

KHOA ĐIỆN – ĐIỆN TỬ – VIỄN THÔNG
BỘ MÔN ĐIỆN – ĐIỆN TỬ



BÁO CÁO THỰC TẬP
VI ĐIỀU KHIỂN

Sinh viên thực hiện:
Thi Minh Nhựt – 1350366

Giảng viên hướng dẫn:
Đường Khánh Sơn

Cần Thơ
Ngày 24 tháng 5 năm 2016

Mục lục

Mở đầu	1
NỘI DUNG BÁO CÁO THỰC TẬP	4
1 LED ĐƠN	5
1.1 Giới thiệu chung	5
1.1.1 Yêu cầu	5
1.1.2 LED	5
1.1.3 Sơ đồ mạch	6
1.2 Cách điều khiển các chân xuất nhập số	6
1.3 Bài tập	7
1.3.1 Bài tập 1.1	7
1.3.2 Bài tập 1.2	9
1.3.3 Bài tập 1.3	9
2 HIỂN THỊ KÝ TỰ TRÊN LCD	11
2.1 Giới thiệu chung	11
2.1.1 Yêu cầu	11
2.1.2 LCD	11
2.1.3 Sơ đồ mạch	12
2.2 Sử dụng các lệnh cơ bản cho LCD	12
2.3 Bài tập	13
2.3.1 Bài tập 2.1	13
2.3.2 Bài tập 2.2	15
2.3.3 Bài tập 2.3	16
3 XỬ LÝ NGẮT	18
3.1 Giới thiệu chung	18
3.1.1 Yêu cầu	18
3.1.2 Hoạt động ngắn	18
3.2 Thiết lập hoạt động ngắn	18
3.3 Bài tập	19
3.3.1 Bài tập 3.1	19

3.3.2	Bài tập 3.2	21
3.3.3	Bài tập 3.3	24
3.3.4	Bài tập 3.4	25
3.3.5	Đọc tín hiệu từ nút nhấn với hàm INPUT	26
4	XỬ LÝ ADC	28
4.1	Giới thiệu chung	28
4.1.1	Yêu cầu	28
4.1.2	ADC	28
4.1.3	Sơ đồ mạch	28
4.2	Cấu hình ADC	29
4.3	Bài tập	29
4.3.1	Bài tập 4.1	29
4.3.2	Bài tập 4.2	31
4.3.3	Bài tập 4.3	34
5	TIMER	37
5.1	Giới thiệu chung	37
5.1.1	Yêu cầu	37
5.1.2	Timer/Counter	37
5.2	Các lệnh của Timer/Counter	37
5.3	Bài tập	38
5.3.1	Bài tập 5.1	38
5.3.2	Bài tập 5.2	40
5.3.3	Bài tập 5.3	42
6	GIAO TIẾP I2C	45
6.1	Giới thiệu chung	45
6.1.1	Yêu cầu	45
6.1.2	Chuẩn I2C	45
6.2	Giao tiếp chuẩn I2C với CCS	45
6.3	Bài tập	46
6.3.1	Bài tập 6.1	46
6.3.2	Bài tập 6.2	49
6.3.3	Bài tập 6.3	56
6.3.4	Mô phỏng cảm biến nhiệt TC74 với Protues	58
7	GIAO TIẾP RS232	60
7.1	Giới thiệu chung	60
7.1.1	Yêu cầu	60
7.1.2	Module RS232	60
7.2	Các lệnh giao tiếp RS232	60
7.3	Bài tập	61

7.3.1	Bài tập 7.1	61
7.3.2	Bài tập 7.2	63
7.3.3	Bài tập 7.3	65
8	ĐIỀU KHIỂN ĐỘNG CƠ DC	68
8.1	Giới thiệu chung	68
8.1.1	Yêu cầu	68
8.1.2	Động cơ DC	68
8.2	Các bài toán điều khiển động cơ	68
8.2.1	Điều khiển chiều quay của động cơ	68
8.2.2	Điều khiển tốc độ quay của động cơ	70
8.3	Bài tập	71
8.3.1	Bài tập 8.1	71
8.3.2	Bài tập 8.2	73
8.3.3	Điều khiển tốc độ động cơ với Transistor	77
9	DO KHOẢNG CÁCH BẰNG CẢM BIẾN SIÊU ÂM SRF05	79
9.1	Giới thiệu chung	79
9.1.1	Yêu cầu	79
9.1.2	Cảm biến siêu âm SRF05	79
9.2	Đo khoảng cách với cảm biến siêu âm	79
9.3	Bài tập	80
9.3.1	Bài tập 9.1	80
9.3.2	Bài tập 9.2	82
PHỤ LỤC		85
A	Sử dụng PICKit 2 Programmer	85
B	Thư viện DEF_887.H	88
C	Thư viện LCD_LIB_4BIT.C	92
D	Hiển thị với LED 7 đoạn	94
E	Thư viện RTC DS3231	107
F	Các khai báo sau #FUSES	112

Danh sách hình vẽ

1	Sơ đồ chân của PIC 16F887	4
1.1	Mạch điều khiển LED nối với PORT E	6
2.1	Mạch điều khiển LCD	12
2.2	Kết quả hiển thị ký tự lên LCD 16x02	14
3.1	Mạch đảo trạng thái LED ở PORT E với ngắt ngoài	20
3.2	Mạch đọc số lần nhấn nút ở chân B0 hiển thị lên LCD	22
3.3	Mạch đọc nút nhấn từ chân B3 – B5 và hiển thị lên LCD	24
4.1	Mạch đọc ADC từ biến trở	28
4.2	Kết quả chương trình ADC hiển thị lên LCD 16x02	30
4.3	Kết quả chương trình ADC xuất ra PORT E	32
4.4	Mạch đọc ADC từ biến trở xuất giá trị ra LED	32
4.5	Mạch đọc ADC từ biến trở gửi lên PC	35
5.1	Mạch đếm số lần nhấn nút sử dụng Timer hiển thị lên LCD	43
6.1	Mạch đọc giá trị nhiệt độ từ IC TC74	47
6.2	Mạch đọc thời gian thực từ module RTC DS3231	51
6.3	Kết quả đọc thời gian thực từ IC DS3231	52
6.4	Kết quả đọc thời gian thực và nhiệt độ từ IC DS3231	56
7.1	Mạch sử dụng module RS232 to TTL nhận lệnh từ PC bật tắt LED ở PORT E	62
7.2	Mạch sử dụng module RS232 to TTL nhận chuỗi từ PC hiển thị lên LCD	64
7.3	Mạch sử dụng module RS232 to TTL gửi nhiệt độ từ IC DS3231 lên PC	66
8.1	Mạch cầu H điều khiển chiều quay của động cơ	69
8.2	Sơ đồ chân IC L293D	69
8.3	Phương pháp điều chế độ rộng xung	70
8.4	Mạch đảo chiều động cơ với IC L293D	72
8.5	Điều khiển tốc độ động cơ với Transistor	77

A.1	Giao diện của phần mềm PICKit 2 Programmer	86
A.2	Giao diện khi nạp thành công chương trình cho PIC	87
D.1	Sơ đồ chân của LED 7 đoạn	94
D.2	Sơ đồ chân của LED 7 đoạn Anode hoặc Kathode chung ngoài thực tế	94
D.3	Sơ đồ chân của IC ghi dịch 74HC595	96
D.4	Sơ đồ mạch điều khiển một LED 7 đoạn	98
D.5	Sơ đồ mạch điều khiển một LED 7 đoạn qua IC 74HC595	100
D.6	Sơ đồ mạch điều khiển 4 LED 7 đoạn	102
D.7	Sơ đồ mạch điều khiển 4 LED 7 đoạn qua IC 74HC595	104

Danh sách bảng

1.1	Điện thế phân cực thuận của một số LED	5
2.2	Sơ đồ chân và chúa năng các chân của LCD	11
6.1	Địa chỉ thanh ghi thời gian của IC DS3231	49
6.2	Cài đặt định dạng thời gian hiển thị	49
6.3	Giải thích các tham số cài đặt thời gian thực cho IC DS3231	50
6.4	IC DS3231 với chức năng đọc nhiệt độ môi trường	56
D.1	Mã LED 7 đoạn Anode chung	95
D.3	Mã LED 7 đoạn Kathode chung	96
F.2	Ý nghĩa của các khai báo trong #FUSES	113

Mở đầu

Trước hết, em xin chân thành cảm ơn thầy đã tạo điều kiện cho chúng em có cơ hội thực hành trên board mạch thực tế, ứng dụng lý thuyết về vi điều khiển PIC 16F887 vào lập trình điều khiển một số thiết bị ngoài vi, khai thác các chức năng của vi điều khiển, từ đó làm cơ sở để giải quyết các bài toán trong thực tế cần đến ứng dụng của vi điều khiển.

Về thiết bị thực tập gồm có: Kit BOOK 1, Adapter, cáp USB, cáp kết nối,... cùng một số ngoại vi thay thế khác mà em chuẩn bị được để thực tập thêm, nhằm nâng cao khả năng lập trình của mình.

Các chương trình trong bài báo cáo được viết cho vi điều khiển PIC 16F887 bằng ngôn ngữ lập trình CCS. Với một số chương trình em có tham khảo thêm trên các diễn đàn, từ đó học hỏi và đúc kết lại để hoàn thành bài tập trong phần báo cáo.

Địa chỉ của một số trang web viết về vi điều khiển PIC và ngôn ngữ lập trình CCS mà em tìm hiểu được:

- picvietnam.com
- codientu.org
- dientuchiase.com
- ytuongnhanh.vn
- chiaseprojects.blogspot.com
- Kênh Youtube¹ của tác giả Nguyễn Thanh Dâng, về PIC16F877A và MikroC.
- ccsinfo.com

Bên cạnh phần cứng đó là các phần mềm hỗ trợ: ngôn ngữ lập trình CCS, chương trình nạp PICkit 2, ngôn ngữ lập trình Matlab, phần mềm mô phỏng Protues. Em sử dụng ngôn ngữ LATEX để hoàn thành bài báo cáo vì khi làm việc nhiều với code thì LATEX sẽ giúp định dạng code đẹp hơn, dễ nhìn hơn và tài liệu viết ra cũng mang tính cấu trúc hơn.

¹https://www.youtube.com/watch?v=NcjcOiDxC5I&list=PLhFjtzzUovr_22rUHg566bYU5s8LDKk7f

Trong bài báo cáo, ở mỗi bài em xin tóm tắt lại một số kiến thức trong tài liệu thực tập vi điều khiển của thầy để làm cơ sở giải bài tập thực hành. Mỗi bài tập em có trình bày định hướng cách giải quyết các bài tập theo cách suy nghĩ của em nên không thể tránh khỏi sai sót, mong thầy cho nhận xét để em có thể hoàn thiện bài báo cáo hơn.

Cuối cùng, qua đợt thực tập vi điều khiển PIC 16F887 em hy vọng bài báo cáo của em có thể làm tài liệu chia sẻ cho người mới bắt đầu học lập trình vi điều khiển với PIC 16F887 cần những bài thực hành cơ bản, theo em khi giải quyết các bài tập trong nội dung thực tập giúp cho em hiểu biết thêm về cách lập trình cho PIC 16F887.

Địa chỉ toàn bộ mã nguồn của bài báo cáo:

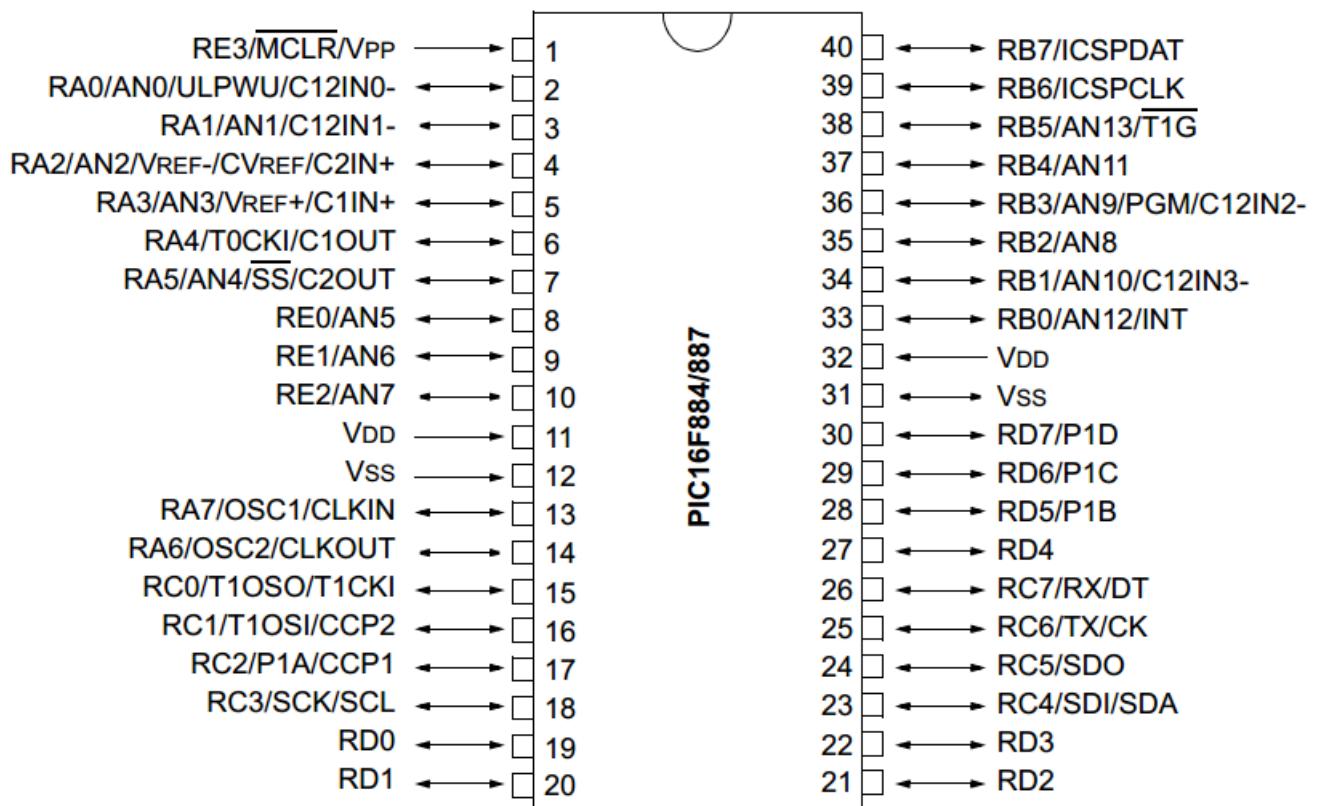
<https://github.com/thinhut/vdk-pic16f887/tree/master/VDK2016>

Sinh viên thực hiện,

Thi Minh Nhựt

NỘI DUNG BÁO CÁO THỰC TẬP

Sơ đồ chân của PIC 16F887



Hình 1: Sơ đồ chân của PIC 16F887

BÀI 1

LED ĐƠN

1.1 Giới thiệu chung

1.1.1 Yêu cầu

Điều khiển các chân I/O của PIC 16F887 với chức năng OUTPUT thông qua chớp tắt các LED được nối với PORT E của vi điều khiển.

1.1.2 LED

LED là diode có khả năng phát ra ánh sáng hay tia hồng ngoại, tia tử ngoại. Có điện thế phân cực thuận từ $1.5 \div 3V$, điện thế phân cực ngược thấp.

Điện thế phân cực thuận của một số loại LED:

Loại LED	Điện thế phân cực thuận
Đỏ	$1.4 \div 1.8V$
Vàng	$2.0 \div 2.5V$
Xanh lá cây	$2 \div 2.8V$

Bảng 1.1: Điện thế phân cực thuận của một số LED

Với các loại LED thường thì: $I_{max} = 30mA$. Còn LED loại siêu sáng thì điện áp phân cực thuận cao hơn LED thường (có loại lên đến $5V$), dòng $I_{max} = 30mA$.

Cách mắc điện trở hạn dòng cho LED:

- Có n LED mắc nối tiếp nhau:

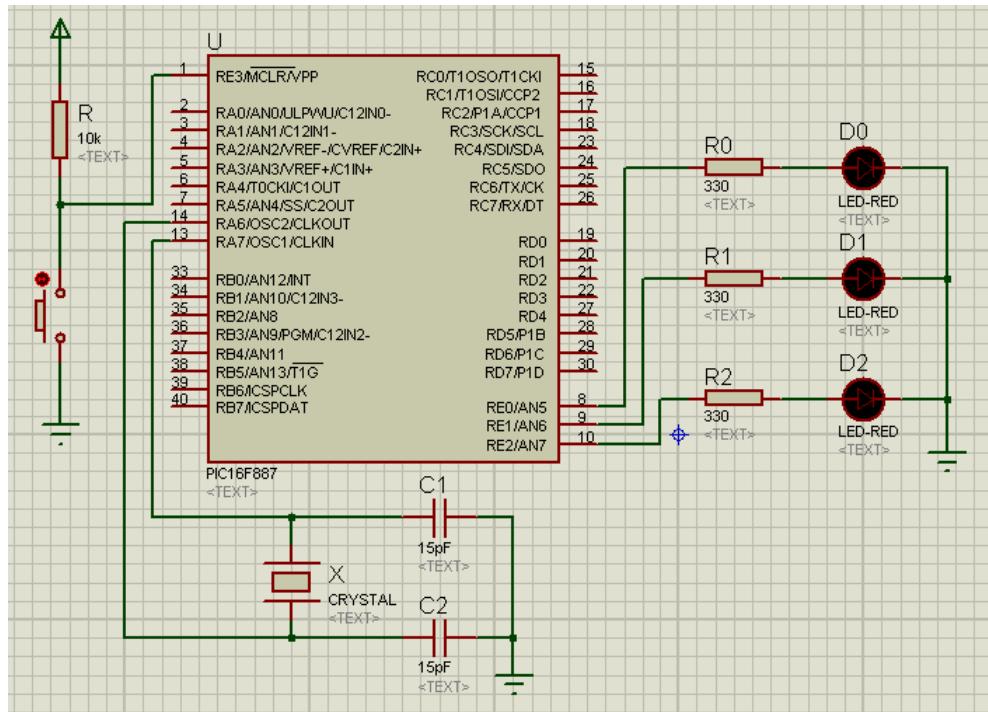
$$R_{nt} = \frac{U_{nguồn} - n \times U_{LED}}{I_{max}}$$

- Có m nhánh đấu mắc song song (mỗi nhánh gồm n LED nối tiếp):

$$R_{ss} = \frac{R_{nt}}{m}$$

- * Tùy theo yêu cầu về độ sáng của LED mà ta chọn giá trị của điện trở cho phù hợp.

1.1.3 Sơ đồ mạch



Hình 1.1: Mạch điều khiển LED nối với PORT E

1.2 Cách điều khiển các chân xuất nhập số

PORT E có chức năng xuất nhập số I/O và chức năng chuyển đổi ADC. Ở đây chúng ta quan tâm đến chức năng xuất nhập số.

Cách điều khiển:

- Xác định các chân của PORT E là chân OUTPUT hay là INPUT, dùng lệnh: TRISA hoặc TRISB hoặc TRISC hoặc TRISD hoặc TRISE, ở đây mình điều khiển PORT E nên dùng TRISE.
 - Mỗi chân sẽ có một trạng thái là 0 (chân OUTPUT) hoặc 1 (chân INPUT).
 - Cả ba chân RE2, RE1, RE0 là chân OUTPUT thì: TRISE = 0b000 = 0x00, thứ tự bit là RE2-RE1-RE0 (tương tự như các PORT khác, theo thứ tự là chân có chỉ số cao đến chân có chỉ số thấp).

Ví dụ: RE2 - INPUT, RE1,RE0 - OUTPUT thì: TRISE = 0b100 = 0x04, tương tự như các trường hợp khác của các PORT còn lại.

- Để đơn giản cho việc lập trình, chúng ta có thể khai báo mã ở dạng nhị phân 0b... hoặc khai báo mã dạng thập lục phân 0x... (mã hex).
 - Nếu là chân OUTPUT thì dùng 0 (mức thấp) hoặc 1 (mức cao) để thể hiện trạng thái của một chân.
- Ví dụ: RE2,RE1,RE0 – chân OUTPUT ở mức cao thì: PORTE = 0b111 = 0x07; RE2 – chân OUTPUT ở mức cao, RE1,RE0 – chân OUTPUT ở mức thấp thì: PORTE = 0b100 = 0x04
- Nếu là chân INPUT, thì phải đọc tín hiệu của chân đó (được xét ở bài sau).
 - Một số hàm hỗ trợ: delay_ms(số mili giây) hoặc delay_us(số micro giây), các cấu trúc lập trình for, while, if, if ... else,...

1.3 Bài tập

Trong chương trình có sử dụng thư viện DEF_887.H (trong *phụ lục B trang 88*).

1.3.1 Bài tập 1.1

Yêu cầu Viết chương trình chớp tắt LED ở PORT E với thời gian delay 250ms.

Hướng giải quyết

- Khai báo các chân ở PORT E là chân OUTPUT: TRISE = 0x00.
- Lặp lại quá trình sau (dùng cấu trúc while): LED tắt (PORTE = 0x00); giữ trạng thái cũ 250ms (delay_ms(250)); LED sáng (PORTE = 0b111 = 0x07); giữ trạng thái cũ 250ms (delay_ms(250)). Theo cách lập trình này ta có *Chương trình 1*.
- * *Cách khác*: đầu tiên cho LED tắt (PORTE = 0x00). Lặp lại quá trình: đảo trạng thái của LED (PORTE = ~PORTE); giữ trạng thái cũ 250ms (delay_ms(250)). Theo cách lập trình này ta có *Chương trình 2*.

Chương trình 1

```
1 /*Yeu cau:  
2     Chuong trinh chop tat cac LED o PORT E voi thoi gian delay 250ms  
3 */  
4 //Ten file: BAI-1-1.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
7 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
8 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
9  
10 void main(){  
11     TRISE = 0x00; //Tat ca cac chan o PORT E la chan xuat (output)  
12  
13     while (True){  
14         PORTE = 0x00; //Tat ca cac chan o muc thap --> LED tat  
15         delay_ms(250); //Duy tri trang thai cu trong 250ms  
16         PORTE = 0x07; //Chan RE2,RE1,RE0 o muc cao-->LED D3,D4,D5 sang  
17         delay_ms(250);  
18     }  
19 }
```

Chương trình 2

```
1 /*Yeu cau:  
2     Chuong trinh chop tat cac LED o PORT E voi thoi gian delay 250ms  
3 */  
4 //Ten file: BAI-1-1v2.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
7 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
8 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
9  
10 void main(){  
11     TRISE = 0x00; //Tat ca cac chan o PORT E la chan xuat (output)  
12     PORTE = 0x00; //Tat ca cac chan o muc thap --> LED tat  
13     while (True){  
14         PORTE = ~PORTE; //Tat ca cac chan o muc thap --> LED tat  
15         delay_ms(250); //Duy tri trang thai cu trong 250ms  
16         //Dao trang thai cua LED  
17     }  
18 }
```

1.3.2 Bài tập 1.2

Yêu cầu Viết chương trình chớp tắt LED ở PORT E với thời gian delay 1s.

Hướng giải quyết Ta sử dụng hàm `delay_ms(số mili giây)` với tham số truyền vào là 1000. Hàm `delay_ms(số mili giây)` có tham số 'số mili giây' có giá trị 0 – 65535 (int16).

Chương trình Sử dụng lại *Chương trình 1* hoặc *Chương trình 2*, thay đổi `delay_ms(250)` thành `delay_ms(1000)`.

1.3.3 Bài tập 1.3

Yêu cầu Viết chương trình chớp tắt LED ở chân RE1 với thời gian delay 1s và LED ở chân RE2 với thời gian delay là 0.5s.

Hướng giải quyết

- Với yêu cầu thì 2 LED thực hiện chớp hoặc tắt cùng một lúc khi mới vào chu kỳ đầu, nhưng phải đảm bảo đúng được chu kỳ của mỗi LED, với thời gian delay khác nhau, nên chia ra các trường hợp sau (để đảm bảo đúng thời gian delay):

Thời gian	LED ở RE2	LED ở RE1	PORTE
0ms	ON	ON	0b110 = 0x06
500ms	OFF	ON	0b010 = 0x02
1000ms	ON	OFF	0b100 = 0x04
1500ms	OFF	OFF	0b000 = 0x00
2000ms	Lặp lại trạng thái		

- Dựa vào bảng trên, ta viết được *chương trình 3* thỏa yêu cầu.

Chương trình 3

```
1 /* Yeu cau:  
2     Chuong trinh chop tat LED - RE1 voi thoi gian delay 1s;  
3     LED - RE2 voi thoi gian delay 0.5s  
4 */  
5 //Ten file: BAI-1-3.C  
6 #include<16F887.h>  
7 #include<def_887.h>  
8  
9 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
10  
11 #use delay(clock = 20000000)//Tan so thach anh 20MHz  
12  
13 void main(){  
14     TRISE = 0x00; //Tat ca cac chan o PORT E la chan xuat (output)  
15  
16     while (True){  
17         PORTE = 0x06; // 0b110;  
18         delay_ms(500);  
19         PORTE = 0x02; // 0b010;  
20         delay_ms(500);  
21         PORTE = 0x04; // 0b100;  
22         delay_ms(500);  
23         PORTE = 0x00; // 0b000;  
24         delay_ms(500);  
25     }  
26 }
```

BÀI 2

HIỂN THỊ KÝ TỰ TRÊN LCD

2.1 Giới thiệu chung

2.1.1 Yêu cầu

Viết chương trình hiển thị ký tự lên LCD.

2.1.2 LCD

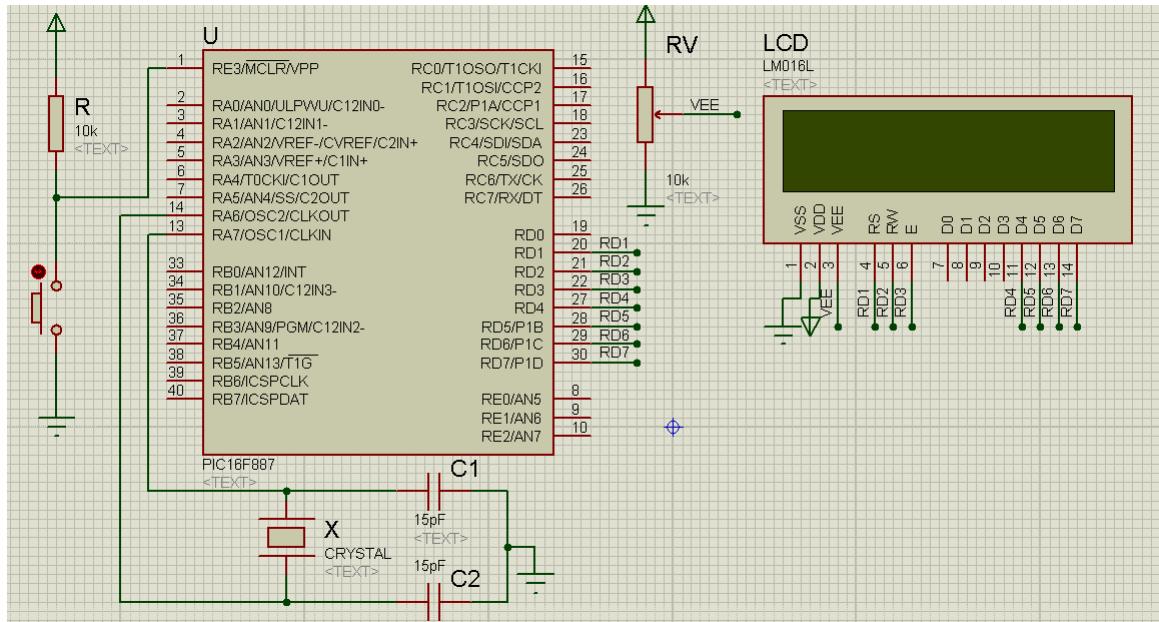
Chức năng các chân của LCD:

STT	Ký hiệu	Mô tả	Giá trị
1	VSS	GND	0V
2	VCC		5V
3	VEE	Tùy chỉnh độ tương phản	
4	RS	Lựa chọn thanh ghi	RS = 0: ghi lệnh RS = 1: ghi dữ liệu
5	R/W	Chọn thanh ghi đọc/viết dữ liệu	R/W = 0: viết dữ liệu R/W = 1: đọc dữ liệu
6	E	Enable	
7 – 10 11 – 14	DB0 – BD3 DB4 – DB7	Chân chuyền dữ liệu	8 bit từ $DB0 \rightarrow DB7$
15	A	Cực dương của LED nền	0 – 5V
16	K	Cực âm của LED nền	0V

Bảng 2.2: Sơ đồ chân và chức năng các chân của LCD

Ngoài sử dụng LCD, người ta còn sử dụng các LED 7 đoạn, LED ma trận để hiển thị dữ liệu (với LED 7 đoạn có trong *phụ lục D trang 94*).

