Bài 2 : Khảo sát cổng xuất nhập - Vòng lặp trễ

Nội dung:

Nd1. Khảo sát hoạt động của nút nhấn, LED.
Nd2. Khảo sát các thanh điều khiển cổng xuất nhập.
Nd3. Sử dụng hàm làm trễ `_delay()` thông qua chu kỳ xung clock F_{OSC}.
Nd4. Sử dụng nút nhấn và hiển thị kết quả ra LED.

Yêu cầu:

Yc1. Viết chương trình xuất dữ liệu ra cổng LED và thực hiện quay trái dữ liệu đó sau mỗi 500 ms.
Yc2. Kiểm tra nút nhấn RA5. Khi nút RA5 được nhấn/nhả thì thực hiện toggle việc đổi chiều quay trái/phải giá trị trên port LED.

2.1 Kiến thức liên quan

2.1.1 Các thanh ghi điều khiển cổng xuất nhập

Mỗi Port có ba thanh ghi điều khiển hoạt động chính:

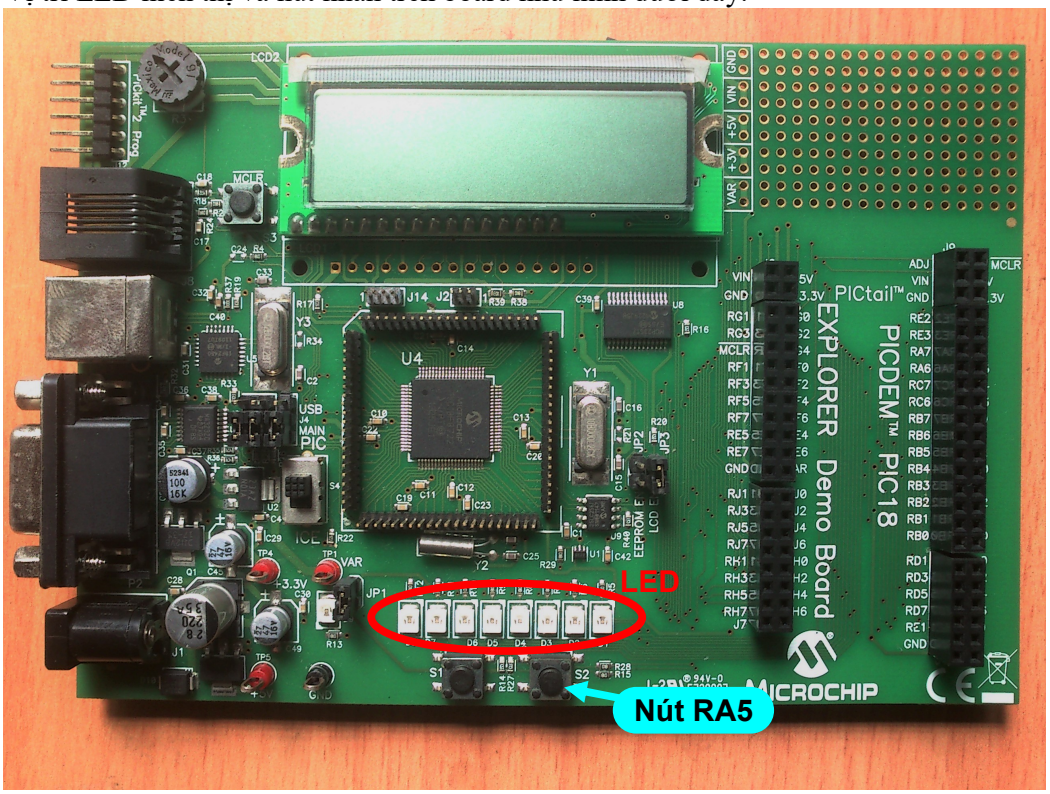
Các bit trong thanh ghi TRIS: thiết lập chân tương ứng là ngõ vào (logic 1) hoặc ngõ ra (logic 0).

Các bit trong thanh ghi PORT: đọc mức logic từ chân tương ứng.

Các bit trong thanh ghi LAT: ghi mức logic ra chân tương ứng.

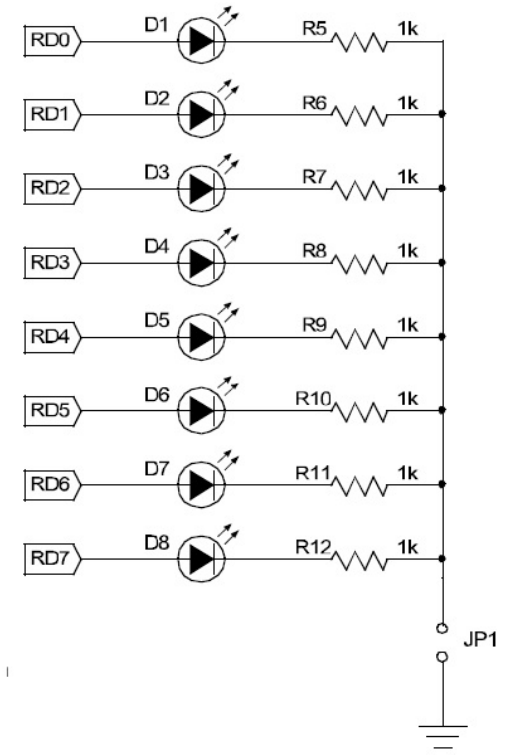
2.1.2 Kết nối mạch

Vị trí LED hiển thị và nút nhấn trên board như hình dưới đây:



