
MỤC LỤC

1	Lập trình vào ra	2
1.1	Giới thiệu về PIC18F4520	2
1.1.1	Thông số PIC18F4520	2
1.1.2	Sơ đồ chân	2
1.2	Các dạng bài tập vào ra	3
1.2.1	Các dạng bài tập và sử dụng lệnh kiểm tra phím	4
1.2.2	Chương trình mẫu kiểm tra lỗi	6
1.2.3	Bài tập phần 1 – vào ra	7
2	Hướng dẫn phần ngắn ngoài Tài liệu tra cứu PIC trang 24-36	15
2.1	Giới thiệu về ngắn ngoài	15
2.2	Viết khung chương trình có sử dụng ngắn	16
2.2.1	Viết các lệnh của chương trình chính:	16
2.2.2	Chương trình mẫu	16
2.3	Bài tập ngắn ngoài	17
2.3.1	Bài tập sử dụng 01 ngắn ngoài	18
2.3.2	Bài tập dùng hơn 1 ngắn ngoài	20
3	Lập trình Timer	23
3.1	Chế độ định thời	23
3.2	Chế độ đếm sự kiện	29
3.2.1	Hiển thị LED đơn	29
3.2.2	Hiển thị LED 7 thanh	31
4	Lập trình UART	33
5	Biến đổi tương tự số ADC	36
5.1	Bài tập	38
5.2	Các dạng bài ADC	40
6	Lập trình PWM	40
6.1	Các công thức PWM	41
6.2	Bài tập	42

Hướng dẫn lập trình MPLAB

Tài liệu được viết bởi Giảng Viên: ThS Nguyễn Văn Dũng – Đại học công nghiệp Hà nội. Mọi thắc mắc và góp ý gửi qua email: VanDungEvn@gmail.com hoặc số điện thoại: 0973291368.

Trong tài liệu sẽ hướng dẫn lập trình Vi điều khiển PIC18F4520 trên **MPLAB IDE** sử dụng trình biên dịch **C18**. Tài liệu này đi kèm và sử dụng **Tài liệu tra cứu PIC** của **Khoa Điện tử, trường Đại học Công nghiệp Hà nội**.

1 Lập trình vào ra

1.1 Giới thiệu về PIC18F4520.

1.1.1 Thông số PIC18F4520.

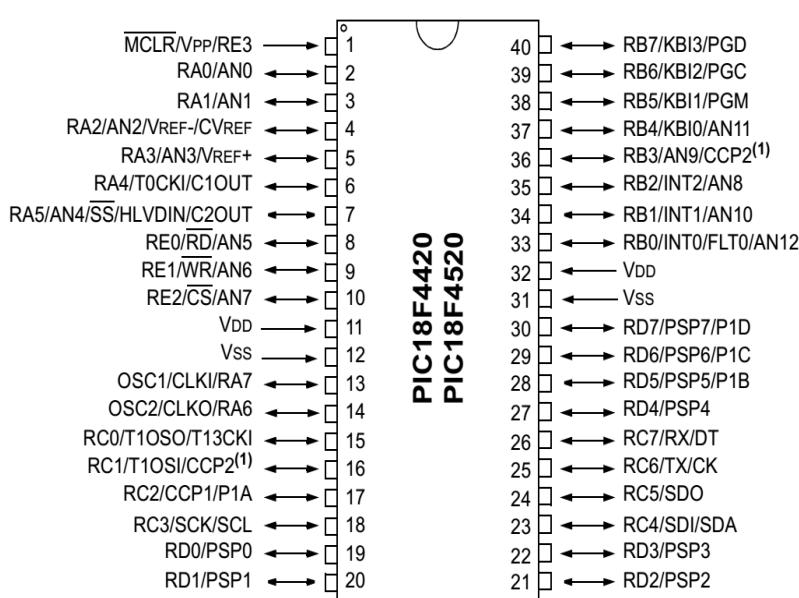
Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I ² C™			
PIC18F2420	16K	8192	768	256	25	10	2/0	Y	Y	1	2	1/3
PIC18F2520	32K	16384	1536	256	25	10	2/0	Y	Y	1	2	1/3
PIC18F4420	16K	8192	768	256	36	13	1/1	Y	Y	1	2	1/3
PIC18F4520	32K	16384	1536	256	36	13	1/1	Y	Y	1	2	1/3

Tần số hoạt động tối đa: 40Mhz

Khả năng xóa, ghi bộ nhớ:

- 100.000 lần cho bộ nhớ Flash
- 1000.000 lần cho bộ nhớ EEPROM

1.1.2 Sơ đồ chân



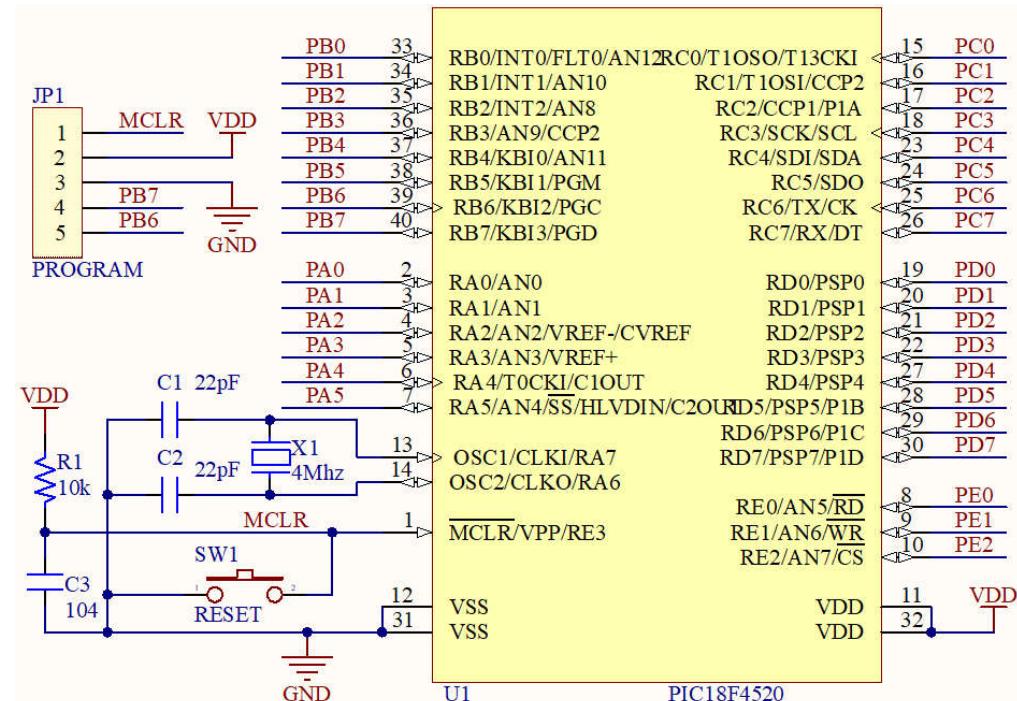
Kiểu đóng gói PDIP40
gồm: 40 chân trong đó
4 chân nguồn và 36
chân IO (5 PORT):
- PORTA-D mỗi
PORT 8 chân Rx0 –
Rx7
- PORTE 4 chân : RE0
– RE3