2.1.3 Sơ đồ mạch



Hình 2.1: Mạch điều khiển LCD

2.2 Sử dụng các lệnh cơ bản cho LCD

Sử dụng thư viện LCD_LIB_4BIT.C (trong *phụ lục trang 92*) để điều khiển LCD. Các bước cơ bản để bắt đầu làm việc với LCD:

- Thêm thư viện LCD vào chương trình: #include<LCD_LIB_4BIT.C>
 - Chúng ta dùng LCD ở chế độ ghi, dùng lệnh: OUTPUT_LOW(LCD_RW); (trong bảng 2.2 trang 11).
 - Khởi tạo LCD, dùng hàm: LCD_Init();
 - Chức năng của một số hàm có trong thư viện được sử dụng:
 - Hàm LCD_Init(); Khởi tạo LCD.
 - Hàm LCD_PutCmd(unsigned int cX); Gửi lệnh lên LCD.
- Ví dụ: LCD_PutCmd(0x01); – lệnh xóa màn hình.

- Hàm `LCD_PutChar(unsigned int cX);` Ghi một chuỗi hoặc một ký tự lên LCD.
- Hàm `LCD_SetPosition(unsigned int cX);` Thiết lập vị trí con trỏ.
 - * Dòng 1: bắt đầu từ vị trí 0x00, tăng giá trị này lên để đến các vị trí khác trên dòng 1.
 - * Dòng 2: bắt đầu từ vị trí 0x40, tăng giá trị này lên để đến các vị trí khác trên dòng 2.
- Cách sử dụng hàm `printf(tham số):`
 - Xuất chuỗi, ký tự: `printf("Chuỗi, ký tự cần xuất");`
 - Xuất số nguyên: `printf("N = %d", n);` số nguyên ngắn dùng `%d`, số nguyên dài dùng `%lu`
 - Xuất số thực: `printf("A = %.2f", a);` quy định số chữ số sau dấu phẩy (trong ví dụ: quy định 2 chữ số sau dấu phẩy).
- Các cấu trúc: `while, for, if, if...else,...`

2.3 Bài tập

2.3.1 Bài tập 2.1

Yêu cầu Viết chương trình hiển thị các ký tự sau lên LCD: DAI HOC KTCN CAN THO

Hướng giải quyết

- Thiết lập LCD ở chế độ ghi: `OUTPUT_LOW(LCD_RW);`
- Khởi tạo LCD: `LCD_Init();`
- Xóa màn hình: `LCD_PutCmd(0x01);`
- Ghi ký tự lên LCD:
 - Cài đặt vị trí con trỏ: ta chọn vị trí thứ 6 – `LCD_SetPosition(0x05);` để ghi chuỗi "DAI HOC" trên dòng 1.
 - Chọn vị trí thứ 3 – `LCD_SetPosition(0x42);` để ghi chuỗi "KTCN CAN THO" trên dòng 2.
 - Ghi chuỗi: `LCD_PutChar("DAI HOC");` và `LCD_PutChar("KTCN CAN THO");`

Kết quả



Hình 2.2: Kết quả hiển thị ký tự lên LCD 16x02

Chương trình 4

```
1 /*Yeu cau
2     Chuong trinh hien thi chuoi "DAI HOC KTCN CAN THO" len LCD
3 */
4 //Ten file BAI-2-1.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #fuses HS,NOWDT, NOPROTECT, NOLVP
8 #use delay(clock=20000000) //Tan so thach anh 20MHz
9 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
10
11 void main(){
12     OUTPUT_LOW(LCD_RW); //Che do ghi
13     LCD_Init();          //Khoi tao LCD
14
15     LCD_PutCmd(0x01);   //Xoa man hinh
16
17     LCD_SetPosition(0x05); //Cot 6 dong 1
18     LCD_PutChar("DAI HOC");
19     LCD_SetPosition(0x42); //Cot 3 dong 2
20     LCD_PutChar("KTCN CAN THO");
21 }
```

2.3.2 Bài tập 2.2

Yêu cầu Viết chương trình hiển thị trên LCD theo yêu cầu sau: Hàng thứ nhất hiển thị họ tên sinh viên; hàng thứ hai hiển thị mã số sinh viên.

Hướng giải quyết Sử dụng lại *chương trình 4* của *bài tập 2.1* với các thay đổi: vị trí con trỏ và chuỗi ký tự để phù hợp với yêu cầu.

Kết quả



Hình 2.3: Kết quả hiển thị họ tên và mã số sinh viên lên LCD 16x02

Chương trình 5

```
1 /*Yeu cau
2  Chuong trinh hien thi len LCD:
3   - Dong 1: Ho ten sinh vien
4   - Dong 2: Ma so sinh vien
5 */
6 //Ten file BAI-2-2.C
7 #include<16f887.h>
8 #include<def_887.h>
9 #fuses HS,NOWDT, NOPROTECT, NOLVP
10 #use delay(clock=20000000) //Tan so thach anh 20MHz
11 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
12
13 void main(){
14     OUTPUT_LOW(LCD_RW); //Che do ghi
15     LCD_Init();          //Khoi tao LCD
16
```

```

17 LCD_PutCmd(0x01); //Xoa man hinh
18 LCD_SetPosition(0x01); //Cot 2 dong 1
19 LCD_PutChar("Thi Minh Nhut");
20 LCD_SetPosition(0x44); //Cot 5 dong 2
21 LCD_PutChar("1350366");
22 }

```

2.3.3 Bài tập 2.3

Yêu cầu Viết chương trình đếm số từ 0 đến 999 hiển thị lên LCD.

Hướng giải quyết

- Phân khai báo giống như *bài tập 2.1*
- Tăng giá trị số đếm: ban đầu ta gán biến đếm là `count = 0`; rồi sử dụng cấu trúc `for` để tăng giá trị biến đếm lên (`count++;`)
- Do cần hiển thị số lên LCD nên ta dùng hàm `LCD_PutChar` kết hợp với hàm `printf` để làm việc này.
 - Hiển thị số nguyên: ví dụ `unsigned long n = 12345` thì dùng:
`printf(LCD_PutChar,"%lu",n);`
 - Hiển thị số thực: ví dụ `float n = 1.2345` thì dùng:
`printf(LCD_PutChar,"% .4f",n);`
- Để hiển thị giá trị biến đếm lên LCD, dùng:
`printf(LCD_PutChar,"%lu",count);`
 khi `count` vượt quá giá trị, đưa biến `count = 0`;

Kết quả



Hình 2.4: Kết quả chương trình đếm số hiển thị lên LCD 16x02

Chương trình 6

```
1 /*Yeu cau
2     Chuong trinh hien dem so tu 0 - 999 hien thi len LCD
3 */
4 //Ten file BAI-2-3.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #fuses HS,NOWDT, NOPROTECT, NOLVP
8 #use delay(clock=20000000) //Tan so thach anh 20MHz
9 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
10
11 void main(){
12     unsigned long count,i, N_max = 999; //Gia tri dem gioi han
13     int16 sleep = 200; //Thoi gian delay (ms)
14
15     OUTPUT_LOW(LCD_RW); //Che do ghi
16     LCD_Init(); //Khởi tạo LCD
17
18     LCD_PutCmd(0x01); //Xoa man hinh
19
20     LCD_SetPosition(0x00); //Cot 1 dong 1
21     LCD_PutChar("Dem so:");
22     while (True){
23         count = 0;
24         for (i = 0; i<= N_max; i++){
25             LCD_SetPosition(0x07); //Cot 8 dong 1
26             printf(LCD_PutChar,"%lu",count);
27
28             delay_ms(sleep);
29             count++;
30
31             if (count > N_max){ //Reset lai bien count
32                 count = 0;
33                 //Thông báo Reset
34                 LCD_SetPosition(0x46); //Cot 7 dong 2
35                 LCD_PutChar("Reset!");
36                 delay_ms(2000);
37
38                 //Reset man hanh
39                 LCD_PutCmd(0x01); //Xoa man hinh
40                 LCD_SetPosition(0x00); //Cot 1 dong 1
41                 LCD_PutChar("Dem so:");
42             }
43         }
44     }
45 }
```

BÀI 3

XỬ LÝ NGẮT

3.1 Giới thiệu chung

3.1.1 Yêu cầu

Viết chương trình xử lý ngắt với nút nhấn.

3.1.2 Hoạt động ngắt

Ngắt là một tín hiệu điều khiển bắt vi điều khiển tạm ngưng công việc đang thực hiện để tiến hành các thao tác khác do ngắt quy định qua chương trình ngắt.

Khi phát hiện ngắt thì vi điều khiển sẽ thực hiện một chương trình độc lập với chương trình chính gọi là chương trình ngắt.

Cấu trúc của một chương trình ngắt:

- Bắt đầu là tên ngắt: #INT_tên_ngắt với tên_ngắt ta có thể xem trong chương trình CCS (trong View/Valid Interrupts).
- Kế tiếp là *chương trình ngắt* (tùy theo mục đích mà chúng ta viết chương trình hoạt động khi xảy ra ngắt).

```
tên_hàm(){  
    //Nội dung chương trình ngắt  
}
```

3.2 Thiết lập hoạt động ngắt

Sử dụng các lệnh sau để thiết lập hoạt động ngắt:

- ENABLE_INTERRUPTS(level); với level là INT_tên_ngắt hoặc GLOBAL (ngắt toàn cục): cho phép ngắt.
- DISABLE_INTERRUPTS(level); với level giống như trên: vô hiệu hóa ngắt.

- CLEAR_INTERRUPT(level); với level không có GLOBAL: xóa cờ ngắt.
- EXT_INT_EDGE(soure, edge); với:
 - soure = 0, 1, 2: nguồn ngắt (ứng với EXT0, EXT1, EXT2).
 - edge = L_TO_H, H_TO_L: cách kích ngắt (mức thấp lên cao hoặc mức cao xuống thấp).

Thiết kế chương trình có dùng ngắt:

- Trong hàm main(): cho phép ngắt cụ thể (tên_ngắt), ngắt toàn cục (GLOBAL) và đợi ngắt (EXT_INT_EDGE).
- Chương trình xử lý ngắt (đặt trước hàm main()): xóa cờ ngắt (CLEAR_INTERRUPT), cấm ngắt toàn cục (DISABLE_INTERRUPTS(GLOBAL)); xử lý chương trình ngắt xong rồi cho phép ngắt toàn cục lại (ENABLE_INTERRUPTS(GLOBAL));

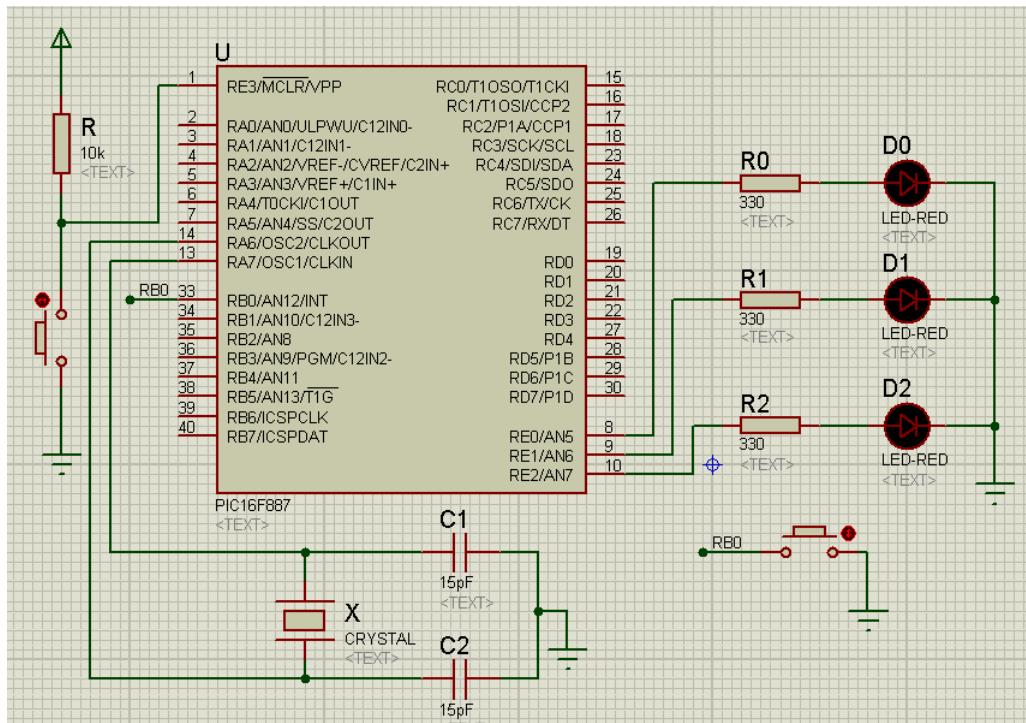
3.3 Bài tập

3.3.1 Bài tập 3.1

Yêu cầu Viết chương trình nhận nút nhấn ở chân B0 của PIC 16F887, cứ mỗi lần nhấn phím sẽ đảo trạng thái các LED ở PORT E.

Hướng giải quyết

- Chân B0 cho phép ngắt ngoài, nên chúng ta sử dụng #INT_EXT.
- Chương trình ngắt: thiết kế như hướng dẫn ở mục 3.2, nội dung ngắt là đảo trạng thái PORT E: PORTE = ~PORTE;
- Chương trình chính:
 - Khai báo PORT B là chân INPUT (nút nhấn): TRISB = 0xFF; còn PORT E là chân OUTPUT (led): TRISE = 0x00;
 - Ở PORT B khi giao tiếp nút nhấn, cần có điện trở mắc lên nguồn cho chân B0, ta khai báo: PORT_B_PULLUPS(0x01);
 - Kích hoạt ngắt ngoài: ENABLE_INTERRUPTS(INT_EXT);
 - Chọn cạnh ngắt: EXT_INT_EDGE(H_TO_L);
 - Kích hoạt ngắt toàn cục ENABLE_INTERRUPTS(GLOBAL);
 - Duy trì hoạt động của vi điều khiển: dùng while.



Hình 3.1: Mạch đảo trạng thái LED ở PORT E với ngắt ngoài

Sơ đồ mạch

Chương trình 7

```

1 /*Yeu cau:
2     Chuong trinh dao trang thai cac LED o PORT E voi ngat ngoai
3 */
4 //Ten file: BAI-3-1.C
5 #include<16F887.h> //Khai bao ten PIC
6 #include<def_887.h> //Thu vien do nguoi dung dung nghia
7
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
9
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz
11
12 #INT_EXT //Ngat ngoai - chan B0
13 void NgatNgoai(){
14     CLEAR_INTERRUPT(INT_EXT); //Xoa co ngat
15     DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc
16
17     //Xu ly chuong trinh ngat
18
19     PORTE = ~PORTE; //Dao trang thai PORT E
20

```

```

21     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
22 }
23
24 void main(){
25     TRISB = 0xFF; //Chan INPUT
26     TRISE = 0x00; //Chan OUTPUT
27
28     PORT_B_PULLUPS(0x01); //Noi dien tro len nguon cho B0
29
30     PORTE = 0x00; //Set gia tri ban dau la muc 0
31
32     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai
33     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap
34     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
35
36     while (True){ //Duy tri hoat dong cua vi dieu khien
37         ;
38     }
39 }
```

3.3.2 Bài tập 3.2

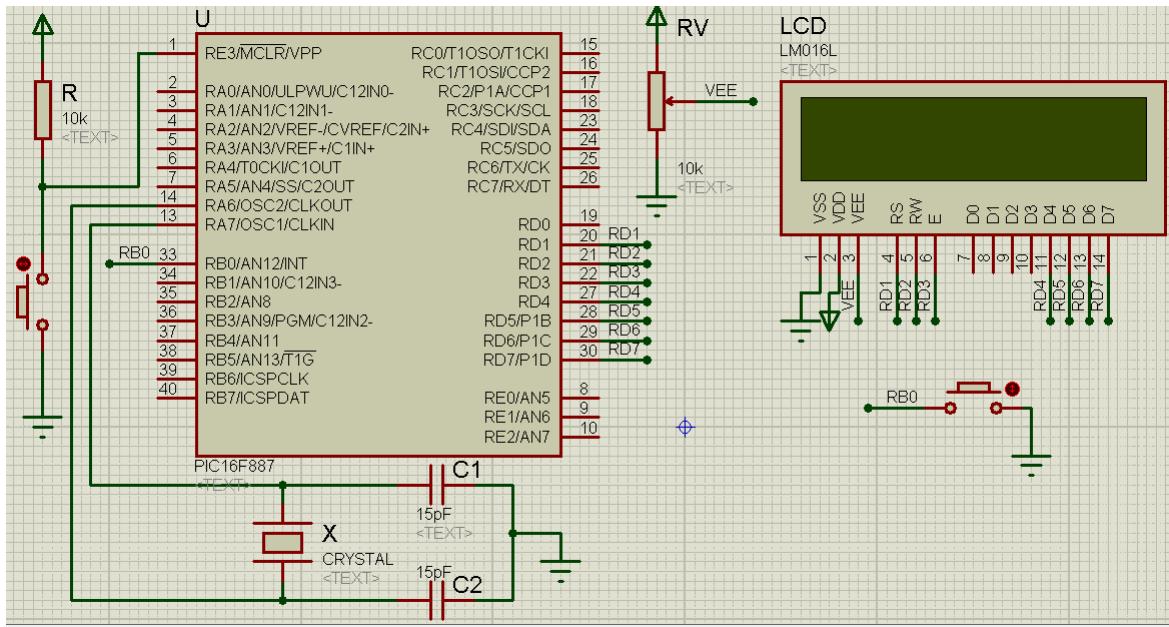
Yêu cầu Viết chương trình hiển thị số lần nhấn phím ở chân B0 của PIC 16F887 lên màn hình LCD 16x02.

Hướng giải quyết

- Sử dụng lại *chương trình 7* với một số thay đổi như sau:
 - Khai báo biến **count** là biến toàn cục (để ảnh hưởng đến toàn chương trình).
 - Thay lệnh **PORTE = ~PORTE** bằng lệnh **count++**.
- Trong chương trình chính ta thực hiện:
 - Khai báo LCD (được trình bày trong *chương trình 6* của *bài tập 2.3*).
 - Sử dụng vòng lặp **while** để duy trì chương trình: trong vòng lặp cho hiển thị biến **count** lên LCD bằng hàm **printf** kết hợp với hàm **LCD_PutChar**.
- Với các khai báo trên thì chương trình sẽ xảy ra nhiều (do dội phím, rung do phần cứng), khi đó phải khắc phục nhiều theo *bài tập 3.4 trang 25*.
 - * *Lưu ý*: Khi chúng ta chưa xử lý chông nhiễu thì nó sẽ xảy ra một ngắt không mong muốn.

- *Hướng giải quyết khác cho bài này:* Ta không dùng ngắn, mà thay vào đó dùng hàm `input(PIN_B0)` để đọc tín hiệu từ nút nhấn. Rồi xử lý chống nhiễu bằng cách thêm hàm `delay_ms(500)` vào lệnh `if` sau khi đọc được nút nhấn.
 - * Trong bài này chúng ta không sử dụng hàm `input` trong *chương trình chính* là do nội dung của bài thực hành là khảo sát ngắn trên vi điều khiển PIC 16F887, nên chọn hàm `input` để sử dụng thì không hợp với nội dung.
 - * Để so sánh chương trình dùng ngắn và chương trình dùng `input`, ta cùng xem chương trình trong *mục 3.3.5 trang 26*.

Sơ đồ mạch



Hình 3.2: Mạch đọc số lần nhấn nút ở chân B0 hiển thị lên LCD

Chương trình 8

```
1 /*Yeu cau:  
2     Chuong trinh dem so lan nhat nut hien thi len LCD su dung ngat  
3     */  
4 //Ten file: BAI-3-2.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
7  
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
10 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao  
11  
12 unsigned long count = 0;  
13  
14 #INT_EXT //Ngat ngoai - chan B0  
15 void NgatNgoai(){  
16     CLEAR_INTERRUPT(INT_EXT); //Xoa co ngat  
17     DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc  
18  
19     //Xu ly chuong trinh ngat  
20     delay_ms(20); //Chong nhieu  
21     if (INPUT(PIN_B0) == 0){  
22         count++; //Tang gia tri bien dem  
23     }  
24  
25     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc  
26 }  
27  
28 void main(){  
29     TRISB = 0xFF; //Chan INPUT  
30     PORT_B_PULLUPS(0x01); //Noi dien tro len nguon  
31  
32     OUTPUT_LOW(LCD_RW); //Che do ghi  
33     LCD_Init(); //Khoi tao LCD  
34     LCD_PutCmd(0x01); //Xoa man hinh  
35     LCD_SetPosition(0x00);  
36     LCD_PutChar("Count= ");  
37     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai  
38     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap  
39     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc  
40  
41     while (True){ //Duy tri hoat dong cua vi dieu khien  
42         LCD_SetPosition(0x06);  
43         printf(LCD_PutChar, "%lu", count);  
44     }  
45 }
```

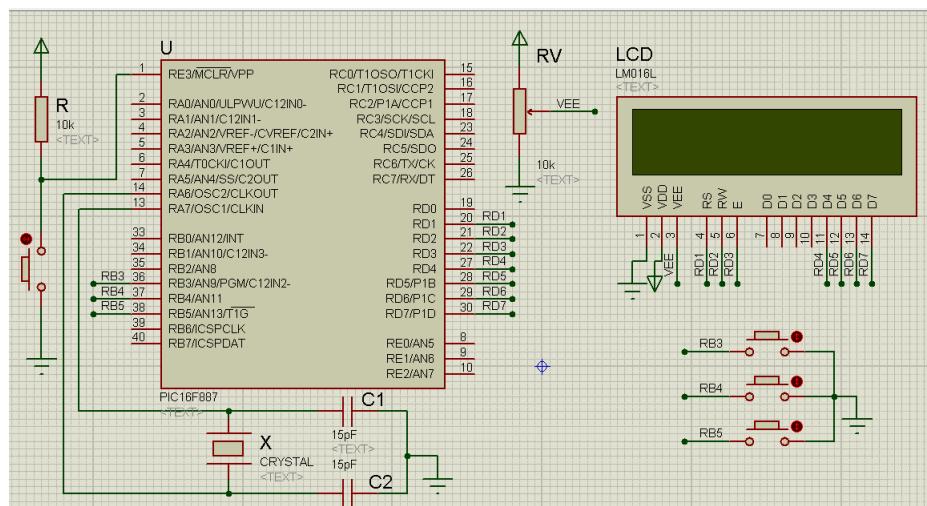
3.3.3 Bài tập 3.3

Yêu cầu Viết chương trình nhận nút nhấn ở chân B3 – B5 của PIC 16F887, hiển thị lên màn hình LCD 16x02.

Hướng giải quyết

- Ở PIC 16F877A thì sử dụng các ngắt nối tiếp (từ chân RB4 – RB7) dễ hơn nhiều so với PIC 16F887.
- Để đơn giản, chúng ta không sử dụng ngắt nối tiếp mà dùng cách đọc tín hiệu từ các chân B3 – B5 thông qua hàm input rồi cho xuất ra LCD.
- Tạo một mảng: {B3, B4, B5} = {3, 4, 5}, nếu nhấn nút nhấn ở chân B3 thì xuất ra biến kt = 3, tương tự cho các chân còn lại.
- Kiểm tra giá trị của biến kt rồi cho xuất ra LCD thông qua hàm `printf(LCD_PutChar)`.
- Sử dụng điện trở nội cho các chân ở PORT B: `PORT_B_PULL(0x38)`; chỉ có 3 chân B3, B4, B5 được mắc trờ lên nguồn.
- * *Kết luận:* Phần trên là ý tưởng giải quyết bài tập của em, nhưng khi chạy thực tế, vì lý do nào đó mà chân RB3 không lên mức cao được! Vấn đề này em chưa giải quyết được.

Sơ đồ mạch



Hình 3.3: Mạch đọc nút nhấn từ chân B3 – B5 và hiển thị lên LCD

3.3.4 Bài tập 3.4

Yêu cầu Viết chương trình nhận nút nhấp ở chân B0 của PIC 16F887 có xử lý chống nhiễu.

Hướng giải quyết

- Khi sử dụng nút nhấn, có xảy ra quá trình dội, nhiễu và rung do phần cứng. Cụ thể là ở bài tập 3.1 khi ta nhấn nút thì LED bị nhiễu. Nên ta cần xử lý chống nhiễu cho tín hiệu.
- Dùng hàm `delay_ms(số mili giây)` với `số mili giây = 10 - 20ms` để bỏ qua xung nhiễu. Rồi lấy mức 0 làm điều kiện có nút nhấn:
`if input(PIN_B0) == 0` thì thực hiện lệnh cần thiết.

Chương trình 9

```
1 /*Yeu cau:
2  Chuong trinh dao trang thai cac LED o PORT E voi xu ly ngat va chong nhieu
3  */
4 //Ten file: BAI-3-4.C
5 #include<16F887.h> //Khai bao ten PIC
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia
7
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
9
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz
11
12 #INT_EXT //Ngat ngoai - chan B0
13 void NgatNgoai(){
14     CLEAR_INTERRUPT(INT_EXT); //Xoa co ngat
15     DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc
16
17     //Xu ly chuong trinh ngat
18     delay_ms(20);
19     if (INPUT(PIN_B0) == 0){
20         PORTE = ~PORTE; //Dao trang thai PORT E
21     }
22     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
23 }
24
25 void main(){
26     TRISB = 0xFF; //Chan INPUT
27     TRISE = 0x00; //Chan OUTPUT
28
29     PORT_B_PULLUPS(0x01); //Noi dien tro len nguon
```

```

30     PORTE = 0x00; //Set gia tri ban dau la muc 0
31
32     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai
33     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap
34     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
35
36     while (True){ //Duy tri hoat dong cua vi dieu khien
37         ;
38     }
39 }
```

3.3.5 Đọc tín hiệu từ nút nhấn với hàm INPUT

Nội dung file INPUT_BUTTON.C (Một cách làm khác của bài tập 3.2 trang 21).

