

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP  
**KHOA ĐIỆN TỬ**  
**BỘ MÔN TIN HỌC CÔNG NGHIỆP**

**BÀI TẬP**  
**MÔN HỌC HỆ THỐNG NHÚNG**



**Thái Nguyên, năm 2022**

## CÁC DẠNG BÀI TẬP

|  |    |
|--|----|
| <b>CCS FOR PIC .....</b>                           | 4  |
| 1. Giới thiệu về CCS.....                          | 4  |
| 2. Giao diện CCS .....                             | 4  |
| 3. Cấu trúc một chương trình C cho PIC .....       | 4  |
| 4. Các lệnh truy xuất trong phần mềm CCS .....     | 7  |
| 5. Chương trình đầu tiên .....                     | 8  |
| 6. Cấu trúc điều khiển .....                       | 10 |
| <b>DẠNG 1: CƠ BẢN.....</b>                         | 15 |
| 1. Giao tiếp Led đơn.....                          | 15 |
| 2. Giao tiếp Led 7 đoạn .....                      | 18 |
| 3. Giao tiếp nút nhấn đơn .....                    | 20 |
| 4. Bài tập sinh viên tự thực hiện .....            | 23 |
| <b>DẠNG 2: MA TRẬN PHÍM .....</b>                  | 25 |
| 1. Nguyên tắc quét phím:.....                      | 25 |
| 2. Cấu tạo ma trận phím từ các phím nhấn đơn ..... | 25 |
| 3. Phương pháp quét phím:.....                     | 26 |
| 4. Bài tập tự thực hiện .....                      | 27 |
| <b>DẠNG 3: LCD .....</b>                           | 28 |
| 1. Giới thiệu LCD 16x2 .....                       | 28 |
| 2. Các lệnh sử dụng trong LCD của PIC 16F877A..... | 28 |
| 3. Ứng dụng.....                                   | 28 |
| 4. Bài tập tự thực hiện .....                      | 30 |
| <b>DẠNG 4: CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ.....</b>    | 31 |
| 1. ADC của Pic 16F877A .....                       | 31 |
| 2. Các thanh ghi sử dụng trong ADC .....           | 31 |
| 3. Các bước chuyển đổi ADC .....                   | 32 |

|                                    |   |    |
|------------------------------------|---|----|
| 4.                                 | <b>Lựa chọn nguồn xung chuyển đổi.....</b>                    | 32 |
| 5.                                 | <b>Các lệnh sử dụng trong chuyển đổi ADC .....</b>            | 32 |
| 6.                                 | <b>Phương pháp tính giá trị ADC từ tín hiệu đầu vào .....</b> | 33 |
| 7.                                 | <b>Ứng dụng.....</b>  | 33 |
| 8.                                 | <b>Bài tập tự thực hiện .....</b>                             | 34 |
| <b>DẠNG 5: TIMER/COUNTER .....</b> |   | 36 |
| 1.                                 | <b>Giới thiệu: .....</b>                                      | 36 |
| 2.                                 | <b>Các thanh ghi sử dụng trong Timer/Counter .....</b>        | 36 |
| 3.                                 | <b>Tính giá trị cho các Timer/Counter .....</b>               | 38 |
| 4.                                 | <b>Các lệnh sử dụng trong Timer/Counter .....</b>             | 39 |
| 5.                                 | <b>Lập trình cho Timer1 .....</b>                             | 40 |
| 6.                                 | <b>Ứng dụng.....</b>  | 40 |
| 7.                                 | <b>Bài tập tự thực hiện .....</b>                             | 42 |



## CCS FOR PIC

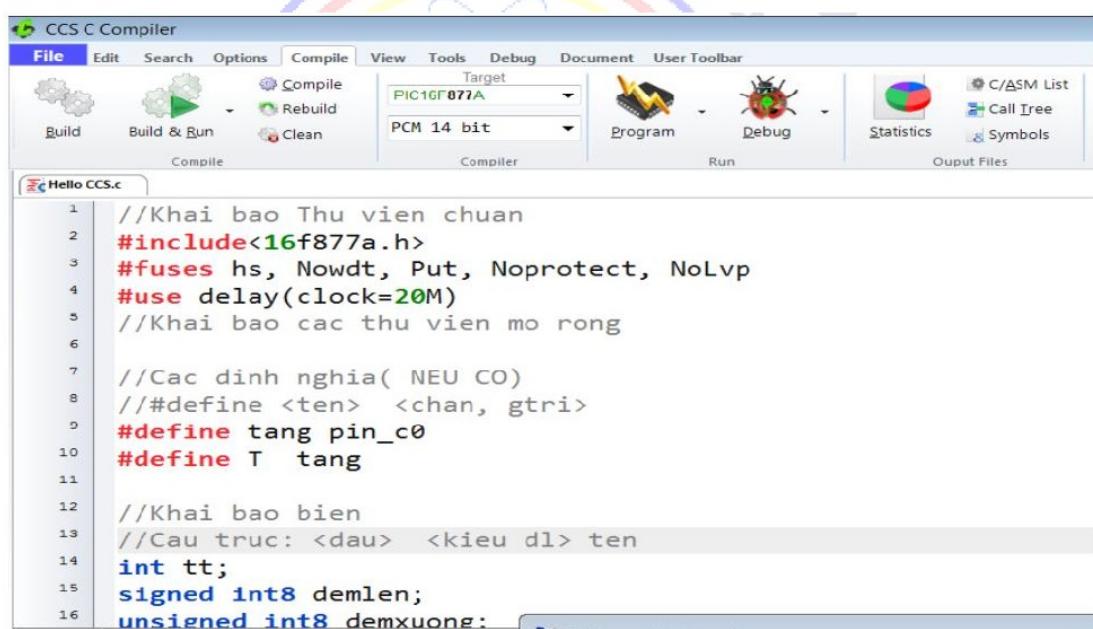
### 1. Giới thiệu về CCS

Trình biên dịch PIC-C sử dụng để biên dịch tập tin mã nguồn C cho họ Vi điều khiển PIC. Khi lập trình cho Vi điều khiển PIC dùng PIC-C thì các thành phần cơ bản cũng giống như các họ Vi điều khiển khác, sự khác biệt chủ yếu nằm ở phần cứng của từng họ Vi điều khiển. Các lệnh liên quan đến phần cứng được trình bày theo phần cứng đó, ví dụ:

- Truy xuất đến các Port thì các lệnh Pic-C liên quan đến các Port
- Sử dụng Timer/Counter thì sử dụng các lệnh liên quan đến Timer

Khi sử dụng phần mềm PIC-C để lập trình cho Vi điều khiển PIC thì bắt buộc phải khai báo thư viện <16F8xxx.h>

### 2. Giao diện CCS



a. Khai báo thư viện

```
#include <Tên thư viện.h>
```

Ví dụ:

```
#include <16F877A.h>
#include <GiaotiepLed_1.h>
```

```
#use delay(clock=20M)
#fuses hs, put, noproTECT, nolVP, nowDT
```

|                  |   |
|------------------|---|
| <b>NOWDT</b>     | Không sử dụng bộ định thời giám sát (No watchdog timer)   |
| <b>PUT</b>       | Sử dụng bộ định thời khi có nguồn để kéo dài thêm thời gian reset vi điều khiển để chờ nguồn điện ổn định, thời gian kéo dài thêm 72ms (Power up timer) |
| <b>HS</b>        | Sử dụng bộ dao động tần số cao từ 4MHz đến 20MHz (High Speed)   |
| <b>NOPROTECT</b> | Không sử dụng bảo vệ mã code nạp vào bộ nhớ flash bên trong   |
| <b>NOLVP</b>     | Không sử dụng chế độ nạp code dùng nguồn điện áp thấp 5V mà dùng nguồn 12,5V.   |

b. Các định nghĩa (nếu có)

**#define tên chân**

Ví dụ:

```
#define Tang Pin_A0
#define Led Pin_B2
```

c. Khai báo biến (nếu có)

**Kiểu DL tên biến [Giá trị khởi tạo]**

➤ Kiểu dữ liệu

| <b>Kiểu</b>    | <b>Kích thước</b>        | <b>Phạm vi biểu diễn</b> |                          |
|----------------|--------------------------|--------------------------|--------------------------|
|                |                          | <b>Unsigned</b>          | <b>Signed</b>            |
| <b>int1</b>    | True hay false (0 hay 1) | 0 ÷ 1                    | Không có                 |
| <b>int8</b>    | Số nguyên 1 byte (8 bit) | 0 ÷ 255                  | -128 ÷ 127               |
| <b>int16</b>   | Số nguyên 16 bit         | 0 ÷ 65535                | -32768 ÷ 32767           |
| <b>int32</b>   | Số nguyên 32 bit         | 0 ÷ 4294967295           | -2147483648 ÷ 2147483647 |
| <b>float32</b> | Số thực 32 bit           | -1.5 x 1045 ÷ 3.4 x 1038 |                          |

➤ Ví dụ:

- **int1** TT; // biến TT là kiểu dữ liệu 1bit chỉ có hai giá trị là 0 và 1.
- **signed int8** dem; // biến dem là kiểu số nguyên giá trị 8bit và có dấu.
- **unsigned int8** tam; // biến tam là kiểu số nguyên giá trị 8bit không dấu

