//end get counter value-----

```

```
//control FAN & HEATER-----  
void control_fan_heater()  
{  
    if(comp_value == cnt) //if weather is normal  
    {  
        portb.b6 = 0; //turn off fan  
        portb.b7 = 0; //turn off heater  
        Glcd_Write_Text("      ", 0, 6, 1); //clear text position  
    }  
    else if(comp_value < cnt) //if weather is cold  
    {  
        portb.b6 = 0; //turn off fan  
        portb.b7 = 1; //turn on heater  
        Glcd_Write_Text("      ", 0, 6, 1); //clear text position  
        Glcd_Write_Text("Heater ON ", 0, 6, 1); //display Heater status  
    }  
    else //if weather is hot  
    {  
        portb.b7 = 0; //turn off heater  
        portb.b6 = 1; //turn on fan  
        Glcd_Write_Text("      ", 0, 6, 1); //clear text position  
        Glcd_Write_Text("Fan ON ", 0, 6, 1); //display fan status  
    }  
}  
//end control FAN & HEATER-----  
  
//main program-----  
void main()  
{  
    _init(); //start with the configurations  
  
    draw(); //draw the circle and rectangle  
  
    cnt = eeprom_read(0); //get counter from eeprom  
    counter(); //display counter at the beginning  
  
    do  
    {  
        get_adc(); //read ADC value
```



```

    if(portb.b0 == 0){increment_switch = 1; counter();} //if increment switch was pressed
    set its flag and increment counter
    if(portb.b1 == 0){decrement_switch = 1; counter();} //if deccrement switch was pressed
    set its flag and decrement counter

    //enter here if temperature changed or any of the switches was pressed
    if(temp_changed_flag == 1 || increment_switch == 1 || decrement_switch == 1)
    {
        progress_bar(); //update progress bar
        control_fan_heater(); //control FAN or HEATER

        //reset the flags
        temp_changed_flag = 0;
        increment_switch = 0;
        decrement_switch = 0;
    }

}while(1);
}
//end main program-----

```

Link project: <http://www.dientudieukhien.net/2015/01/dieu-khien-quat-lo-suo-i-giao-dien-dieu-khien-glcd.html>

VI- Xem thêm nhiều project

Ở trên là 5 ví dụ để bạn làm quen với phần mềm lập trình và biên dịch mikroC PRO for Pic, nhìn chung đây cũng là lập trình theo ngôn ngữ C nên bạn sẽ sớm làm quen, điều thú vị ở trình biên dịch này là đã hỗ trợ sẵn rất nhiều thư viện, giúp bạn dễ dàng hơn trong việc lập trình vi điều khiển. Các ví dụ ở trên là không đủ để bạn thực hành hết các chức năng của một cấu trúc vi điều khiển, bạn sẽ cần nhiều hơn các project để làm quen và rút kinh nghiệm, bạn dễ dàng **tải các project** có cả mô phỏng bằng protues tại địa chỉ web site: <http://dientudieukhien.net>.

TÀI LIỆU THAM KHẢO

- Quyển ebook được thực hiện với mong muốn chia sẻ những điều tác giả biết, trong khi viết khó tránh sai sót, mọi góp ý đóng góp xin gửi về địa chỉ mail: huynhminhtrung.ctv@gmail.com
- Có tham khảo thông tin trang <http://mikroe.com>
- Các Project được tham khảo từ <http://dientudieukhien.net> và <http://libstock.com>
- Tải nhiều project nâng cao + isis tại <http://dientudieukhien.net>