Chương trình 10

```

1 /*Yeu cau:
2     Chuong trinh dem so lan nhat nut hien thi len LCD su dung ngat
3 */
4 //Ten file: INPUT_BUTTON.C
5 #include<16F887.h> //Khai bao ten PIC
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia
7
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz
10 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
11
12 unsigned long count = 0;
13
14 void main(){
15     TRISB = 0xFF; //Chan INPUT
16     PORT_B_PULLUPS(1); //Noi dien tro len nguon
17
18     OUTPUT_LOW(LCD_RW); //Che do ghi
19     LCD_Init(); //Khoi tao LCD
20
21     LCD_PutCmd(0x01); //Xoa man hinh
22     LCD_SetPosition(0x00); //Cot 1 dong 1
23     LCD_PutChar("Count= ");
24
25     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai
26     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap
27     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
28
29     while (True){ //Duy tri hoat dong cua vi dieu khien
```

```
30     LCD_SetPosition(0x06); //Cot 7 dong 1
31     printf(LCD_PutChar,"%lu",count);
32     if (INPUT(PIN_B0) == 0){//Da nhan nut nhan
33         delay_ms(500);
34         count++;
35     }
36 }
37 }
```

BÀI 4

XỬ LÝ ADC

4.1 Giới thiệu chung

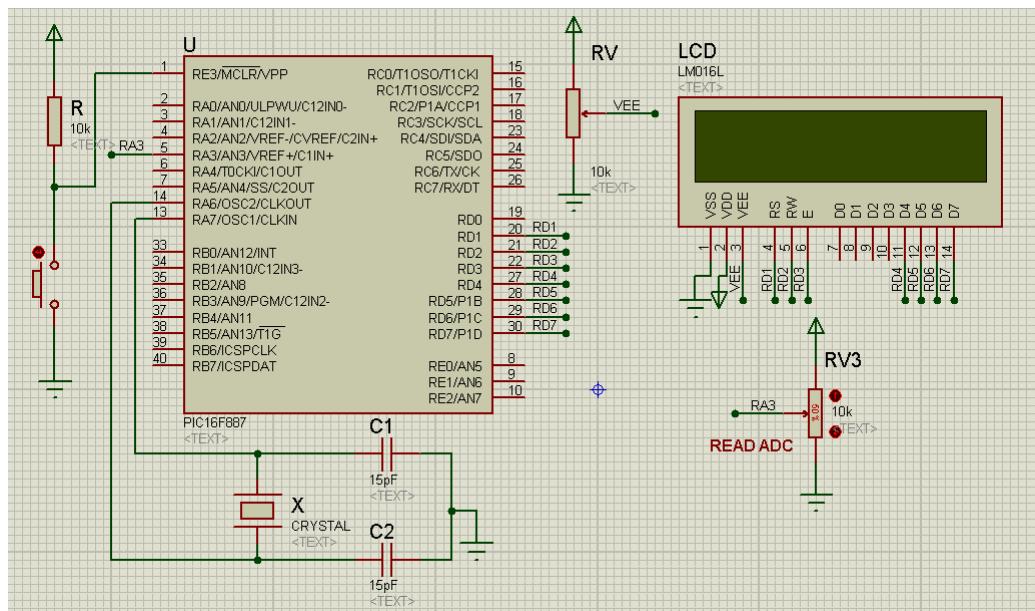
4.1.1 Yêu cầu

Đọc giá trị ADC từ biến trở.

4.1.2 ADC

PIC 16F887 có 2 PORT hỗ trợ tính năng ADC là: PORT A và PORT E. Có thể đọc ADC ở chế độ 10 bit (0 – 1023) hoặc 8 bit (0 – 255) tùy mục đích sử dụng. Điện áp tham chiếu của bộ ADC thường là 5V.

4.1.3 Sơ đồ mạch



Hình 4.1: Mạch đọc ADC từ biến trở

4.2 Cấu hình ADC

- Khai báo chế độ đọc ADC là 10 hoặc 8 bit:
`#DEVICE *= 16 ADC 10 hoặc #DEVICE *= 16 ADC 8`
- Xác định cách thức hoạt động của bộ biến đổi ADC: `SETUP_ADC(mode)`;
- Xác định chân lấy tín hiệu ADC và điện thế sử dụng: `SETUP_ADC_PORTS(value)`;
- Chọn chân để đọc Analog với lệnh `READ_ADC`, ta khai báo:
`SET_ADC_CHANNEL(channel);` với `channel` có giá trị từ 0 – 7 theo thứ tự A0 – A5; E0 – E2. Sau hàm này chúng ta nên dùng `delay_us(10)` để cho kết quả đúng.
- Đọc giá trị ADC từ chân đã khai báo ở hàm `SET_ADC_CHANNEL` với lệnh `READ_ADC(mode)`; với `mode` không bắt buộc.
 - * Các tham số (`mode`, `value` và `channel`) của những hàm trên được định nghĩa trong thư mục `DEVICES` ví dụ: `16F887.h`, cách sử dụng những hàm này có thể xem trong phần Help của phần mềm CCS.

4.3 Bài tập

4.3.1 Bài tập 4.1

Yêu cầu Viết chương trình đọc giá trị ADC từ biến trỏ R7 và hiển thị lên LCD.

Hướng giải quyết

- Khởi tạo LCD (trình bày trong *chương trình 4 – bài tập 2.1* của *bài 2*).
- Cấu hình bộ ADC:
 - Khai số bit đọc ADC, ta chọn 10 bit: `#device *= 16 ADC 10`
 - Xác định hoạt động của bộ ADC, chọn *thời gian lấy mẫu bằng xung clock*: `SETUP_ADC(ADC_CLOCK_INTERNAL)`;
 - Xác định chân đọc ADC, dùng *chân A3*: `SET_ADC_CHANNEL(3)`; và `delay_us(10)` (để bảo đảm giá trị đọc chính xác).
- Thực hiện vòng lặp (dùng `while`) đọc giá trị ADC (dùng `READ_ADC`) và cho hiển thị giá trị lên LCD (dùng hàm `printf` kết hợp với hàm `LCD_PutChar`).

Kết quả



Hình 4.2: Kết quả chương trình ADC hiển thị lên LCD 16x02

Chương trình 11

```
1 /*Yeu cau
2     Chuong trinh doc ADC tu bien tro va hien thi len LCD
3 */
4 //Ten file BAI-4-1.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #device *= 16 ADC = 10
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
11
12 void main(){
13     int16 adc;
14
15     OUTPUT_LOW(LCD_RW); //Che do ghi
16     LCD_Init();          //Khoi tao LCD
17
18     LCD_PutCmd(0x01); //Xoa man hinh
19
20     //Xac dinh cach thuc hoat dong cua bo ADC
21     //Thoi gian lay mau bang xung clock
22     SETUP_ADC(ADC_CLOCK_INTERNAL);
23     SET_ADC_CHANNEL(3);    //Su dung chan A3 doc ADC
24     delay_us(10);         //Dam bao doc gia gia ADC chinh xac
25
26     while (True){
27         LCD_PutCmd(0x01);
```

```

28     adc = READ_ADC(); //Doc ADC tu chan A3
29     LCD_SetPosition(0x00); //Cot 1 dong 1
30     LCD_PutChar("Read ADC:");
31     printf(LCD_PutChar, "%lu", adc); //Hien thi len LCD
32     delay_ms(1000);
33 }
34 }
```

4.3.2 Bài tập 4.2

Yêu cầu Viết chương trình đọc giá trị ADC từ biến trỏ R7 và xuất ra giá trị các LED ở PORT E.

Hướng giải quyết

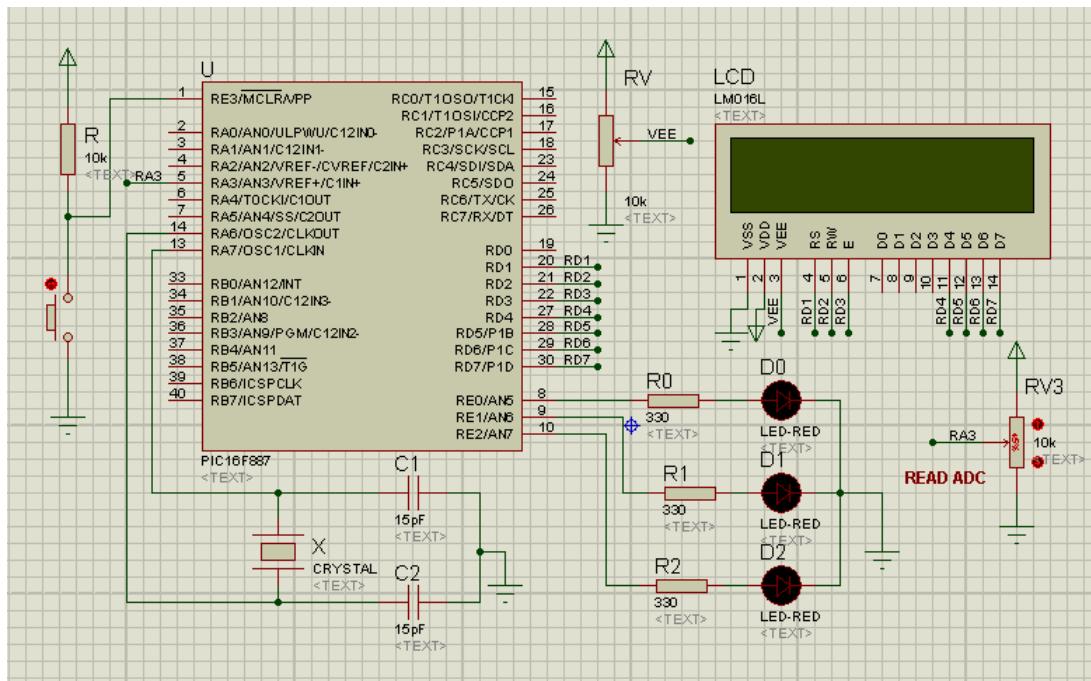
- Đọc giá trị ADC như ở *bài tập 4.1*.
- Sau khi đọc được giá trị ADC từ biến trỏ, dùng hàm `OUTPUT_E(value)` ; với `value` là giá trị ADC đọc được.
 - Giá trị ADC chúng ta đọc được trên LCD là ở dạng số nguyên, nên muốn hiểu rõ trạng thái của LED cần đổi giá trị ADC đọc được sang mã nhị phân.
 - Do PORT E chỉ có 3 chân, nên chỉ quan tâm 3 số cuối của mã nhị phân.
Ví dụ: $ADC = 511 = 0b01\ 1111\ 1111$ (10 bit), chúng ta qua tâm số giá trị cuối của mã nhị phân là 111, khi đó cả 3 LED đều sáng.
- Viết thêm lệnh chuyển từ số thập phân sang số nhị phân cho hiển thị lên LCD để dễ quan sát.

Kết quả



Hình 4.3: Kết quả chương trình ADC xuất ra PORT E

Sơ đồ mạch



Hình 4.4: Mạch đọc ADC từ biến trở xuất giá trị ra LED

Chương trình 12

```
1 /*Yeu cau
2     Chuong trinh doc ADC tu bien tro va xuat ra LED
3 */
4 //Ten file BAI-4-2.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #device *= 16 ADC = 10
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
11
12 void main(){
13     int16 adc, a[10];
14     int count, i, pos;
15
16     TRISE = 0x00;
17     OUTPUT_LOW(LCD_RW); //Che do ghi
18     LCD_Init(); //Khoi tao LCD
19
20     LCD_PutCmd(0x01); //Xoa man hinh
21
22     //Xac dinh cach thuc hoat dong cua bo ADC
23     //Thoi gian lay mau bang xung clock
24     SETUP_ADC(ADC_CLOCK_INTERNAL);
25     SET_ADC_CHANNEL(3); //Su dung chan A3 doc ADC
26     delay_us(10); //Dam bao doc gia gia ADC chinh xac
27
28     while (True){
29         adc = READ_ADC(); //Doc ADC tu chan A3
30         OUTPUT_E(adc); //Dua gia tri ADC vao cong E
31
32         LCD_SetPosition(0x00); //Cot 1 dong 1
33         LCD_PutChar("Read ADC:");
34         LCD_SetPosition(0x09); //Cot 10 dong 1
35         printf(LCD_PutChar, "%lu", adc); //Hien thi len LCD
36
37         //Chuyen sang nhi phan hien thi LCD
38         count = 0;
39
40         //Chuyen sang ma nhi phan
41         while (adc > 0){ //Con chia duoc
42             a[count] = adc%2;
43             adc = adc/2;
44             count++;
45         }
46 }
```

```

47
48     //Hien len LCD
49     LCD_SetPosition(0x40);
50     LCD_PutChar("Bin:"); //Gia tri nhi phan
51     LCD_SetPosition(0x44);
52     pos = 68; //0x44 = 64
53     for (i = count - 1; i>=0; i--){
54         printf(LCD_PutChar,"%lu",a[i]);
55         pos++;
56         LCD_SetPosition(pos);
57         if (i == 0){
58             break;
59         }
60     }
61     delay_ms(1000);
62 }
63 }

```

4.3.3 Bài tập 4.3

Yêu cầu Viết chương trình đọc ADC từ biến trở và gửi lên máy tính.

Hướng giải quyết

- Phần code cho vi điều khiển:

- Để gửi dữ liệu từ vi điều khiển lên PC, chúng ta sử dụng chuẩn giao tiếp RS232 (được giới thiệu trong bài 7 trang 60).
- Thêm vào khai báo sau để giao tiếp RS232:


```
#use rs232(baud = 9600, xmit = PIN_C6, rcv = PIN_C7)
```
- Sau khi đọc được giá trị từ biến trở (giống như bài tập 4.1), chúng ta gửi dữ liệu lên máy tính bằng lệnh: `printf("%lu", adc)`.
- Sử dụng lại chương trình 11 của bài tập 4.1 cho hiển thị lên LCD để kiểm chứng lại kết quả trên LCD và kết quả gửi lên máy tính.

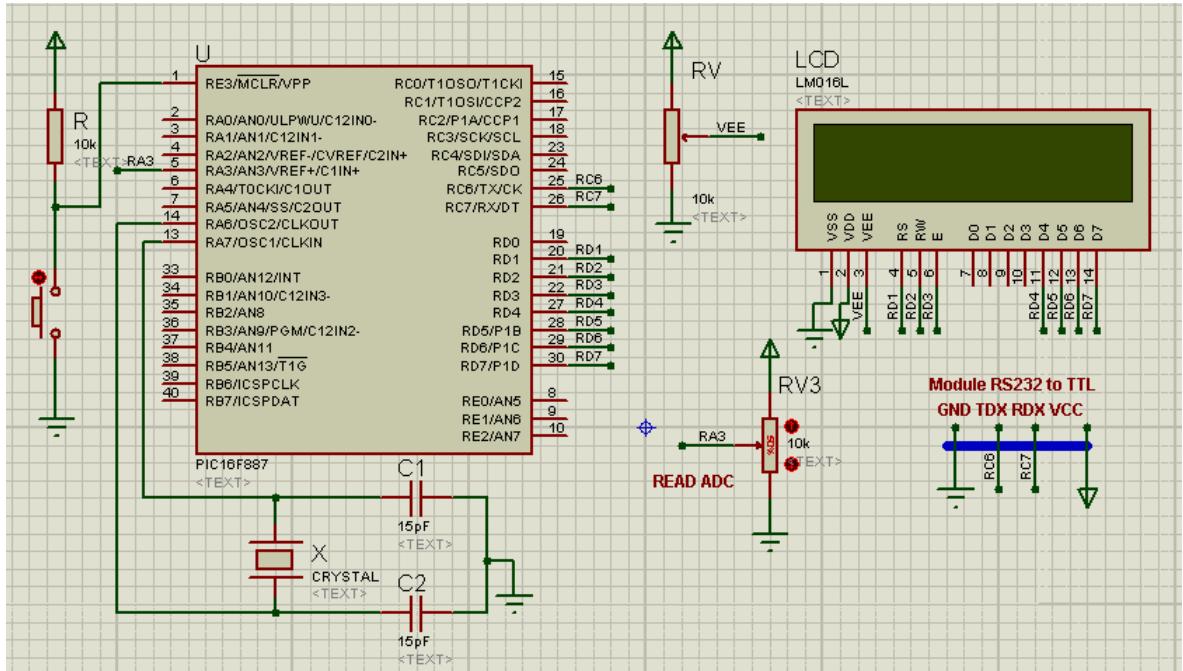
- Phần code cho Matlab¹

- Dữ liệu do vi điều khiển gửi lên máy tính sẽ được lưu trong bộ nhớ đệm, để đọc giá trị từ bộ nhớ đệm ra ta phải thiết lập cho tham số BytesAvailableFcn.
- Khi có một byte được nhận ở bộ nhớ đệm thì tham số này sẽ được gọi.

¹Tham khảo tại: <http://www.picvietnam.com/forum/showthread.php?t=752>

- Do đó cần viết hàm (hàm `Serial_Callback` trong file `Serial_Callback.m`) để đáp ứng sự kiện trên và cho hiển thị trực tiếp lên cửa sổ command line.

Sơ đồ mạch



Hình 4.5: Mạch đọc ADC từ biến trở gửi lên PC

Chương trình 13

```

1 /*Yeu cau
2   Chuong trinh doc ADC tu bien tro va gui len may tinh
3 */
4 //Ten file BAI-4-3.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #device *= 16 ADC = 10
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #use rs232(baud = 9600, xmit = PIN_C6, rcv = PIN_C7)
11 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
12
13 void main(){
14     int16 adc;

```

```

15     OUTPUT_LOW(LCD_RW);    //Che do ghi
16     LCD_Init();           //Khoi tao LCD
17
18     LCD_PutCmd(0x01);     //Xoa man hinh
19
20     //Xac dinh cach thuc hoat dong cua bo ADC
21     //Thoi gian lay mau bang xung clock
22     SETUP_ADC(ADC_CLOCK_INTERNAL);
23     SET_ADC_CHANNEL(3);   //Su dung chan A3 doc ADC
24     delay_us(10);         //Dam bao doc gia gia ADC chinh xac
25
26     while (True){
27         LCD_PutCmd(0x01);
28         adc = READ_ADC();           //Doc ADC tu chan A3
29         printf("%lu\n",adc);        //Gui gia tri do duoc len PC
30         LCD_SetPosition(0x00);      //Cot 1 dong 1
31         LCD_PutChar("Read ADC:");
32         printf(LCD_PutChar,"%lu",adc); //Hien thi len LCD
33         delay_ms(1000);
34     }
35 }

```

Hàm Serial_Callback dùng để thiết lập cho tham số BytesAvailableFcn trước khi mở cổng COM:

```

1 %%Ten file: Serial_Callback.m
2 function Serial_Callback(obj,event)
3     ind = fscanf(obj)

```

Lệnh Matlab nhận dữ liệu từ vi điều khiển gửi lên:

```

1 s = serial('COM6'); %Thay doi gia tri cong COM
2 %Thiet lap gia tri cho tham so BytesAvailableFcn de hien thi gia tri do vi
3 % dieu khien gui len
4 s.BytesAvailableFcn = @Serial_Callback;
5 fopen(s); %Mo cong COM de giao tiep
6 %%Du lieu duoc tu dong gui len va hien thi
7 %%Khi khong giao tiep, dong cong COM lai:
8 fclose(s);

```

BÀI 5

TIMER

5.1 Giới thiệu chung

5.1.1 Yêu cầu

Sử dụng timer/counter trên vi điều khiển PIC 16F887 tạo thời gian trễ và tạo bộ đếm.

5.1.2 Timer/Counter

Timer được dùng để tạo ra khoảng thời gian trễ và nó cũng hoạt động như một bộ đếm (đếm số xung đi vào một chân cụ thể trên vi điều khiển).

Vi điều khiển PIC 16F887 có 3 bộ timer:

- Timer0: 8 bit, đếm được 255, có chế độ định thời và bộ đếm.
- Timer1: 16 bit, đếm được 65535, có chế độ định thời và bộ đếm.
- Timer2: 8 bit, có chức năng điều PWM – điều chế độ rộng xung.

5.2 Các lệnh của Timer/Counter

Gồm các lệnh sau:

- `SETUP_TIMER_X(mode)`; với $X = 1$ hoặc $X = 2$ (được định nghĩa trong `device.h` (`device` là tên chip)): khởi tạo TIMER.
- `SET_TIMERx(value)`; với $x = 0$ hoặc $x = 1$ hoặc $x = 2$: thiết lập giá trị bắt đầu cho TIMER.
- `GET_TIMERx()`; với $x = 0$ hoặc $x = 1$ hoặc $x = 2$: đọc giá trị của TIMER/COUNTER.

Xét cho từng bộ Timer:

- TIMER0: ta có các lệnh

```
SETUP_TIMER_0(mode); SETUP_COUNTERS(rtcc_state, ps_state);
SET_TIMER0(value); GET_TIMER0();
```

- TIMER1: ta có các lệnh

```
SETUP_TIMER_1(mode); SET_TIMER1(value); GET_TIMER1();
```

- TIMER2: ta có các lệnh

```
SETUP_TIMER_2(mode, period, postscale); SET_TIMER2(value);
GET_TIMER2();
```

- * Các tham số (mode, value, rtcc_state, ps_state, period và postscale) của những hàm trên được định nghĩa trong thư mục DEVICES ví dụ: 16F887.h, cách sử dụng những hàm này có thể xem trong phần Help của phần mềm CCS.

5.3 Bài tập

5.3.1 Bài tập 5.1

Yêu cầu Viết chương trình chớp tắt LED ở PORT E với thời gian delay 1s sử dụng timer của PIC 16F887.

Hướng giải quyết

- Đây là chương trình tạo thời gian trễ bằng cách sử dụng timer.
- Chọn Timer0 8 bit để tạo thời gian trễ.
- Tiến hành tính toán giá trị ban đầu cho thanh ghi TMR:

Thông số	Ký hiệu	Giá trị	Bộ chia	Giá trị	TMR0
Tần số thạch anh	F	$20MHz$	2	2500	
Tần số lệnh	$F' = \frac{F}{4}$	$5MHz$	4	1250	
Số xung trong 1s		5×10^6	8	625	
Số xung trong 1ms		5×10^3	16	312.5	
			32	156.25	$256 - 156$
			64	78.125	$= 100$
			128	39.0625	
			256	19.53125	

- Trong $1ms$ thực hiện 5×10^3 xung, mà Timer0 chỉ đếm được tới 255. Nên chúng ta biến đổi số 5×10^3 qua các bộ chia trước.
- Trong đó bộ chia trước 32 là cho kết quả ít sai lệch so với các bộ chia khác. Chọn bộ chia 1 : 32.
- Thiết lập giá trị cho Timer0: bộ Timer0 có 256 giá trị, theo như cách chọn trên thì phải Timer0 phải đếm 156 lần mới được $1ms$, nên cần phải cài đặt giá trị bắt đầu cho Timer0 là: $256 - 156 = 100$:
`SET_TIMER0(100);`
- Sau khoảng thời gian đếm từ $100 \div 256$ thì Timer0 sẽ tràn, nên chúng ta sử dụng ngắt #INT_TIMER0 để xác định tràn trên Timer0.
- Để thời gian delay là $1s$, xác định số lần tràn như sau: $count = 1 \times 1000$ lần.
- Khai báo cho Timer0: chọn xung kích nội và bộ chia trước 32, giá trị ban đầu là 100 rồi đếm lên, nên: `SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_32);`
`SET_TIMER0(100);`
- Khai báo ngắt tràn trên Timer0: `ENABLE_INTERRUPTS(INT_TIMER0);` và thực hiện thiết kế chương trình ngắt như hướng dẫn ở mục 3.2 trong trang 18.
- Chương trình ngắt:
 - Mỗi lần tràn Timer0 ta tăng biến `count`.
 - So sánh biến `count` với 1000, nếu thỏa: `count = 0`; và đảo trạng thái PORT E: `PORT_E = ~PORT_E`.
- Thực hiện vòng lặp (dùng `while`) để giữ cho vi điều khiển hoạt động.

Sơ đồ mạch Sơ đồ mạch giống sơ đồ mạch của bài tập 1.3 trong bài 1.

Chương trình 14

```

1 /*Yeu cau:
2   Chuong trinh chop tat LED o PORT voi thoi gian delay 1s su dung Timer
3 */
4 //Ten file: BAI-5-1.C
5 #include<16F887.h> //Khai bao ten PIC
6 #include<def_887.h> //Thu vien do nguoi dung dung nghia
7
8 #FUSES NOWDT, PUT, NOPROTECT
9
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz
11

```

```

12 int32 count = 0;
13
14 #INT_TIMER0 //Ngat TIMER0
15 void Ngat_Timer0(){
16     SET_TIMER0(100);
17
18     //Xu ly chuong trinh ngat
19     ++count; //Tang gia tri bien dem
20     if (count == 1000){
21         CLEAR_INTERRUPT(INT_TIMER0); //Xoa co ngat
22         DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc
23         count = 0;
24         PORTE = ~PORTE;
25         ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
26     }
27 }
28
29 void main(){
30     TRISE = 0x00; //Chan OUTPUT
31
32     ENABLE_INTERRUPTS(INT_TIMER0); //Kich hoat ngat tran Timer0
33     SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_32); //Xung kich noi va chia truoc 32
34     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
35     SET_TIMER0(100); //Bat dau dem tu 100, khi tran Timer0 duoc 1ms
36
37     while (True){ //Duy tri hoat dong cua vi dieu khien
38         ;
39     }
40 }
```

5.3.2 Bài tập 5.2

Yêu cầu Viết chương trình đếm từ 0 đến 9999 sử dụng Timer của PIC 16F887, hiển thị trên LCD 16x02.

Hướng giải quyết

- Chúng ta sử dụng lại *chương trình 14* để tạo thời gian trễ là `sleep`, ví dụ trong chương trình khai báo `int32 sleep = 500;` (tạo thời gian trễ 500ms).
- Chương trình ngắn: cứ sao một khoảng thời gian `sleep` thì chúng ta tăng giá trị của biến `dем` lên.
- Chương trình chính: thêm vào vòng lặp `while` công việc: hiển thị giá trị biến `dем` lên LCD (sử dụng hàm `printf` kết hợp với hàm `LCD_PutChar`). Thêm vào điều kiện so sánh, nếu vượt quá giới hạn, đặt lại biến `dем = 0`;

Sơ đồ mạch Giống sơ đồ mạch bài tập 2.3 trong bài 2.