- **const unsigned int8** ma7doan [10] =  
 $\{0XC0,0XF9,0xA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90\};$   
//Mảng ma7doan gồm 10 phần tử có giá trị được xác định trong {.....}
- **float** doc; //biến doc là kiểu số thực tức có dấu , phía sau

d. Các toán tử

| TT | Toán tử | Chức năng                                    | Ví dụ                      |
|----|---------|--|----------------------------|
| 1  | +       | Toán tử cộng                                 |                            |
| 2  | +=      | Toán tử cộng và gán                          | $x+=y$ tương đương $x=x+y$ |
| 3  | &=      | Toán tử and và gán                           | $x&=y$ tương đương $x=x&y$ |
| 4  | @       | Toán tử địa chỉ                              |                            |
| 5  | &       | Toán tử and                                  |                            |
| 6  | ^=      | Toán tử xor và gán                           | $x^=y$ tương đương $x=x^y$ |
| 7  | ^       | Toán tử xor                                  |                            |
| 8  | =       | Toán tử or và gán                            | $x =y$ tương đương $x=x y$ |
| 9  |         | Toán tử or                                   |                            |
| 10 | --      | Toán tử giảm                                 |                            |
| 11 | /=      | Toán tử chia và gán                          | $x/=y$ tương đương $x=x/y$ |
| 12 | /       | Toán tử chia                                 |                            |
| 13 | ==      | Toán tử bằng dùng trong phép so sánh         |                            |
| 14 | >       | Toán tử lớn hơn                              |                            |
| 15 | >=      | Toán tử lớn hơn hoặc bằng                    |                            |
| 16 | ++      | Toán tử tăng                                 |                            |
| 17 | *       | Toán tử truy xuất gián tiếp đi trước con trỏ |                            |
| 18 | !=      | Toán tử không bằng                           |                            |
| 19 | <=      | Toán tử dịch trái và gán                     | $x<=y$ tương đương $x=x<y$ |
| 20 | <       | Toán tử nhỏ hơn                              |                            |
| 22 | <<      | Toán tử dịch trái                            |                            |
| 23 | &&      | Toán tử and dùng trong phép so sánh          |                            |

|    |     |                                     |                                |
|----|-----|-------------------------------------|--------------------------------|
| 24 | !   | Toán tử not dùng trong phép so sánh |                                |
| 25 |     | Toán tử or dùng trong phép so sánh  |                                |
| 26 | %=  | Toán tử chia lấy số dư và gán       | x% = y tương đương x = x%y     |
| 27 | %   | Toán tử modul                       |                                |
| 28 | *=  | Toán tử nhân và gán                 | x * = y tương đương x = x * y  |
| 29 | *   | Toán tử nhân                        |                                |
| 30 | ~   | Toán tử bù 1                        |                                |
| 31 | >>= | Toán tử dịch phải và gán            | x >>= y tương đương x = x >> y |
| 32 | >>  | Toán tử dịch phải                   |                                |
| 33 | ->  | Toán tử con trả cấu trúc            |                                |
| 34 | --= | Toán tử trừ và gán                  |                                |
| 35 | -   | Toán tử trừ                         |                                |

e. Viết chương trình con (nếu có)

```
Void ten_chuongtrinh_con([khai_bao_bien])
{
    Code here
}
```

f. Viết chương trình chính

```
Void main()
{
    Set_tris_cổng(giá_trị);
    Set_tris_A(0x00);
    Set_tris_B(0x80)
    While(true)
    {
        Code here
    }
}
```

#### 4. Các lệnh truy xuất trong phần mềm CCS

| Lệnh | Chức năng | Ví dụ |
|------|-----------|-------|
|      |           |       |

|                                |   |  |
|--------------------------------|---|--|
| <b>OUTPUT_LOW(pin)</b>         | có chức năng cho 1 chân của port xuống mức 0          | <b>output_low(Pin_A0);</b> // cho chân A0 xuống mức thấp;  |
| <b>OUTPUT_HIGH(pin)</b>        | có chức năng cho 1 chân của port lên mức 1            | <b>output_high(pin_a0);</b> //cho chân A0 lên mức cao.   |
| <b>OUTPUT_BIT (pin, value)</b> | có chức năng xuất giá trị value ra 1 chân của port.   | <b>output_bit(pin_a0,1);</b> //xuất tín hiệu mức cao cho chân A0.  |
| <b>OUTPUT_X(value)</b>         | có chức năng xuất dữ liệu 8 bit ra port X             | <b>output_a(0x0f);</b> // xuất tín hiệu 0000 1111 cho Port A.  |
| <b>SET_TRIS_X(value)</b>       | : có chức năng định cấu hình cho port X               | <b>set_tris_a(0xf0);</b> // định cấu hình cho port A là 1111 0000 có nghĩa là từ chân A7 - A4 là tín hiệu đi vào vi điều khiển, từ chân A3 - A0 là tín hiệu đi ra vi điều khiển. |
| <b>INPUT(pin)</b>              | có chức năng đọc giá trị 1 chân của port gán cho biến | <b>input(pin_a0);</b> // đọc tín hiệu vào vi điều khiển là mức thấp hay mức cao.   |

## 5. Chương trình đầu tiên

- a. Viết chương trình để đếm từ 00-99 trên 2 Led 7 đoạn

- **Không sử dụng chương trình con**

```
#include <16f877a.h>
#use delay(clock=20M)
unsigned int8 Maled[10]={0xC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8,
0X80, 0X90};
int8 i, dem;
void main()
{
    set_tris_B(0x00);
    set_tris_C(0x00);
    while(TRUE)
    {
        output_B(0x00);
        output_C(0x00);
        dem=0;
        for(i=0; i<=99; i++)
        {

```

```

        output_B(maled[dem/10]);lấy giá trị Led hàng chục
        output_C(maled[dem%10]);lấy giá trị cho led hàng đơn
        delay_ms(200);
        dem++;
    }
}
}

```

- **Sử dụng chương trình con**

```

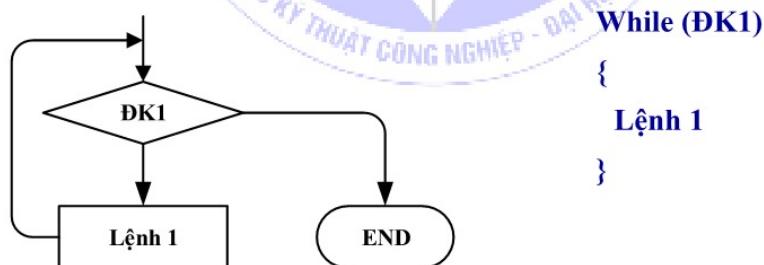
#include <7doan_1.h>
#use delay(clock=20M)
const unsigned int8 maled[10]={0xC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82,
0XF8, 0X80, 0X90};
INT8 I;
VOID GAIMA()
{
    OUTPUT_B(MALED[I/10]);
    OUTPUT_C(MALED[I%10]);
}
void main()
{
SET_TRIS_B(0X00);
SET_TRIS_C(0X00);
while(TRUE)
{
FOR(I=0; I<=99; I=I+5)
{
    GAIMA();
    DELAY_MS(200);
}
}
}

```

b. Led kết nối vào chân C0, viết chương trình để led sáng/tắt sau mỗi 500ms

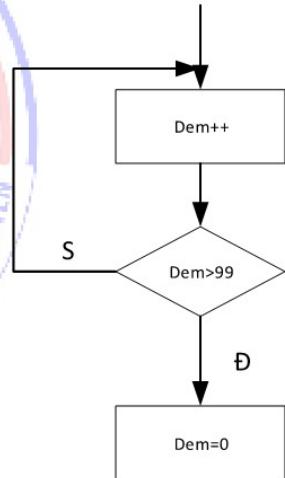
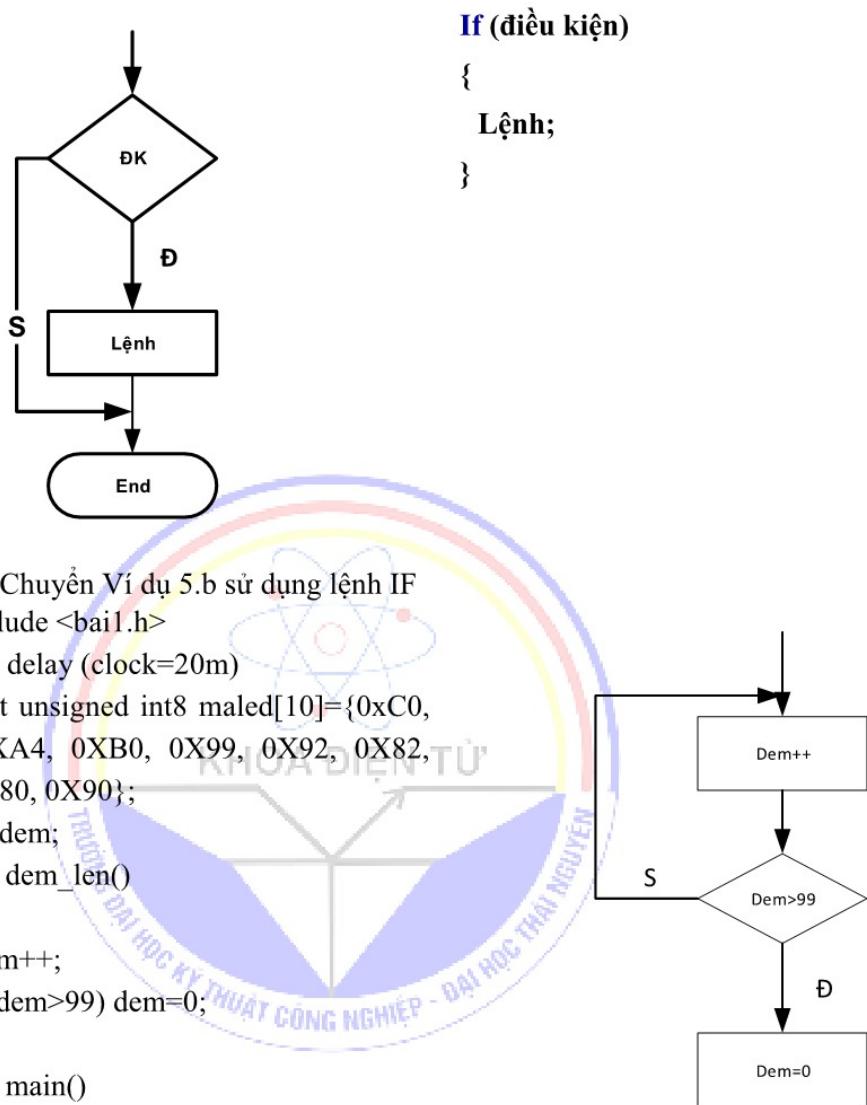
```
#include <GiaotiepLed_1.h>
#use delay(clock=20M)
void nhapnhay()
{
    output_high(pin_C0);
    delay_ms(500);
    output_low(pin_C0);
    delay_ms(500);
}
void main()
{
    set_tris_C(0x00);
    while(TRUE)
    {
        nhapnhay();
    }
}
```

6. Cấu trúc điều khiển  
a. Cấu trúc While



b. Cấu trúc If

- **Chỉ có If:** Cấu trúc này chỉ thực hiện <Lệnh> khi đúng điều kiện còn đúng nếu sai thì bỏ qua kết thúc lệnh đầu tiên.