Hình 1.1 Sơ đồ chân PIC18F4520

- 4 chân nguồn : VDD (11, 32) cấp điện áp 2.0 – 5.5V, GND (12, 31) 0V

Kết nối tối thiểu để MCU hoạt động:

- Nguồn cấp: 4 chân nguồn VDD và VSS
- RESET (MCLR) như mạch điện, tụ C3 thay 104 khi bao gồm cả nạp chip.
- Thạch anh, có thể sử dụng thạch anh nội bộ thì không cần nối như chân RA6, RA7
- Kết nối nạp chip (Program) cần thiết trong quá trình nạp kiểm tra.



Hình 1.2 Sơ đồ kết nối cơ bản PIC18F4520

1.2 Các dạng bài tập vào ra

Tạo Project => Microchip C18 gồm 2 file **code.c** và **18f4520.lkr**

```

.LED don.mcw
1 //khai bao thu vien
2 #include <P18f4520.h>
3 #include <delays.h>
4 //Cau hinh cho vi dieu khien
5 #pragma config OSC = HS
6 #pragma config MCLRE = ON
7 #pragma config WDT = OFF
8 void main() {
9     ADCON1 = 0x0F; // Cac chan vao ra so
10    // Khoi tao vao ra cac PORT TRIS.....
11    while(1)
12    {
13        // Chuong trinh chinh
14    }
15 }

```

Hình 1.3 Chương trình mẫu khởi tạo

1.2.1 Các dạng bài tập và sử dụng lệnh kiểm tra phím.

1.2.1.1 Kiểm tra phím nhấn – nhả thực hiện nhiệm vụ

- Kiểm tra trạng thái phím nhấn hay nhả.

Sử dụng cấu trúc if .. else

Kiểm tra nếu nhấn phím có thể để 0

Thuật toán và xử lý lệnh



Hình 1.4 Lưu đồ thuật toán kiểm tra phím

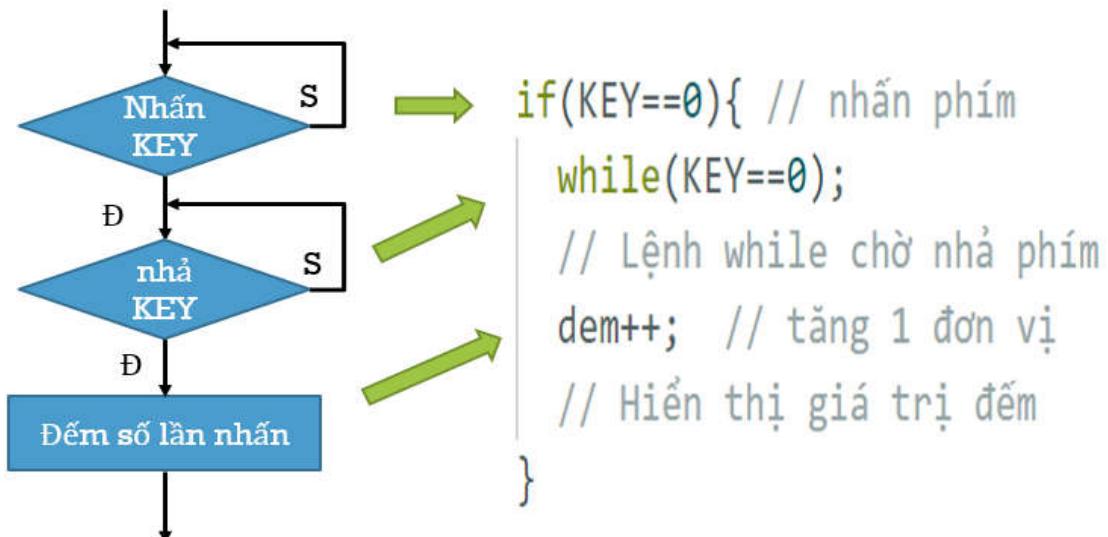
1.2.1.2 Đếm số lần nhấn phím

Đếm số lần nhấn phím:

Sử dụng kết hợp if và while

Sử dụng **while(...)** để chờ nhả phím, nếu không có lệnh này thì giá trị dem sẽ tăng liên tục khi nhấn và giữ phím.

