Bài 3 : Ngắt quãng và truy xuất ROM, RAM

Nội dung:

Nd1. Khảo sát ngắt quãng ngoài INT0 (thông qua nút nhấn RB0) gồm các phần: bố cục chương trình có vector ngắt, các điều kiện sử dụng ngắt quãng, viết chương trình xử lý ngắt quãng (isr: interrupt service routine).

Nd2. Xử lý tra bảng dữ liệu trong ROM và RAM.

Yêu cầu:

Yc1. Viết chương trình khởi tạo ngắt ngoài INT0 để nhận sự kiện nhấn nút RB0.

- Tổ chức bảng dữ liệu trong ROM (có 4 dữ liệu).
- Mỗi lần nhấn nút RB0, xử lý trong **isr** lấy từng byte dữ liệu trong bảng dữ liệu ROM theo thứ tự tăng dần để xuất ra port LED 8 bit.

Yc2. Làm lại chương trình với bảng dữ liệu trong RAM (cần chép trước từ ROM sang RAM).

Yc3. Kết hợp thêm gọi hàm delay500ms để hiện dữ liệu tự động.

Yc4. Kiểm tra nút nhấn để đổi quy luật.

3.1 Kiến thức liên quan

3.1.1 Tóm tắt các thanh ghi điều khiển ngắt

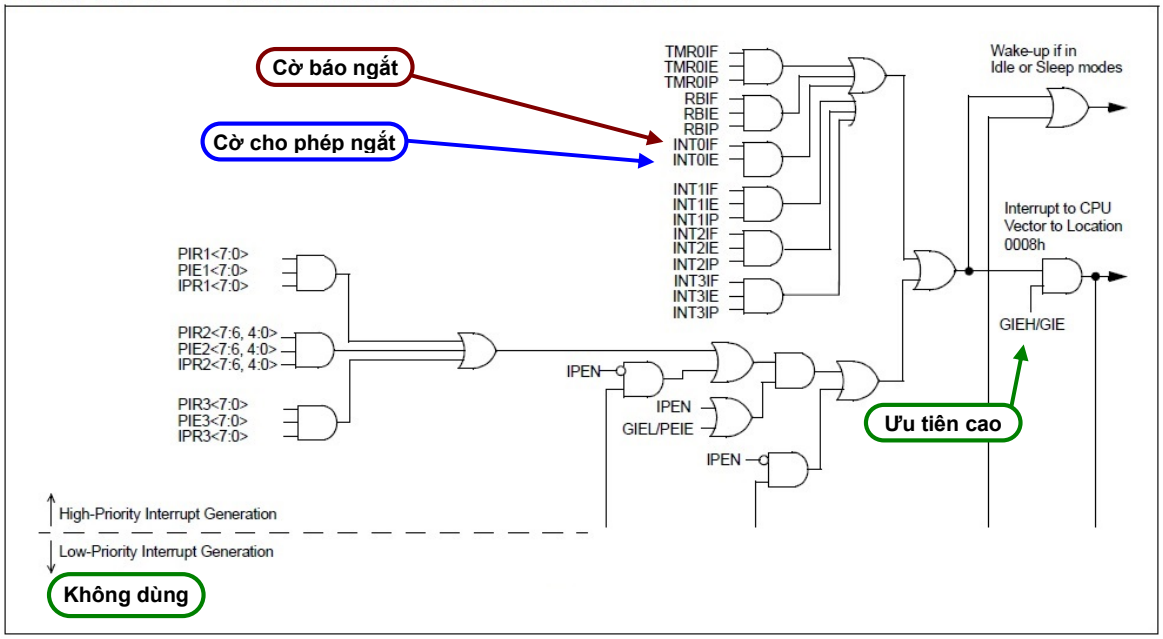
Thanh ghi INTCON:

REGISTER 10-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

Sơ đồ điều khiển ngắt:

FIGURE 10-1: PIC18F8722 FAMILY INTERRUPT LOGIC



3.2 Các bước hiện thực yêu cầu 1

Bước 1. Tạo dự án mới Tn03, thêm file nguồn Interrupt.c.

Bước 2. Trong cửa sổ code của Interrupt.c, định nghĩa bảng dữ liệu trong ROM như sau:

```
//Dinh nghia du lieu ROM
const unsigned char dl_rom[4]={0x55,0xAA,0x3C,0xC3};
```

Bước 3. Viết hàm khởi động chung thực hiện cấu hình: port LED xuất 8 bit, nút nhấn RB0 là ngõ nhập. Hàm **`init()`** như sau:

```
#define XUAT    0
#define NHAP    1
volatile unsigned char led @ 0xF8C;
volatile unsigned char led_io @ 0xF95;

void init()
{   ADCON1=0x0F;     //Chon digital
    led_io=XUAT;     //port LED xuat
    TRISB0=NHAP;     //Nut nhan RB0 nhap
    idx=0;           //Dau bang ROM
    xuatled();
}
```

Bước 4. Cấu hình **cho phép** dùng ngắt ngoài INT0. Độ ưu tiên của INT0 được định sẵn là ưu tiên **cao** (vector H'08'). Hàm **`int0_init()`** như sau:

```
void int0_init()
{   IPEN=1;          //Su dung uu tien
    INT0IF=0;        //Xoa co ngat
    INT0IE=1;        //Cho phep ngat
    GIEH=1;          //Cho phep ngat toan cuc
}
```

Bước 5. Viết chương trình phục vụ cho ngắt ngoài INT0 (**int0_isr**), trong đó thực hiện :