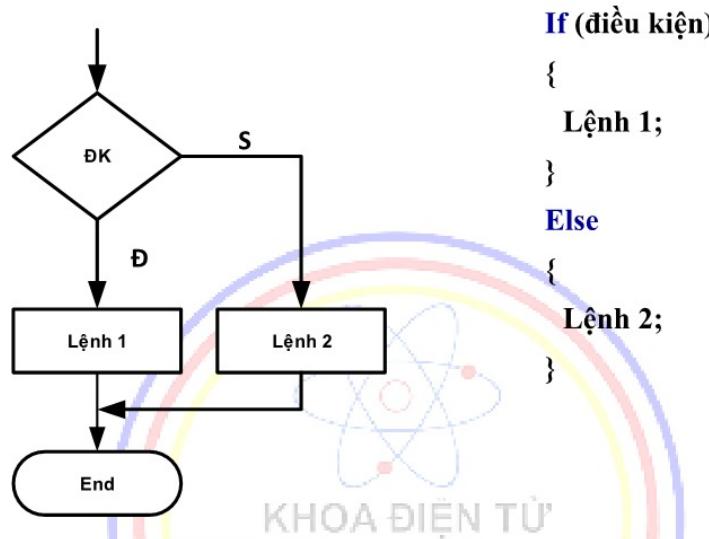


```

        delay_ms(200);
    }
}

```

- **If....Else:** Cấu trúc này kiểm tra điều kiện có đúng không, nếu đúng thì thực hiện 1 và kết thúc, còn ngược lại nếu sai thì chạy tiếp chương trình thực hiện hai và kết thúc.



- Viết chương trình để đếm từ 00-99 và từ 99-00 sử dụng 2 Led 7 đoạn kết nối đến PortB và PortC (**không sử dụng chương trình con**)

```

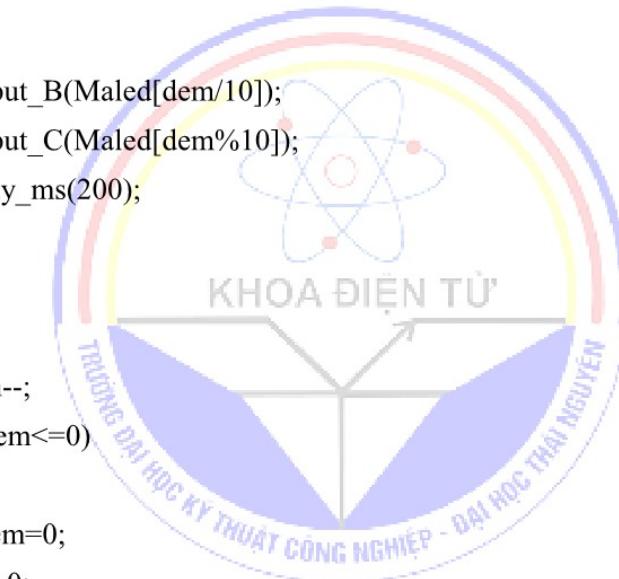
#include <16f877a.h>
#use delay(clock=20M)
unsigned int8 Maled[10]={0xC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8,
0X80, 0X90};
int1 TT;
int8 dem;
void main()
{
    Set_tris_B(0x00);
    Set_tris_C(0x00);
    dem=0;
    TT=0;

```

```

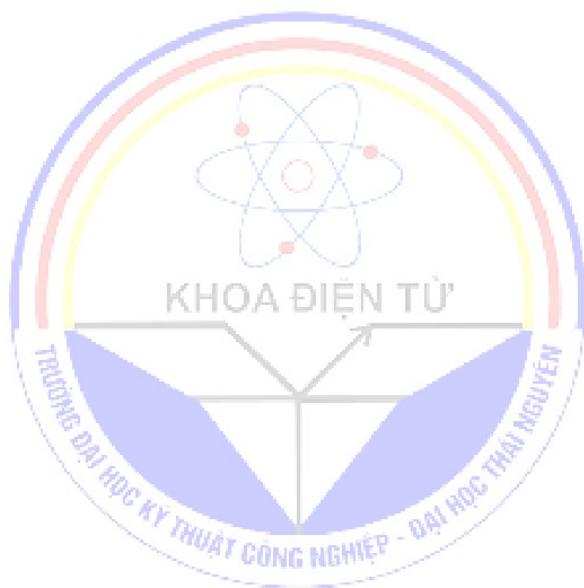
while(TRUE)
{
    if(TT==0)
    {
        dem++;
        if(dem>99)
        {
            dem=99;
            TT=1;
        }
        output_B(Maled[dem/10]);
        output_C(Maled[dem%10]);
        delay_ms(200);
    }
    else
    {
        dem--;
        if(dem<=0)
        {
            dem=0;
            tt=0;
        }
        output_B(Maled[dem/10]);
        output_C(Maled[dem%10]);
        delay_ms(200);
    }
}
}

```



## 7. Bài tập tự thực hiện

- Viết chương trình để đếm từ 00-99 và từ 99-00 sử dụng 2 Led 7 đoạn kết nối đèn PortB và PortC (**sử dụng chương trình con**)

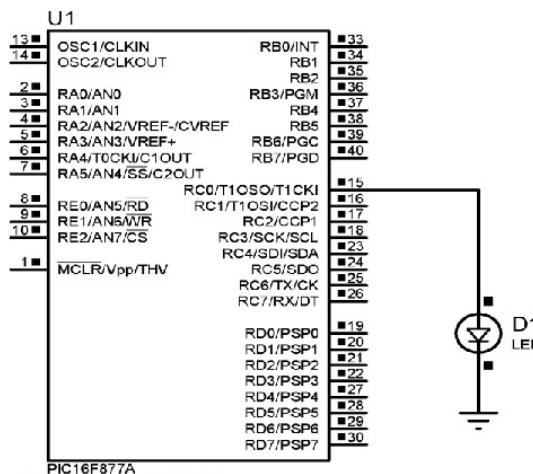


## DẠNG 1: CƠ BẢN

### 1. Giao tiếp Led đơn

#### a. Led kết nối vào chân C0, viết chương trình để led sáng/tắt sau mỗi 500ms

```
#include <GiaotiepLed_1.h>
#use delay(clock=20M)
void nhaphay()
{
    output_high(pin_C0);
    delay_ms(500);
    output_low(pin_C0);
    delay_ms(500);
}
void main()
{
    set_tris_C(0x00);
    while(TRUE)
    {
        nhaphay();
    }
}
```



#### b. 8 Led sang đuôi từ Trái qua Phải

##### Xây dựng bảng trạng thái Led sáng

- Tắt đuôi: cho Led sáng hết, sau đó cho Led số một (kết nối chân B0) tắt, Led số 2 tắt, .... Led số 7 tắt, Led số 8(kết nối chân B7) sáng

Trạng thái các Led được thể hiện theo bảng dưới

| STT<br>Led sáng | Trạng thái (bit 1: Sáng) |   |   |   |   |   |   |  | Hex |
|-----------------|--------------------------|---|---|---|---|---|---|--|-----|
|                 | 7                        | 6 | 5 | 4 | 3 | 2 | 1 |  |     |
| 1               | 0                        | 0 | 0 | 0 | 0 | 0 | 0 |  | 01h |
| 2               | 0                        | 0 | 0 | 0 | 0 | 0 | 1 |  | 02h |
| 3               | 0                        | 0 | 0 | 0 | 0 | 1 | 0 |  | 04h |

|   |   |   |   |   |   |   |   |  |     |
|---|---|---|---|---|---|---|---|--|-----|
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |  | 08h |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |  | 10h |
| 6 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |  | 20h |
| 7 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |  | 40h |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  | 80h |

- Sáng dòn

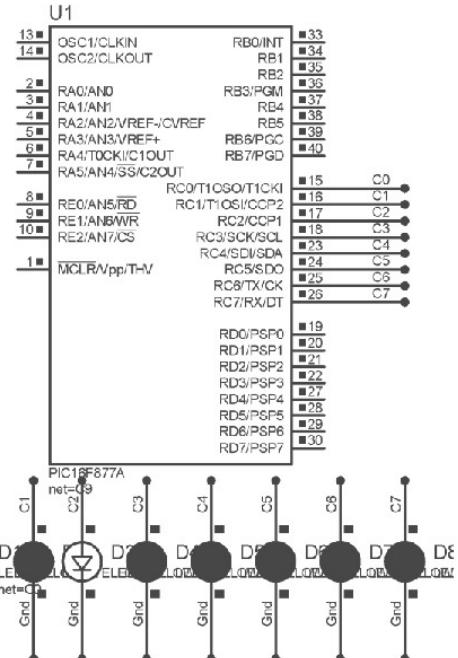
- Cho 8 led đơn kết nối đến Port C của PIC, viết chương trình cho phép Led sáng dòn từ chân C0 đến C7 theo bảng mô tả dưới đây:

| STT<br>Led sáng | Trạng thái (bit 1: Sáng) |   |   |   |   |   |   |   |  | Hex |
|-----------------|--------------------------|---|---|---|---|---|---|---|--|-----|
|                 | 7                        | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |     |
| 1               | 0                        | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  | 01h |
| 2               | 0                        | 0 | 0 | 0 | 0 | 0 | 1 | 1 |  | 03h |
| 3               | 0                        | 0 | 0 | 0 | 0 | 1 | 1 | 1 |  | 07h |
| 4               | 0                        | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  | 0Fh |
| 5               | 0                        | 0 | 0 | 1 | 1 | 1 | 1 | 1 |  | 1Fh |
| 6               | 0                        | 0 | 1 | 1 | 1 | 1 | 1 | 1 |  | 3Fh |
| 7               | 0                        | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  | 7Fh |
| 8               | 1                        | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  | FFh |

```

#include<GiaotiepLed_1.h>
#use delay(clock=20M)
void sang_duoit()
{
    int x, i;
    x=0x80;           //1000.0000
    output_C(x);
    delay_ms(500);
    for(i=0; i<=7; i++)
    {
        x=x>>1;
        output_c(x);
        delay_ms(500);
    }
}
void main()
{
    set_tris_C(0x00);
    while(TRUE)
    {
        output_C(0x00);
        sang_duoit();
    }
}