Thuật toán và xử lý lệnh



Hình 1.5 Lưu đồ thuật toán đếm số lần nhấn phím

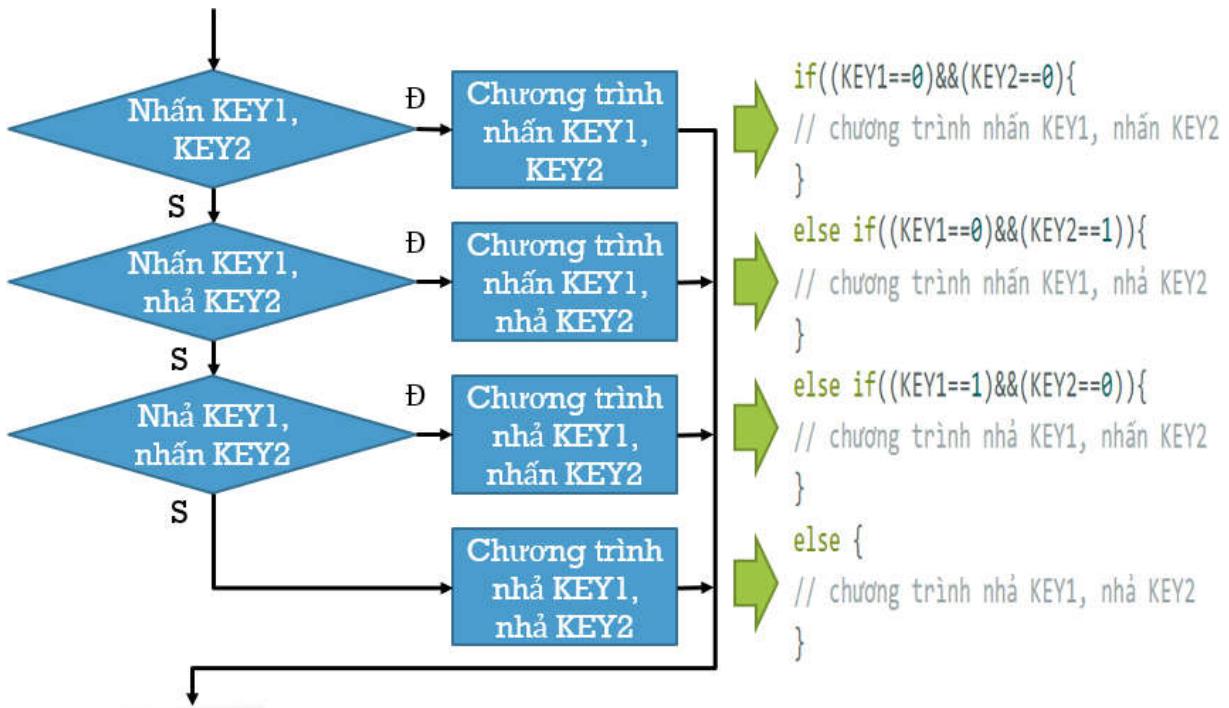
1.2.1.3 Kiểm tra đồng thời nhiều phím nhấn.

- Sử dụng 2 phím KEY1 và KEY2:

Sử dụng kết hợp if và else

Kiểm tra trạng thái các trường hợp của 2 phím xử lý lệnh: KEY, KEY

Thuật toán và xử lý lệnh



Hình 1.6 Lưu đồ thuật toán xử lý nhiều phím ấn

Chương trình có thể sử dụng 4 lệnh if cho 4 trường hợp, lúc đó thuật toán sẽ có 4 hình thoi tương ứng với 4 lệnh if.

1.2.2 Tính thời gian trên sử dụng hàm trong thư viện delays.h

Tính giá trị a trong các hàm trong thư viện hỗ trợ:

Delay100TCYx(a); Delay1KTCYx(a); Delay10KTCYx(a)

Yêu cầu đề bài: Tạo thời gian trễ Tdelay, lưu ý: 1 chu kỳ thực hiện lệnh là 4 chu kỳ dao động: $TCY = 4 * Tosc = 1/Fosc \Rightarrow$

$$a = \frac{Tdelay}{TCY \text{ Hàm trễ}} = \frac{Fosc}{4 \text{ Hàm trễ}} Tdelay$$

Tosc: Chu kỳ bộ dao động; **Fosc:** Tần số bộ dao động; **TCY** thời gian 1 chu kỳ hoạt động của MCU

Ví dụ: với $Fosc = 4\text{Mhz}$, $Tdelay = 0.5 \text{ giây}$

- Với hàm $\text{Delay1KTCYx} \Rightarrow a = \frac{Fosc}{4 \text{ Hàm trễ}} Tdelay = \frac{4M}{4 \text{ 1K}} 0.5 = 500$
- Với hàm $\text{Delay10KTCYx} \Rightarrow a = \frac{Fosc}{4 \text{ Hàm trễ}} Tdelay = \frac{4M}{4 \text{ 10K}} 0.5 = 50$

Vậy $\text{Delay1KTCYx}(500) = \text{Delay10KTCYx}(50) = 0.5 \text{ giây}$

1.2.3 Các thanh ghi điều khiển PORT.

PIC18F4520 có 5 PORT A, B, C, D, E; mỗi PORT gồm có 3 thanh ghi:

- Thanh ghi **TRISx** điều hướng vào ra.
- Thanh ghi **PORTx** xuất nhập dữ liệu
- Thanh ghi **LATx** chốt (xuất) dữ liệu.

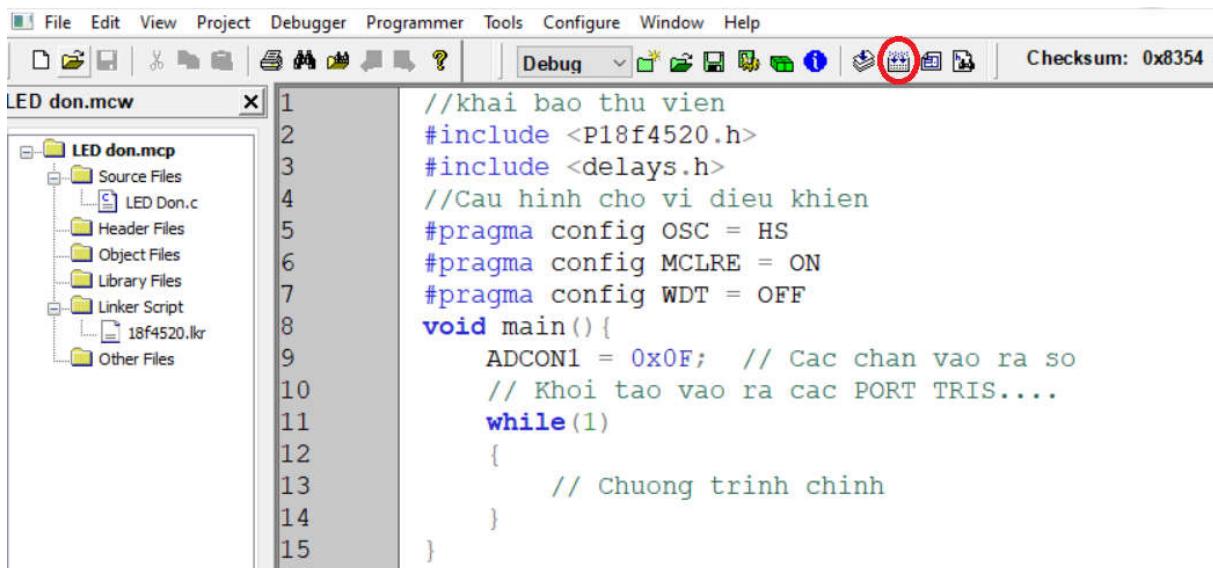
1.2.4 Chương trình mẫu kiểm tra lỗi

Tạo project trong MPLAB:

- Project => Project Wizard... => Next
- Step One => PIC18F4520
- Step Two => Active Toolsuite => Microchip C18

Lưu Project và tạo file .C => add to Project

Add file linker: 18f4520.lkr



```
//khai bao thu vien
#include <P18f4520.h>
#include <delays.h>
//Cau hinh cho vi dieu khien
#pragma config OSC = HS
#pragma config MCLRE = ON
#pragma config WDT = OFF
void main(){
    ADCON1 = 0x0F; // Cac chan vao ra so
    // Khoi tao vao ra cac PORT TRIS....
    while(1)
    {
        // Chuong trinh chinh
    }
}
```

Khi tạo Project xong => mở file “Tai lieu tra cuu PIC.pdf” mở trang 16 hoặc gõ code nhu hình trên.