Chương trình 15

```
1 /*Yeu cau:  
2     Chuong trinh dem so tu 0 den 9999 su dung Timer va cho hien thi len LCD.  
3     */  
4 //Ten file: BAI-5-2.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
7  
8 #FUSES NOWDT,HS, PUT, NOPROTECT  
9  
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
11 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao  
12  
13 int32 count = 0, sleep = 500, dem = 0, N_max = 9999;  
14  
15 #INT_TIMER0 //Ngat TIMER0  
16 void Ngat_Timer0(){  
17     SET_TIMER0(100);  
18  
19     //Xu ly chuong trinh ngat  
20     count++; //Tang gia tri bien dem  
21     if (count == sleep){  
22         CLEAR_INTERRUPT(INT_TIMER0); //Xoa co ngat  
23         DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc  
24         count = 0;  
25         dem++; //Tang gia tri cua bien dem len  
26         ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc  
27     }  
28 }  
29  
30 void main(){  
31  
32     OUTPUT_LOW(LCD_RW); //Che do ghi  
33     LCD_Init(); //Khoi tao LCD  
34  
35     LCD_PutCmd(0x01); //Xoa man hinh  
36  
37     ENABLE_INTERRUPTS(INT_TIMER0); //Kich hoat ngat ngoai  
38     SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_32); //Xung kich noi va chia truoc 32  
39     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc  
40     SET_TIMER0(100); //Bat dau dem tu 100, khi tran Timer0 duoc 1ms  
41  
42     while (True){ //Duy tri hoat dong cua vi dieu khien  
43 }
```

```

44     if (dem > N_max){
45         dem = 0;
46         LCD_PutCmd(0x01);
47     }
48
49     LCD_SetPosition(0x00); //Cot 1 dong 1
50     LCD_PutChar("Dem so:");
51     LCD_SetPosition(0x07); //Cot 8 dong 1
52     printf(LCD_PutChar, "%lu", dem);
53
54 }
55 }
```

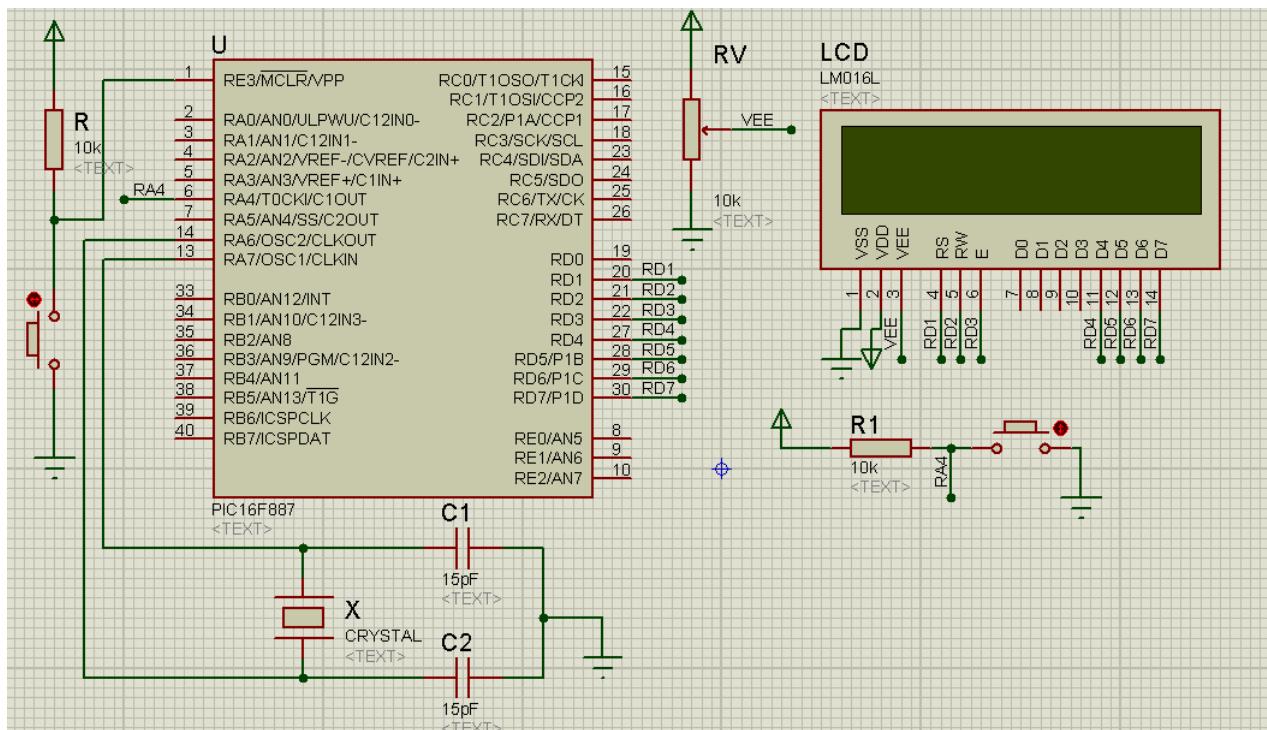
5.3.3 Bài tập 5.3

Yêu cầu Viết chương trình đếm số lần nhấn phím sử dụng Timer của PIC 16F887, hiển thị trên LCD 16x02.

Hướng giải quyết

- Timer 0 nhận được xung kích từ chân A4 của vi điều khiển (tùy thuộc vào cách ta cài đặt Timer 0).
- Cài đặt Timer 0: SETUP_TIMER_0(RTCC_EXT_H_TO_L|RTCC_DIV_1); chọn bit cạnh xuống trên chân A4 và sử dụng bộ chia 1 (do khi nhấn thì đếm và hiển thị lên LCD, bộ chia 1 là thích hợp nhất).
- Cứ mỗi lần nhấn nút nhấn thì giá trị của Timer 0 tăng lên một giá trị. Sử dụng tính chất này để thực hiện đếm.
- Thiết lập giá trị đầu cho Timer 0: SET_TIMER0(0); bắt đầu đếm từ 0.
- Đọc giá trị của Timer 0: value = GET_TIMER0();. Chưa vội cho hiển thị giá trị value lên LCD.
- Do gặp phải vấn đề sau: Timer 0 chỉ đếm được tới 255, vượt quá 255 thì chúng ta đếm sai, thực hiện kỹ thuật cộng dồn vào biến đếm: count = value + over*255, với over là số lần tràn (khi bắt đầu đếm thì over = 0;)
 - Chúng ta so sánh giá trị đọc được từ Timer 0 là biến value với 255 (giá trị tối đa của Timer 0. Nếu value == 255 thì cho tăng biến over lên.
 - Đồng thời cũng cần reset lại bộ đếm:SET_TIMER0(0);
- Phần còn lại, chúng ta chỉ việc cấu hình cho PORT A (chân INPUT) và LCD rồi cho hiển thị giá trị của biến count (đã thực hiện trong bài 2).

Sơ đồ mạch



Hình 5.1: Mạch đếm số lần nhấn nút sử dụng Timer hiển thị lên LCD

Chương trình 16

```
1 /*Yeu cau:  
2     Chuong trinh dem so lan nhat nut voi Timer va hien thi LCD  
3 */  
4 //Ten file: BAI-5-3.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
7 #FUSES NOWDT,HS, PUT, NOPROTECT  
8  
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
10 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao  
11  
12 int32 count, value, over = 0;  
13 void main(){  
14  
15     TRISA = 0xFF; //PORT A la INPUT  
16  
17     OUTPUT_LOW(LCD_RW); //Che do ghi  
18     LCD_Init(); //Khoi tao LCD  
19  
20     LCD_PutCmd(0x01); //Xoa man hinh  
21  
22     //Cai dat Timer 0 voi che do la canh xuong (H to L) tu chan A4  
23     //Su dung bo chia 1 (khong thay doi)  
24     SETUP_TIMER_0(RTCC_EXT_H_TO_L|RTCC_DIV_1);  
25     SET_TIMER0(0); //Bat dau dem tu 0.  
26  
27  
28     LCD_SetPosition(0x00); //Cot 1 dong 1  
29     LCD_PutChar("Count:");  
30  
31     while (True){ //Duy tri hoat dong cua vi dieu khien  
32         if (value == 255){ //Bat dau tran  
33             over++; //Tang so lan tran len  
34             SET_TIMER0(0); //Reset la bo dem  
35             value = GET_TIMER0(); //Cap nhan lai gia tri value  
36         }  
37         else{ //Khi khong tran thi doc gia tri tu Timer  
38             value = GET_TIMER0(); //Doc gia tri cua Timer  
39  
40             //Gia tri cua bien dem = gia_tri_doc_duoc + so_lan_tran*255  
41             count = value + over*255;  
42             LCD_SetPosition(0x06); //Cot 7 dong 1  
43             printf(LCD_PutChar, "%lu", count); //Hien thi len LCD  
44         }  
45     }  
46 }
```

BÀI 6

GIAO TIẾP I2C

6.1 Giới thiệu chung

6.1.1 Yêu cầu

Viết chương trình giao tiếp I2C giữa vi điều khiển PIC 16F887 với các module có chuẩn giao tiếp I2C.

6.1.2 Chuẩn I2C

Chuẩn I2C là chuẩn giao tiếp 2 dây là **Serial Data (SDA)** (truyền dữ liệu 2 hướng) và **Serial Clock (SCL)** (truyền xung đồng hồ theo một hướng).

Cách kết nối: chân SDA, SCL của module lần lượt nối đến chân SDA, SCL của vi điều khiển PIC 16F887. Ta có thể kết nối điều khiển nhiều thiết bị I2C với nhau thông qua 2 chân SDA và SCL của vi điều khiển. Mỗi thiết bị I2C sẽ được phân biệt bằng một *địa chỉ duy nhất address* hoặc *quan hệ chủ tớ master - slave*.

6.2 Giao tiếp chuẩn I2C với CCS

Để giao tiếp ta dùng khai báo và các lệnh sau:

- Khai báo: `#use I2C(mode, speed, SDA = PIN_C4, SLC = PIN_C3)`, với:
 - mode: master hoặc slave.
 - speed: slow ($100kHz$) hoặc fast ($400kHz$).
 - Trên PIC 16F887, chân C4 và chân C3 lần lượt là chân SDA và SCL.
- Các hàm giao tiếp được CCS định nghĩa:
 - `I2C_ISR_STATE()`: Thông báo trạng thái giao tiếp I2C.
 - `I2C_START()`: Tạo điều kiện START.
 - `I2C_STOP()`: Tạo điều kiện STOP.

- I2C_READ(mode): Đọc giá trị 8 bit thiết bị I2C. Với mode = 0 (không chỉ ra ACK) và mode = 1 (chỉ ra ACK)
 - I2C_WRITE(Device_address): Ghi giá trị 8 bit đến thiết bị I2C.
- * Để hiểu rõ hơn các hàm này, có thể xem trong phần Help của CCS.

6.3 Bài tập

6.3.1 Bài tập 6.1

Yêu cầu Viết chương trình đọc giá trị nhiệt độ từ cảm biến nhiệt TC 47 và hiển thị lên LCD 16x02.

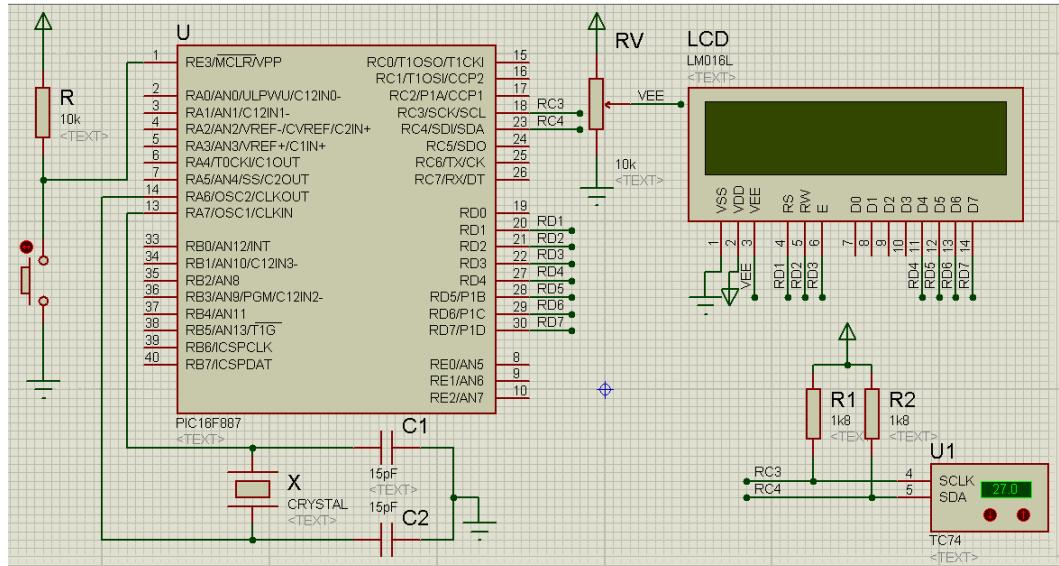
Hướng giải quyết

- Xác định chế độ truyền là **Master** (quan hệ chủ – tớ \longleftrightarrow PIC (chủ) – TC74 (tớ)).
- Xác định địa chỉ của thiết bị I2C TC74 là: 0x48 (Do thiết bị thực tập là TC74A0-5.0VAT, tùy dòng IC mà chúng ta xác định địa chỉ cho thích hợp, *tham khảo trang 9 của datasheet¹ TC74*).
- Để giao tiếp với ngoại vi chuẩn I2C theo quan hệ chủ tớ, ta thực hiện như sau:
 - Thiết bị chủ (PIC) tạo điều kiện Start: `i2c_start();`
 - Thiết bị chủ gửi địa chỉ của thiết bị tớ cùng với 1 bit 0, tức là gửi địa chỉ 0x90 ($0x48 = 1001000$, thêm bit 0 vào cuối: $1001000\ 0 = 0x90$), dùng lệnh: `i2c_write(0x90);` và đợi xung ACK phản hồi từ thiết bị tớ (phần này đã giải thích tại sao phải thêm số 0 vào cuối 7 bit địa chỉ ở ý trên).
 - Khi thiết bị tớ đã nhận được đúng địa chỉ của nó (phản hồi xung ACK), thiết bị chủ gửi lệnh thông báo truy cập vào thanh ghi đọc nhiệt độ - lệnh 00h, dùng lệnh: `i2c_write(0x00);` (phần lệnh của TC74 được mô tả ở trang 8 trong datasheet).
 - Ta đã hoàn thành việc *gửi yêu cầu đọc nhiệt độ* từ thiết bị tớ – TC74, tiếp theo là thiết bị chủ (PIC) sẽ *đọc giá trị nhiệt độ* từ thiết tớ (TC74).
 - Tạo lại điều kiện Start để thực hiện giao tiếp mới: `i2c_start();`
 - Thiết bị chủ gửi địa chỉ của thiết bị tớ cùng với bit 1 (lấy địa chỉ thiết bị I2C thêm vào bit 1: $1001000\ 1 = 0x91$), dùng lệnh: `i2c_write(0x91);`

¹<http://www.alldatasheet.com/>

- Phần còn lại là đọc giá trị nhiệt độ từ thiết bị tớ, thiết bị chủ gửi xung Not-ACK dùng lệnh: `i2c_read(0)`; (giá trị là số nguyên 8 bit).
 - Tạo điều kiện Stop: `i2c_stop()`; để kết thúc quá trình giao tiếp.
 - Để giao tiếp được nhiều lần, ta thực hiện vòng lặp `while` để lặp lại quá trình này (từ lệnh `i2c_start()`; đến `i2c_stop()`).
- Chúng ta lấy được giá trị nhiệt độ từ cảm biến TC74 và gửi lên LCD để quan sát.
 - Phần mô phỏng chương trình với phần mềm Protues em trìn bày trong phần *phuc luc trang 58*.
- * Khi làm phần cứng, cần măc thêm 2 điện trở pullup lên nguồn.
 - * Ngoài cách giải thích trên, chúng ta có thể tham khảo cách giải thích khác trong phần chú thích 2 ở bên dưới².

Sơ đồ mạch



Hình 6.1: Mạch đọc giá trị nhiệt độ từ IC TC74

²<http://embedded-lab.com/blog/using-tc74-microchip-thermal-sensor-for-temperature-measurement/>

Chương trình 17

```
1 /*Yeu cau
2     Doc nhiet do tu cam bien nhiet TC74
3 */
4 //Ten file BAI-6-1.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #fuses HS,NOWDT, NOPROTECT, NOLVP
8 #use i2c(master, SDA = PIN_C4, SCL = PIN_C3)
9 #use delay(clock=20000000)
10 #include<lcd_lib_4bit.c>
11
12 int temp;
13
14 void main(){
15     Output_low(LCD_RW); //Che do ghi
16     LCD_init(); //Khoi tao LCD
17
18     lcd_putcmd(0x01);
19     delay_ms(100);
20     printf(lcd_putchar,"Nhiet do: ");
21     while (true){
22         i2c_start(); //Tao dieu kien start de giao tiep
23         //Dia chi thiet bi I2C(trong trang 9 cua Datasheet) sau khi da them bit 0
24         //vao
25         //0x48 (1001000 them bit 0 vao 10010000 = 0x90
26         i2c_write(0x90);
27         i2c_write(0x00); //Gui lenh yeu cau doc nhiet do
28         i2c_start(); //Tao lai dieu kien start
29         //Dia chi thiet bi I2C(trong trang 9 cua Datasheet) sau khi da them bit 1
30         //vao
31         //0x48 (1001000 them bit 1 vao 10010001 = 0x91
32         i2c_write(0x91); //Thiet bi chu yeu cau duoc doc du lieu cua thiet bi to
33         temp = i2c_read(0); //Gui lenh doc gia tri nhiet do
34         i2c_stop(); //Tao dieu kien Stop, ket thuc giao tiep
35         LCD_SetPosition(0x09);
36         printf(lcd_putchar,"%u",temp); //Hien thi nhiet do doc duoc len LCD
37         lcd_putchar(223);
38         printf(lcd_putchar,"C");
39         delay_ms(1000);
40     }
41 }
```

6.3.2 Bài tập 6.2

Yêu cầu Viết chương trình đọc giá trị thời gian từ IC DS1307 và hiển thị lên LCD 16x02.

Hướng giải quyết Do không có IC DS1307 nên em thay thế bằng phần cứng khác là module RTC DS3231.

- * So với IC DS1307 thì IC DS3231 có thời gian chạy chính xác hơn. Module RTC DS3231 ngoài chức năng lấy thời gian thực, nó còn chức năng đo được giá trị nhiệt độ.
- Chúng ta quan tâm đến các địa chỉ sau:
 - Địa chỉ của thiết bị I2C là 0x68.
 - Địa chỉ của các thanh ghi lưu giá trị thời gian:

Thanh ghi	Địa chỉ	Thanh ghi	Địa chỉ
secondREG	0x00	dayREG	0x03
minuteREG	0x01	dateREG	0x04
hourREG	0x02	monthREG	0x05
		yearREG	0x06

Bảng 6.1: Địa chỉ thanh ghi thời gian của IC DS3231

- Cài đặt định dạng thời gian hiển thị:

Định dạng	Cài đặt	Định dạng	Cài đặt
_24_hour_format	0	am	0
_12_hour_format	1	pm	1

Bảng 6.2: Cài đặt định dạng thời gian hiển thị

- Giá trị lưu trong thanh ghi ở dạng BCD. Nên muốn ghi giá trị vào thanh ghi cần chuyển sang mã BCD.
- Sử dụng chương trình DS3231 High Precision I2C RTC Driver do tác giả sshahryiar trên diễn đàn <https://www.ccsinfo.com>³ viết để làm bài này.

³<https://www.ccsinfo.com/forum/viewtopic.php?t=50256>

- Về thư viện chương trình gồm có 2 file: DS3231.h và DS3231.C (nội dung của các file trong phần phụ lục trang 107, do trong bài viết tác giả sshahryiar không giải thích nội dung code, gây khó cho việc theo dõi, nên trong phần phụ lục em xin giải thích lại cách làm của tác giả để tiện cho việc theo dõi).
- Với file DS3231.h chứa định nghĩa các địa chỉ thanh ghi, cách định dạng thời gian và các hàm trong file DS3231.C.
- Chúng ta quan tâm đến việc sử dụng các hàm trong thư viện do tác giả sshahryiar viết:
 - * Đầu tiên cần cài đặt thời gian thực vào IC bằng 2 hàm sau: (do thời gian trong IC không đúng với thời gian thực hiện tại).


```
setTime(hr, min, s, am_pm, hr_format);
setDate(dy, dt, mt, yr);
```

Ví dụ: Thời gian thực hiện tại là: 10:30:00 AM Tus 19-04-2016, khi đó ta cài đặt thời gian như sau:

```
setTime(10, 30, 0, 0, _12_hour_format);
setDate(3, 19, 4, 16);
```

Giải thích ý nghĩa của các tham số :

setTime	Giải thích	setDate	Giải thích
hr	Giờ: 1-12 (12h) hoặc 00-23 (24h)	dy	Thứ: 1-7 (từ chủ nhật đến thứ 7)
min	Phút: 00-59	dt	Ngày: 01-31
s	Giây: 00-59	mt	Tháng: 01-12
am_pm	Giờ: 0 hoặc 1 với am - 0 hoặc pm - 1	yr	Năm: 00-99
hr_format	Giờ 12: _12_hour_format Giờ 24: _24_hour_format		

Bảng 6.3: Giải thích các tham số cài đặt thời gian thực cho IC DS3231

- * Đọc thời gian trong IC ra, ta dùng 2 hàm sau:

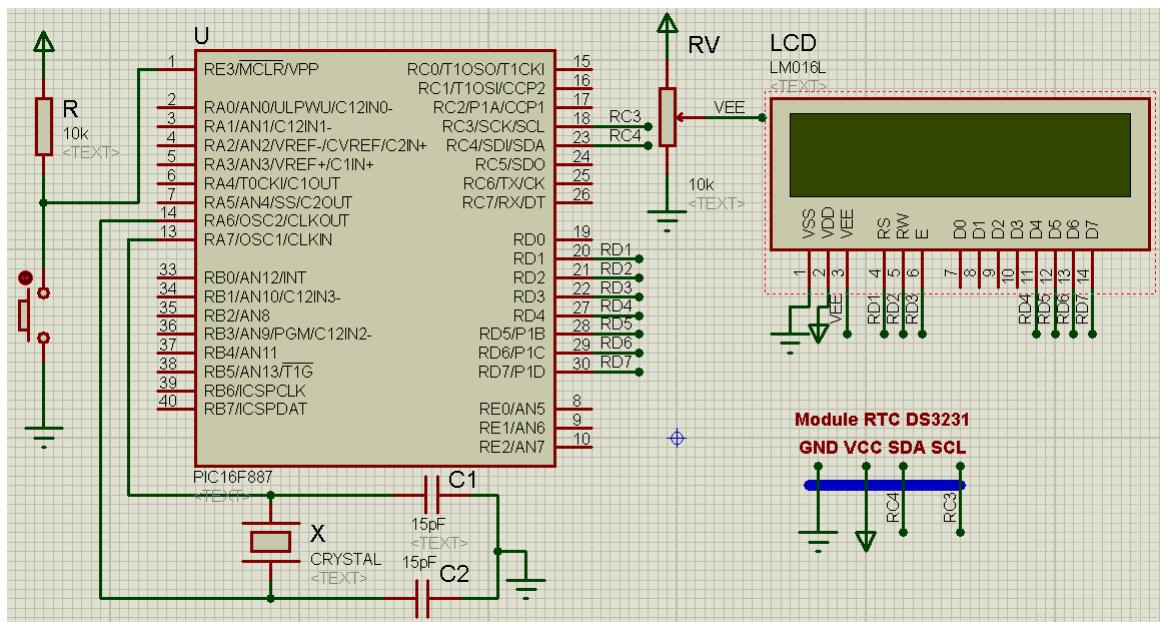
```
getTime(hr, min, s, am_pm, hr_format);
getDate(dy, dt, mt, yr);
```

Với các tham số kết quả ra tùy thuộc vào cách ta đã cài đặt tham số đầu vào theo định nghĩa trong bảng 6.3.

- Do ta truyền tham số vào ở dạng số, nên muốn hiển thị thứ hoặc giờ am hay giờ pm thì ta có thể kết hợp với cấu trúc switch - case để lựa chọn giá trị xuất ra.

- Trong chương trình cần thêm vào `#include<DS3231.C>` để thêm thư viện vào chương trình. Tương tự như các bài tập trước ta khai báo LCD và thực hiện vòng lặp `while` để giữ chương trình.
- Cách chạy chương trình để IC DS3231 tự động cập nhật thời gian: chúng ta thực hiện qua 2 bước sau:
 - Bước 1:* Chạy chương trình 18 – 1 cài đặt thời gian thực vào IC. Sau lần chạy đầu tiên, ta rút nguồn ra, nạp chương trình 18 – 2 vào IC.
 - Bước 2:* Chạy chương trình 18 – 2 để lấy thời gian trong IC ra – thời gian lúc này mới là thời gian thực. Đến đây, quá trình nạp xem như thành công.
- * Thật ra ở *bước 2* chúng ta chỉ chú thích đi 2 dòng cài đặt thời gian ban đầu vào IC, sau khi chạy lần đầu tiên, IC sẽ tính giờ từ thời điểm ta cài vào (trường hợp ngắt nguồn thì IC đã có nguồn Pin 3V giúp IC nhớ được giờ), nên cấp nguồn lại thì thời gian vẫn đúng với thời gian thực. Do đó *chương trình 18 – 1* và *chương trình 18 – 2* chỉ là một chương trình (do đó ta thấy ở *chương trình 18 – 1* có một số lệnh chưa cần đến, làm cho chương trình dài lên).