-

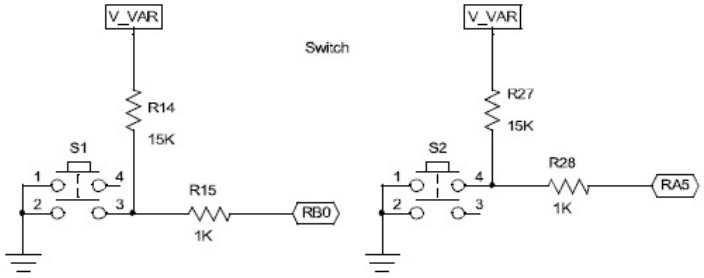
LED hiển thị có sơ đồ mạch như sau:



Muốn các LED sáng, các chân RDi tương ứng phải lên mức 1.

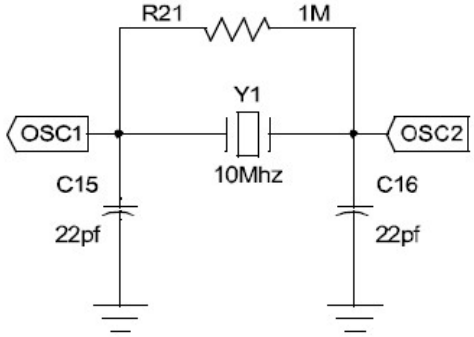
Jumper JP1 cho phép sử dụng PORTD với mục đích khác khi hở ra.

Các nút nhấn có kết nối như sau:



Nút nhấn RA5, RB0 khi được nhấn sẽ làm cho chân tương ứng ở mức ***logic 0***. Cần thiết lập các chân RA5 và RB0 là ***ngõ nhập - digital***. Các thanh ghi liên quan là TRISA (TRISA5=1), TRISB (TRISB0=1) và ADCON1=0x0F.

Mạch sử dụng dao động bên trong với thạch anh Y1 có tần số 10 MHz.



2.2 Các bước hiện thực yêu cầu

Bước 1. Tạo dự án mới Tn02. Tạo file Tn02.c và đánh code ở các bước kế tiếp vào (xem lại bài Tn01).

Bước 2. Định nghĩa port LED và chương trình con **init**.

```
volatile unsigned char led @ 0xF8C;     //LATD
volatile unsigned char led_io @ 0xF95;  //TRISD
```

```
void init(void)
{   ADCON1=0x0F;     //Khong dung Analog
    led_io=0;        //Cong D xuat 8 bit (LED x 8)
    led=0x01;
    TRISA5=1;
}
```

Bước 3. Sử dụng MACRO có sẵn của hệ thống :

```
// NOTE: To use the macros below, YOU must have previously defined _XTAL_FREQ
#define __delay_us(x) _delay((unsigned long)((x)*(_XTAL_FREQ/4000000.0)))
#define __delay_ms(x) _delay((unsigned long)((x)*(_XTAL_FREQ/4000.0)))
#define __delaywdt_us(x) _delaywdt((unsigned long)((x)*(_XTAL_FREQ/4000000.0)))
#define __delaywdt_ms(x) _delaywdt((unsigned long)((x)*(_XTAL_FREQ/4000.0)))
```

Bước 4. Cần định nghĩa tần số dao động trước :

```
#define _XTAL_FREQ 10000000
```

và chương trình làm trễ 500ms như sau :

```
#define _XTAL_FREQ 10000000
void lamtre()
{   unsigned char dem;
    for(dem=0;dem<250;dem++) __delay_ms(2);
}
```

Bước 5. Viết chương trình cho hàm **main** thực hiện quay trái giá trị hiển thị ra port LED sau mỗi nửa giây :

```
void main(void)
{   init();
    while(1)
    {   lamtre();quaytrai();}
}
```

Bước 6. Hàm quay trái port LED có thể viết bằng C hay hợp ngữ như sau:

```
void quaytrai()
{   unsigned char tam,bit7led;
    tam=led; bit7led=tam&0x80; tam<<=1;
    if ((bit7led)!=0) tam|=1;
    led=tam;
}
```

```
void quaytrai_asm() {asm("RLNCF LATD");}
```

Bước 7. Sinh viên viết lại chương trình hoàn chỉnh, dịch chương trình, nạp xuống mạch và chạy thử để kiểm tra.

2.3 Các bước hiện thực yêu cầu 2

Bước 8. Tiếp tục yêu cầu 1, thêm vào code cần thiết để sử dụng nút nhấn RA5 như trong Tn01.

Bước 9. Để đổi chiều trái/phải, ta cần định nghĩa thêm biến **`chieu`**. Biến **`chieu`**=0 thì quay trái, **`chieu`**=1 thì quay phải. Sinh viên tự viết hàm quay phải.

Bước 10. Sửa lại chương trình, dịch và chạy thử.

2.4 Bài làm thêm

a) Sửa lại code để giảm thời gian đếm xuống 200ms/lần đếm.
b) Tạo biến đếm tăng/giảm trên port LED. Dùng nút RA5 để đổi chiều đếm.