Để biên dịch code bấm nút **2 mũi tên**, hoặc nhấn **F10** hoặc Project=> Make



Màn hình Output hiển thị biên dịch thành công.

Như vậy trước khi viết chương trình mới thì cần tạo Project và code mẫu như trên rồi biên dịch để xác định phần tạo Project không bị lỗi.

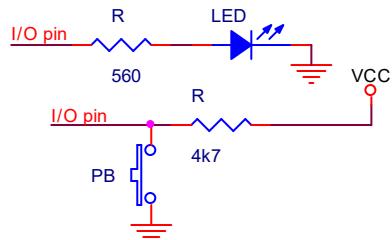
1.2.5 Bài tập phần 1 – vào ra

Câu 1. Vẽ mạch điện và lập trình theo yêu cầu:

a. Vẽ mạch điện như sau:

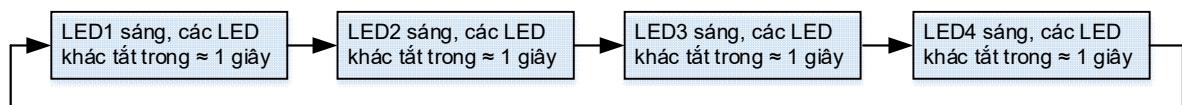
- 04 LED (LED1 - LED4) nối lần lượt với các chân RD4 – RD7 theo sơ đồ như hình bên;

- Nút nhấn PB1 nối với RB0 theo sơ đồ như hình bên.



b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu:

- Khi PB1 ở trạng thái nhả: các LED sáng/tắt theo chu trình sau:

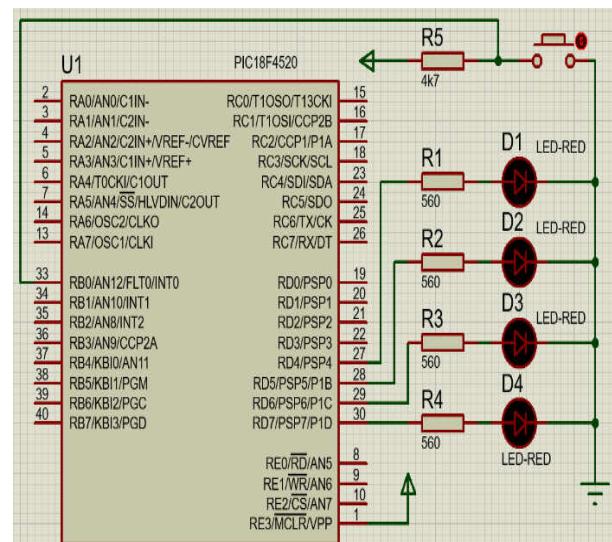
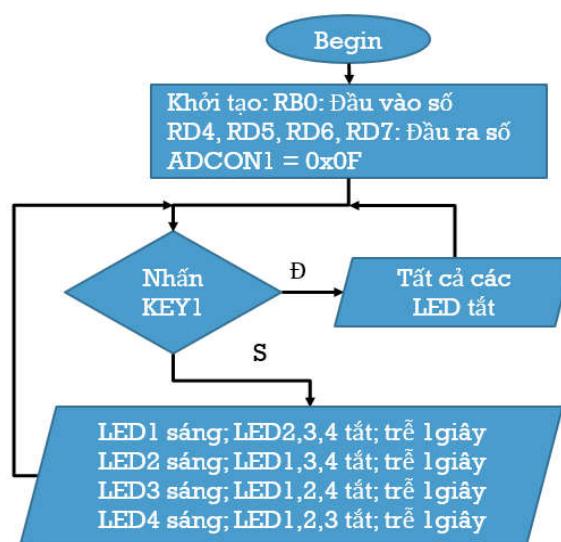


- Khi PB1 ở trạng thái nhấn, tắt cả các LED tắt.

Phân tích mạch:

Chân nối LED => đầu ra số, Chân nối nút nhấn => đầu vào số.

+ Lưu đồ thuật toán + Sơ đồ mạch



Lập trình code: có 2 phương pháp ghi dữ liệu theo byte hoặc theo bit:

+ Viết theo byte

```
//Khai bao thu vien
#include <P18f4520.h>
#include <delays.h>
//Cau hinh cho vi dieu khien
#pragma config OSC = HS
#pragma config MCLRE = ON
#pragma config WDT = OFF
```

```

void main(){
    ADCON1 = 0x0F; // IO digital mode
    TRISB = 0b01; // RB0 Input
    TRISD = 0; // RD0-RD7 Output
    while(1)
        { // Nha phim LED1-4 Lan luot sang trong 1 giay
            if(PORTBbits.RB0){ // Nha phim
                PORTD = 0b00010000; // RD4=1 LED1 Sang
                Delay1KTCYx(1000); // Tre 1 giay
                PORTD = 0b00100000; // RD5=1 LED2 Sang
                Delay1KTCYx(1000); // Tre 1 giay
                PORTD = 0b01000000; // RD6=1 LED3 Sang
                Delay1KTCYx(1000); // Tre 1 giay
                PORTD = 0b10000000; // RD7=1 LED4 Sang
                Delay1KTCYx(1000); // Tre 1 giay
            } // Fosc = 4Mhz, tre 1 giay
            else{ // nhan phim, cac LED tat
                PORTD = 0 ; // Cac LED tat
            }
        }
}