Sơ đồ mạch



Hình 6.2: Mạch đọc thời gian thực từ module RTC DS3231

Kết quả



Hình 6.3: Kết quả đọc thời gian thực từ IC DS3231

Chương trình 18 – 1: Cài thời gian thực vào IC

```
1 /*Yeu cau
2     Chuong trinh cai thoi gian thuc vao IC
3 */
4 //Ten file BAI-6-2-1.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #include<string.h>
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #use I2C(Master, SDA = pin_C4, SCL = pin_C3) //Giao tiep I2C
11 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
12 #include<DS3231.C>
13 unsigned char s, min, hr, dy, dt, mt, yr;
14 short am_pm;
15 char day[3],ampm[2];
16 void main(){
17     OUTPUT_LOW(LCD_RW); //Che do ghi
18     LCD_Init(); //Khoi tao LCD
19     DS3231_init();
20
21     setTime(10,30,0,0, _12_hour_format);
22     setDate(3,19,4,16);
23     getTime(hr, min, s, am_pm, _12_hour_format);
24     getDate(dy, dt, mt, yr);
25     LCD_PutCmd(0x01); //Xoa man hinh
26     while (True){
27         getTime(hr, min, s, am_pm, _12_hour_format);
28         getDate(dy, dt, mt, yr);
29
30         switch(dy){ //Tu cac so 1 - 7: dung cau truc switch - case tinh ra thu
```

```

31     case 1:{  
32         strcpy(day, "Sun"); //Copy chuoi "Sun" vao chuoi day  
33         break;  
34     }  
35     case 2:{  
36         strcpy(day, "Mon");  
37         break;  
38     }  
39     case 3:{  
40         strcpy(day, "Tus");  
41         break;  
42     }  
43     case 4:{  
44         strcpy(day, "Wed");  
45         break;  
46     }  
47     case 5:{  
48         strcpy(day, "Thu");  
49         break;  
50     }  
51     case 6:{  
52         strcpy(day, "Fri");  
53         break;  
54     }  
55     case 7:{  
56         strcpy(day, "Sat");  
57         break;  
58     }  
59 }  
60 switch (am_pm){ //Tinh ra gio AM hay gio PM  
61     case 0:{  
62         strcpy(ampm,"AM"); //Copy chuoi "AM" vao chuoi ampm  
63         break;  
64     }  
65     case 1:{  
66         strcpy(ampm,"PM");  
67         break;  
68     }  
69 }  
70  
71 //Hien thi ket qua len LCD  
72 LCD_SetPosition(0x03);  
73 printf(LCD_PutChar, "%02d:%02d:%02d %c%c", hr,min,s,ampm[0],ampm[1]);  
74 LCD_SetPosition(0x41);  
75 printf(LCD_PutChar, "%c%c%c-%02d-%02d-20%02d", day[0],day[1],day[2],dt,mt,yr);  
76 }  
77  
78 }

```

Chương trình 18 – 2: Lấy thời gian thực từ IC ra

```
1 /*Yeu cau
2   Chuong trinh lay thoi gian thuc tu IC DS3231
3 */
4 //Ten file BAI-6-2-2.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #include<string.h>
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #use I2C(Master, SDA = pin_C4, SCL = pin_C3) //Giao tiep I2C
11 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
12 #include<DS3231.C>
13 unsigned char s, min, hr, dy, dt, mt, yr;
14 short am_pm;
15 char day[3],ampm[2];
16 void main(){
17   OUTPUT_LOW(LCD_RW); //Che do ghi
18   LCD_Init(); //Khoi tao LCD
19   DS3231_init();
20
21   //Ta danh dau chu thich di 2 dong cai dat thoi gian ban dau,
22   //de lay thoi gian thuc
23   //setTime(10,30,0,0, _12_hour_format);
24   //setDate(3,19,4,16);
25   getTime(hr, min, s, am_pm, _12_hour_format);
26   getDate(dy, dt, mt, yr);
27   LCD_PutCmd(0x01); //Xoa man hinh
28
29   while (True){
30     getTime(hr, min, s, am_pm, _12_hour_format);
31     getDate(dy, dt, mt, yr);
32
33     switch(dy){ //Tu cac so 1 - 7: dung cau truc switch - case tinh ra thu
34       case 1:{
35         strcpy(day,"Sun"); //Copy chuoi "Sun" vao chuoi day
36         break;
37       }
38       case 2:{
39         strcpy(day,"Mon");
40         break;
41       }
42       case 3:{
43         strcpy(day,"Tus");
44         break;
45       }
46       case 4:{
```

```

47         strcpy(day, "Wed");
48         break;
49     }
50 case 5:{
51     strcpy(day, "Thu");
52     break;
53 }
54 case 6:{
55     strcpy(day, "Fri");
56     break;
57 }
58 case 7:{
59     strcpy(day, "Sat");
60     break;
61 }
62 }

63
64 switch (am_pm){ //Tinh ra gio AM hay gio PM
65     case 0:{
66         strcpy(ampm,"AM"); //Copy chuoi "AM" vao chuoi ampm
67         break;
68     }
69     case 1:{
70         strcpy(ampm,"PM");
71         break;
72     }
73 }

74 //Hien thi ket qua len LCD
75 LCD_SetPosition(0x03);
76 printf(LCD_PutChar,"%02d:%02d:%02d %c%c",hr,min,s,ampm[0],ampm[1]);
77 LCD_SetPosition(0x41);
78 printf(LCD_PutChar,"%c%c%c-%02d-%02d-20%02d",day[0],day[1],day[2],dt,mt,yr);
79
80 }
81
82 }

```

6.3.3 Bài tập 6.3

Yêu cầu Viết chương trình đồng hồ số hiển thị trên LCD 16x02 bao gồm giờ, phút, giây và nhiệt độ môi trường.

Hướng giải quyết

- * Do sử dụng phần cứng là module RTC DS3231 nên ngoài chức năng lưu thời gian thực nó còn chức năng đọc nhiệt độ môi trường.
- Địa chỉ của thanh ghi nhiệt độ là:

Thanh ghi	Địa chỉ	Hàm	Mô tả
tempMSBREG	0x11	getTemp()	Đọc vào nhiệt độ môi trường
tempLSBREG	0x12		

Bảng 6.4: IC DS3231 với chức năng đọc nhiệt độ môi trường

- Với phần thời gian thì chương trình giống như *chương trình 18 – 1* và *chương trình 18 – 2* của *bài tập 6.2*. Phần nhiệt độ thì chúng ta thêm dòng lệnh sau vào vòng lặp `while` và cho hiển thị giá trị đọc được lên LCD: `getTemp();`

Sơ đồ mạch Cách kết nối giống như *bài tập 6.2*

Kết quả



Hình 6.4: Kết quả đọc thời gian thực và nhiệt độ từ IC DS3231

Chương trình 19

```
1 /*Yeu cau
2     Chuong trinh hien thi nhiet do va thoi gian len LCD
3 */
4 //Ten file BAI-6-3.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #include<string.h>
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #use I2C(Master, SDA = pin_C4, SCL = pin_C3) //Giao tiep I2C
11 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
12 #include<DS3231.C>
13 unsigned char s, min, hr, dy, dt, mt, yr;
14 short am_pm;
15 char day[3];
16
17 void main(){
18     OUTPUT_LOW(LCD_RW); //Che do ghi
19     LCD_Init(); //Khoi tao LCD
20     DS3231_init();
21
22     //Ta danh dau chu thich di 2 dong cai dat thoi gian ban dau,
23     //de lay thoi gian thuc
24     //setTime(11,15,0,0, _24_hour_format);
25     //setDate(3,19,4,16);
26     getTime(hr, min, s, am_pm, _24_hour_format);
27     getDate(dy, dt, mt, yr);
28     LCD_PutCmd(0x01); //Xoa man hinh
29
30     while (True){
31         getTime(hr, min, s, am_pm, _24_hour_format);
32         getDate(dy, dt, mt, yr);
33
34         switch(dy){ //Tu cac so 1 - 7: dung cau truc switch - case tinh ra thu
35             case 1:{
36                 strcpy(day,"Sun"); //Copy chuoi "Sun" vao chuoi day
37                 break;
38             }
39             case 2:{
40                 strcpy(day,"Mon");
41                 break;
42             }
43             case 3:{
44                 strcpy(day,"Tus");
45                 break;
46             }
47         }
48     }
49 }
```

```

47     case 4:{  
48         strcpy(day, "Wed");  
49         break;  
50     }  
51     case 5:{  
52         strcpy(day, "Thu");  
53         break;  
54     }  
55     case 6:{  
56         strcpy(day, "Fri");  
57         break;  
58     }  
59     case 7:{  
60         strcpy(day, "Sat");  
61         break;  
62     }  
63 }  
64  
65 //Hien thi ket qua len LCD  
66 LCD_SetPosition(0x01);  
67 printf(LCD_PutChar, "%02d:%02d:%02d %.1f", hr, min, s, getTemp());  
68 LCD_PutChar(223);  
69 printf(LCD_PutChar, "C");  
70 LCD_SetPosition(0x41);  
71 printf(LCD_PutChar, "%c%c%c-%02d-%02d-20%02d", day[0], day[1], day[2], dt, mt, yr);  
72 }  
73  
74 }

```

6.3.4 Mô phỏng cảm biến nhiệt TC74 với Protues

Trong quá trình tìm hiểu, em thấy có nhiều bạn gặp vấn đề khi mô phỏng cảm biến nhiệt độ TC74 với phần mềm protues. Em kiểm thử và tìm ra được là cảm biến nhiệt TC74 trong phần mềm mô phỏng Protues 7.8 nó tương thích với địa chỉ I2C là 0x9A (đa phần các bạn làm trên địa chỉ 0x90 nên khi mô phỏng thì ra giá trị nhiệt độ không đúng như đã cài đặt).

Về sơ đồ mạch và cách giải thích chương trình tương tự *bài tập 6.1 trang 46.*

Chương trình sau mô phỏng được với phần mềm Protues 7.8

```
1 //Yeu cau: Mo phong cam bien nhiet TC74 voi Protues
2 //Ten file BAI-6-1V2.C - Mo phong duoc voi Protues
3 #include<16f887.h>
4 #include<def_887.h>
5 #fuses HS,NOWDT, NOPROTECT, NOLVP
6 #use i2c(master, SDA = PIN_C4, SCL = PIN_C3)
7 #use delay(clock=20000000)
8 #include<lcd_lib_4bit.c>
9
10 int temp;
11 void main(){
12     Output_low(LCD_RW); //Che do ghi
13     LCD_init(); //Khoi tao LCD
14     lcd_putcmd(0x01);
15     delay_ms(100);
16     printf(lcd_putchar, "Nhiet do: ");
17     while (true){
18         i2c_start(); //Tao dieu kien start de giao tiep
19         i2c_write(0x9A); //Dia chi thiet bi I2C, tra trang 9 cua Datasheet
20         i2c_write(0x00); //Gui lenh yeu cau doc nhiet do
21         i2c_start(); //Tao lai dieu kien start
22         i2c_write(0x9B); //Thiet bi chu yeu cau duoc doc du lieu cua thiet bi to
23         temp = i2c_read(0); //Gui lenh doc gia tri nhiet do
24         i2c_stop(); //Tao dieu kien Stop, ket thuc giao tiep
25         LCD_SetPosition(0x09);
26         printf(lcd_putchar, "%u",temp); //Hien thi nhiet do doc duoc len LCD
27         lcd_putchar(223);
28         printf(lcd_putchar, "C");
29         delay_ms(1000);
30     }
31 }
```

BÀI 7

GIAO TIẾP RS232

7.1 Giới thiệu chung

7.1.1 Yêu cầu

Giao tiếp giữa máy tính và vi điều khiển PIC 16F887 thông qua module RS232.

7.1.2 Module RS232

Cổng nối tiếp RS232 được sử dụng để kết nối các thiết bị ngoại vi với máy tính.

Ưu điểm: Khả năng chống nhiễu cao, tháo rời ra khỏi máy tính dễ dàng, cổng nối tiếp còn là nguồn cấp cho vi điều khiển,...

Cách kết nối: Ta sử dụng 2 chân RX và TX của module kết nối lần lượt với 2 chân RX và TX của vi điều khiển (kết nối nối tiếp RX - RX và TX - TX).

7.2 Các lệnh giao tiếp RS232

Ta sử dụng khai báo và các lệnh sau để giao tiếp với RS232 qua cổng nối tiếp:

- Khai báo:

- Khai báo tần số thạch anh: #use delay(clock = 20000000), ví dụ tần số là 20MHz
- Khai báo RS232:

```
#use RS232(BAUD = 9600, PARITY = N, XMIT = PIN_C6, RCV = PIN_C7)
```

với BAUD = 9600: tốc độ truyền; PARITY = N: không kiểm tra chẵn lẻ;
XMIT = PIN_C6: chân truyền là C6; RCV = PIN_C7: chân nhận là C7.

- Các lệnh giao tiếp: printf, getc, getch, getchar, fgetc, gets, fgets, puc, putchar, fputc, puts, fputs, kbhit, assert, perror, set_uart_speed, setup_uart.

7.3 Bài tập

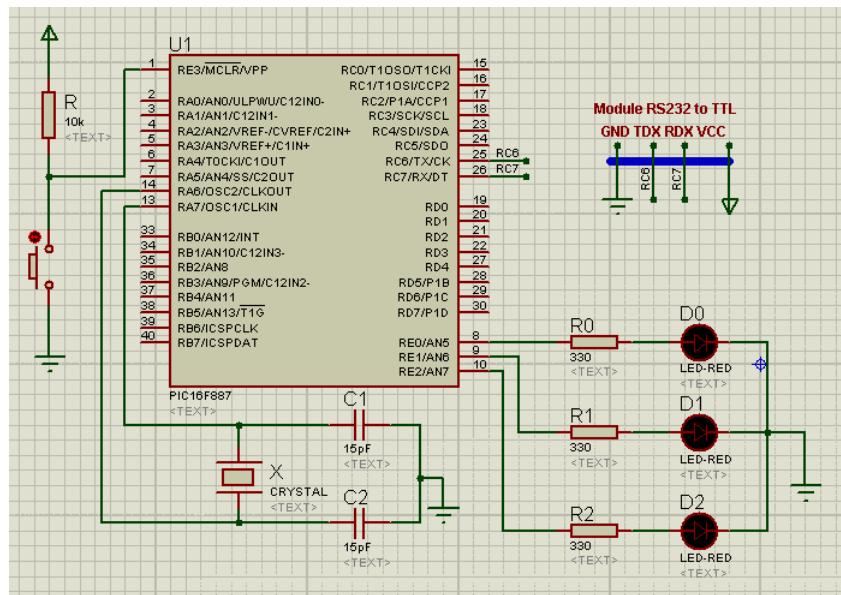
7.3.1 Bài tập 7.1

Yêu cầu Viết chương trình truyền dữ liệu giữa PC và PIC 16F887 với yêu cầu sau: Truyền ký tự 'b' và 't' từ PC thông qua chương trình MATLAB, khi PIC nhận được ký tự 'b' thì bật các LED ở PORT E, khi nhận được ký tự 't' thì tắt các LED ở PORT E.

Hướng giải quyết

- Chúng ta thực hiện truyền dữ liệu qua cổng nối tiếp RS232 để thực hiện giao tiếp giữa PC và PIC.
- Khai báo RS232: chọn tốc độ truyền là 9600
`#use RS232(BAUD = 9600, PARITY = N, XMIT = PIN_C6, RCV = PIN_C7)`
- Sử dụng ngắt nhận dữ liệu từ RS232 là #INT_RDA.
 - Khai báo ngắt:
ENABLE_INTERRUPTS(INT_RDA); và ENABLE_INTERRUPTS(GLOBAL);
 - Chương trình ngắt:
 - * Nội dung chương trình ngắt:
 - + Đọc giá trị từ PC gửi qua PIC: value = getc();
 - + Nếu value = 'b' thì đưa PORT E lên mức cao: PORTE = 0xFF;
 - + Nếu value = 't' thì đưa PORT E lên mức thấp: PORTE = 0x00;
- Khai báo PORT E là OUTPUT: TRISE = 0x00;
- Sử dụng phần mềm MATLAB để gửi dữ liệu đến PIC:
 - Xác định tên cổng nối tiếp, ta dùng s = serial(tên_port, tốc độ truyền) với Windows thì tên_port = 'com1' (còn những hệ điều hành khác, chúng ta có thể xem hướng dẫn về serial trong phần Help của MATLAB); tốc độ truyền bằng tốc độ truyền ta khai báo trong chương trình CCS.
 - Mở port dùng: fopen(s).
 - Gửi giá trị qua port dùng: fprintf(s, 'giá trị gửi').
Ví dụ: fprintf(s, 'b').
 - Nếu không gửi dữ liệu tiếp, ta thực hiện đóng port dùng: fclose(s).

Sơ đồ mạch



Hình 7.1: Mạch sử dụng module RS232 to TTL nhận lệnh từ PC bật tắt LED ở PORT E

Chương trình 20

```

1 /* Yeu cau:
2     Viet chuong bat tat cac LED o PORT E qua Matlab: b--bat led; t--tat led
3 */
4 //Ten file: BAI-7-1.C
5 #include<16f887.h>
6 #include<def_887.h>
7
8 #fuses HS, NOWDT, NOPROTECT, NOLVP
9 #use delay(clock = 20000000)
10 #use rs232(baud = 9600, xmit = PIN_C6, rcv = PIN_C7)
11 char value;
12
13 #INT_RDA
14 void RDA_interrupts(){
15     value =getc();
16     if (value == 'b')
17         PORTE = 0xFF;
18     if (value == 't')
19         PORTE = 0x00;
20 }
21

```

```

22 void main(){
23     TRISE = 0x00; //Pin OUTPUT
24     ENABLE_INTERRUPTS(INT_RDA); //Cho phep ngat RS232
25     ENABLE_INTERRUPTS(GLOBAL); // Cho phep ngat toan cuc
26
27     while (True){
28         ;
29     }
30 }
```

Chương trình nhận lệnh điều khiển từ PC với code Matlab (thực hiện lần lượt các lệnh sau bằng Command line):

```

1 %Mo cong COM, thiet lap toc do truyen phai phu hop voi code dieu kien PIC
2 s = serial('com6','BaudRate',9600);%Thay doi gia tri cong COM
3 fopen(s); %Mo cong COM
4 fprintf(s,'b'); %Gui lenh thu nhat: bat LED
5 %Tiep tuc gui lenh thu 2
6 fprintf(s,'t') %Lenh thu 2: tat LED
7
8 %Khi khong con lam viec, ta dong cong COM lai
9 fclose(s)
```

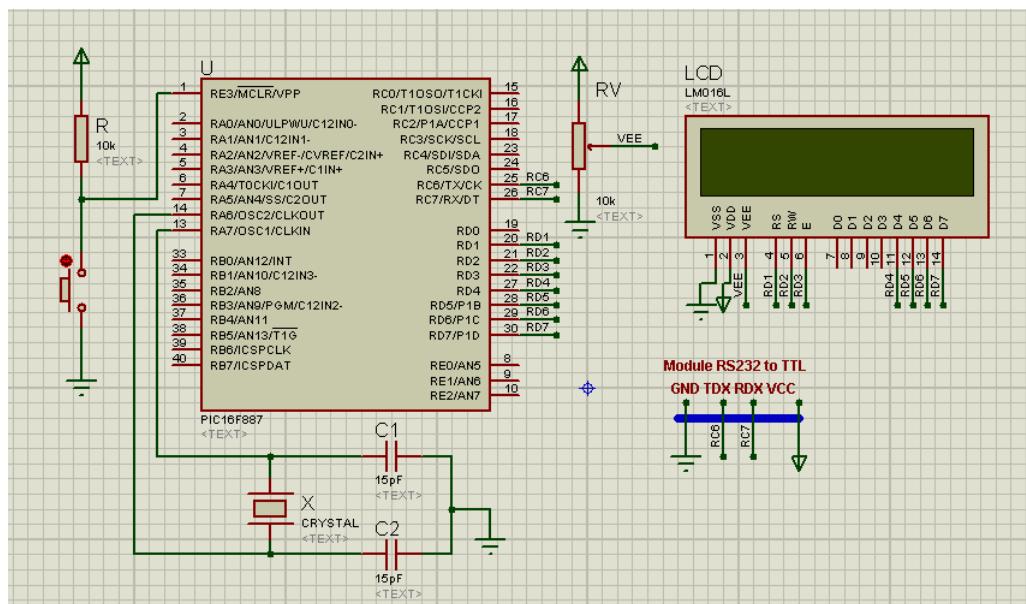
7.3.2 Bài tập 7.2

Yêu cầu Viết chương trình truyền chuỗi "TT VI DIEU KHIEN" từ PC xuống máy tính và hiển thị chuỗi nhận được lên LCD 16x02.

Hướng giải quyết

- * Chúng ta dựa vào *chương trình 20*, nhưng thay đổi một số chỗ cho phù hợp.
- *Chương trình ngắt*: Để đọc được một chuỗi từ PC truyền xuống, dùng hàm `gets(string)` thay cho hàm `getc()` (chỉ đọc được 1 ký tự).
- *Chương trình chính*: ta thêm các lệnh khởi tạo LCD (thêm vào thư viện `LCD_LIB4BIT.C` ở phần khai báo đầu). Trong vòng lặp `while` cho hiển thị chuỗi "TT VI DIEU KHIEN" nhận được từ PC lên LCD.
- * Khi chạy thực tế, đôi khi phần cứng bị nhiễu, đã nhận được chuỗi ký tự nhưng mà chưa hiển thị lên LCD thì nhấn nút Reset là chuỗi được hiển thị lên LCD.

Sơ đồ mạch



Hình 7.2: Mạch sử dụng module RS232 to TTL nhận chuỗi từ PC hiển thị lên LCD

Chương trình 21

```

1 /*Yeu cau
2   Chuong trinh nhan chuoi "TT VI DIEU KHIEN" tu PC hien thi len LCD
3 */
4 //Ten file BAI-7-2.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #fuses HS,NOWDT, NOPROTECT, NOLVP
8 #use delay(clock=20000000) //Tan so thach anh 20MHz
9 #use rs232(baud = 9600, xmit = PIN_C6, rcv = PIN_C7)
10 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
11
12 char string[30];
13
14 #INT_RDA
15 void RDA_interrupts(){
16     gets(string); //Doc vao mot chuoi tu PC
17 }
18
19
20 void main(){
21     ENABLE_INTERRUPTS(INT_RDA); //Cho phep ngat RS232
22     ENABLE_INTERRUPTS(GLOBAL); // Cho phep ngat toan cuc

```

```

23     OUTPUT_LOW(LCD_RW); //Che do ghi
24     LCD_Init(); //Khoi tao LCD
25
26     while (True){
27         LCD_PutCmd(0x01); //Xoa man hinh
28         LCD_SetPosition(0x00); //Cot 6 dong 1
29         printf(LCD_PutChar,"%s",string);
30         delay_ms(3000);
31     }
32 }
```

Chương trình nhận lệnh điều khiển từ PC với code Matlab (thực hiện lần lượt các lệnh sau bằng Command line):

```

1 %Mo cong COM, thiet lap toc do truyen phai phu hop voi code dieu kien PIC
2 s = serial('com6','BaudRate',9600);%Thay doi gia tri cong COM
3 fopen(s); %Mo cong COM
4 fprintf(s,'TT VI DIEU KHIEN'); %Gui chuoi xuong vi dieu khien
5 %Khi khong con lam viec, ta dong cong COM lai
6 fclose(s)
```

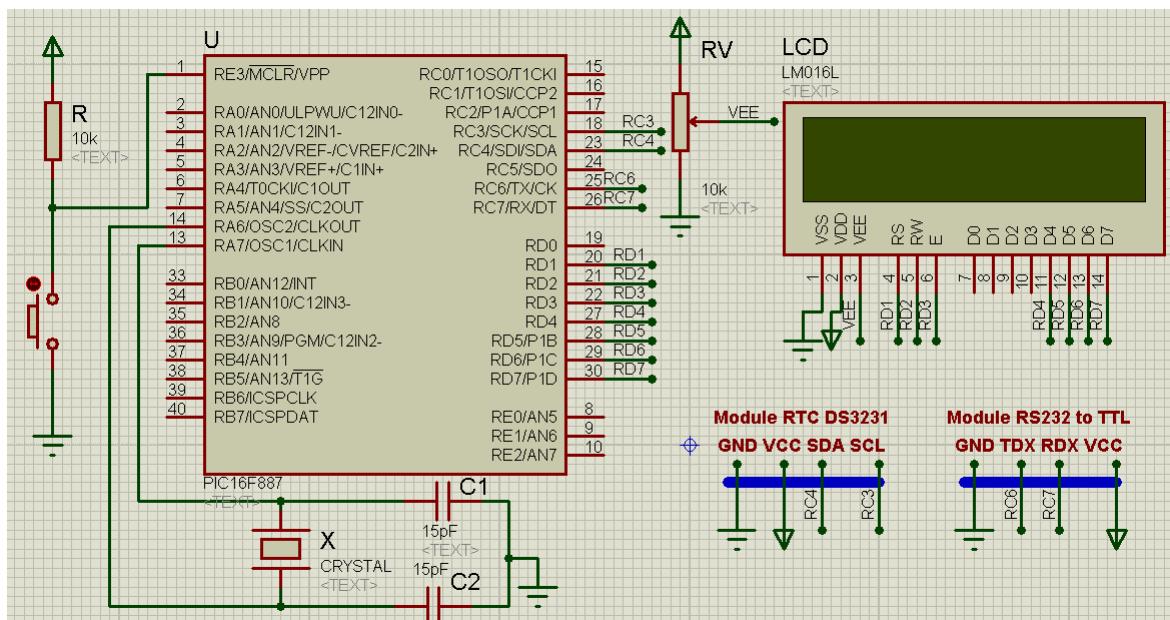
7.3.3 Bài tập 7.3

Yêu cầu Viết chương trình đọc giá trị nhiệt độ từ IC TC74 sau đó truyền giá trị đọc được lên PC qua cổng RS232.

Hướng giải quyết

- Do không có IC TC74, nên em thay thế bằng phần cứng khác là IC DS3231 (trong bài tập 6.2 và bài tập 6.3 trang 49 và trang 56), nó có chức năng lưu thời gian thực và đọc nhiệt độ môi trường.
- Sử dụng IC với chức năng đọc nhiệt độ (trong bài tập 6.3 trang 56) nên chúng ta rút ngắn lại chương trình 19 trang 57.
- Sau khi đọc được nhiệt độ dùng lệnh `printf("%.1fC", getTemp());` để gửi lên máy tính.
- Sử dụng LCD để kiểm chứng lại kết quả gửi lên máy tính.
- Phần code Matlab hoàn toàn giống bài tập 4.3 trang 34.

Sơ đồ mạch



Hình 7.3: Mạch sử dụng module RS232 to TTL gửi nhiệt độ từ IC DS3231 lên PC

Chương trình 22

```

1 /*Yeu cau
2   Chuong trinh hien thi nhiet do len PC
3 */
4 //Ten file BAI-7-3.C
5 #include<16f887.h>
6 #include<def_887.h>
7 #include<string.h>
8 #fuses HS,NOWDT, NOPROTECT, NOLVP
9 #use delay(clock=20000000) //Tan so thach anh 20MHz
10 #use I2C(Master, SDA = pin_C4, SCL = pin_C3) //Giao tiep I2C
11 #use rs232(baud = 9600, xmit = PIN_C6, rcv = PIN_C7)
12 #include<LCD_LIB_4BIT.C> //Them thu vien LCD vao
13 #include<DS3231.C>
14
15 void main(){
16   OUTPUT_LOW(LCD_RW); //Che do ghi
17   LCD_Init(); //Khoi tao LCD
18   DS3231_init();
19
20   LCD_PutCmd(0x01); //Xoa man hinh
21

```

```

22     while (True){
23         LCD_SetPosition(0x00);
24         printf(LCD_PutChar,"% .1f",getTemp()); //Hien thi nhiet do len LCD
25         printf("% .1fC\n",getTemp());           //Ghi nhiet do len may tinh
26         LCD_PutChar(223);
27         printf(LCD_PutChar, "C");
28     }
29
30 }
```

Hàm Serial_Callback dùng để thiết lập cho tham số BytesAvailableFcn trước khi mở cổng COM:

```

1 %%Ten file: Serial_Callback.m
2 function Serial_Callback(obj,event)
3     ind = fscanf(obj)
```

Lệnh Matlab nhận dữ liệu từ vi điều khiển gửi lên:

```

1 s = serial('COM6'); %Thay doi gia tri cong COM
2 %Thiet lap gia tri cho tham so BytesAvailableFcn de hien thi gia tri do vi
3 % dieu khien gui len
4 s.BytesAvailableFcn = @Serial_Callback;
5 fopen(s); %Mo cong COM de giao tiep
6 %%Du lieu duoc tu dong gui len va hien thi
7 %%Khi khong giao tiep, dong cong COM lai:
8 fclose(s);
```

BÀI 8

ĐIỀU KHIỂN ĐỘNG CƠ DC

8.1 Giới thiệu chung

8.1.1 Yêu cầu

Điều khiển chiều quay và tốc độ động cơ DC.

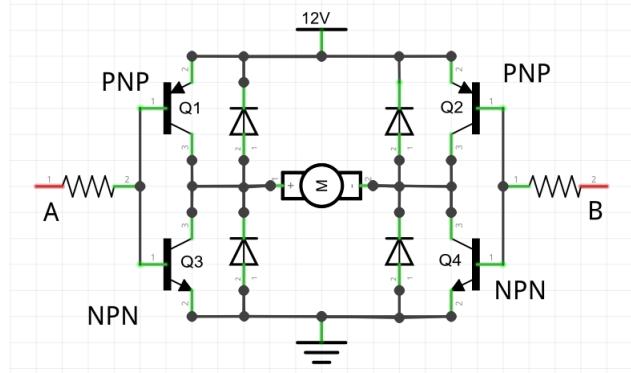
8.1.2 Động cơ DC

Để động cơ hoạt động được, ta cấp điện áp vào động cơ, xuất hiện dòng điện chạy các cuộn dây trong động cơ, làm động cơ quay theo một chiều xác định, để đổi chiều quay ta đổi chiều điện áp. Khi điều khiển động cơ: không được cấp điện áp và dòng điện quá giá trị định mức ghi trên động cơ (cấp thấp hơn động cơ vẫn hoạt động được).