```



### c. Mở rộng chân cho Led đơn

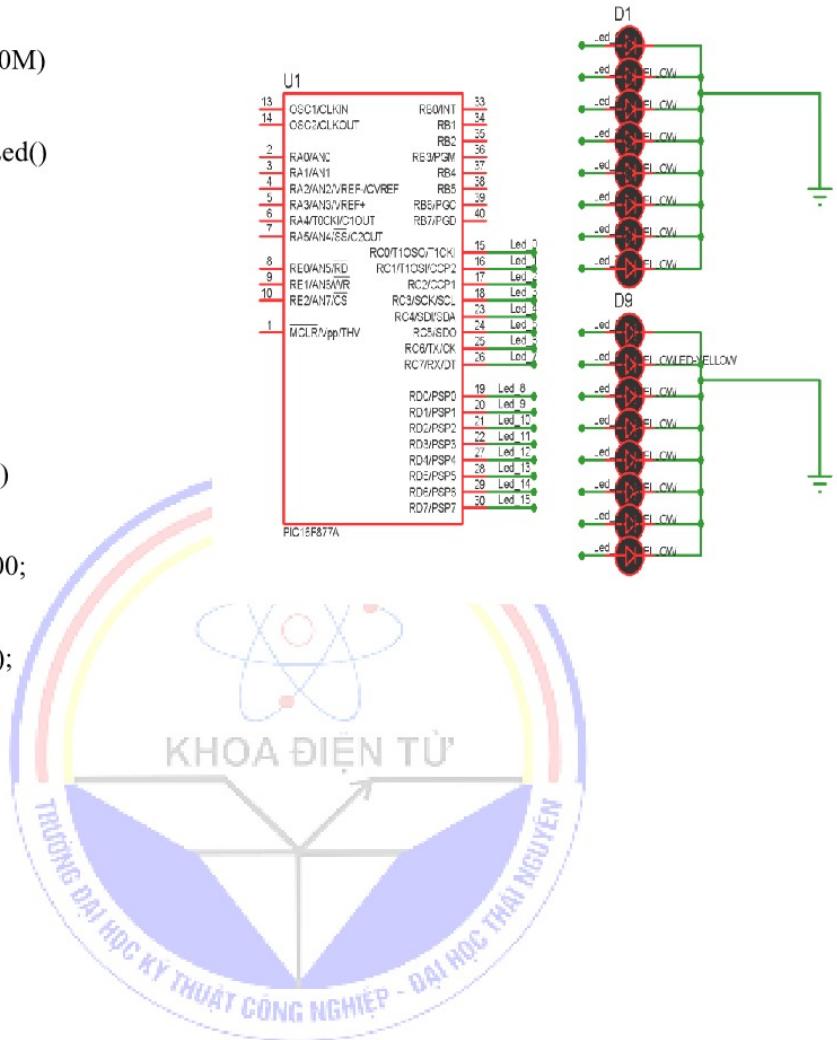
Cho 16 led đơn kết nối đến Port C và D của PIC, viết chương trình cho phép Led sáng dần từ chân D7 đến D0 và C7 đến C0 theo bảng mô tả dưới đây

```

#include <Vidu.h>
#use delay(clock=20M)
unsigned int16 i, x;
void sang_don_16Led()
{
    x=0x8000;
    output_C(x);
    output_D(x>>8);
    delay_ms(200);

    for(i=0; i<16; i++)
    {
        x=(x>>1)|0x8000;
        output_C(X);
        output_D(x>>8);
        delay_ms(200);
    }
}
void main()
{
    set_tris_C(0x00);
    output_C(0x00);
    output_d(0x00);
    while(TRUE)
    {
        sang_don_16Led()
    }
}

```



## 2. Giao tiếp Led 7 đoạn

### a. Kết nối 2 Led 7 đoạn vào Port B và C, cho Led đếm dần từ 00-99

```

#include <7doan_1.h>
#use delay(clock=20M)
const      unsigned      int8
maled[10]={0xC0, 0XF9, 0XA4,
0XB0, 0X99, 0X92, 0X82, 0XF8,
0X80, 0X90};

INT8 I;
VOID GAIMA()
{
OUTPUT_B(MALED[I/10]);
OUTPUT_C(MALED[I%10]);
}

```

```
void main()
```

```
{
SET_TRIS_B(0X00);
SET_TRIS_C(0X00);
while(TRUE)
```

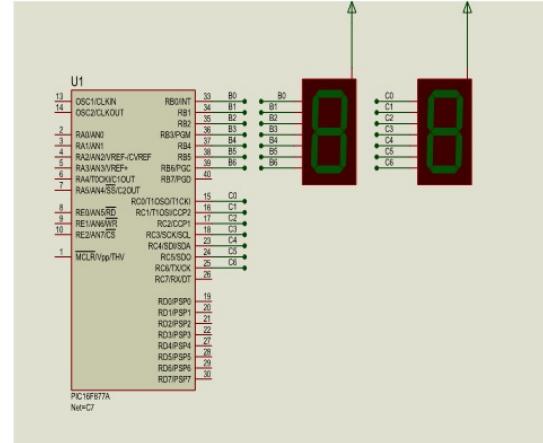
```
{
FOR(I=0; I<=99; I=I+5)
```

```
{
GAIMA();
DELAY_MS(200);
```

```
}
```

```
}
```

```
}
```



**KHOA ĐIỆN TỬ**

**TỰ HỌC KỸ THUẬT CÔNG NGHIỆP - ĐẠI HỌC THÁI NGUYỄN**

### b. Kết nối 4 Led 7 đoạn vào Port B và C, cho Led hiển thị bốn số cuối của MSSV.

```

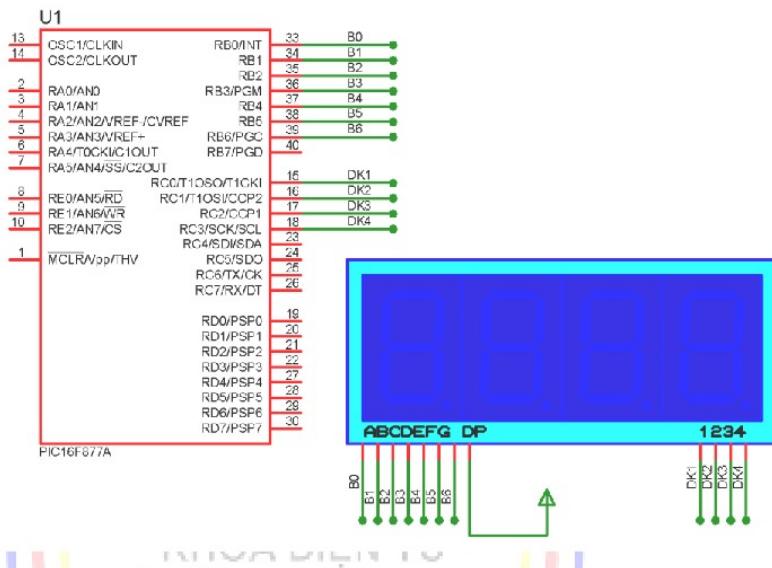
#include <4_led_7_doan.h>
#use delay(clock=20M)
void main()
{
    while(TRUE)
    {
        output_B(0xc0); output_high(pin_C0); delay_ms(1); output_low(pin_C0);
        output_B(0xf9); output_high(pin_C1); delay_ms(1); output_low(pin_C1);
    }
}

```

```

        output_B(0xa4); output_high(pin_C2); delay_ms(1); output_low(pin_C2);
        output_B(0x99); output_high(pin_C3); delay_ms(1); output_low(pin_C3);
    }
}

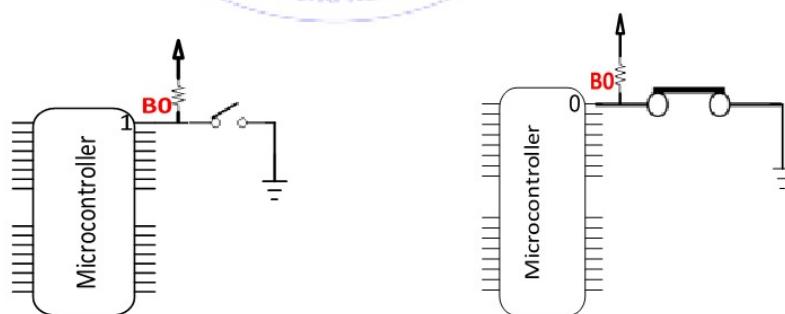
```



### 3. Giao tiếp nút nhấn đơn

#### a. Nguyên tắc hoạt động của nút nhấn

- Chân B<sub>0</sub> được nối đến chân của vi điều khiển nên B<sub>0</sub>=1, chân còn lại được nối Mass

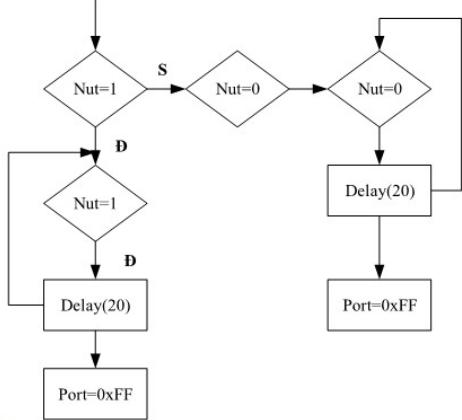


- Khi nhấn phím, chân B<sub>0</sub> sẽ được nối Mass và B<sub>0</sub>=0

#### Ví dụ

- Nút nhấn xuống Led sáng, nhả ra Led tắt

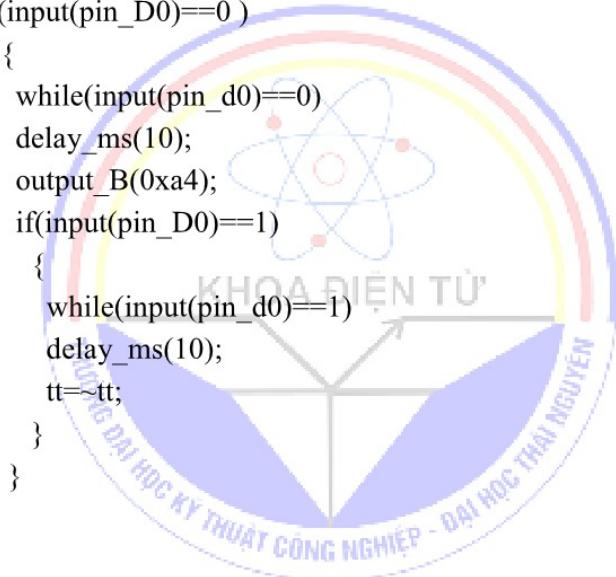
```
#include <Button_Led.h>
#use delay(clock=20M)
void main()
{
    set_tris_B(0xff);
    set_tris_C(0x00);
    output_C(0x00);
    while(TRUE)
    {
        if(input(pin_b0)==1)
        {
            while(input(pin_b0)==1);
            delay_ms(100);
            output_C(0xff);
        }
        else
        {
            while(input(pin_b0)==0);
            delay_ms(100);
            output_C(0x00);
        }
    }
}
```

