Bài 4 : Khảo sát bộ định thời - Modules

Nội dung:

Nd1. Khảo sát các chế độ hoạt động của các bộ định thời timer0 và timer1.
Nd2. Tạo module Timer.c để khởi động và sử dụng bộ định thời.
Nd3. Khai thác module Lcd.c để hiển thị ra màn hình LCD (tải từ BKEL).

Yêu cầu:

Yc1. Sử dụng bộ Timer0 tạo ngắt quãng thời gian thực 10ms theo công thức tính t.
Yc2. Trên cơ sở ngắt thời gian 10ms, kết hợp thêm biến đếm phụ dem1s (với số lần đếm SODEM500MS*2), viết hàm **`tre1s()`** để có thời gian trễ 1s dùng cho việc điều khiển hiển thị LED.
Yc3. Thay đổi giá trị prescaler (4, 8), tính lại số đếm để thời gian xảy ra ngắt không đổi.
Yc4. Sử dụng thêm module Lcd.c có sẵn để hiển thị ra màn hình LCD các ký tự sau mỗi giây.

4.1 Các bước hiện thực yêu cầu 1

Bước 1. Tạo dự án mới Tn04_Timer, tập tin Tn04_Timer.c.
Bước 2. Khởi động port LED.
Bước 3. Tạo mới module định thì có tên *Timer.c*.
Bước 4. Tham khảo thanh ghi điều khiển bộ định thì Timer0 **T0CON** :

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Trong module Timer.c, viết hàm khởi động **`timer0_init`**`()` gồm các bước cần làm như sau:

- Chọn mức độ ưu tiên thấp cho ngắt Timer0 [IPEN=1, TMR0IP=0].
- Cho phép sử dụng ngắt Timer0 [TMR0IE=1].
- Cho phép ngắt tổng thể [GIEH=1, GIEL=1].
- Gọi hàm **`timer0_reset()`**.

```
#include <xc.h>
#include "Tn04.h"
#define SODEM10MS   -00000
void timer0_reset();
void timer0_init()
{   RCONbits.IPEN=1;          //cho phep dung uu tien
    INTCON2bits.TMR0IP=0;     //Timer0 uu tien thap
    INTCONbits.TMR0IE=1;      //cho phep ngat Timer0
    INTCONbits.GIEH=1;        //cho phep ngat uu tien cao
    INTCONbits.GIEL=1;        //cho phep ngat uu tien thap
    T0CON = 0;                //dung Fosc/4, prescaler=2
    timer0_reset();
}
```

Sinh viên tự tính

Bước 5. Viết hàm **`timer0_reset()`** thực hiện :

- Xoá cờ ngắt.
- Ngưng đếm.
- Nạp số đếm vào bộ đếm.
- Bắt đầu đếm lại.

```
void timer0_reset()
{   TMR0IF=0;           //xoa co ngat
    TMR0ON=0;           //Ngung dem
    TMR0=SODEM10MS;     //Nap so dem
    TMR0ON=1;           //Bat dau dem
}
```

Bước 6. Số đếm SODEM10MS được tính từ công thức sau :

$$t_{timer} = (1/(F_{OSC}/4))*prescaler*SODEM10MS$$

trong đó:

- t_{timer} là đơn vị thời gian cần tạo ra, trong ví dụ này là 10ms.
- F_{OSC} là tần số xung clock cấp cho CPU (với PICDEM PIC18 là 10 MHz).
- ***prescaler*** = 2 (do lập trình các bit $T0PS_{2,1,0}$ trong T0CON quy định)

Sinh viên tính ra giá trị của SODEM10MS và thay vào 00000 ở dòng code:

```
#define     SODEM10MS -00000
```

Do bộ đếm thực hiện **đếm tăng** khi có xung clock nên giá trị nạp vào bộ đếm sử dụng số âm.

Ngoài ra, việc định nghĩa hằng số SODEM10MS thay vì dùng số trực tiếp giúp ta dễ dàng thay đổi giá trị đếm sau này khi muốn thay đổi thời gian xảy ra ngắt quãng.