8.2 Các bài toán điều khiển động cơ

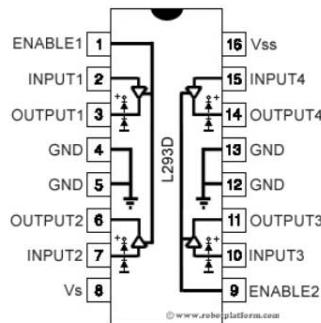
8.2.1 Điều khiển chiều quay của động cơ

Để điều khiển chiều quay của động cơ, người ta sử dụng mạch cầu H để làm điều này.



Hình 8.1: Mạch cầu H điều khiển chiều quay của động cơ

Chọn IC L293D để thực hiện chức năng của mạch cầu H điều khiển chiều quay của động cơ. Cụ thể như sau:



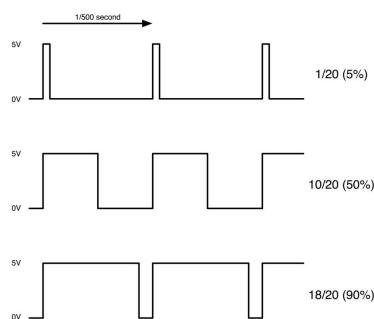
Hình 8.2: Sơ đồ chân IC L293D

- Số chân của IC: 16 chân (6 chân điều khiển tín hiệu; 4 chân điều khiển động cơ; 2 chân cấp nguồn; 4 chân đất), IC điều khiển được 2 động cơ.
- Các chân điều khiển: ENABLE1, INPUT1, INPUT2 (động cơ 1) và ENABLE2, INPUT3, INPUT4 (động cơ 2). Cụ thể chiều quay được xác định như sau:
 - Ta xác định cách điều khiển cho động cơ thứ nhất.
 - Chiều ngược chiều kim đồng hồ: $\text{ENABLE1} = 1, \text{INPUT1} = 1, \text{INPUT2} = 0$.
 - Chiều cùng chiều kim đồng hồ: $\text{ENABLE1} = 1, \text{INPUT1} = 0, \text{INPUT2} = 1$.
 - Dừng động cơ khi $\text{ENABLE1} = 0$ (bắt kẽ INPUT1 và INPUT2 ở mức cao hay mức thấp).
 - Động cơ thứ 2 cũng thực hiện tương tự như vậy: ta dùng các chân ENABLE2, INPUT3, INPUT4.

- Cấp nguồn: chân VSS lấy nguồn của vi điều khiển, chân Vs lấy nguồn ngoài cấp cho động cơ (không nên dùng chung nguồn với vi điều khiển).
- Chân nối động cơ: OUTPUT1, OUTPUT2 cho động cơ 1; OUTPUT3, OUTPUT4 cho động cơ 2.

8.2.2 Điều khiển tốc độ quay của động cơ

Ta sử dụng phương pháp điều chế độ rộng xung – PWM để điều khiển tốc độ động cơ. Ví dụ như trong các giản đồ xung *hình 8.3*:



Hình 8.3: Phương pháp điều chế độ rộng xung

Khối PWM trên PIC 16F887 có 2 mạch so sánh: 8 bit (so sánh với giá trị đếm của timer2) và 10 bit.

Ta thực hiện điều chế độ rộng xung bằng phần cứng (dùng chân CCP1 hoặc CCP2). Tiến hành tính toán và thiết lập các giá trị sau:

- Xác định tần số điều xung PWM qua khai báo sau:

```
setup_timer_2(mode, period, postscale);
```

suy ra: $f = \frac{f_{osc}}{4 \times mode \times (period + 1)}$ với:

- mode: T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16.
- period: 0 – 255.
- postscale: 1.

- Kích hoạt chế độ PWM: `setup_ccp1(cpp_pwm)` hoặc `setup_ccp2(cpp_pwm)`.

- Xác định hệ số chu kỳ xung thông qua khai báo:

```
set_pwm1_duty(value) hoặc set_pwm2_duty(value)
```

- Nếu `value` là số nguyên 8 bit: $dutycycle = \frac{value}{period + 1}$.

- Nếu `value` là số nguyên 16 bit: $dutycycle = \frac{value \times 1023}{4 \times (period + 1)}$.
- * Nếu ứng dụng không cần điều chế xung mịn thì ta có thể dùng 8 bit.

8.3 Bài tập

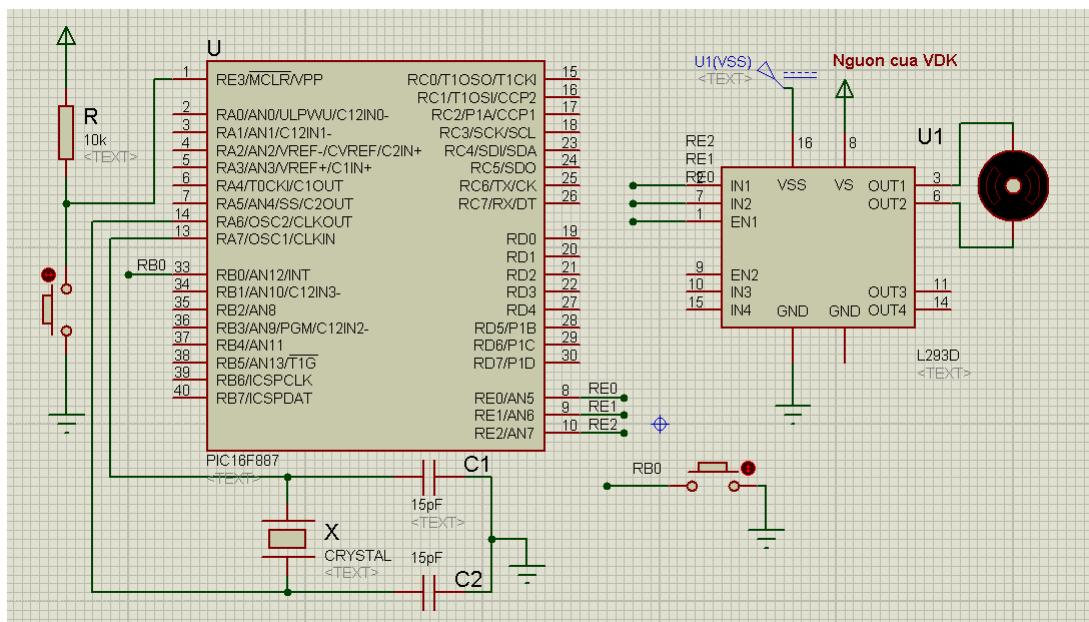
8.3.1 Bài tập 8.1

Yêu cầu Viết chương trình điều khiển chiều quay động cơ bằng nút nhấn nối với ngắt ngoài của vi điều khiển PIC 16F887.

Hướng giải quyết

- Ta sử dụng ngắt ngoài #INT_EXT (ngắt trên chân B0) để điều khiển chiều quay của động cơ thông qua số lần nhấn nút `count`:
- Khi chưa nhấn nút nhấn (`count = 0`): động cơ chưa quay.
- Khi nhấn nút lần đầu tiên (`count%2 = 1`): động cơ quay theo chiều thuận: `ENABLE1 = 1, INPUT1 = 1, INPUT2 = 0`.
- Khi nhấn nút lần thứ 2 (`count%2 = 0`): động cơ quay theo chiều nghịch: `ENABLE1 = 1, INPUT1 = 0, INPUT2 = 1`.
- Lặp lại quá trình trên.
- *Lưu ý*: tùy vào phần cứng của chúng ta có bị nhiễu hay không mà ta có thể khai báo giá trị ban đầu của biến `count` là: `count = 0` hay `count = -1`.

Sơ đồ mạch



Hình 8.4: Mạch đảo chiều động cơ với IC L293D

Chương trình 23

```

1 /*Yeu cau:
2   Chuong trinh dao chieu dong co voi nut nhan
3 */
4 //Ten file: BAI-8-1.C
5 #include<16F887.h> //Khai bao ten PIC
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia
7
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz
10
11 //Do phan cung bi nhieu, nen xay ra 1 ngat truoc.
12 unsigned long count = -1;
13
14 #INT_EXT //Ngat ngoai - chan B0
15 void NgatNgoai(){
16   CLEAR_INTERRUPT(INT_EXT); //Xoa co ngat
17   DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc
18
19 //Xu ly chuong trinh ngat
20 delay_ms(500); //Chong nhieu
21 count++; //Tang gia tri bien dem

```

```

22     /* E0 - ENABLE1
23     * E1 - INPUT1
24     * E2 - INPUT2
25     */
26     if (count == 0){ //Dung quay
27         PORTE = 0x00; // Chan ENABLE = 0
28     }
29     else
30         if ((count % 2 == 0) & (count>0)){//Quay thuan
31             PORTE = 0x03; //011 - INPUT2 = 0; INPUT1 = 1; ENABLE = 1
32         }
33     else
34         if (count % 2 == 1){ //Quay nghich
35             PORTE = 0x05; //101 - INPUT2 = 1; INPUT1 = 0; ENABLE = 1
36         }
37
38     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
39 }
40
41 void main(){
42     TRISB = 0xFF; //Chan INPUT
43     TRISE = 0x00; //Chan tin hieu dieu khien la OUTPUT
44     PORT_B_PULLUPS(1); //Noi dien tro len nguon
45
46     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai
47     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap
48     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
49
50     while (True){ //Duy tri hoat dong cua vi dieu khien
51         ;
52     }
53 }
```

8.3.2 Bài tập 8.2

Yêu cầu Viết chương trình điều khiển tốc độ và chiều quay của động cơ.

Hướng giải quyết

- Do đề bài không đưa ra yêu cầu cụ thể là điều khiển như thế nào, nên ta đặt ra yêu cầu cụ như sau:
 - Khi chưa nhấn nút nhấn trạng thái 0: động cơ không quay.
 - Khi nhấn nút nhấn lần đầu tiên – trạng thái 1: động cơ quay theo chiều của kim đồng hồ (quay thuận) và tốc độ tăng dần sau mỗi 1s, đến khi ổn định tốc độ thì giảm dần tốc độ sau mỗi 1s.

- Nhấn nút nhấn lần thứ 2 – trạng thái 2: động cơ quay ngược chiều của chiều kim đồng hồ (quay nghịch) và tốc độ tăng dần sau mỗi 1s, đến khi ổn định tốc độ thì giảm dần tốc độ sau mỗi 1s.
- Ví dụ, ta cần điều chế một xung có tần số điều xung là $f = 10kHz$ thì ta cần chọn: $mode \times (period + 1) = \frac{f_{osc}}{4 \times f} = \frac{20 \times 10^6}{4 \times 10 \times 10^3} = 500$, ta có bảng:

<i>mode</i>	<i>period</i>
1	499
4	124
16	30.25

Chọn cách khai báo sau: `setup_timer_2(T2_DIV_BY_4, 124, 1)`; sẽ tạo ra được một xung có tần số là $10kHz$.

- Tăng hoặc giảm tốc độ, ta cần tăng hoặc giảm giá trị của tham số `value` trong hàm `set_pwm1_duty(value)`.

Ta giả sử `set_pwm1_duty(dutycycle) = 100%`, thì xem `value` có giá trị là bao nhiêu:

$$\begin{aligned} value &= \frac{1}{f} \times dutycycle \times \frac{f_{osc}}{mode} \\ &= \frac{1}{10kHz} \times 100\% \times \frac{20MHz}{4} \\ &= \frac{1}{10 \times 10^3} \times 1 \times \frac{20 \times 10^6}{4} = 500 \end{aligned}$$

- Thực hiện vòng lặp `for` tăng dần giá trị của `value` nhưng phải luôn kiểm tra điều kiện $value \leq 500$.
- Phần trên ta chỉ điều khiển được tốc độ, tiếp theo ta sẽ điều khiển chiều quay của động cơ: sử dụng IC L293D với chân `ENABLE1` nối tắt lên nguồn, còn 2 chân `INPUT1` và `INPUT2` ta nối với 2 chân `CCP1` và `CCP2` để điều chế rộng xung cấp cho động cơ, việc đảo chiều tương tự như bài tập 8.1 trong chương trình 23.
- Cấu hình các chân điều khiển: `PORTB = 0xFF` (chân `INPUT`) và `PORTC = 0x00` (chân `OUTPUT`). Ta vẫn sử dụng ngắt ngoài ở B0 để dùng lại chương trình 23.

Sơ đồ mạch giống sơ đồ bài tập 8.1 hình 8.4 trang 72.

Chương trình 24

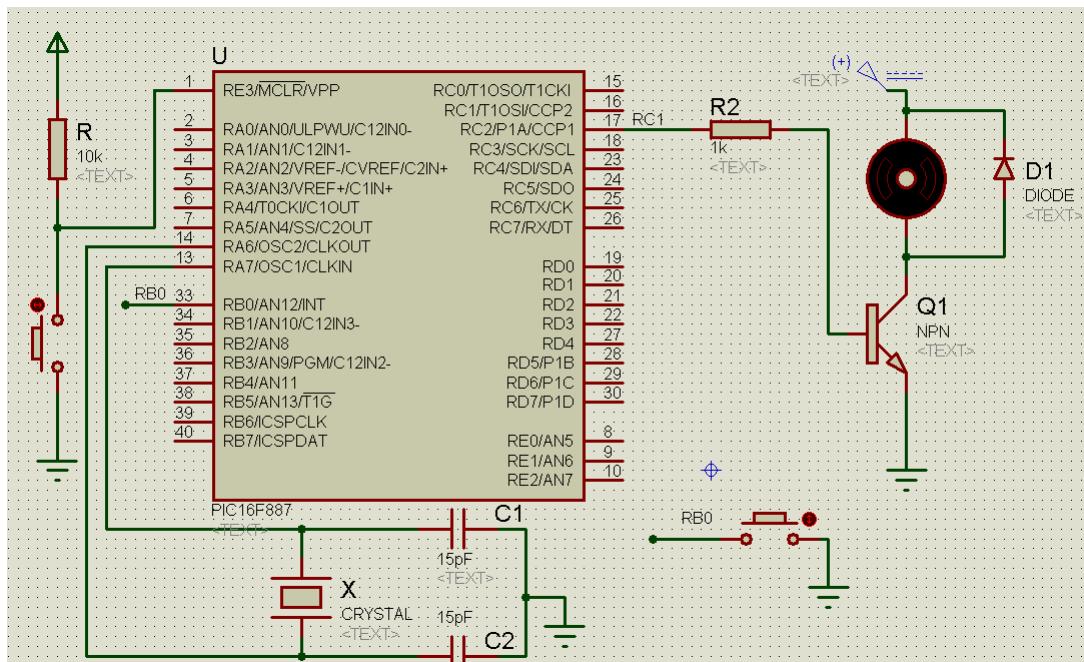
```
1 /*Yeu cau:  
2     Chuong trinh dieu khien toc do va dao chieu dong co voi nut nhan.  
3 */  
4 //Ten file: BAI-8-2.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
7  
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
10  
11 //Do phan cung bi nhieu, nen xay ra 1 ngat truoc.  
12 unsigned long count = -1, dc_max = 500,sleep = 100, dc = 0;  
13  
14 #INT_EXT //Ngat ngoai - chan B0  
15 void NgatNgoai(){  
16     //CLEAR_INTERRUPT(INT_EXT); //Xoa co ngat  
17     //DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc  
18  
19     //Xu ly chuong trinh ngat  
20     delay_ms(500); //Chong nhieu  
21     count++; //Tang gia tri bien dem  
22  
23     if (count == 0){  
24         PORTC = 0xF9; //Dung dong co  
25     }  
26     else  
27         if (count % 2 == 1){  
28             dc = 0;  
29             SET_PWM2_DUTY(0); //Keo chan INPUT2 = 0  
30  
31             while (dc < dc_max){ //Tang toc  
32                 dc++;  
33                 SET_PWM1_DUTY(dc);  
34                 delay_ms(sleep);  
35             }  
36  
37             delay_ms(2000);  
38  
39             while (dc > 0){  
40                 dc--;  
41                 SET_PWM1_DUTY(dc);  
42                 delay_ms(sleep);  
43             }  
44     }  
45     else
```

```

47     if (count % 2 == 0){ //Dao chieu
48         dc = 0;
49         SET_PWM1_DUTY(0); //Keo chan INPUT1 = 0
50
51         while (dc < dc_max){
52             dc++;
53             SET_PWM2_DUTY(dc);
54             delay_ms(sleep);
55         }
56
57         delay_ms(2000);
58
59         while (dc >0){
60             dc--;
61             SET_PWM2_DUTY(dc);
62             delay_ms(sleep);
63         }
64     }
65     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
66 }
67
68 void main(){
69     TRISB = 0xFF; //Chan INPUT
70     TRISC = 0x00; //Chan OUTPUT
71     PORT_B_PULLUPS(1); //Noi dien tro len nguon
72
73     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai
74     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap
75     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
76
77     //Cai dat tan so dieu xung la 10kHz
78     SETUP_TIMER_2(T2_DIV_BY_4,124,1);
79
80     SETUP_CCP1(CCP_PWM); //Su dung ca 2 chan PWM
81     SETUP_CCP2(CCP_PWM);
82
83     //INPUT2 = 0; INPUT1 = 0; ---> Dong co khong quay
84     SET_PWM1_DUTY(0);
85     SET_PWM2_DUTY(0);
86
87     while (True){
88         ;
89     }
90 }
```

8.3.3 Điều khiển tốc độ động cơ với Transistor

Sơ đồ mạch



Hình 8.5: Điều khiển tốc độ động cơ với Transistor

Nội dung của chương trình:

Chương trình 25

```
1 /*Yeu cau:  
2     Chuong trinh toc do va dao chieu dong co voi nut nhan.  
3 */  
4 //Ten file: BAI-8-2-1.C  
5 #include<16F887.h> //Khai bao ten PIC  
6 #include<def_887.h> //Thu vien do nguoi dung dung nghia  
7  
8 #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
10  
11  
12 #INT_EXT //Ngat ngoai - chan B0  
13 void NgatNgoai(){  
14     int i;  
15     CLEAR_INTERRUPT(INT_EXT); //Xoa co ngat
```

```

16 DISABLE_INTERRUPTS(GLOBAL); //vo hieu hoa ngat toan cuc
17
18 //Xu ly chuong trinh ngat
19 delay_ms(500); //Chong nhieu
20 //count++; //Tang gia tri bien dem
21 SETUP_TIMER_2(T2_DIV_BY_4,124,1);
22 SETUP_CCP1(CCP_PWM);
23 while (i < 125){
24     i++;
25     SET_PWM1_DUTY(i);
26     delay_ms(10);
27 }
28 ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
29 }

30
31 void main(){
32     TRISB = 0xFF; //Chan INPUT
33     TRISC = 0x00; //Chan OUTPUT
34     PORT_B_PULLUPS(1); //Noi dien tro len nguon
35
36     ENABLE_INTERRUPTS(INT_EXT); //Kich hoat ngat ngoai
37     EXT_INT_EDGE(H_TO_L); //Chon canh ngat, cao xuong thap
38     ENABLE_INTERRUPTS(GLOBAL); //Cho phep ngat toan cuc
39
40     //Cai dat tan so dieu xung la 10kHz
41     SETUP_TIMER_2(T2_DIV_BY_4,124,1);
42
43     SETUP_CCP1(CCP_PWM); //Su dung ca 2 chan PWM
44     SETUP_CCP2(CCP_PWM);
45
46     //INPUT2 = 0; INPUT1 = 0; ---> Dong co khong quay
47     SET_PWM1_DUTY(0);
48     //SET_PWM2_DUTY(0);
49
50     while (True){
51
52 }
53
54 }

```

BÀI 9

ĐO KHOẢNG CÁCH BẰNG CẢM BIẾN SIÊU ÂM SRF05

9.1 Giới thiệu chung

9.1.1 Yêu cầu

Sử dụng vi điều khiển PIC 16F887 lập trình đo khoảng cách với cảm biến siêu âm SRF05.

9.1.2 Cảm biến siêu âm SRF05

Nguyên lý đo khoảng cách của cảm biến siêu âm SRF05:

- Dùng nguyên lý phản xạ sóng để đo vật cản.
- Cảm biến phát đi một lúc 8 xung với tần số $40kHz$, gặp vật cản xung phát đi sẽ dội về. Từ đây, có thể tính được khoảng cách.
- Khi phát xung đi, chân Echo ở mức cao, nhận xung dội về, chân Echo xuống mức thấp (hoặc sau $30ms$ không nhận được cũng xuống mức thấp).

Cảm biến có 2 chế độ hoạt động: chân Trigger và chân Echo dùng riêng hoặc dùng chung.

- Chân Trigger và chân Echo dùng riêng;
- Chân Trigger và chân Echo dùng chung;

9.2 Đo khoảng cách với cảm biến siêu âm

Do khoảng cách với cảm biến siêu âm chính là đo thời gian chân Echo ở mức cao. Ta thực hiện theo các bước sau:

- Kích chân Trigger: Xuất ra mức cao ở chân Trigger và delay tối thiểu $10\mu s$.
- Dợi chân Echo lên mức cao.
- Khi chân Echo lên mức cao, kích hoạt Timer: có 2 cách thực hiện:
 - Dợi chân Echo xuống mức thấp.
 - Cho phép ngắt cạnh xuống (sử dụng với ngắt).
- Khi chân Echo xuống mức thấp, dừng Timer, tính thời gian từ Timer rồi suy ra khoảng cách.
 - Có $v = 344m/s = 344 \times 10^2 \times 10^{-6} = 0.0344cm/\mu s$. Thời gian t đơn vị là μs .
 - Công thức tính khoảng cách:
$$S = 2 \times d \iff d = \frac{S}{2} = \frac{v \times t}{2} = \frac{0.0344 \times t}{2} = \frac{t}{58.14} \text{ cm}$$

* Nếu sau $30ms$ mà không gặp vật cản thì chân *Echo* cũng sẽ xuống mức thấp.
- Reset lại giá trị của Timer để chuẩn bị cho các lần đo tiếp.

9.3 Bài tập

9.3.1 Bài tập 9.1

Yêu cầu Viết chương trình đo khoảng cách bằng cảm biến siêu âm kết nối với vi điều khiển PIC 16F887.

Chương trình 26

```

1  /*Yeu cau:
2  Chuong trinh do khoang cach tu cam bien sieu am
3  */
4  //Ten file: BAI-9-1.C
5  #include<16F887.h> //Khai bao ten PIC
6  #include<def_887.h> //Thu vien do nguoi dung dung nghia
7  #include<string.h>
8  #FUSES HS, NOWDT, NOPROTECT, NOLVP
9
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz
11 #include<LCD_LIB_4BIT.C>
12 #define PIN_TRIGGER PIN_A3

```

```

13
14 int1 echo = 0;
15 int16 value = 0;
16
17 void Trigger(){
18     OUTPUT_HIGH(PIN_TRIGGER);
19     delay_us(12);
20     OUTPUT_LOW(PIN_TRIGGER);
21 }
22
23 #INT_CCP1
24 void CCP1_ISR(void){
25     if (echo == 1){
26         SETUP_CCP1(CCP_CAPTURE_FE);
27         SET_TIMER1(0);
28         echo = 0;
29     }
30     else{
31         SETUP_CCP1(CCP_CAPTURE_RE);
32         value = CCP_1;
33         echo = 1;
34     }
35 }
36
37 void main(){
38     float distance = 0;
39
40     OUTPUT_LOW(LCD_RW);
41     LCD_Init();
42     LCD_PutCmd(0x01);
43
44     SETUP_TIMER_1(T1_INTERNAL|T1_DIV_BY_4);
45     SETUP_CCP1(CCP_CAPTURE_RE);
46
47     ENABLE_INTERRUPTS(INT_CCP1);
48     ENABLE_INTERRUPTS(GLOBAL);
49
50
51     while (True){
52         Trigger();
53
54         while (echo == 0){
55             ;
56         }
57
58         distance = value*0.8/58;
59         LCD_PutCmd(0x01);
60         LCD_SetPosition(0x00);

```

```

61     LCD_PutChar("Distance:");
62     LCD_SetPosition(0x40);
63     printf(LCD_PutChar, "%.2fcm", distance);
64     delay_ms(1000);
65 }
66 }
```

9.3.2 Bài tập 9.2

Yêu cầu Viết chương trình đo khoảng cách bằng cảm biến siêu âm kết nối với vi điều khiển PIC 16F887 và gửi giá trị đo được lên máy tính.

Chương trình 27

```

1 /*Yeu cau:
2 Chuong trinh do khoang cach tu cam bien sieu am
3 */
4 //Ten file: BAI-9-1.C
5 #include<16F887.h> //Khai bao ten PIC
6 #include<def_887.h> //Thu vien do nguoi dung dung nghia
7 #include<string.h>
8 #FUSES HS, NOWDT, NOPROTECT, NOLVP
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz
10 #include<LCD_LIB_4BIT.C>
11 #define PIN_TRIGGER PIN_A3
12
13 int1 echo = 0;
14 int16 value = 0;
15
16 void Trigger(){
17     OUTPUT_HIGH(PIN_TRIGGER);
18     delay_us(12);
19     OUTPUT_LOW(PIN_TRIGGER);
20 }
21
22 #INT_CCP1
23 void CCP1_ISR(void){
24     if (echo == 1){
25         SETUP_CCP1(CCP_CAPTURE_FE);
26         SET_TIMER1(0);
27         echo = 0;
28     }
29     else{
30         SETUP_CCP1(CCP_CAPTURE_RE);
31         value = CCP_1;
32         echo = 1;
```

```

33     }
34 }
35
36 void main(){
37     float distance = 0;
38     OUTPUT_LOW(LCD_RW);
39     LCD_Init();
40     LCD_PutCmd(0x01);
41
42     SETUP_TIMER_1(T1_INTERNAL|T1_DIV_BY_4);
43     SETUP_CCP1(CCP_CAPTURE_RE);
44
45     ENABLE_INTERRUPTS(INT_CCP1);
46     ENABLE_INTERRUPTS(GLOBAL);
47
48     while (True){
49         Trigger();
50         while (echo == 0){
51             ;
52         }
53         distance = value*0.8/58;
54         LCD_PutCmd(0x01);
55         LCD_SetPosition(0x00);
56         LCD_PutChar("Distance:");
57         LCD_SetPosition(0x40);
58         printf(LCD_PutChar, "%.2fcm", distance);
59         printf("%f\n", distance);
60         delay_ms(1000);
61     }
62 }
```

Hàm Serial_Callback dùng để thiết lập cho tham số BytesAvailableFcn trước khi mở cổng COM:

```

1 %%Ten file: Serial_Callback.m
2 function Serial_Callback(obj,event)
3     ind = fscanf(obj)
```

Lệnh Matlab nhận dữ liệu từ vi điều khiển gửi lên:

```

1 s = serial('COM6'); %Thay doi gia tri cong COM
2 %Thiet lap gia tri cho tham so BytesAvailableFcn de hien thi gia tri do vi
3 % dieu khien gui len
4 s.BytesAvailableFcn = @Serial_Callback;
5 fopen(s); %Mo cong COM de giao tiep
6 %%Du lieu duoc tu dong gui len va hien thi
7 %%Khi khong giao tiep, dong cong COM lai:
8 fclose(s);
```

PHỤ LỤC

Phụ lục A

Sử dụng PICKit 2 Programmer

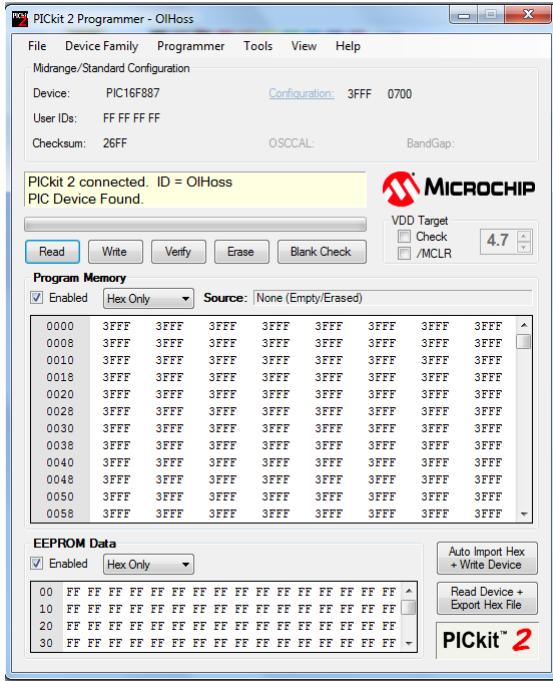
A.1 Giới thiệu chung

- Giao tiếp qua cổng USB, không cần cài đặt Driver.
- Hỗ trợ các loại PIC: PIC10F, PIC12F5xx, PIC16F5xx, PIC12F6xx, PIC16F, PIC18F, PIC24, dsPIC30, dsPIC33, PIC32 với 8 bit, 16 bit, 32 bit và một số sản phẩm Serial Eeprom của Microchip.