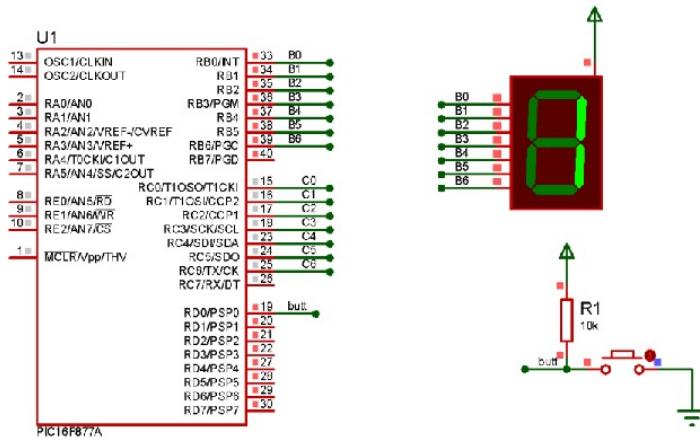


### b. Nhấn nút lần 1 Led sáng số 1, nhấn lần 2 led sáng số 2

```
#include <7seg_1Butt_00_99.h>
#use delay(clock=20m)
0X90, 0X88, 0X83, 0XC6, 0XA1, 0X86, 0X8E};
int8 tt=1;
void main()
{
    output_d(0xff);
    while(TRUE)
    {
        if(input(pin_D0)==0 )
```

```
{  
    while(input(pin_d0)==0)  
    delay_ms(10);  
    output_B(0xf9);  
    if(input(pin_D0)==1)  
    {  
        while(input(pin_d0)==1)  
        delay_ms(10);  
        tt=~tt;  
    }  
}  
if(input(pin_D0)==0 )  
{  
    while(input(pin_d0)==0)  
    delay_ms(10);  
    output_B(0xa4);  
    if(input(pin_D0)==1)  
    {  
        while(input(pin_d0)==1)  
        delay_ms(10);  
        tt=~tt;  
    }  
}  
}
```





#### 4. Bài tập sinh viên tự thực hiện

BT1. Kết nối 8 Led đơn vào Port B, cho Led thực hiện:

- a. Sáng đuôi
- b. Tắt dần

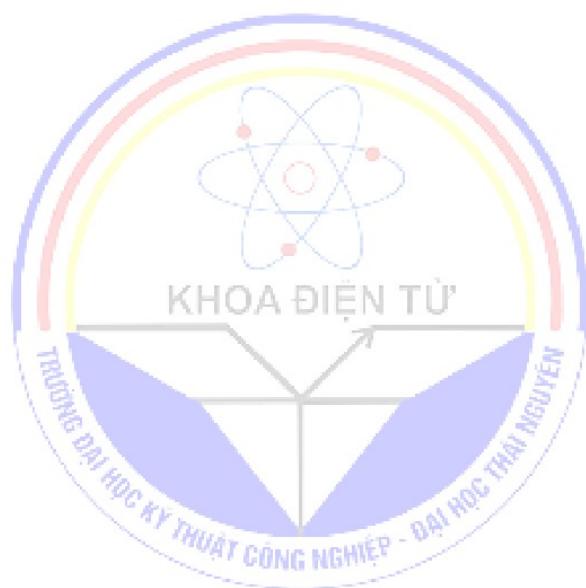
BT2. Kết nối 16 Led đơn vào Port C và D của PIC, viết chương trình cho phép Led sáng dần từ chân D7 đến D0 và C7 đến C0 theo bảng mô tả dưới đây:

| STT<br>Led sáng | Trạng thái (bit 1: Sáng) |   |   |   |   |   |   |   | Hex |
|-----------------|--------------------------|---|---|---|---|---|---|---|-----|
|                 |                          |   |   |   |   |   |   |   |     |
| 1               |                          |   |   |   |   |   |   |   | 1 1 |
| 2               |                          |   |   |   |   |   |   | 1 | 1   |
| 3               |                          |   |   |   |   |   | 1 | 1 |     |
| 4               |                          |   |   |   | 1 | 1 |   |   |     |
| 5               |                          |   |   | 1 | 1 |   |   |   |     |
| 6               |                          |   | 1 | 1 |   |   |   |   |     |
| 7               | 1                        | 1 |   |   |   |   |   |   |     |
| 8               |                          |   |   |   |   |   | 1 | 1 |     |
| 9               |                          |   |   |   |   | 1 | 1 |   |     |
| 10              |                          |   |   |   |   | 1 | 1 |   |     |
| 11              |                          |   |   |   | 1 | 1 |   |   |     |
| 12              |                          |   |   | 1 | 1 |   |   |   |     |

|    |   |   |   |  |  |  |  |  |
|----|---|---|---|--|--|--|--|--|
| 13 |   | 1 | 1 |  |  |  |  |  |
| 14 | 1 | 1 |   |  |  |  |  |  |
| 15 |   |   |   |  |  |  |  |  |
| 16 |   |   |   |  |  |  |  |  |

BT3. Kết nối 8Led đơn vào PortC, Led 7 đoạn vào PortB. Viết chương trình để Led 7 đoạn sáng các số 1, 2, 3.. mỗi khi thay đổi kiểu của 8Led đơn

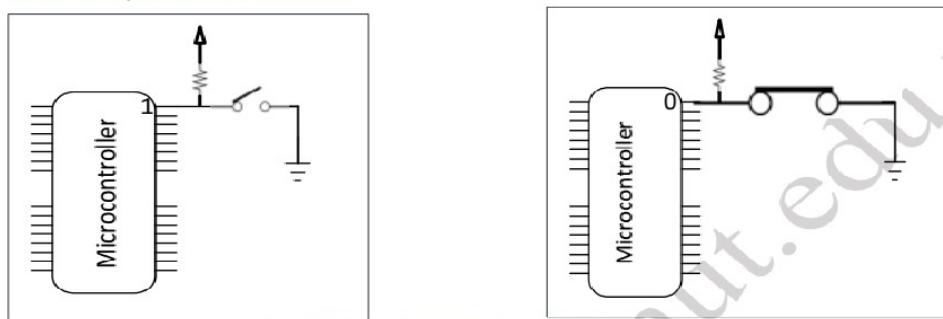
BT4. Kết nối 3 Led 7 đoạn, Nhấn nút tăng 1 đơn vị bắt đầu từ 000-999



## DẠNG 2: MA TRẬN PHÍM

### 1. Nguyên tắc quét phím:

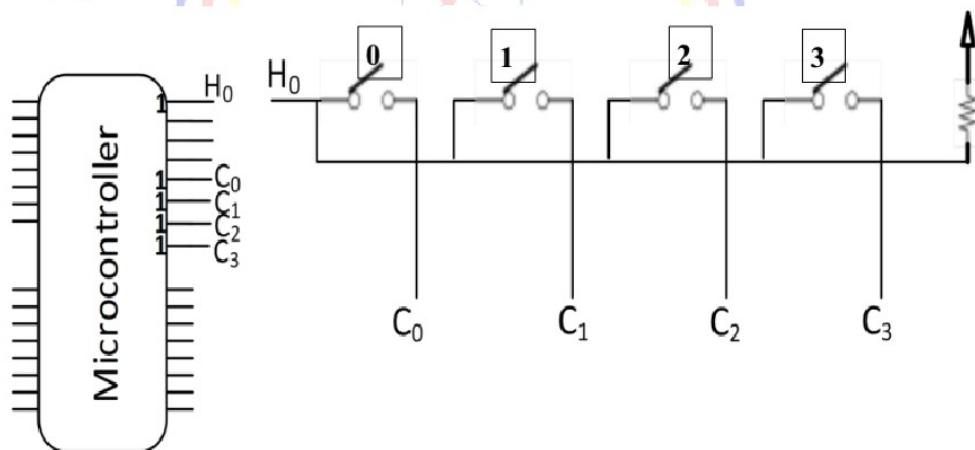
Cấu tạo ma trận phím: ma trận phím được cấu tạo nên từ các phím nhấn đơn, phím nhấn đơn hoạt động như sau:



Chân H0 được nối đến chân của vi điều khiển nên  $H0=1$ , chân còn lại được nối Mass

### 2. Cấu tạo ma trận phím từ các phím nhấn đơn

Ma trận phím 1 hàng 4 cột 1x4 được cho như hình dưới đây:



Khi chưa được nhấn, các chân nối hàng  $H_0 = 1$  và các chân nối cột  $C_0C_1C_2C_3$

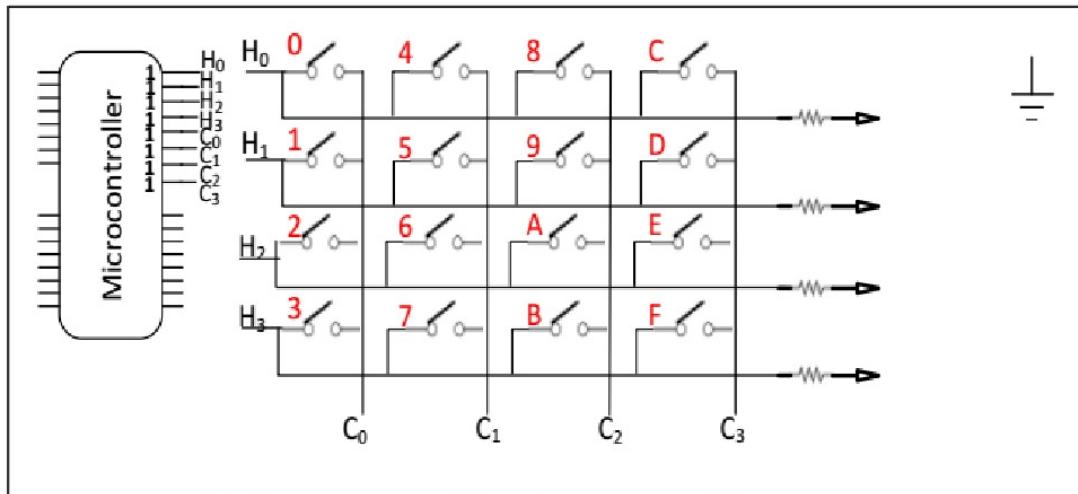
đều có giá trị =1 (do đều được nhấn đến các chân của vi điều khiển). Để kiểm tra phím nào được nhấn bằng cách cho 1 trong các chân nối đến cột từ  $C_0 \div C_3$  lần lượt = 0