```

+ Chương trình viết theo bit

```

//Khai bao thu vien
#include <P18f4520.h>
#include <delays.h>
//Cau hinh cho vi dieu khien
#pragma config OSC = HS
#pragma config MCLRE = ON
#pragma config WDT = OFF
// Dinh nghia chan LED va nut nhan
#define LED1 PORTDbits.RD4
#define LED2 PORTDbits.RD5
#define LED3 PORTDbits.RD6
#define LED4 PORTDbits.RD7
#define PB1 PORTBbits.RB0
void main(){
    ADCON1 = 0x0F; // IO digital mode
    TRISB = 0b01; // RB0 Input
    TRISD = 0; // RD0-RD7 Output

```

```

while(1)
{
    if(PB1==0) {          // Nhan phim
        LED1=LED2=LED3=LED4=0; // Cac LED tat
    }
    else{   // nha phim, cac LED sang chu trinh
        LED1 = 1;           // LED1 Sang
        LED2=LED3=LED4=0;   // LED2,3,4 tat
        Delay1KTCYx(1000); // Tre 1 giay
        LED2 = 1;           // LED2 Sang
        LED1=LED3=LED4=0;   // LED1,3,4 tat
        Delay1KTCYx(1000); // Tre 1 giay
        LED3 = 1;           // LED3 Sang
        LED2=LED1=LED4=0;   // LED1,2,4 tat
        Delay1KTCYx(1000); // Tre 1 giay
        LED4 = 1;           // RD4=1 LED1 Sang
        LED2=LED3=LED1=0;   // LED1,2,3 tat
        Delay1KTCYx(1000); // Tre 1 giay
    } // Fosc = 4Mhz, tre 1 giay
}
}

```

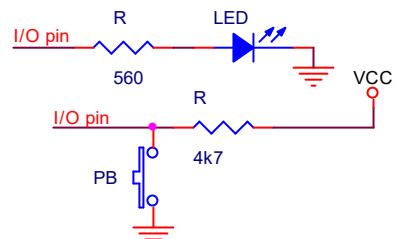
Lưu ý khi lập trình :

Dựa theo chương trình mẫu phần **1.2.2** tạo Project biên dịch kiểm tra lỗi trước khi viết chương trình, để tránh bị lỗi do trình biên dịch hoặc quá trình tạo Project.

Câu 2. Vẽ mạch điện và lập trình theo yêu cầu:

a. Vẽ mạch điện như sau:

- 04 LED (LED1 - LED4) nối lần lượt với các chân RD4 – RD7 theo sơ đồ như hình bên;
- Nút nhấn PB1 nối với RB0 theo sơ đồ như hình bên.



b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu:

- Khi bật nguồn hoặc reset: 4 LED tắt;
- Nhấn PB1 một lần, LED1 sáng, các LED khác tắt; nhấn PB1 hai lần, LED2 sáng, các LED khác tắt; nhấn PB1 ba lần, LED3 sáng, các LED khác tắt; nhấn PB1 bốn lần, LED4 sáng, các LED khác tắt; nhấn PB1 từ năm lần trở lên, tắt cả các LED sáng.

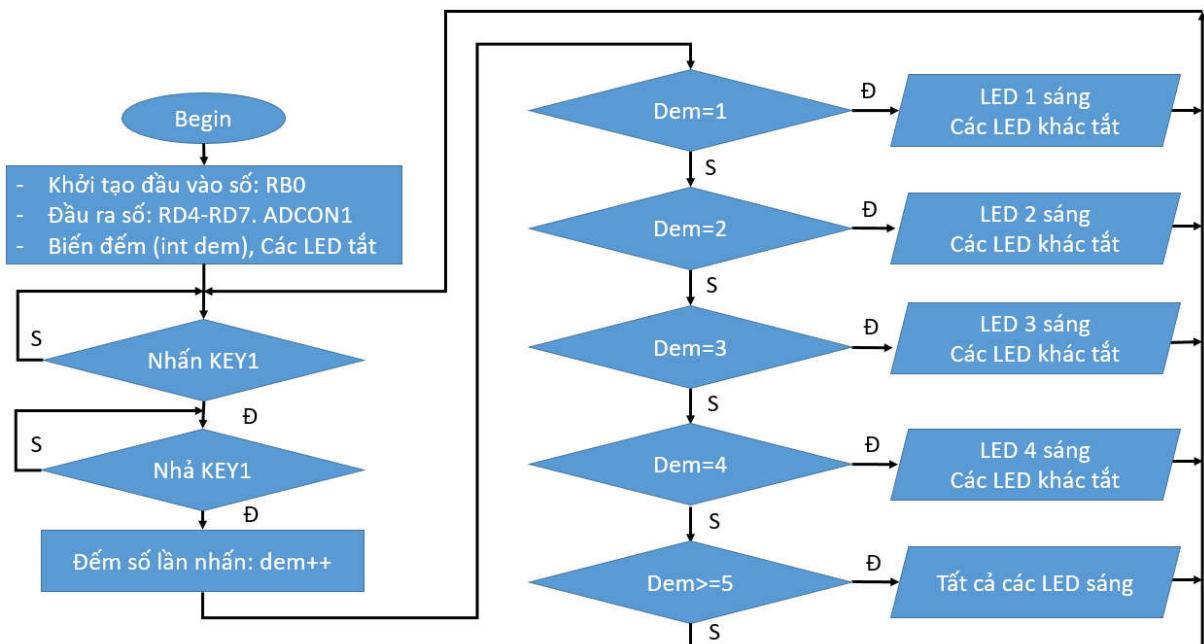
Phân tích mạch:

Chân nối LED => đầu ra số, Chân nối nút nhấn => đầu vào số.

Yêu cầu lập trình:

- Đếm số lần nhấn phím dùng cấu trúc **if ... while** tại mục 1.2.1.2.
- Hiển thị theo giá trị biến đếm được viết sau lệnh đếm số lần nhấn.
- Sử dụng lệnh if để kiểm tra giá trị đếm và hiển thị theo yêu cầu.