A.2 Phần mềm giao tiếp với PICkit 2

Sử dụng phần mềm PICKit 2 Programmer, các chức năng chính thường dùng trong phần mềm (giao diện trong *hình A.1*):

- Thẻ Tab:
 - Import Hex: Tải một file Hex vào bộ nhớ tạm.
 - Export Hex: Xuất ra một file Hex có trong bộ nhớ tạm.
- Thẻ Programmer:
 - Read Device: đọc nội dung chip như Program memory, EEPROM data đưa vào bộ nhớ tạm.
 - Write Device: nạp nội dung như Program memory, EEPROM data đưa vào bộ nhớ tạm vào chip.
 - Verify: so sánh nội dung của chip với nội dung trong bộ nhớ tạm.
 - Erase: Xóa toàn bộ nội dung trong chip.
- Thẻ Tools:
 - Enable Code Protect: khóa chương trình, chống sao chép.
 - Enable Data Protect: khóa bộ nhớ EEPROM data chống sao chép.



Hình A.1: Giao diện của phần mềm PICKit 2 Programmer

- **Fast programming:** nếu chọn chức năng này thì PICKit 2 sẽ nạp nhanh hơn, bình thường (chưa chọn) thì sẽ nạp chậm và độ tin cậy cao hơn.
- **Check Communication:** Kiểm tra giao tiếp giữa PC với PICkit 2 và ICSP giúp dò tìm chip.

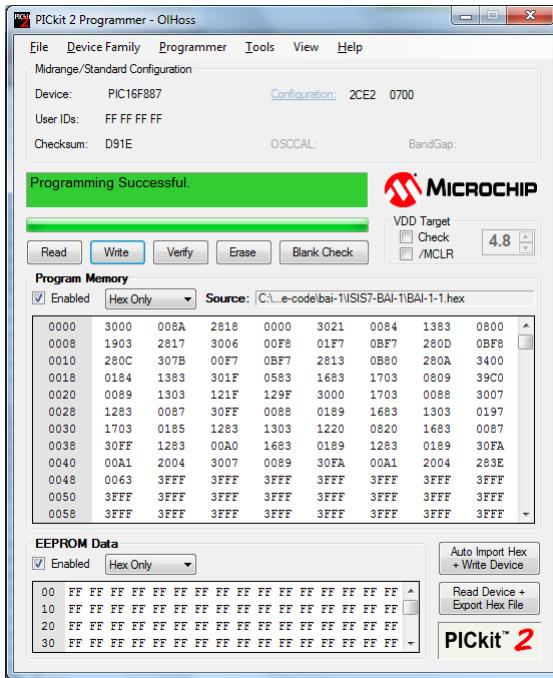
A.3 Nạp chương trình cho PIC

Thực hiện theo các bước:

- *Bước 1:* Vào thẻ Tools chọn Check Communication để kiểm tra và dò tìm PIC tự động:
 - + Nếu nhận được PIC thì dòng Device sẽ hiện tên PIC (như trong *hình A.1* là PIC16F887).
 - + Không nhận được chip thì kiểm tra lại kết nối USB, cách gắn PIC lên Adapter có đúng chiều hay chưa.
- *Bước 2:* Vào thẻ File chọn Import Hex rồi đi đến địa chỉ lưu file Hex cần nạp cho PIC. Rồi Click chọn Open để mở file Hex lưu vào bộ nhớ tạm.
- *Bước 3:* Click chọn Write để ghi nội dung từ bộ nhớ tạm vào PIC.

Nếu nạp thành công sẽ có giao diện như *hình A.2*, các trường hợp khác là chưa nạp thành công (không xuất hiện màu xanh lá như *hình A.2* mà xuất hiện màu

đó).



Hình A.2: Giao diện khi nạp thành công chương trình cho PIC

Ngoài ra cũng cần lưu ý ô điện áp VDD Target thông thường từ 4.7V – 5V.

Phụ lục B

Thư viện DEF_887.H

Nội dung file DEF_887.H (tên định danh của các PORT, các chân và thanh ghi).

```
1 // register definitions
2 #define W 0
3 #define F 1
4 // register files
5 #byte INDF      =0x00
6 #byte TMRO     =0x01
7 #byte PCL       =0x02
8 #byte STATUS    =0x03
9 #byte FSR       =0x04
10 #byte PORTA   =0x05
11 #byte PORTB   =0x06
12 #byte PORTC   =0x07
13 #byte PORTD   =0x08
14 #byte PORTE   =0x09
15
16 #byte EEDATA   =0x10C
17 #byte EEADR   =0x10D
18 #byte EEDATH   =0x10E
19 #byte EEADRH   =0x10F
20 #byte ADCONO   =0x1F
21 #byte ADCON1   =0x9F
22 #byte ADRESH   =0x9F
23 #byte ADSESL   =0x9F
24
25 #byte PCLATH   =0x0a
26 #byte INTCON   =0x0b
27 #byte PIR1     =0x0c
28 #byte PIR2     =0x0d
29 #byte PIE1     =0x8c
30 #byte PIE2     =0x8d
31
32 #byte OPTION_REG =0x81
33 #byte TRISA     =0x85
34 #byte TRISB     =0x86
35 #byte TRISC     =0x87
36 #byte TRISD     =0x88
```

```

37 #byte TRISE      =0x89
38
39 #byte EECON1     =0x18C
40 #byte EECON2     =0x18D
41 //DINH NGHIA BIT
42 #bit ra5  =0x05.5
43 #bit ra4  =0x05.4
44 #bit ra3  =0x05.3
45 #bit ra2  =0x05.2
46 #bit ra1  =0x05.1
47 #bit ra0  =0x05.0
48
49 #bit rb7  =0x06.7
50 #bit rb6  =0x06.6
51 #bit rb5  =0x06.5
52 #bit rb4  =0x06.4
53 #bit rb3  =0x06.3
54 #bit rb2  =0x06.2
55 #bit rb1  =0x06.1
56 #bit rb0  =0x06.0
57
58 #bit rC7  =0x07.7
59 #bit rC6  =0x07.6
60 #bit rC5  =0x07.5
61 #bit rC4  =0x07.4
62 #bit rC3  =0x07.3
63 #bit rC2  =0x07.2
64 #bit rC1  =0x07.1
65 #bit rC0  =0x07.0
66
67 #bit rD7  =0x08.7
68 #bit rD6  =0x08.6
69 #bit rD5  =0x08.5
70 #bit rD4  =0x08.4
71 #bit rD3  =0x08.3
72 #bit rD2  =0x08.2
73 #bit rD1  =0x08.1
74 #bit rD0  =0x08.0
75
76 #bit rE2  =0x09.2
77 #bit rE1  =0x09.1
78 #bit rE0  =0x09.0
79
80
81 #bit trisa5 =0x85.5
82 #bit trisa4 =0x85.4
83 #bit trisa3 =0x85.3
84 #bit trisa2 =0x85.2

```

```

85  #bit trisa1  =0x85.1
86  #bit trisa0  =0x85.0
87
88  #bit trisb7  =0x86.7
89  #bit trisb6  =0x86.6
90  #bit trisb5  =0x86.5
91  #bit trisb4  =0x86.4
92  #bit trisb3  =0x86.3
93  #bit trisb2  =0x86.2
94  #bit trisb1  =0x86.1
95  #bit trisb0  =0x86.0
96
97  #bit trisc7  =0x87.7
98  #bit trisc6  =0x87.6
99  #bit trisc5  =0x87.5
100 #bit trisc4  =0x87.4
101 #bit trisc3  =0x87.3
102 #bit trisc2  =0x87.2
103 #bit trisc1  =0x87.1
104 #bit trisc0  =0x87.0
105
106 #bit trisd7  =0x88.7
107 #bit trisd6  =0x88.6
108 #bit trisd5  =0x88.5
109 #bit trisd4  =0x88.4
110 #bit trisd3  =0x88.3
111 #bit trisd2  =0x88.2
112 #bit trisd1  =0x88.1
113 #bit trisd0  =0x88.0
114
115 #bit trise2  =0x89.2
116 #bit trise1  =0x89.1
117 #bit trise0  =0x89.0
118 // INTCON Bits for C
119 #bit gie    = 0x0b.7
120 #bit peie   = 0x0b.6
121 #bit tmr0ie = 0x0b.5
122 #bit int0ie = 0x0b.4
123 #bit rbie   = 0x0b.3
124 #bit tmr0if  = 0x0b.2
125 #bit int0if  = 0x0b.1
126 #bit rbif   = 0x0b.0
127 // PIR1 for C
128 #bit pspif  = 0x0c.7
129 #bit adif   = 0x0c.6
130 #bit rcif   = 0x0c.5
131 #bit txif   = 0x0c.4
132 #bit sspif  = 0x0c.3

```

```

133 #bit ccp1if  = 0x0c.2
134 #bit tmr2if  = 0x0c.1
135 #bit tmr1if  = 0x0c.0
136 //PIR2 for C
137 #bit cmif    = 0x0d.6
138 #bit eEIF    = 0x0d.4
139 #bit bclif   = 0x0d.3
140 #bit CCP2IF  = 0x0d.0
141
142 // PIE1 for C
143 #bit adie    = 0x8c.6
144 #bit rcie    = 0x8c.5
145 #bit txie    = 0x8c.4
146 #bit sspie   = 0x8c.3
147 #bit CCP1IE  = 0x8c.2
148 #bit tmr2ie  = 0x8c.1
149 #bit tmr1ie  = 0x8c.0
150 //PIE2 for C
151 #bit osfie   = 0x8d.7
152 #bit cmie    = 0x8d.6
153 #bit eeie    = 0x8d.4
154 // OPTION Bits
155 #bit not_rbpu = 0x81.7
156 #bit intedg  = 0x81.6
157 #bit t0cs    = 0x81.5
158 #bit t0se    = 0x81.4
159 #bit psa     = 0x81.3
160 #bit ps2     = 0x81.2
161 #bit ps1     = 0x81.1
162 #bit ps0     = 0x81.0
163 // EECN1 Bits
164 #bit eepgd   = 0x18c.7
165 #bit free    = 0x18C.4
166 #bit wrerr   = 0x18C.3
167 #bit wren    = 0x18C.2
168 #bit wr      = 0x18C.1
169 #bit rd      = 0x18C.0
170 //ADCON0
171 #bit CHS0    =0x1F.3
172 #bit CHS1    =0x1F.4
173 #bit CHS2    =0x1F.5

```

Phụ lục C

Thư viện LCD_LIB_4BIT.C

Nội dung file LCD_LIB_4BIT.C

```
1 #include <stddef.h>
2 #define LCD_RS PIN_D1
3 #define LCD_RW PIN_D2
4 #define LCD_EN PIN_D3
5
6 #define LCD_D4 PIN_D4
7 #define LCD_D5 PIN_D5
8 #define LCD_D6 PIN_D6
9 #define LCD_D7 PIN_D7
10
11 #define Line_1 0x80
12 #define Line_2 0xC0
13 #define Clear_Scr = 0x01
14
15
16 #separate void LCD_Init(void); //Ham khai tao LCD
17 #separate void LCD_SetPosition(unsigned int cX); //Thiet lap vi tri con tro
18 #separate void LCD_PutChar(unsigned int cX); //Ham viet mot ky tu hoac mot
     chuoi len LCD
19 #separate void LCD_PutCmd(unsigned int cX); //Ham gui lenh len LCD
20 #separate void LCD_PulseEnable(void); //Xung kich hoat
21 #separate void LCD_setData(unsigned int cX); //Dat du lieu len chan Data
22
23 //Dinh nghia cong
24 #use standard_io(C)
25 #use standard_io(D)
26
27 #separate void LCD_Init(void){
28     LCD_SetData(0x00);
29     delay_ms(200);
30     output_low(LCD_RS);
31     LCD_SetData(0x03);
32     LCD_PulseEnable();
```

```

33     LCD_PulseEnable();
34     LCD_PulseEnable();
35     LCD_SetData(0x02);
36     LCD_PulseEnable();
37     LCD_PutCmd(0x2C);
38     LCD_PutCmd(0x0C);
39     LCD_PutCmd(0x06);
40     LCD_PutCmd(0x01);
41 }
42
43 #separate void LCD_SetPosition(unsigned int cX){
44     LCD_setData(swap(cX)|0x08);
45     LCD_PulseEnable();
46     LCD_setData(swap(cX));
47     LCD_PulseEnable();
48 }
49
50 #separate void LCD_PutChar(unsigned int cX){
51     output_high(LCD_RS);
52     LCD_PutCmd(cX);
53     output_low(LCD_RS);
54 }
55
56 #separate void LCD_PutCmd(unsigned int cX){
57     LCD_SetData(swap(cX));
58     LCD_PulseEnable();
59     LCD_SetData(swap(cX));
60     LCD_PulseEnable();
61 }
62
63 #separate void LCD_PulseEnable(void){
64     output_high(LCD_EN);
65     delay_us(3);
66     output_low(LCD_EN);
67     delay_ms(3);
68 }
69
70 #separate void LCD_SetData(unsigned int cX){
71     output_bit(LCD_D4,cX & 0x01);
72     output_bit(LCD_D5,cX & 0x02);
73     output_bit(LCD_D6,cX & 0x04);
74     output_bit(LCD_D7,cX & 0x08);
75 }

```

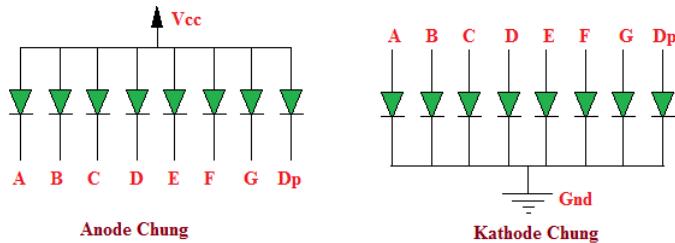
Phụ lục D

Hiển thị với LED 7 đoạn

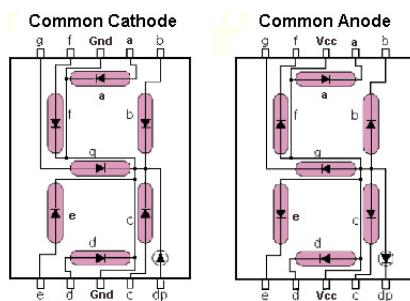
D.1 Giới thiệu

LED 7 đoạn có 2 loại là Anode chung hoặc Kathode chung, ở dạng một LED 7 đoạn hoặc nhiều LED 7 đoạn ghép lại với nhau.

Gồm 7 LED đơn mắc với nhau theo thứ tự có thể hiển thị được các số từ 0 – 7 (ngoài ra có thể có thêm dấu "." để phân cách nhiều LED 7 đoạn với nhau), ta gọi các LED đơn này là a, c, c, d, e, f, g (có thể có thêm d_p).



Hình D.1: Sơ đồ chân của LED 7 đoạn



Hình D.2: Sơ đồ chân của LED 7 đoạn Anode hoặc Kathode chung ngoài thực tế

D.1.1 LED 7 đoạn có Anode chung

- Để hiển thị thanh LED nào, ta chỉ cần nối xuống chân K của thanh LED đó xuống mass.

- Sử dụng điện trở để hạn dòng cho LED.
- Mã LED 7 đoạn Anode chung:

Số	Mã nhị phân								Mã HEX
	7	6	5	4	3	2	1	0	
	dp	g	f	e	d	c	b	a	dp
0	dp	1	0	0	0	0	0	0	40 C0
1	dp	1	1	1	1	0	0	1	79 F9
2	dp	0	1	0	0	1	0	0	24 A4
3	dp	0	1	1	0	0	0	0	30 B0
4	dp	0	0	1	1	0	0	1	19 99
5	dp	0	0	1	0	0	1	0	12 92
6	dp	0	0	0	0	0	1	0	02 82
7	dp	1	1	1	1	0	0	0	78 F8
8	dp	0	0	0	0	0	0	0	00 80
9	dp	0	0	1	0	0	0	0	10 90
A	dp	0	0	0	1	0	0	0	08 88
B	dp	0	0	0	0	0	1	1	03 83
C	dp	1	0	0	0	1	1	0	46 C6
D	dp	0	1	0	0	0	0	1	21 A1
E	dp	0	0	0	0	1	1	0	06 86
F	dp	0	0	0	1	1	1	0	0E 8E

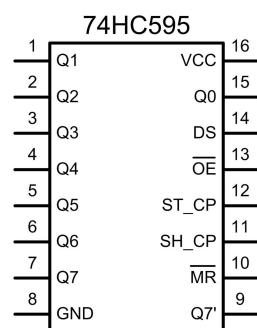
Bảng D.1: Mã LED 7 đoạn Anode chung

D.1.2 LED 7 đoạn có Kathode chung

- Để hiển thị thanh LED nào, ta chỉ cần nối xuống chân *A* của thanh LED đó lên nguồn.
- Sử dụng điện trở để hạn dòng cho LED.
- Mã LED 7 đoạn Kathode chung:

Số	Mã nhị phân									Mã HEX
	7	6	5	4	3	2	1	0		
	dp	g	f	e	d	c	b	a	dp	
0	dp	0	1	1	1	1	1	1	BF	3F
1	dp	0	0	0	0	1	1	0	86	06
2	dp	1	0	1	1	0	1	1	DB	5B
3	dp	1	0	0	1	1	1	1	CF	4F
4	dp	1	1	0	0	1	1	0	E6	66
5	dp	1	1	0	1	1	0	1	ED	6D
6	dp	1	1	1	1	1	0	1	FD	7D
7	dp	0	0	0	0	1	1	1	87	07
8	dp	1	1	1	1	1	1	1	FF	7F
9	dp	1	1	0	1	1	1	1	EF	6F
A	dp	1	1	1	0	1	1	1	F7	77
B	dp	1	1	1	1	1	0	0	FC	7C
C	dp	0	1	1	1	0	0	1	B9	39
D	dp	1	0	1	1	1	1	0	DE	5E
E	dp	1	1	1	1	0	0	1	F9	79
F	dp	1	1	1	0	0	0	1	F1	71

Bảng D.3: Mã LED 7 đoạn Kathode chung



Hình D.3: Sơ đồ chân của IC ghi dịch 74HC595

D.1.3 IC ghi dịch 74HC595

Nguyên lý hoạt động của IC 74HC595:hình D.3

- Mô tả chức năng các chân:

- Input: chân DS : đầu vào dữ liệu nối tiếp. Tại mỗi thời điểm chỉ đưa vào 1 bit.
- Output: từ chân $Q0 - Q7$. Xuất dữ liệu khi chân \overline{OE} tích cực ở mức thấp và có một xung tích cực ở sườn âm tại chân chốt ST_CP .
- Output – Enable: chân \overline{OE} : chân cho phép tích cực ở mức thấp. Khi nó ở mức cao thì không đầu ra nào được cho phép.
- SQH: chân $Q7'$: chân dữ liệu nối tiếp. Dùng để nối tiếp với IC 75HC595 khác, dữ liệu sẽ truyền cho IC tiếp theo khi nó đã nhận đủ được 8 bit.
- Shift clock: chân SH_CP : khi có một xung clock tích cực ở sườn dương thì 1 bit được dịch vào IC.
- Latch clock: chân ST_CP : xung chốt dữ liệu. Khi có một xung clock tích cực ở sườn dương thì cho phép xuất dữ liệu.
- Chiều dịch bit: từ $Q0 \rightarrow Q7$.
- Reset: chân \overline{MR} : chân này ở mức thấp thì dữ liệu sẽ bị xóa.

- Nguyên lý hoạt động:

- Khi 1 bit được đưa vào chân DS , muốn đẩy bit này vào thanh ghi thì phải kích một xung vào chân SH_CP (xung sườn dương từ 0 lên 1). Làm như vậy, cho đến khi thanh ghi đủ 8 bit.
- Dữ liệu 8 bit trong thanh ghi vẫn chưa thể xuất ra được, cần tạo một xung lên chân ST_CP (xung sườn dương từ 0 lên 1).
- Khi dữ liệu quá 8 bit thì số bit còn dư sẽ được đưa đến chân $Q7'$. Kết nối chân $Q7'$ với chân DS của IC tiếp theo để mở rộng chân.

D.2 Phương pháp điều khiển

Ta xét 2 phương pháp điều khiển: trực tiếp qua các chân I/O và gián tiếp qua IC ghi dịch.

- Phương pháp điều khiển trực tiếp qua các chân I/O:

- Ưu điểm: đơn giản.
- Nhược điểm: tốn nhiều chân của vi điều khiển, không thể điều khiển số lượng lớn các LED 7 đoạn.

- Phương pháp điều khiển gián tiếp qua IC ghi dịch:

- Ưu điểm: Dùng ít chân để điều khiển LED 7 đoạn (với IC 74HC595 chỉ dùng 3 chân của vi điều khiển có thể điều khiển được nhiều LED qua các IC ghi dịch này và có thể kết nối nhiều IC ghi dịch lại với nhau).
- Khuyết điểm: phức tạp hơn phương pháp điều khiển trực tiếp.

Khi sử dụng nhiều LED 7 đoạn được ghép chung với nhau, ta nên sử dụng phương pháp quét LED (bật tắt các LED một cách liên tục).

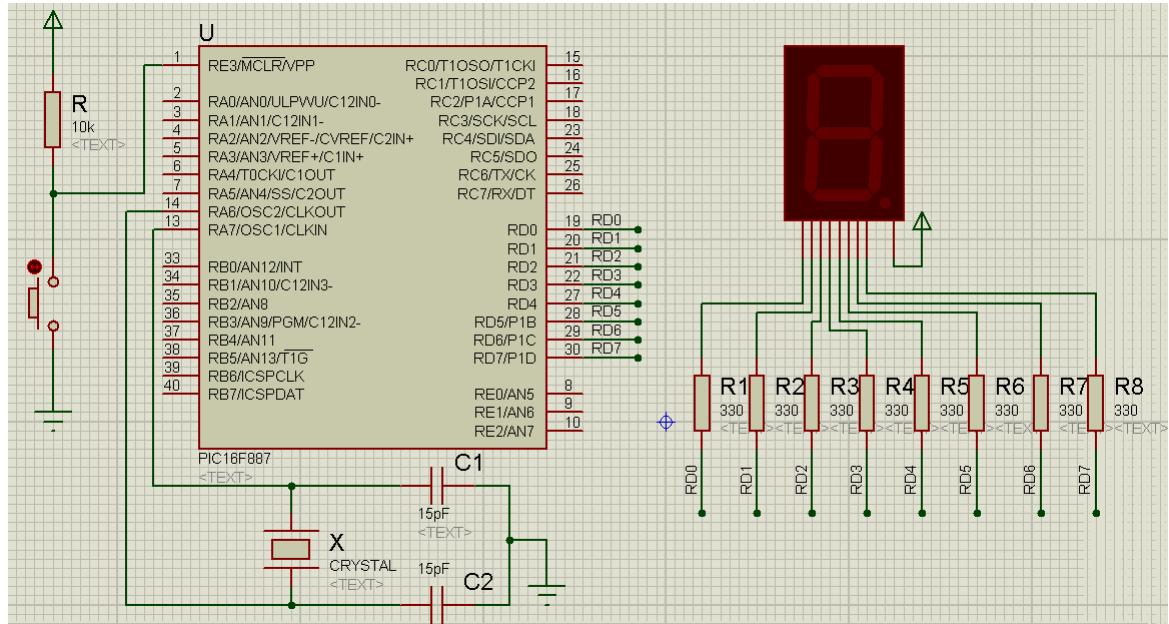
D.3 Yêu cầu

Yêu cầu 1 Sử dụng LED 7 đoạn đếm các số từ 0 – 9 và từ A – F bằng 2 phương pháp (điều khiển trực tiếp và điều khiển thông qua IC 74HC595).

D.3.1 Phương pháp điều khiển trực tiếp

Hướng giải quyết Ta xuất giá trị của mã code ứng với loại LED 7 đoạn có Anode chung hay Kathode chung cho trong bảng D.1 hoặc bảng D.3.