Ví dụ:

- Nếu  $C_0 = 0$ , phím số “0” được nhấn (do chân  $C_0$  được xem như nối Mass)
- Nếu  $C_1 = 0$ , phím số “1” được nhấn (do chân  $C_1$  được xem như nối Mass)

- Nếu  $C_2 = 0$ , phím số “2” được nhấn (do chân  $C_2$  được xem như nút Mass)
- Nếu  $C_2 = 0$ , phím số “3” được nhấn (do chân  $C_3$  được xem như nút Mass)

*Mã trận phím 4 hàng 4 cột 4x4 được cho như hình dưới đây:*



- Cho hàng là ngõ vào và cột là ngõ ra. Khi chưa có phím nào được nhấn trạng thái các chân tương ứng với các Hàng và các Cột như sau:

| H3 | H2 | H1 | H0 | C3 | C2 | C1 | C0 |
|----|----|----|----|----|----|----|----|
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

- Thông thường, để kiểm tra phím ta sử dụng phương pháp quét Cột, đó là lần lượt cho các cột từ C0 đến C3 bằng 0 và kiểm tra xem nếu Hàng nào có mức 0 thì phím tương ứng sẽ được nhấn. Mã phím được tính theo công thức sau:

$$MP = Cột * 4 + Hàng$$

Ví dụ:

- Nếu  $H2 = 0$  và  $C1 = 0$ , thì phím có mã “6” được nhấn ( $MP = 1*4+2=6$ )
- Nếu  $H0 = 0$  và  $C3 = 0$ , thì phím có mã “C” được nhấn ( $MP = 3*4+0=12=C$ )

### 3. Phương pháp quét phím:

- Nguyên tắc quét:

- Lần lượt kiểm tra 4 Cột, xem có phím nào được nhấn không?
- Nếu có trả về MP là 1 trong 15 phím (từ phím “0” đến phím “15 hay F”
- Nếu không trả về **MP=0xFF**
- Xây dựng mã quét phím

| H <sub>3</sub> | H <sub>2</sub> | H <sub>1</sub> | H <sub>0</sub> | C <sub>3</sub> | C <sub>2</sub> | C <sub>1</sub> | C <sub>0</sub> | Mã Hex |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|--------|
| 1              | 1              | 1              | 1              | 1              | 1              | 1              | 0              | 0xFE   |
| 1              | 1              | 1              | 1              | 1              | 1              | 0              | 1              | 0xFD   |
| 1              | 1              | 1              | 1              | 1              | 0              | 1              | 1              | 0xFB   |
| 1              | 1              | 1              | 1              | 0              | 1              | 1              | 1              | 0xF7   |

- Chương trình con quét phím

```
unsigned int8 quet_phim()
{
    unsigned int8 cot, mp=0xff;
    unsigned int8 maquet[4]={0xEF, 0xDF, 0xBF, 0x7F};
    for(cot=0; cot<=3; cot++)
    {
        output_B(maquet[cot]);
        while(input(pin_b0)==0) mp=cot*4+0;
        while(input(pin_b1)==0) mp=cot*4+1;
        while(input(pin_b2)==0) mp=cot*4+2;
        while(input(pin_b3)==0) mp=cot*4+3;
    }
    return mp;
}
```

#### 4. Bài tập tự thực hiện

Kết nối một Keypad 1 hàng 4 cột và một Led 7 đoạn đến Pic, viết chương trình để mỗi khi nhấn nút Led sẽ sáng các số tương ứng.

## DẠNG 3: LCD

### 1. Giới thiệu LCD 16x2

LCD 16x2 là loại LCD có 2 Hàng và 16 ký tự trên 1 hàng nối đến chân Mass



➤ Mô tả các chân LCD 16x2

| TT | Tên chân        | Mô tả              |
|----|-----------------|--------------------|
| 1  | VSS             | GND                |
| 2  | V <sub>DD</sub> | +5V                |
| 3  | V <sub>EE</sub> | Điện áp tương phản |
| 4  | RS              | Register Select    |
| 5  | RW              | Read/Write         |
| 6  | E               | Enable             |
| 7  | D <sub>0</sub>  | Data0              |
| 8  | D <sub>1</sub>  | Data1              |
| 9  | D <sub>2</sub>  | Data2              |
| 10 | D <sub>3</sub>  | Data3              |
| 11 | D <sub>4</sub>  | Data4              |
| 12 | D <sub>5</sub>  | Data5              |
| 13 | D <sub>6</sub>  | Data6              |
| 14 | D <sub>7</sub>  | Data7              |

### 2. Các lệnh sử dụng trong LCD của PIC 16F877A

- Khai báo thư viện của LCD: #include<lcd.c>
- Định nghĩa các chân của LCD: #Define LCD\_TenPin PIN\_ChânVDK
- Khởi tạo LCD: LCD\_init();
- Xác định vị trí con trỏ: LCD\_gotoxy (x,y)
- Xuất Dữ liệu ra LCD: LCD\_putc 0

### 3. Ứng dụng

Viết chương trình hiển thị ra LCD đồng hồ đếm giờ **hh: mm: ss**

**Code:**

```
#include <LCD_Dongho.h>
#use delay(clock=20M)
#define LCD_RS_PIN PIN_B0
#define LCD_RW_PIN PIN_B1
#define LCD_ENABLE_PIN PIN_B2
#define LCD_DATA4_PIN_B4
#define LCD_DATA5_PIN_B5
#define LCD_DATA6_PIN_B6
#define LCD_DATA7_PIN_B7
#include <lcd.c>
//*****const unsigned char Hang1[20]={"He thong nhung*"};
unsigned int8 gio, phut, giay;
//*****
void LCD_GIAIMA()
{
    LCD_gotoxy(5,2);
    LCD_putc((gio/10)+0x30);
    LCD_putc((gio%10)+0x30);
    LCD_putc(":");
    LCD_putc((phut/10)+0x30);
    LCD_putc((phut%10)+0x30);
    LCD_putc(":");
    LCD_putc((giay/10)+0x30);
    LCD_putc((giay%10)+0x30);
}
void Hienthi()
{
    for(gio=0; gio<=24; gio++)
    {
        for(phut=0; phut<=60; phut++)
        {
            for(giay=0; giay<=60; giay++)

```

```

    {
        LCD_Giaima();
        Delay_ms(1000);
    }
}

void main()
{
    set_tris_B(0x00);
    LCD_init();// khai tao LCD

    while(TRUE)
    {
        lcd_gotoxy(1,1);
        lcd_putc(hang1);
        hienthi();
    }
}

```

#### 4. Bài tập tự thực hiện

- D1.1) Viết chương trình để nhấn phím bất kỳ trên ma trận phím và hiển thị ra LCD.
- D1.2) Hiển thị chiều quay của động cơ ra LCD như sau:
- Nếu động cơ quay thuận hiển thị ra LCD: **QUAY THUAN**
  - Nếu động cơ quay ngược hiển thị ra LCD: **QUAY NGUOC**
  - Nếu động cơ quay thuận hiển thị ra LCD: **DUNG**
- D1.3) Kết nối 16 Led đơn (đánh số thứ tự từ 0-15) và ma trận phím, viết chương trình để khi nhấn phím bất kỳ Led tương ứng sẽ sáng.
- D1.4) Viết chương trình để hiển thị ra LCD dạng máy tính điện tử thực hiện các phép toán +, -, \*, / cho các số từ 0-9

## DẠNG 4: CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ

### 1. ADC của Pic 16F877A

Là bộ chuyển đổi ADC 10 bit đa hợp 8 kênh, ứng dụng giao tiếp với các tín hiệu tương tự nhận được từ các cảm biến như: cảm biến nhiệt độ LM35, cảm biến độ ẩm, cảm biến khoảng cách, v.v..

### 2. Các thanh ghi sử dụng trong ADC

Thanh ghi điều khiển ADC 0: **ADCON0**

Thanh ghi điều khiển ADC 1: **ADCON1**

Thanh ghi lưu kết quả Byte cao: **ADRESH**

Thanh ghi lưu kết quả Byte thấp: **ADRESL**

#### ➤ Thanh ghi ADCON0

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| ADCS1 | ADCS1 | CHS3  | CHS2  | CHS1  | CHS0  | G/D   | ADON  |

Chức năng các Bit trong thanh ghi ADCON0:

**ADCS1: ADC0: Lựa chọn xung chuyển đổi**

00=F<sub>xtal</sub>/2

01=F<sub>xtal</sub>/8

10=F<sub>xtal</sub>/32

11=F<sub>xtal\_nội</sub>

**CHS3 : CHS2 : CHS1 : CHS0: lựa chọn kênh tương tự**

|              |              |              |               |
|--------------|--------------|--------------|---------------|
| 0000: Kênh 0 | 0100: Kênh 4 | 1000: Kênh 8 | 1100: Kênh 12 |
|--------------|--------------|--------------|---------------|

|              |              |              |               |
|--------------|--------------|--------------|---------------|
| 0001: Kênh 1 | 0101: Kênh 5 | 1001: Kênh 9 | 1101: Kênh 13 |
|--------------|--------------|--------------|---------------|

|              |              |               |             |
|--------------|--------------|---------------|-------------|
| 0010: Kênh 2 | 0110: Kênh 6 | 1010: Kênh 10 | 1110: CVREF |
|--------------|--------------|---------------|-------------|

|              |              |               |               |
|--------------|--------------|---------------|---------------|
| 0011: Kênh 3 | 0111: Kênh 7 | 1011: Kênh 11 | 1111: điện áp |
|--------------|--------------|---------------|---------------|

tham chiếu cố định bằng 0,6V

**GO/DGONE: =1: bắt đầu quá trình chuyển đổi**

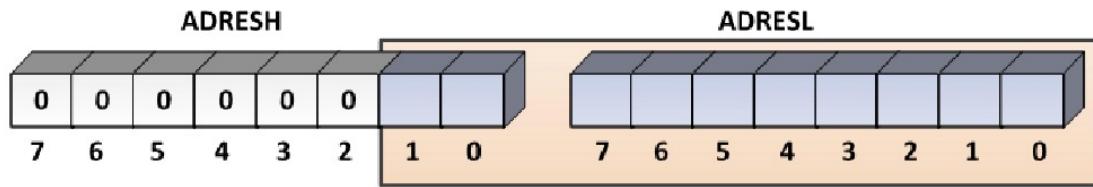
**ADON :=1-** mở nguồn cho khối chuyển đổi ADC hoạt động