Bước 7. Viết chương trình phục vụ ngắt quãng cho Timer0 (**`timer0_isr()`**) thực hiện các việc sau :

- Gọi hàm timer0_reset() đã viết ở bước 5 để khởi động lại ngắt Timer0.
- Gọi hàm xử lý định kỳ **`timer_process()`** (10ms thực hiện 1 lần).

Tham khảo code sau:

```
void interrupt low_priority timer0_isr()
{   if (TMR0IE & TMR0IF)
    {   timer0_reset();     //nap lai so dem
        timer_process();
    }
}
```

Hàm **`timer_process()`** sẽ được viết trong module chính Tn04.c nên ta cần tạo ra các file header kèm theo các module code như bước kế tiếp.

Bước 8. Trong cây dự án, vào mục Header files chọn new→C Header File thêm vào file Timer.h. Sửa lại nội dung file thành như sau:

```
// File:   Timer.h
#ifndef TIMER_H
#define TIMER_H
extern  void timer0_init();
#endif  /* TIMER_H */
```

Bước 9. Thêm header file Tn04.h với nội dung như sau:

```
// File:   Tn04.h
#ifndef TN04_H
#define TN04_H
extern void timer_process();
#endif  /* TN04_H */
```

Bước 10. Thêm vào đầu module Timer.c dòng: **`#include "Tn04.h"`**

Bước 11. Module chính Tn04_Timer.c sẽ chứa các hàm **`init()`**, **`timer_process()`** và **`main()`**, phần khai báo như sau:

```
// File Tn04.c
#include <xc.h>
#pragma config OSC=HS, WDT=OFF, LVP=OFF
#include "Timer.h"
volatile unsigned char led @ 0xF8C;      //LATD
volatile unsigned char led_io @ 0xF95;   //TRISD
#define XUAT         0
#define NHAP         1
#define SODEM500MS   50
unsigned char dem500ms;
```

Bước 12. Hàm **timer_process()** thực hiện tăng giá trị trên port LED sau mỗi 500ms.

```
#define SODEM500MS  50
unsigned char dem500ms;
void timer_process()
{   if ((--dem500ms)==0) //kiem tra du 500ms moi lam
    { led++;
      dem500ms=SODEM500MS;
    }
}
```

Bước 13. Hàm **init()** và **main()**.

```
void init()
{   ADCON1=0x0F;
    led_io=XUAT;
    led=0x00;
    dem500ms=SODEM500MS;
}
void main()
{
    init();
    timer0_init();
    while(1);
}
```

Bước 14. Dịch và chạy.

4.2 Các bước hiện thực yêu cầu 2

Bước 15. Trong module Tn04B.c, định nghĩa thêm biến **dem1s**, thêm vào hàm **tre1s()** và sửa lại hàm **timer_process()** như sau:

```
unsigned char dem1s;
void timer_process()
{   if (dem1s>0) dem1s--;}
void tre1s()
{   dem1s=SODEM500MS*2;
    while(dem1s>0);
}
```

Bước 16. Hàm **main()** bây giờ sử dụng hàm **tre1s()** để hiển thị led như sau:

```
void main()
{
    init();
    timer0_init();
    while(1)
    { tre1s();led++;}
}
```

4.3 Các bước hiện thực yêu cầu 3

Bước 17. Tham khảo thanh ghi **T0CON** để thay đổi giá trị **prescaler** = 8.

Bước 18. Tính lại hằng số **SODEM10MS** để giữ nguyên thời gian t_{timer} = 10ms.

Bước 19. Chạy lại chương trình.