Sơ đồ mạch



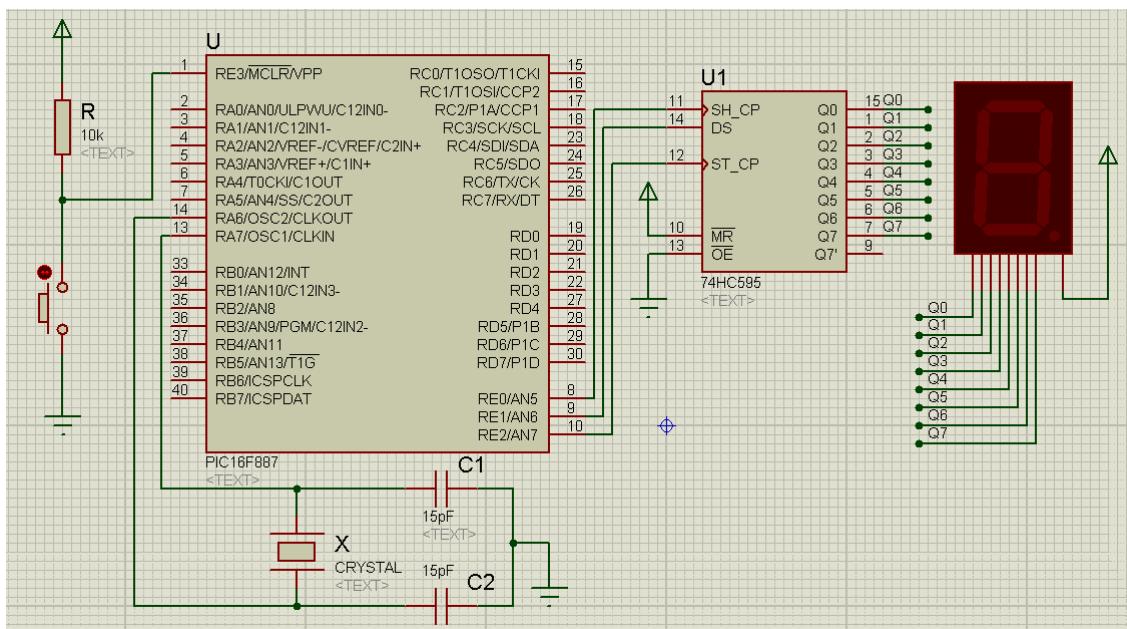
Hình D.4: Sơ đồ mạch điều khiển một LED 7 đoạn

Chương trình

```
1 /*Yeu cau:  
2     Chuong trinh hien thi so tu 0 - 9, A - F hien thi len LED 7 DOAN  
3     ( Phuong phap dieu khien truc tiep)  
4 */  
5 //Ten file: BAI-2-PHU-LUC-LED-7-SEG-V1.C  
6 #include<16F887.h> //Khai bao ten PIC  
7 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
8 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
10  
11 //Khai bao ma code cho LED 7 doan co Anode chung  
12 //Cac so tu 0 - 9; cac chu tu A - F  
13 unsigned char code7seg[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,  
    0x80, 0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E};  
14 void main(){  
15     int count;  
16     TRISD = 0x00; //Tat ca cac chan o PORT D la chan xuat (output)  
17     PORTD = 0xFF;//Cho Off tat ca cac LED  
18  
19     while (True){  
20         for (count = 0; count <= 16; count ++){  
21             PORTD = code7seg[count]; //Lay vi tri thu count trong mang  
                Code7seg  
22             delay_ms(1000);  
23         }  
24     }  
25 }
```

D.3.2 Phương pháp điều khiển thông qua IC ghi dịch 74HC595

Sơ đồ mạch



Hình D.5: Sơ đồ mạch điều khiển một LED 7 đoạn qua IC 74HC595

Chương trình

```

1 /*Yeu cau:
2   Chuong trinh hien thi so tu 0 - 9, A - F hien thi len LED 7 DOAN
3   ( Phuong phap dieu khien thong qua IC ghi dich 74HC595)
4 */
5 //Ten file: BAI-2-PHU-LUC-LED-7-SEG-V2.C
6 #include<16F887.h> //Khai bao ten PIC
7 #include<def_887.h> //Thu vien do nguoi dung dung nghia
8
9 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz
11
12 #define Clock PIN_E0 //Chan SH_CP
13 #define Data PIN_E1 //Chan SD
14 #define Latch PIN_E2 //Chan ST_CP
15
16 void add_byte_74hc595(unsigned char data_byte); //Ghi 1 byte vao IC
17 void output_byte_74hc595(); //Cho phep xuat du lieu ra cac chan Q0 - Q7

```

```

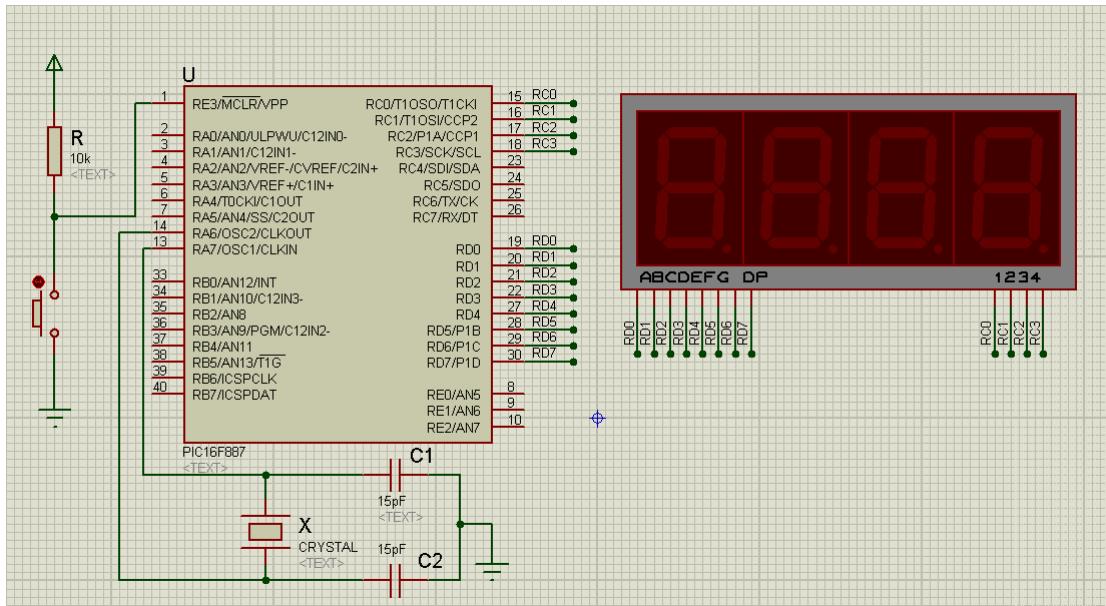
18
19 //Khai bao ma code cho LED 7 doan co Anode chung
20 //Cac so tu 0 - 9; cac chu tu A - F
21 unsigned char code7seg[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,
22   0x80, 0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E};
23 void main(){
24   int i;
25   TRISE = 0x00; //Tat ca cac chan o PORT E la chan xuat (output)
26
27   while (True){
28     for (i = 0; i <= 16; i++){
29       add_byte_74hc595(Code7seg[i]);
30       output_byte_74hc595();
31       delay_ms(1000);
32     }
33   }
34 }
35
36 void add_byte_74hc595(unsigned char data_byte){//Ghi 1 byte vao IC
37   int i;
38   output_low(Clock); //Muc thap
39   for (i=0; i<8; i++){
40     //Lay ra tung bit de ghi vao chip
41     if ((data_byte & 0x80) == 0){ //So sanh bit
42       output_low(Data); //ghi vao IC bit 0
43     }
44     else{
45       output_high(Data); //ghi vao IC bit 1
46     }
47
48     data_byte = data_byte << 1; //Bo di bit da duoc o tren
49     output_high(Clock); //Muc cao
50     //Ghi duoc 1 bit vao IC
51
52     //Tiep tuc ghi cac bit tiep theo
53     output_low(Clock); //Muc thap
54   }
55 }
56
57 void output_byte_74hc595(){//Cho phep xuat du lieu trong ra cac chan Q0 - Q7
58   output_low(Latch); //Muc thap
59   delay_us(5);
60   output_high(Latch); //Muc cao
61   //Cho phep xuat du lieu
62 }

```

Yêu cầu 2 Sử dụng 4 LED 7 đoạn hiển thị các số từ 0 – 9999 bằng 2 phương pháp (điều khiển trực tiếp bằng các chân và điều khiển thông qua IC 74HC595).

D.3.3 Phương pháp quét LED – Điều khiển trực tiếp qua các chân của vi điều khiển

Sơ đồ mạch



Hình D.6: Sơ đồ mạch điều khiển 4 LED 7 đoạn

Hướng giải quyết

- Thường các LED loại này các thanh a, b, c, d, e, f, g (có thể có dp) là chung với nhau, nó chỉ phân biệt nhau bằng số tự của LED 7 đoạn (mỗi LED 7 đoạn sẽ có một chân điều khiển).
- Do đó, để hiển thị với nhiều LED 7 đoạn, ta dùng phương pháp quét LED: lặp lại quá trình bậc tắt các LED, khi đó sẽ đánh lừa được thị giác con người (do mắt người chỉ nhìn thấy được 24 điểm ảnh trên 1s).
- Với yêu cầu 2, ta cần tách ra các chữ số riêng lẻ từ số ban đầu, rồi cho mỗi LED 7 đoạn hiển thị một số.

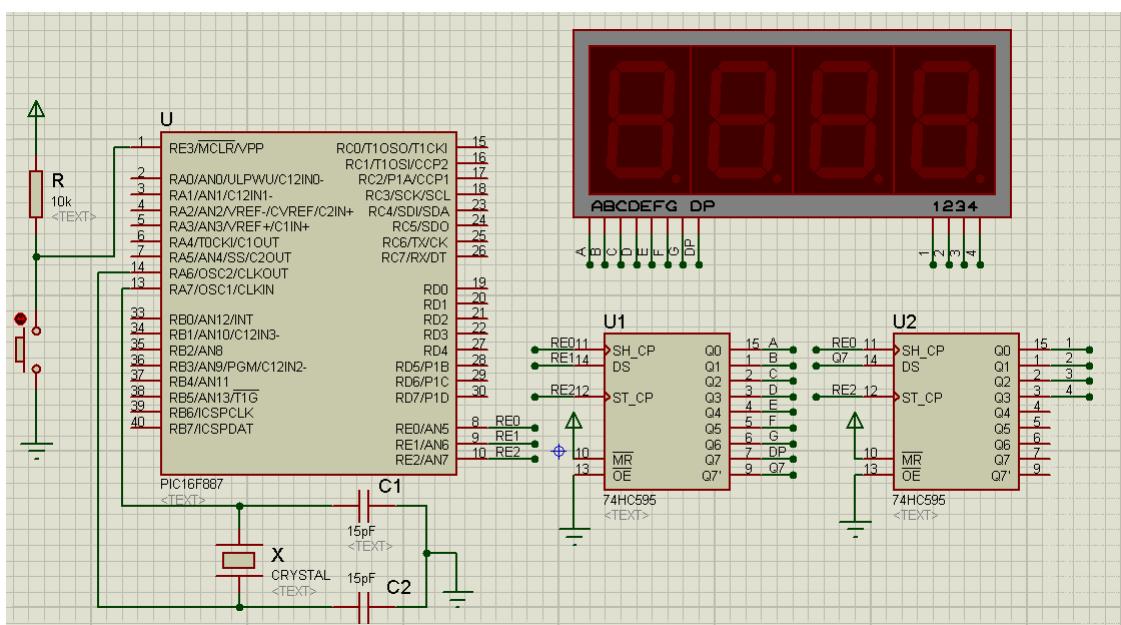
Chương trình

```
1 /*Yeu cau:
2     Chuong trinh hien thi so tu 0 - 9, A - F hien thi len LED 7 DOAN
3     ( Phuong phap dieu khien truc tiep)
4 */
5 //Ten file: BAI-2-PHU-LUC-4-LED-7-SEG-V1.C
6 #include<16F887.h> //Khai bao ten PIC
7 #include<def_887.h> //Thu vien do nguoi dung dinh nghia
8 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
9 #use delay(clock = 20000000) //Tan so thach anh 20MHz
10
11 //Khai bao ma code cho LED 7 doan co Anode chung
12 //Cac so tu 0 - 9; cac chu tu A - F
13 unsigned char code7seg[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,
14     0x80, 0x90};
15 unsigned char digits[] = {PIN_C0, PIN_C1, PIN_C2, PIN_C3};
16 void main(){
17     unsigned long count,i;
18     int a,b,c,d;
19
20     TRISD = 0x00; //Chan Ouput, dieu khien 7 thanh LED
21     TRISC = 0x00; // Chan Output, dieu khien 4 LED
22     PORTC = 0x00;
23
24     while (True){
25         for (count = 0; count <=9999; count++){
26             a = count / 1000;
27             b = (count%1000)/100;
28             c = (count%100)/10;
29             d = (count%100)%10;
30
31             for (i=0; i<250; i++){
32                 output_high(PIN_C3);
33                 PORTD = Code7seg[d];
34                 delay_ms(1);
35                 output_low(PIN_C3);
36
37                 output_high(PIN_C2);
38                 PORTD = Code7seg[c];
39                 delay_ms(1);
40                 output_low(PIN_C2);
41
42                 output_high(PIN_C1);
43                 PORTD = Code7seg[b];
44                 delay_ms(1);
45                 output_low(PIN_C1);
46
47                 output_high(PIN_C0);
48                 PORTD = Code7seg[a];
```

```
48           delay_ms(1);
49           output_low(PIN_C0);
50       }
51   }
52 }
53 }
```

D.3.4 Phương pháp quét LED – Điều khiển thông qua IC 74HC595

Sơ đồ mạch



Hình D.7: Sơ đồ mạch điều khiển 4 LED 7 đoạn qua IC 74HC595

Hướng giải quyết

- Để tiết kiệm số chân điều khiển của vi điều khiển, ta dùng 2 IC 75HC595 để điều khiển 4 LED 7 đoạn.
 - Hoạt động theo nguyên tắc ngăn xếp: đưa vào sau sẽ lấy ra trước.
 - IC đầu tiên khi nhận đủ 8 bit, nếu tiếp tục gửi bit thì nó sẽ chuyển số bit dữ sang chân $Q7'$ để chuyển sang IC thứ 2.

Chương trình

```
1 /*Yeu cau:  
2     Chuong trinh hien thi so tu 0-9999 hien thi len 4 LED 7 DOAN  
3     ( Phuong phap dieu khien thong qua IC ghi dich 74HC595)  
4 */  
5 //Ten file: BAI-2-PHU-LUC-4-LED-7-SEG-V2.C  
6 #include<16F887.h> //Khai bao ten PIC  
7 #include<def_887.h> //Thu vien do nguoi dung dinh nghia  
8  
9 #FUSES NOWDT, HS, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT  
10 #use delay(clock = 20000000) //Tan so thach anh 20MHz  
11  
12 #define Clock PIN_E0 //Chan SH_CP  
13 #define Data PIN_E1 //Chan SD  
14 #define Latch PIN_E2 //Chan ST_CP  
15  
16 void add_byte_74hc595(unsigned char data_byte); //Ghi 1 byte vao IC  
17 void output_byte_74hc595(); //Cho phep xuat du lieu trong ra cac chan Q0 -  
    Q7  
18  
19  
20 //Khai bao ma code cho LED 7 doan co Anode chung  
21 //Cac so tu 0 - 9; cac chu tu A - F  
22 unsigned char code7seg[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,  
    0x80, 0x90};  
23 unsigned char digits[] = {0x01, 0x02, 0x04, 0x08};  
24 void main(){  
25     unsigned long count, i;  
26     int a,b,c,d;  
27     TRISE = 0x00; //Tat ca cac chan o PORT E la chan xuat (output)  
28  
29     while (True){  
30         for (count = 0; count <= 9999; count++){  
31             a = count / 1000;  
32             b = (count%1000)/100;  
33             c = (count%100)/10;  
34             d = (count%100)%10;  
35             for (i = 0; i < 1500; i ++){  
36                 add_byte_74hc595(digits[0]);  
37                 add_byte_74hc595(Code7seg[a]);  
38                 output_byte_74hc595();  
39  
40                 add_byte_74hc595(digits[1]);  
41                 add_byte_74hc595(Code7seg[b]);  
42                 output_byte_74hc595();  
43  
44                 add_byte_74hc595(digits[2]);
```

```

45         add_byte_74hc595(Code7seg[c]);
46         output_byte_74hc595();
47
48         add_byte_74hc595(digits[3]);
49         add_byte_74hc595(Code7seg[d]);
50         output_byte_74hc595();
51     }
52 }
53 }
54 }
55 }

56 void add_byte_74hc595(unsigned char data_byte){//Ghi 1 byte vao IC
57     int i;
58     output_low(Clock); //Muc thap
59     for (i=0; i<8; i++){
60         //Lay ra tung bit de ghi vao chip
61         if ((data_byte & 0x80) == 0){ //So sanh bit
62             output_low(Data); //ghi vao IC bit 0
63         }
64         else{
65             output_high(Data); //ghi vao IC bit 1
66         }
67
68         data_byte = data_byte << 1; //Bo di bit da duoc o tren
69         output_high(Clock); //Muc cao
70         //Ghi duoc 1 bit vao IC
71
72         //Tiep tuc ghi cac bit tiep theo
73         output_low(Clock); //Muc thap
74     }
75 }
76 }

77 void output_byte_74hc595(){//Cho phep xuat du lieu trong ra cac chan Q0 - Q7
78     output_low(Latch); //Muc thap
79     delay_us(5);
80     output_high(Latch); //Muc cao
81     //Cho phep xuat du lieu
82
83 }

```

Phụ lục E

Thư viện RTC DS3231

Thư viện gồm 2 file DS3231.H và DS3231.C được viết bởi tác giả sshahryiar trên diễn đàn <https://www.ccsinfo.com>¹

Nội dung file DS3231.H

```
1 #define DS3231_Address           0x68 //Dia chi cua thiet bi I2C
2
3 //Them bit 1 vao cuoi 7 bit dia chi de doc du lieu tu thiet bi I2C
4 #define DS3231_Read_addr         ((DS3231_Address << 1) | 0x01) //0xD1
5 //Them bit 0 vao cuoi 7 bit dia chi de ghi lenh len thiet bi I2C
6 #define DS3231_Write_addr        ((DS3231_Address << 1) & 0xFE) //0xD0
7
8 #define secondREG                0x00 //Dia chi cua thanh ghi giay
9 #define minuteREG               0x01 //Dia chi cua thanh ghi phut
10 #define hourREG                 0x02 //Dia chi cua thanh ghi gio
11 #define dayREG                  0x03 //Dia chi cua thanh ghi thu
12 #define dateREG                 0x04 //Dia chi cua thanh ghi ngay
13 #define monthREG                0x05 //Dia chi cua thanh ghi thang
14 #define yearREG                 0x06 //Dia chi cua thanh ghi nam
15 #define alarm1secREG            0x07
16 #define alarm1minREG            0x08
17 #define alarm1hrREG              0x09
18 #define alarm1dateREG            0x0A
19 #define alarm2minREG            0x0B
20 #define alarm2hrREG              0x0C
21 #define alarm2dateREG            0x0D
22 #define controlREG               0x0E
23 #define statusREG                0x0F
24 #define ageoffsetREG             0x10
25 #define tempMSBREG               0x11 //Dia chi 1 cua thanh ghi nhiet do
26 #define tempLSBREG               0x12 //Dia chi 2 cua thanh ghi nhiet do
27
28 #define _24_hour_format          0 //Dinh dang 24h
29 #define _12_hour_format          1 //Dinh dang 12h
```

¹<https://www.ccsinfo.com/forum/viewtopic.php?t=50256>

```

30 #define am          0 //Gio am
31 #define pm          1 //Gio pm
32
33 //Do gia tri luu trong IC dang BCD, nen viet cac ham chuyen doi
34 unsigned char bcd_to_decimal(unsigned char d); //Chueyn so BCD sang so thap
      phan, dung khi doc duu lieu
35 unsigned char decimal_to_bcd(unsigned char d); //Chuyen so thap phan sang
      so BCD, dung khi viet du lieu
36 unsigned char DS3231_Read(unsigned char address); //Doc duu lieu tu dia chi
      cua thanh ghi
37 void DS3231_Write(unsigned char address, unsigned char value); //Ghi gia
      tri value vao dia chi cua thanh ghi
38 void DS3231_init(); //Kich hoat IC
39 //Lay thoi gian tu IC ra
40 void getTime(unsigned char &p3, unsigned char &p2, unsigned char &p1, short
      &p0, short hour_format);
41 void getDate(unsigned char &p4, unsigned char &p3, unsigned char &p2,
      unsigned char &p1);
42 //Cai dat thoi gian vao IC
43 void setTime(unsigned char hSet, unsigned char mSet, unsigned char sSet,
      short am_pm_state, short hour_format);
44 void setDate(unsigned char daySet, unsigned char dateSet, unsigned char
      monthSet, unsigned char yearSet);
45 float getTemp(); //Lay nhiet do tu IC ra

```

Nội dung file DS3231.C

```

1 #include "DS3231.h"
2
3 // Thuc hien chuye so BCD sang so thap phan
4 //Tach 8 bit cua ma BCD: chuyen 4 bit do thanh mo so thap phan
5 //Ghep 2 so thap phan do lai voi nhau.
6 //Ham nay dung chuyen doi khi doc du lieu
7 unsigned char bcd_to_decimal(unsigned char d){
8     return ((d & 0x0F) + (((d & 0xF0) >> 4) * 10));
9 }
10
11 //Chuyen so thap phan sang ma BCD
12 //Bien doi tung so thap phan thanh ma nhi phan
13 //Ghep cac ma nhi phan nay theo thu tu duoc ma BCD
14 //Ham nay dung khi muon ghi vao IC
15 unsigned char decimal_to_bcd(unsigned char d){
16     return (((d / 10) << 4) & 0xF0) | ((d % 10) & 0x0F);
17 }
18

```

```

19
20 //Doc du lieu tu thanh ghi dia chi
21 unsigned char DS3231_Read(unsigned char address){
22     unsigned char value = 0;
23     I2C_start();           //Tao dieu kien Start
24     I2C_write(DS3231_Write_addr); //Ghi len dia chi (da them bit 1) vao cuoi
25         7 bit
26     I2C_write(address);      //Gui lenh yeu cau (tuy thuoc vao gia tri
27         address)
28     I2C_start();           //Tao dieu kien Start, giao tiep moi
29     I2C_write(DS3231_Read_addr); //Ghi len dia chi (da them bit 1) vao cuoi 7
30         bit
31     value = I2C_read(0);    //Gui lenh doc du lieu tu IC
32     I2C_stop();            //Tao dieu kien Stop, ket thuc giao tiep
33     return value;
34 }
35
36 //Ghi du lieu tu thanh ghi dia chi
37 //Giai thich tuong tu nhu ham DS3231_Read
38 void DS3231_Write(unsigned char address, unsigned char value){
39     I2C_start();
40     I2C_write(DS3231_Write_addr);
41     I2C_write(address);
42     I2C_write(value);
43     I2C_stop();
44 }
45
46 //Khoi tao hoat dong cho IC
47 void DS3231_init(){
48     DS3231_Write(controlREG, 0x00);
49     DS3231_Write(statusREG, 0x08);
50 }
51
52 //Lay thoi gian (h,m,s) tu IC, doc duoc o gia tri BCD --> so thap phan
53 void getTime(unsigned char &p3, unsigned char &p2, unsigned char &p1, short
54     &p0, short hour_format){
55     unsigned char tmp = 0;
56     p1 = DS3231_Read(secondREG);
57     p1 = bcd_to_decimal(p1);
58     p2 = DS3231_Read(minuteREG);
59     p2 = bcd_to_decimal(p2);
60     switch(hour_format){
61         case 1:{           tmp = DS3231_Read(hourREG);
62             tmp &= 0x20;
63             p0 = (short)(tmp >> 5);
64             p3 = (0x1F & DS3231_Read(hourREG));

```

```

63         p3 = bcd_to_decimal(p3);
64         break;
65     }
66     default: {
67         p3 = (0x3F & DS3231_Read(hourREG));
68         p3 = bcd_to_decimal(p3);
69         break;
70     }
71 }
72 }

73 //Lay thoi gian (year,month,date,day) tu IC, doc duoc o gia tri BCD --> so
74 //thap phan
75 void getDate(unsigned char &p4, unsigned char &p3, unsigned char &p2,
76             unsigned char &p1){
77     p1 = DS3231_Read(yearREG);
78     p1 = bcd_to_decimal(p1);
79     p2 = (0x1F & DS3231_Read(monthREG));
80     p2 = bcd_to_decimal(p2);
81     p3 = (0x3F & DS3231_Read(dateREG));
82     p3 = bcd_to_decimal(p3);
83     p4 = (0x07 & DS3231_Read(dayREG));
84     p4 = bcd_to_decimal(p4);
85 }
86 //Cai dat gio, phut, giay vao IC
87 void setTime(unsigned char hSet, unsigned char mSet, unsigned char sSet,
88              short am_pm_state, short hour_format)
89 {
90     unsigned char tmp = 0;
91     DS3231_Write(secondREG, (decimal_to_bcd(sSet)));
92     DS3231_Write(minuteREG, (decimal_to_bcd(mSet)));
93     switch(hour_format){
94         case 1:{
95             switch(am_pm_state){
96                 case 1:{
97                     tmp = 0x60;
98                     break;
99                 }
100                default:{
101                    tmp = 0x40;
102                    break;
103                }
104            }
105            DS3231_Write(hourREG, ((tmp | (0x1F &
106                (decimal_to_bcd(hSet))))));
107            break;
108        }
109    }

```

```

107
108     default:{  

109         DS3231_Write(hourREG, (0x3F & (decimal_to_bcd(hSet))));  

110         break;  

111     }  

112 }  

113 }  

114  

115 //Cai dat thu, ngay, thang, nam vao IC  

116 void setDate(unsigned char daySet, unsigned char dateSet, unsigned char  

    monthSet, unsigned char yearSet){  

117     DS3231_Write(dayREG, (decimal_to_bcd(daySet)));  

118     DS3231_Write(dateREG, (decimal_to_bcd(dateSet)));  

119     DS3231_Write(monthREG, (decimal_to_bcd(monthSet)));  

120     DS3231_Write(yearREG, (decimal_to_bcd(yearSet)));  

121 }
122  

123 //Lay gia tri nhiet do tu IC  

124 float getTemp(){  

125     register float t = 0.0;  

126     unsigned char lowByte = 0;  

127     signed char highByte = 0;  

128     lowByte = DS3231_Read(tempLSBREG);  

129     highByte = DS3231_Read(tempMSBREG);  

130     lowByte >>= 6;  

131     lowByte &= 0x03;  

132     t = ((float)lowByte);  

133     t *= 0.25;  

134     t += highByte;  

135     return t;  

136 }

```

Phụ lục F

Các khai báo sau #FUSES

<i>STT</i>	<i>Từ viết tắt</i>	<i>Ý nghĩa</i>
<i>WATCH DOG TIMER</i>		
1	NOWDT	Không sử dụng bộ Watch Dog Timer
2	WDT	Sử dụng bộ Watch Dog Timer
<i>HIGH SPEED OSC</i>		
3	LP	Sử dụng nguồn dao động tần số thấp $f < 200kHz$
4	XT	Dao động thạch anh $f < 4MHz$ với PCM/PCH
		Dao động thạch anh $f = 3MHz - 10MHz$ với PCD
5	RC	Dao động RC với CLKOUT
6	HS	Dao động tần số cao
		$f > 4MHz$ với PCM/PCH hoặc $f > 10MHz$ với PCD
<i>POWER UP TIMER</i>		
7	NOPUT	Không sử dụng Power Up Timer
8	PUT	Sử dụng Power Up Timer
<i>BROWN OUT</i>		
9	NOBROWNOUT	Không reset chip khi BrownOut
10	BROWNOUT	Reset chip khi BrownOut
<i>LOW VOLTAGE PROGRAM</i>		
11	NOLVP	Không lập trình điện áp thấp, B3 (PIC16); B5 (PIC18) là I/O
12	LVP	Lập trình điện áp thấp trên B3 (PIC16); B5 (PIC18)
<i>CODE PROTECED EEPROM</i>		
13	NOCPD	Không bảo vệ dữ liệu EEPROM
14	CPD	Bảo vệ dữ liệu EEPROM
<i>PROGRAM WRITE PROTECED</i>		
15	WRT	Bộ nhớ chương trình viết được bảo vệ
16	WRT_50%	Nửa phần dưới của bộ nhớ chương trình viết được bảo vệ
17	WRT_5%	Bộ nhớ chương trình viết ít hơn 255 byte thì được bảo vệ
18	NOWRT	Bộ nhớ chương trình viết không được bảo vệ
<i>DEBUG FOR ICD</i>		
19	NODEBUG	Không sử dụng chế độ Debug với ICD
20	DEBUG	Sử dụng chế độ Debug với ICD
<i>CODE PROTECED FROM READING</i>		
21	NOPROTECT	Cho phép đọc lại code
22	PROTECT	Không cho phép đọc lại code

Bảng F.2: Ý nghĩa của các khai báo trong #FUSES
113