Lưu đồ thuật toán



Chương trình có thể viết theo byte hoặc bit, chương trình mẫu theo bit

```
//Khai bao thu vien
#include <P18f4520.h>
#include <delays.h>
//Cau hinh cho vi dieu khien
#pragma config OSC = HS
#pragma config MCLRE = ON
#pragma config WDT = OFF
// Dinh nghia chan LED va nut nhan
#define LED1      PORTDbits.RD4
#define LED2      PORTDbits.RD5
#define LED3      PORTDbits.RD6
#define LED4      PORTDbits.RD7
#define P1       PORTBbits.RB0
int dem; // Khai bao bien
```

```

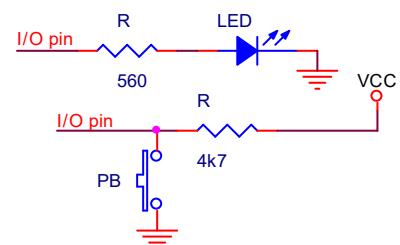
void main() {
    ADCON1 = 0x0F; // IO digital mode
    TRISB = 0b01; // RB0 Input
    TRISD = 0; // RD0-RD7 Output
    LED1=LED2=LED3=LED4=0; // Bat nguon cac LED tat
    while(1)
    {
        if(PB1==0) { // Nhan phim
            while(PB1==0); // Cho nha phim
            dem++;
            if(dem==1) {
                // LED 1 sang, cac LED khac tat
                // Sang muc 1, tat muc 0
            }
            // lenh if voi dem == tu 2...4
            if(dem>=5) {
                // Tat ca LED sang
            }
        }
    }
}

```

Câu 3. Vẽ mạch điện và lập trình theo yêu cầu:

a. Vẽ mạch điện như sau:

- 04 LED (LED1 - LED4) nối lần lượt với các chân RD4 – RD7 theo sơ đồ như hình bên;
- Nút nhấn PB1 nối với RB0 theo sơ đồ như hình bên.



b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu:

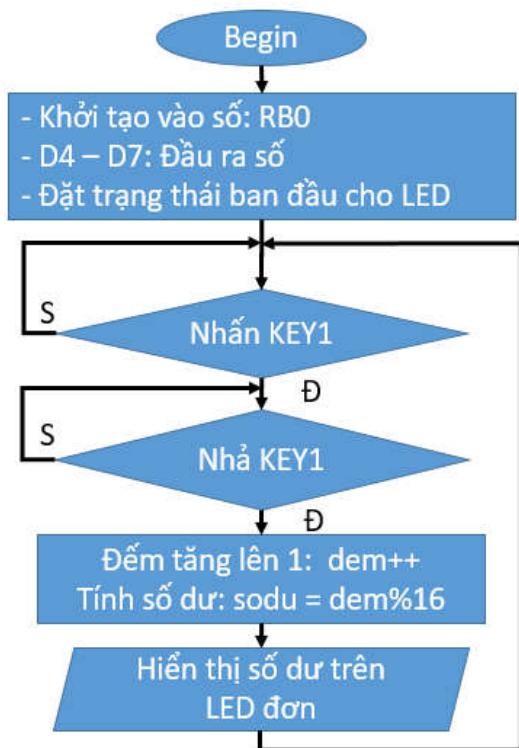
- Khi bật nguồn hoặc reset: 4 LED tắt;
- Đếm số lần nhấn PB1 và hiển thị số dư của phép chia $\frac{\text{Số lần nhấn PB1}}{16}$ dưới dạng số nhị phân trên 04 LED (LED sáng tương ứng bit “1”, tắt tương ứng bit “0”). Giả thiết số lần nhấn không lớn hơn 255 lần.

Phân tích yêu cầu:

- Phản hiển thị giá trị nhị phân trên LED đơn, tức là giá trị đếm được hiển thị theo bảng mã nhị phân.

- Theo cách thủ công giống câu 2 áp dụng vào bài câu 3 cần có 16 lệnh if vì số dư phép chia 16 là 0-15. Để chương trình ngắn gọn hơn thì sử dụng phương pháp hiển thị theo PORT được giải thích bên dưới.

Lưu đồ thuật toán



Giải thích phần hiển thị dữ liệu.

- Giá trị hiển thị là số dư của 16 (0-15) 16 là 2^4 , như vậy cần 4 bit để hiển thị giá trị 0-15.
- Nếu sử dụng lệnh **PORTD = dem%16;** thì dữ liệu sẽ được hiển thị từ bit D0-D3.
- Để bài yêu cầu từ D4-D7 nên (D7 D6 D5 D4 D3 D2 D1 D0) tức là cần chuyển dữ liệu từ bit D3-D0 sang trái thành D7-D4 \Rightarrow lấy $4-0 = 4 \Rightarrow$ lệnh đúng dịch trái 4 bit **PORTD = (dem%16) <<4;**

Chương trình (khởi tạo ban đầu như mục 1.2.2, dưới đây là chương trình bổ xung hoàn thiện)

```

// Định nghĩa chan LED va nut nhan
#define PB1      PORTBbits.RB0
int dem; // Khai bao bien
void main() {
    ADCON1 = 0x0F; // IO digital mode
    TRISB = 0b01; // RB0 Input
    TRISD = 0; // RD0-RD7 Output
    PORTD = 0; // Cac LED tat D4-D7=0
    while(1)
    {
        if(PB1==0) { // Nhan phim
            while(PB1==0); // Cho nha phim
            dem++;
            // LED D7--D4
            PORTD = (dem%16)<<4; // Dich trai 4 bit
        }
    }
}

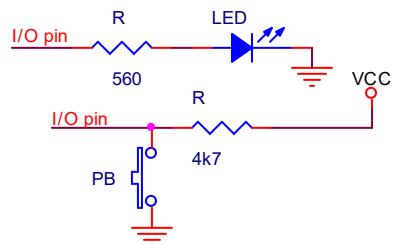
```

Câu 4. Vẽ mạch điện và lập trình theo yêu cầu:

a. Vẽ mạch điện như sau:

- 04 LED (LED1 - LED4) nối lần lượt với các chân RD4 – RD7 theo sơ đồ như hình bên;

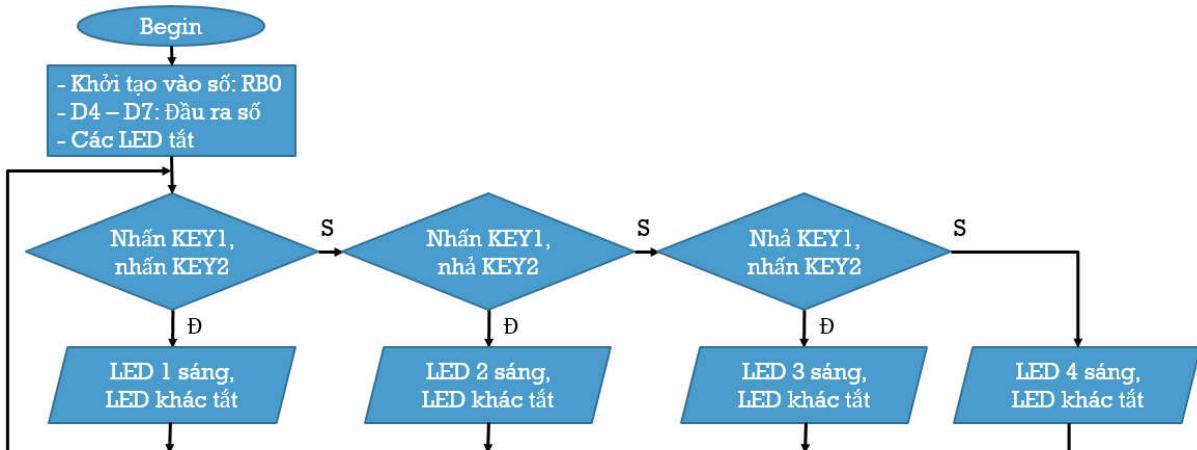
- Nút nhấn PB1 nối với RB0; nút nhấn PB2 nối với RB1 theo sơ đồ như hình bên.