- Xóa cờ ngắt INT0IF.
- Gọi chương trình con **`xuat_led()`** tra bảng lấy dữ liệu trong ROM (theo biến chỉ số **idx**) xuất ra LED.

```
/*//////////////////////////////////////
 * Chuong trinh xu ly ngat isr
 *//////////////////////////////////////
void interrupt int0_isr()
{   INT0IF=0;        // Xoa co ngat
    xuatled();       // Goi ham xu ly
}
```

Bước 6. Viết chương trình con **`xuat_led()`** thực hiện công việc :

- Tra bảng lấy dữ liệu từ ROM theo chỉ số.
- Xuất byte nội dung ROM ra port LED.
- Tăng chỉ số và thực hiện xoay vòng từ 0 đến (MAXIDX -1) và trở về 0.

```
#define MAXIDX  4
unsigned char idx;

void xuatled()
{   led=dl_rom[idx++]; //Xuat ROM ra LED
    idx%=MAXIDX;       //Xoay vong
}
```

3.3 Chương trình mẫu yêu cầu 1

Bước 7. Sinh viên viết lại chương trình hoàn chỉnh theo bố cục gợi ý sau:

```
// File:   Tn03_Interrupt.c
#include <xc.h>
#pragma config OSC=HS, WDT=OFF, LVP=OFF
//Dinh nghia XUAT, NHAP, MAXIDX
//Dinh nghia dia chi port led, led_io
//Dinh nghia du lieu ROM
//Dinh nghia bien chi so cua bang
void xuatled()
{
    //Than ham xuatled()
}
void init()
{
    //Than ham init()
}
void int0_init()
{
    //Than ham int0_init()
}
void interrupt int0_isr()
{
    //Than ham int0_isr()
}
void main(void)
{
    //Than ham main()
}
```

Bước 8. Chạy thử chương trình.

3.4 Các bước hiện thực yêu cầu 2

Bước 9. Thêm vào định nghĩa bảng 4 byte trong RAM như sau :

```
const unsigned char dl_rom[4]={0x99,0x66,0x18,0xE7};
unsigned char dl_ram[4];
```

Bước 10. Trong phần init, thêm lệnh gọi hàm **`rom2ram()`** để chép bảng dữ liệu từ ROM sang RAM.

```
void rom2ram()
{   unsigned char i;
    for(i=0;i<MAXIDX;i++) dl_ram[i]=dl_rom[i];
}
```

Bước 11. Viết hàm **`xuatled2()`** thực hiện tra bảng lấy dữ liệu từ RAM

```
unsigned char idx;
void xuatled2()
{   led=dl_ram[idx++];idx%=MAXIDX;}
```

Bước 12. Dịch và chạy thử chương trình đã sửa.

3.5 Thực hiện yêu cầu 3

Bước 13. Thêm hàm **`lamtre()`** để làm trễ 500ms như trong bài 2.

Bước 14. Định nghĩa 4 bảng quy luật 8-byte và sửa lại hàm **`xuatled()`** như sau:

```
#define MAXIDX  8
#define MAXROM  4
const unsigned char dl_rom[MAXROM][MAXIDX]={
    {0xFF,0x7E,0x3C,0x18,0x00,0x18,0x3C,0x7E},
    {0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF},
    {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80},
    {0x81,0xC3,0xE7,0xFF,0xFF,0xE7,0xC3,0x81},
    };
void xuatled()
{   led=dl_ram[ramidx++];ramidx%=MAXIDX;}
```

Bước 15. Chỉnh lại hàm **`main()`** để gọi **`xuatled()`** và **`lamtre()`** như sau:

```
void main(void)
{
    init();
    int0_init();
    while(1)
    {   xuatled();
        lamtre();
    }
}
```

Bước 16. Dữ liệu trong RAM sẽ được chép từ ROM do hàm **`doi_ql()`** như sau:

```
unsigned char dl_ram[MAXIDX],ramidx,romidx,so_ql;
void doi_ql()
{   int i;
    for(i=0;i<MAXIDX;i++) dl_ram[i]=dl_rom[romidx][i];
    romidx++;romidx%=MAXROM;
}
```

Bước 17. Như vậy, khi nhấn nút RB0, hàm **`int0_isr()`** chỉ thực hiện đổi qui luật bằng cách gọi hàm **`doi_ql()`**. Còn việc hiển thị qui luật ra LED là do hàm **`main()`** thực hiện.

```
void interrupt int0_isr()
{   INT0IF=0;
    doi_ql();
    ramidx=0;
}
```

Bước 18. Hoàn chỉnh chương trình và chạy thử.

3.6 Bài làm thêm

a) Viết chương trình thực hiện chạy LED theo các quy luật 8 trạng thái và 16 trạng thái sau:

Trạng thái 1:	○○○○○○○○	○○○○○○○○
Trạng thái 2:	●○○○○○○●	●○○○○○○○
Trạng thái 3:	●●○○○○●●	●●○○○○○○
Trạng thái 4:	●●●○○●●●	●●●○○○○○
Trạng thái 5:	●●●●●●●●	●●●●○○○○
Trạng thái 6:	●●●○○●●●	●●●●●○○○
Trạng thái 7:	●●○○○○●●	●●●●●●○○
Trạng thái 8:	●○○○○○○●	●●●●●●●○
Trạng thái 9:	trở về 1	○●●●●●●●
Trạng thái 10:		○○●●●●●●
Trạng thái 11:		○○○●●●●●
Trạng thái 12:		○○○○●●●●
Trạng thái 13:		○○○○○●●●
Trạng thái 14:		○○○○○○●●
Trạng thái 15:		○○○○○○○●
Trạng thái 16:		trở về 1

b) Viết chương trình đổi 2 quy luật ở câu a) qua lại khi nhấn nút RB0.