Bước 20. Kết hợp tra bảng dữ liệu như trong bài Tn03 để xuất ra các quy luật chạy led.

```
#define MAXIDX      8
#define MAXROM      4
#define SODEM500MS  10
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},
    {0x00,0x81,0xC3,0xE7,0xFF,0xE7,0xC3,0x81}};
unsigned char dl_ram[MAXIDX],ramidx,romidx;
unsigned char dem500ms;
void xuatled()
{   led=dl_ram[ramidx++];ramidx%=MAXIDX;}
void timer_process()
{   if ((--dem500ms)==0)
    {   xuatled();
        dem500ms=SODEM500MS;
    }
}
void doi_ql()
{   unsigned char i;
    for(i=0;i<MAXIDX;i++) dl_ram[i]=dl_rom[romidx][i];
    romidx++;romidx%=MAXROM;
    ramidx=0;
}
```

4.4 Các bước hiện thực yêu cầu 4

Bước 21. Module Lcd.c và Lcd.h cho phép sử dụng màn hình LCD với các hàm hỗ trợ sau đây:

- Hàm **`lcd_init()`** : khởi động màn hình LCD, gọi 1 lần ở đầu chương trình.
- Hàm **`lcd_cls()`** : xoá màn hình.
- Hàm **`lcd_gotoxy(d,c)`** với tham số **d** là dòng (0÷1) và **c** là cột (0÷15) : để dời con trỏ màn hình đến vị trí (dòng,cột). Màn hình LCD có 2 dòng, 16 cột.
- Hàm **`lcd_putc()`** : hiện ký tự ASCII trong biến **lcd_wr** ra vị trí con trỏ.
- Ngoài ra, nhờ định nghĩa lại hàm **`putch()`** thông qua hàm **`lcd_putc()`** nên ta có thể sử dụng các hàm xuất màn hình của thư viện **stdio.lib** để xuất số ra LCD.

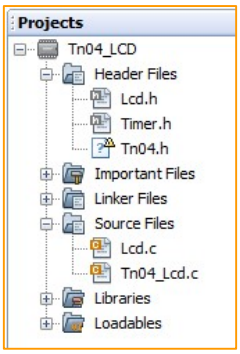
Ví dụ: ta có thể tính và in số 20! như sau:

```
// File Tn04_Lcd.c
#include <xc.h>
#include <stdio.h>
#include "lcd.h"
#pragma config OSC=HS, WDT=OFF, LVP=OFF
#define N       20
void init()
{   ADCON1=0x0F;    //nhap digital
}
void main()
{   float x;unsigned char i;
    init();
    lcd_init();
    for(i=1,x=1;i<=N;i++) x*=i;
    printf("\f%d!=\n%e",N,x);
    while(1);
}
```

Bước 22. Viết chương trình hiện ký tự tăng dần (0÷255) và chạy khắp màn hình (200ms chạy 1 vị trí) với các quy luật chạy:

- Từ trái sang phải, từ trên xuống dưới.
- Từ trái sang phải, từ dưới lên trên.
- Từ phải sang trái, từ trên xuống dưới.

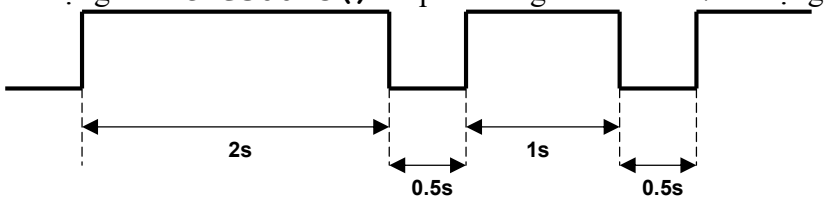
Cây dự án gồm các module gợi ý như hình bên.



4.5 Bài tập

a) Làm lại bài thí nghiệm nhưng sử dụng Timer1 thay vì Timer0 (thêm vào module Timer.c các hàm **`timer1_init()`**, **`timer1_reset()`**, **`timer1_isr()`** và tạo thêm file Timer1.h).

b) Viết và sử dụng hàm **`tre500ms()`** để phát xung ra chân RD7 có dạng như sau:



c) Dùng bộ định thời tạo xung vuông chu kì 10ms, duty cycle 30%.

d) Viết hàm tính cơ số e theo triển khai Mac-Laurin.
e=1+(1/1!)+(1/2!)+...+(1/n!)
tính n bước (tính càng nhiều bước càng chính xác).