**: =0-tắt nguồn cho khối chuyển đổi ADC để giảm công suất tiêu thụ**

#### ➤ Thanh ghi ADCON1

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| ADFM  | -     | VCFG1 | VCFG0 | -     | -     | -     | -     |

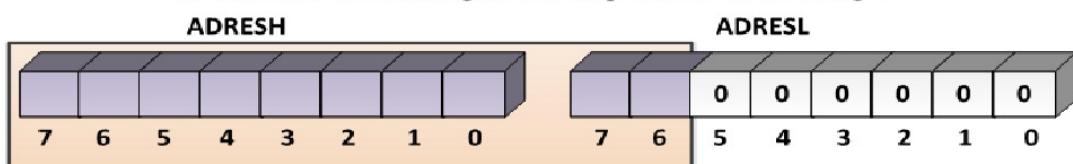
**ADFM: lựa chọn định dạng kết quả của ADC**



KẾT QUẢ 10 BIT, CANH LỀ PHẢI ADFM = 1

=1: căn lề phải cho thanh ghi, 6bit cao của ADRESH bằng 0

=0: căn lề trái cho thanh ghi, 6bit thấp của ADRESL bằng 0



KẾT QUẢ 10 BIT, CANH LỀ TRÁI ADFM = 0

VCFG: lựa chọn điều áp tham chiếu

- Thanh ghi lưu kết quả Byte cao: **ADRESH**
- Thanh ghi lưu kết quả Byte thấp: **ADRESL**

### 3. Các bước chuyển đổi ADC

- Cấu hình cho Port
- Cấu hình cho module ADC
- Thiết lập cấu hình ngắt ADC
- Chờ hết thời gian ổn định theo yêu cầu
- Bắt đầu chuyển đổi bằng cách cho Bit G/D lên 1
- Kiểm tra chuyển đổi ADC kết thúc bằng cách: nếu bit G/D = 0 thì quá trình chuyển đổi kết thúc; Nếu dùng ngắt thì chờ ngắt ADC xảy ra
- Đọc thanh ghi kết quả (ADRESH; ADRESL); xóa bit ADIF nếu dùng ngắt
- Thực hiện chuyển đổi kế tiếp

### 4. Lựa chọn nguồn xung chuyển đổi

Tần số xung ADC được lựa chọn bằng phần mềm dùng bit ADCS trong thanh ghi ADCON0. Có các nguồn xung:

**F<sub>xtal</sub>/2; F<sub>xtal</sub>/8; F<sub>xtal</sub>/32; F<sub>xtal</sub>** lấy từ bộ dao động bên trong ;

### 5. Các lệnh sử dụng trong chuyển đổi ADC

**SETUP\_ADC(MODE):** định cấu hình cho ADC

**SETUP\_ADC\_PORT(VALUE):** Định cấu hình cho Port

**SET\_ADC\_CHANNEL(PIN): Chọn kênh ADC**  
**VALUE=READ\_ADC(MODE): Đọc kết quả chuyển đổi ADC**

**Ví dụ:**

```
#device ADC=10 //khai bao ADC
Setup_ADC(ADC_Clock_div_32);//
Setup_ADC_Ports(AN0); // cho chan vao ADC la chan A0
Set_ADC_channel(0);
```

## 6. Phương pháp tính giá trị ADC từ tín hiệu đầu vào

$$Value_{ADC} = \frac{V_{IN}}{V_{REF} * 2^{bit\_ADC}}$$

Trong đó:  $V_{IN}$ : điện áp đầu vào cần chuyển đổi

$V_{REF}$ : điện áp chuẩn lấy mẫu (thường được lấy từ điện áp cấp cho chip)

$bit\_ADC$ : Số bit của ADC

$Value_{ADC}$ : giá trị ADC (làm tròn đến số nguyên)

Ví dụ: giả sử nguồn cấp cho chip là 5V và điện áp đầu vào ADC sẽ là 2,5V ta áp dụng công thức để lấy giá trị ADC tại thời điểm điện áp đầu vào  $V_{in}=2,5V$

$$Value_{ADC} = (2,5 / 5) \times 255 = 127,5 \text{ làm tròn thành } 127$$

## 7. Ứng dụng

**Ví dụ: Đọc LM35 hiển thị ra Led 7 đoạn**

Do điện áp tham chiếu của ADC là:  $V_{ref+}=5V$ ;  $V_{ref-}=0V$ ;

$$\text{Độ phân giải SS(Step Size)} = \frac{V_{ref+}-V_{ref-}}{2^{10}-1} = \frac{5.000-0}{1023} = 4,887mV$$

(Điện áp trong LM35 cứ thay đổi mỗi 1mV thì điện áp của ADC sẽ thay đổi 4,887mV)

$$\text{Độ phân giải: } \frac{10mV}{4,887mV} = 2,046$$

**Code:**

```
#include<16f877a.h>
#device ADC=10 //khai bao ADC
```

```

#use delay(clock=20M)

const unsigned int8 maled[10]={0xC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8, 0X80,
0X90};

int16 kq, i;

void main()
{
    set_tris_C(0x00);
    set_tris_D(0x00);
    set_tris_A(0x01);

    Setup_ADC(ADC_Clock_div_32);
    Setup_ADC_ports(An0);
    Set_ADC_channel(0);

    while(TRUE)
    {
        KQ=0;
        For (i=0; i<=100;i++)
        {
            kq=kq+read_adc();
            delay_ms(1);
        }
        kq=kq/2.046;
        kq=kq/100;
        output_C(maled[kq/10]);
        output_D(maled[kq%10]);
    }
}

```

## 8. Bài tập tự thực hiện

**D2.1)** Đọc tín hiệu nhiệt độ từ LM35 và hiển thị ra LCD như sau “**Nhiệt độ hiện thời: xx<sup>0</sup>C**”

**D2.2)** Đọc tín hiệu nhiệt độ từ LM35 hiển thị ra Led 7 đoạn và thực hiện công việc:

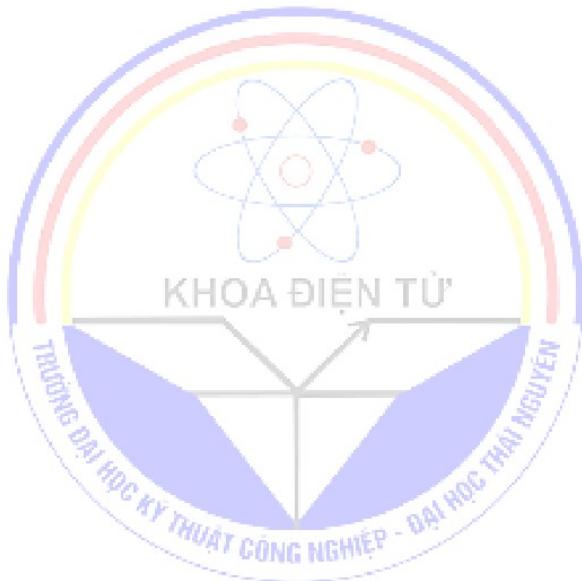
- a. Nếu nhiệt độ  $> 50^{\circ}\text{C}$ , bật Led đỏ
- b. Nếu nhiệt độ  $\leq 50^{\circ}\text{C}$ , bật Led xanh

D2.3) Đọc tín hiệu nhiệt độ từ LM35 và thực hiện công việc:

- a. Nếu nhiệt độ  $> 50^{\circ}\text{C}$ , cho động cơ quay thuận và báo ra Led 7 đoạn ký tự “H”

- b. Nếu nhiệt độ  $\leq 50^{\circ}\text{C}$ , cho động cơ dừng và báo ra Led 7 đoạn ký tự “L”

D2.4) Đặt một ngưỡng nhiệt độ bất kỳ từ nút nhấn. Nếu  $t^0_{\text{đặt}} > t^0_{\text{LM35}}$  bật Led đỏ, còn lại bật Led xanh



## DẠNG 5: TIMER/COUNTER

### 1. Giới thiệu:

Pic16F877A có 03 Timer/Counter: Timer0, Timer1, Timer2. Đặc tính kỹ thuật của các Timer như sau

- Timer\_0:

- Timer/Counter **8 bit**
- Bộ chia trước **8bit**
- Giá trị tràn: **từ FF về 00**
- Dùng xung nội hoặc ngoại



- Timer\_1:

- Timer/Counter **16 bit**
- Bộ chia trước: **1, 2, 4, 8bit**
- Giá trị tràn: **từ FFFF về 0000**
- Dùng xung nội hoặc ngoại

- Timer\_2:

- Timer/Counter **8 bit**
- Bộ chia trước và bộ chia sau: **1, 4, 16bit**
- Giá trị tràn: **tù giá trị lưu trong thanh ghi PR2 về 0000**
- Dùng xung nội hoặc ngoại

### 2. Các thanh ghi sử dụng trong Timer/Counter

- Timer\_0:

- Option\_Reg:

|             |               |             |             |            |            |            |            |
|-------------|---------------|-------------|-------------|------------|------------|------------|------------|
| Bit 7       | Bit 6         | Bit 5       | Bit 4       | Bit 3      | Bit 2      | Bit 1      | Bit 0      |
| <b>RBPU</b> | <b>INTEDG</b> | <b>T0CS</b> | <b>T0SE</b> | <b>PSA</b> | <b>PS2</b> | <b>PS1</b> | <b>PS0</b> |

Trong đó:

RBPU: Bit điều khiển điện trở treo của portB

INTEDG:

T0CS: Bit lựa chọn nguồn xung cho TMR

1=: đếm xung ngoại đưa đến chân T0CKI.

0=: đếm xung clock nội bộ bên trong.

T0SE: Bit lựa chọn cạnh tích cực T0SE-TMR0

1=: tích cực cạnh xuống ừ chân T0CKI.