b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như bảng trạng thái sau:

Trạng thái PB1	Trạng thái PB2	LED1	LED2	LED3	LED4
Nhấn	Nhấn	Sáng	Tắt	Tắt	Tắt
Nhấn	Nhả	Tắt	Sáng	Tắt	Tắt
Nhả	Nhấn	Tắt	Tắt	Sáng	Tắt
Nhả	Nhả	Tắt	Tắt	Tắt	Sáng

Thuật toán (bài này sử dụng thuật toán theo cấu trúc mục 1.2.1.3)



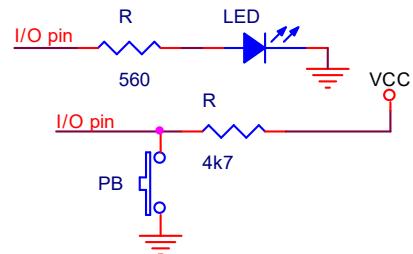
Theo ví dụ trước và cấu trúc lệnh mục 1.2.1.3, các em tự hoàn thiện chương trình bài này.

Câu 5. Vẽ mạch điện và lập trình theo yêu cầu:

a. Vẽ mạch điện như sau:

- LED1 nối với chân RD5; LED2 nối với chân RD6 theo sơ đồ như hình bên;

- Nút nhấn PB1 nối với RB0; nút nhấn PB2 nối với RB1 theo sơ đồ như hình bên.



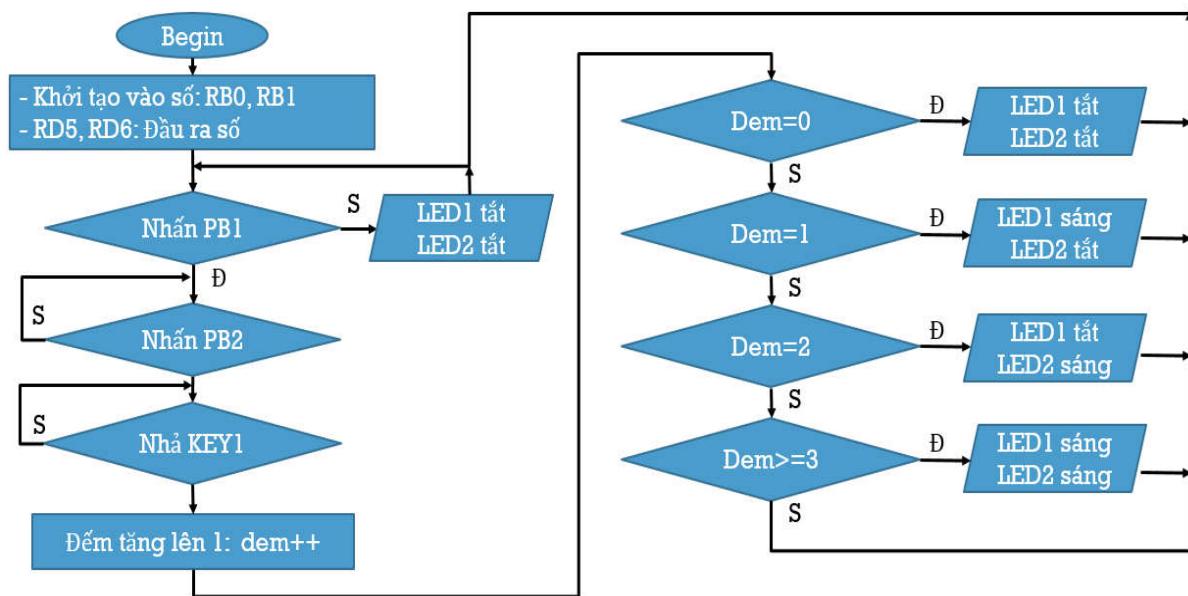
b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- Khi PB1 ở trạng thái nhả: LED1 và LED2 tắt;

- Khi PB1 ở trạng thái nhấn:

Số lần nhấn PB2	LED1	LED2
0 lần	Tắt	Tắt
1 lần	Tắt	Sáng
2 lần	Sáng	Tắt
Từ 3 lần trở lên	Sáng	Sáng

Thuật toán



Dựa theo code có các bài trước vận dụng làm hoàn thiện code bài này.

2 Hướng dẫn phần ngắt ngoài Tài liệu tra cứu PIC trang 24-36

2.1 Giới thiệu về ngắt ngoài

Ngắt ngoài trên PIC18F4520 gồm 3 chân **INT0** (RB0), **INT1** (RB1), **INT2** (RB2) và ngắt PORTB **RBI** (RB4-RB7). Sử thanh ghi điều khiển ngắt ngoài để khởi tạo các ngắt. Các thanh ghi điều khiển ngắt xem chi tiết từ **trang 24-32 trong tài liệu tra cứu PIC**, dưới đây là tổng hợp về các ngắt ngoại **INT0**, **INT1**, **INT2**