0=: đếm xung clock nội bộ bên trong.

PSA: Bit gán bộ chia trước

1=: gán bộ chia cho WDT.

0=: gán bộ chia cho Timer0

PS2 – PS0: các Bit lựa chọn tỉ lệ bộ chia trước

| Bit lựa chọn | Tỉ lệ TMR0 | Tỉ lệ WDT |
|--------------|------------|-----------|
| 000          | 1 : 2      | 1 : 1     |
| 001          | 1 : 4      | 1 : 2     |
| 010          | 1 : 8      | 1 : 4     |
| 011          | 1 : 16     | 1 : 8     |
| 100          | 1 : 32     | 1 : 16    |
| 101          | 1 : 64     | 1 : 32    |
| 110          | 1 : 128    | 1 : 64    |
| 111          | 1 : 256    | 1 : 128   |

- **TMR0:** thanh ghi giá trị 8bit đếm lên; 00h-FFh

- Timer\_1: tương tự

| Bit 7         | Bit 6        | Bit 5          | Bit 4          | Bit 3          | Bit 2         | Bit 1         | Bit 0         |
|---------------|--------------|----------------|----------------|----------------|---------------|---------------|---------------|
| <b>T1GINV</b> | <b>TMRGE</b> | <b>T1CKPS1</b> | <b>T1CKPS0</b> | <b>T1OSCEN</b> | <b>T1SYNC</b> | <b>TMR1CS</b> | <b>TMR1ON</b> |

Trong đó:

T1GINV: Bit đảo cổng của Timer1

1=: Cổng Timer1 tích cực ở mức 1.

0=: Cổng Timer1 tích cực ở mức 1.

TMRGE: Bit cho phép cổng của Timer1, chỉ có tác dụng khi bit TMR1ON = 1,

1=: Timer1 mở nếu cổng Timer1 không tích cực.

0=: Timer1 mở.

T1CKPS1: T1CKPS0: các Bit lựa chọn tỉ lệ bộ chia trước

| Bit lựa chọn | Tỉ lệ chia |
|--------------|------------|
| 00           | 1 : 1      |
| 01           | 1 : 2      |
| 10           | 1 : 4      |
| 11           | 1 : 8      |

T1OSCEN: Bit điều khiển cho phép bộ dao động Timer1

1=: Cho phép bộ dao động.

0=: Không cho phép bộ dao động.

T1SYNC: Bit điều khiển đồng bộ ngõ vào xung CLK ngoài của Timer1, chỉ có tác dụng khi bit TMR1CS = 1

1=: Không đồng bộ ngõ vào xung CLK từ bên ngoài.

0=: Đồng bộ ngõ vào xung CLK từ bên ngoài.

TMR1CS: lựa chọn nguồn xung CLK của Timer1

1=: nguồn xung từ bên ngoài

0=: Xung nội ( $F_{xtal}/4$ )

TMR1ON: điều khiển Timer1

1=: cho phép Timer1 đếm

0=: cho phép Timer1 dừng

- **TMR1H, TMR1L:** thanh ghi giá trị 16bit đếm lên: 0000h-FFFFh
- Timer\_2: tương tự

### 3. Tính giá trị cho các Timer/Counter

- **Timer\_0:**

Các giá trị sử dụng trong Timer0:

- Tần số dao động của thạch anh: **Fxtal (MHz)**
- Tần số dao động của Pic: **F<sub>Pic</sub> (MHz)**
- Tần số dao động của Timer: **F<sub>Timer</sub> (MHz)**
- Thời gian của 1 chu kỳ Timer: **T<sub>timer</sub>**

- Thời gian trễ:  $T_{delay}$

**Tính toán giá trị nạp cho thanh ghi TMR0:**

$$GTN = 256 - (T_{delay}/T_{timer})$$

Ví dụ:

- **Timer\_1:**

**Các giá trị sử dụng trong Timer1**

- Tần số dao động của thạch anh:  $F_{xtal}$  (MHz)
- Tần số dao động của Pic:  $F_{Pic}$  (MHz) =  $\frac{1}{4}F_{xtal}$
- Tần số dao động của Timer:  $F_{Timer}$  (MHz) =  $\frac{1}{8}F_{pic}$
- Thời gian của 1 chu kỳ Timer:  $T_{timer}$
- Thời gian trễ:  $T_{delay}$

**Tính toán giá trị nạp cho thanh ghi TMR0:**

$$GTN = 65.536 - (T_{delay}/T_{timer})$$

Ví dụ: tìm giá trị nạp cho Timer 1 để tạo trễ 100ms, biết  $F_{xtal} = 20$ (MHz) và sử dụng bộ chia 8:

$$F_{pic} = \frac{1}{4} * F_{xtal} = \frac{1}{4} * 20 \text{Mhz} = 5 \text{Mhz}$$

$$F_{timer} = \frac{1}{8} * F_{pic} = \frac{1}{8} * 5 \text{Mhz} = 0,625 \text{MHz}$$

$$T_{timer} = \frac{1}{F_{timer}} = 1,6(\mu\text{m})$$

$$\text{Số xung} = T_{delay}/T_{timer} = \frac{100ms}{1,6*10^3} = 62.500$$

$$GTN = 65.536 - 62.500 = 3036$$

#### 4. Các lệnh sử dụng trong Timer/Counter

- SETUP\_TIMER\_X()
- SET\_TIMER\_X()
- SETUP\_COUNTERS()
- SETUP\_WDT()
- RESTART\_WDT()
- GET\_TIMER\_X()

| Lệnh | Timer 0 | Timer 1 | Timer 2 |
|------|---------|---------|---------|
|------|---------|---------|---------|

|   |  |  |   |
|---|--|--|---|
| <b>Setup_timer_x(mode)</b><br>Định cấu hình cho Timer           | <b>Setup_timer_0(mode)</b><br><b>Mode</b> : RTCC_INTERNAL, RTCC_EXT_L_H, RTCC_EXT_H_L, RTCC_DIV_2 (4,6,8,16,32,64,128,256) | <b>Setup_timer_1(mode)</b><br><b>Mode</b> : T1_DISABLED, T1_INTERNAL, T1_EXTERNAL, T1_EXTERNAL_SYNC, TC_CLK_OUT, T1_DIV_BY_1 (2,4,8) | <b>Setup_timer_2(mode)</b><br><b>Mode</b> : T2_DISABLED, T1_DIV_BY_1 (4,16) |
| <b>Set_TimerX(Value)</b><br>Thiết lập giá trị bắt đầu cho Timer | <b>Set_Timer0(int8 hoặc int16)</b>   | <b>Set_Timer1(int8 hoặc int16)</b>   | <b>Set_Timer2(int8 hoặc int16)</b>  |
| <b>Get_TimerX()</b><br>Đọc giá trị của Timer/Counter            | <b>Value=Get_Timer0()</b>  | <b>Value=Get_Timer1()</b>  | <b>Value=Get_Timer2()</b>   |

## 5. Lập trình cho Timer1

- Khởi tạo cho Timer1: SETUP\_TIMER\_1(T1\_INTERNAL|T1\_DIV\_BY\_8);
- Thiết lập giá trị bắt đầu đếm cho Timer1
- Kiểm tra cờ TRÀN cho Timer\_1: bit TMR1IF, ở địa chỉ **0x0C.0**. Nếu cờ TRAN=1, đặt cờ TRAN=0 và thiết lập lại GTN cho Timer

## 6. Ứng dụng

- Viết chương trình hiển thị đồng hồ điện tử theo định dạng HH-MM-SS lên 8 Led 7 đoạn. Biết Fxtal=20Mhz, tần số chia 8.
- Code:

```
#include <7doan_hh_mm_ss.h>
#use delay(clock=20M)
#bit bit_tran=0x0c.0
unsigned int8 gio, phut, giay, dem;
const unsigned int8 maled[10]={0xC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82,
0XF8, 0X80, 0X90};
void hienthi ()
{
    output_B(maled[gio/10]);
    output_high(pin_d0);
    delay_ms (1);
    output_low(pin_d0);

    output_B(maled[gio%10]);
```

```
output_high(pin_d1);
delay_ms(1);
output_low(pin_d1);

output_B(0x3f);
output_high(pin_d2);
delay_ms(1);
output_low(pin_d2);

output_B(maled[phut/10]);
output_high(pin_d3);
delay_ms(1);
output_low(pin_d3);
output_B(maled[phut%10]);
output_high(p
output_low(pin_d4);

output_B(0x3f);
output_high(pin_d5);
delay_ms(1);
output_low(pin_d5);

output_B(maled[giay/10]);
output_high(pin_d6);
delay_ms(1);
output_low(pin_d6);
output_B(maled[giay%10]);
output_high(pin_d7);
delay_ms(1);
output_low(pin_d7);
}

void main()
{
    gio=0;
    phut=0;
    giay=0;
    dem=0;
    setup_timer_1(T1_internal|T1_div_by_8);
    set_timer1(59286);
    while(TRUE)
    {
        if(bit_tran==1)
        {
```

```

bit_tran=0;
set_timer1(59286); // tao tre 1ms
dem++;
hienthi();
if(dem==100)
{
    giay++;
    if(giay==60)
    {
        giay=0;
        phut++;
        if(phut==60)
        {
            phut=0;
            gio++;
            if(gio==24)
            {
                gio=0;
            }
        }
    }
}
dem=0;
}
}

```



## 7. Bài tập tự thực hiện

- D3.1) Viết chương trình hiển thị đồng hồ điện tử theo định dạng HH-MM-SS lên LCD.  
Biết Fxtal=16Mhz, tần số chia 4, sử dụng Timer 1
- D3.2) Viết chương trình hiển thị đồng hồ điện tử theo định dạng HH-MM-SS lên LCD.  
Biết Fxtal=16Mhz, tần số chia 4, sử dụng Timer 1
- D3.3) Viết chương trình để cứ sau mỗi 20 giây 8Led đơn thay đổi kiểu sáng, số giây  
được đếm lùi trên Led 2 Led 7 đoạn. Biết Fxtal=16Mhz, tần số chia 8, sử dụng Timer  
1
- D3.4) Viết chương trình tạo xung vuông có tần số  $F_{xung}=100\text{Khz}$  với Duty Cycle =60%.  
Kết nối động cơ để quan sát tốc độ tương ứng với tần số đã tạo ra. Biết Fxtal=20Mhz,  
tần số chia 8, sử dụng Timer 1