Thanh ghi RCON

R/W-0	R/W-1 ⁽¹⁾	U-0	R/W-1	R-1	R-1	R/W-0 ⁽²⁾	R/W-0
IPEN	SBOREN	—	RI	TO	PD	POR	BOR
bit 7							bit 0

- Thanh ghi điều khiển ngắt: INTCON

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

- Thanh ghi điều khiển ngắt 2: INTCON2

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

- Thanh ghi điều khiển ngắt 3: INTCON3

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7							bit 0

Loại bit \ Nguồn ngắt	INT0	INT1	INT2	RB - KBI (RB4-RB7)
Cho phép ngắt toàn cục (Global Interrupt Enable)	GIE_GIEH Thanh ghi INTCON			
Cho phép nguồn ngắt Thanh ghi	INT0IE INTCON	INT1IE INTCON3	INT2IE INTCON3	RBIE INTCON
Ngắt bằng sườn âm (=0) hoặc sườn dương (=1)	INTEDG0 Thanh ghi INTCON2	INTEDG1 Thanh ghi INTCON2	INTEDG2 Thanh ghi INTCON2	
Còn ngắt Thanh ghi	INT0IF INTCON	INT1IF INTCON3	INT2IF INTCON3	RBIF INTCON
Ưu tiên ngắt (=1 ưu tiên cao, =0 ưu tiên thấp)	Luôn ưu tiên cao	INT1IP INTCON3	INT2IP INTCON3	RBIP INTCON2

Khi sử dụng các ngắt ngoại **INT0**, **INT1**, **INT2** cần chú ý các bit điều khiển của mỗi ngắt ở các thanh ghi khác nhau theo bảng trên.

2.2 Viết khung chương trình có sử dụng ngắt

2.2.1 Viết các lệnh của chương trình chính:

- Khởi tạo các PORT (ADCON1, TRIS)
- Set bit ngắt toàn cục **INTCONbits.GIE=1;**
- Set bit cho phép nguồn ngắt tương ứng: **INTCONbits.INT0IE = 1;** hoặc **INTCON3bits.INT1IE = 1;** hoặc **INTCON3bits.INT2IE = 1;**
- Chọn ngắt bằng sườn âm/dương bằng cách xóa/set các bit tương ứng: **INTCON2bits.INTEDG0=0** hoặc **INTCON2bits.INTEDG1 =0** hoặc **INTCON2bits.INTEDG2=0** (*=0 ngắt sườn âm, =1 ngắt sườn dương*)
- Chọn mức ưu tiên ngắt bằng cách xóa/set các bit tương ứng: **INT0** luôn ưu tiên cao, **INTCON3bits.INT1IP =0** hoặc **INTCON3bits.INT2IP=0** (*=0 ưu tiên mức thấp, =1 ưu tiên mức cao*)
- Xóa cờ ngắt **INTxIF=0**, chương trình thực hiện khi ngắt.
- Chương trình chính.

3. Viết các lệnh của CTCPVN:

- Xóa cờ ngắt **INTxIF=0**, chương trình thực hiện khi ngắt.

2.2.2 Chương trình mẫu

Chương trình mẫu, tham khảo code trong Tài liệu tra cứu PIC: từ dòng 1 – 9 trang 32, dòng 12-23 trang 33 và bô xung **void main() {}**

```
1  /*****  
2  Chuong trinh mau ngat ngoai  
3  *****/  
4  #include <p18f4520.h>  
5  #include <delays.h>  
6  
7  #pragma config OSC=HS  
8  #pragma config WDT=OFF  
9  #pragma config MCLRE=ON  
10 // Khai bao chan IO va cac bien  
11 // Khai bao chuong trinh ngat  
12 void ngat_uu_tien_cao(void);  
13 void ngat_uu_tien_thap(void);  
14 // CTCPVN uu tien cao UTC, vector 0008H  
15 #pragma code uu_tien_cao=0x08  
16 void ngat_cao(void)  
17 {  
18     ngat_uu_tien_cao();  
19 }
```

```

20     #pragma code
21     #pragma interrupt ngat_uu_tien_cao
22     void ngat_uu_tien_cao(void)
23     {
24         .....INTXIF = 0; // Xoa co ngat
25         // Thuc hien chuong trinh ngat
26     }
27     void main()
28     {
29         TRIS .....; // Khai bao cac chan vao ra
30         ADCON1 = 0x0F; // IO vao ra so
31         INTCONbits.GIE = 1; // Cho phep MCU ngat
32         INTCON.....INTXIE = 1; // cho phep ngat INTX
33         INTCON.....INTEDGx = 0; // ngat suon am
34         .....INTXIF = 0; // xoa co ngat INTX
35         while(1){
36             // Chuong trinh chinh
37         }

```

Các phần điều chỉnh các thanh ghi và bit tương ứng với các nguồn ngắt khác nhau.

Trước khi thực hiện lập trình chương trình ngắt hãy tạo chương trình mẫu rồi biên dịch kiểm tra xem tạo Project đã đúng chưa. Theo chương trình trên thì **bỏ dòng 24, 28, 31, 32, 33** rồi tiến hành biên dịch. Hoặc để chương trình mẫu như mục **1.2.2**.

2.3 Bài tập ngắt ngoài

Các bài ngắt ngoài có thể sử dụng từ 1 đến 3 ngắt, đối với chương trình chỉ có 1 ngắt thì không cần kiểm tra cờ ngắt. Chương trình dùng nhiều hơn 1 ngắt thì cần kiểm tra cờ ngắt trong chương trình ngắt để phân biệt các ngắt khác nhau.

Nội dung phần này chỉ đề cập đến ngắt ngoài, còn với tất cả các bài ngắt thì cần sử dụng 02 lưu đồ thuật toán:

- Thuật toán cho chương trình chính
- Thuật toán cho chương trình phục vụ ngắt

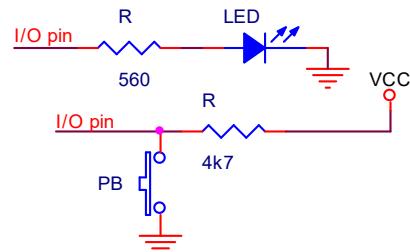
Khi viết các lưu đồ cần lưu ý sử dụng đúng các ký hiệu theo quy chuẩn.

Các dạng bài tập áp dụng cho ngắt ngoài sẽ từ bài 1 ngắt trở lên theo chương trình sau:

2.3.1 Bài tập sử dụng 01 ngắt ngoài.

Bài 1: a. Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Các điện trở; 02 LED; 01 nút nhấn thường mở (KEY).
- LED1 nối với chân RC2; LED2 nối với chân RC4; Nút nhấn KEY được nối với chân RB1/INT1.



b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- Khi bật nguồn hoặc reset: LED1 sáng/ tắt liên tục.
- Khi nhấn KEY, LED1 tắt, LED2 sáng/tắt 10 lần sau đó tắt và LED1 tiếp tục sáng/tắt.

Thời gian sáng/ tắt của LED ≈ 1 giây.

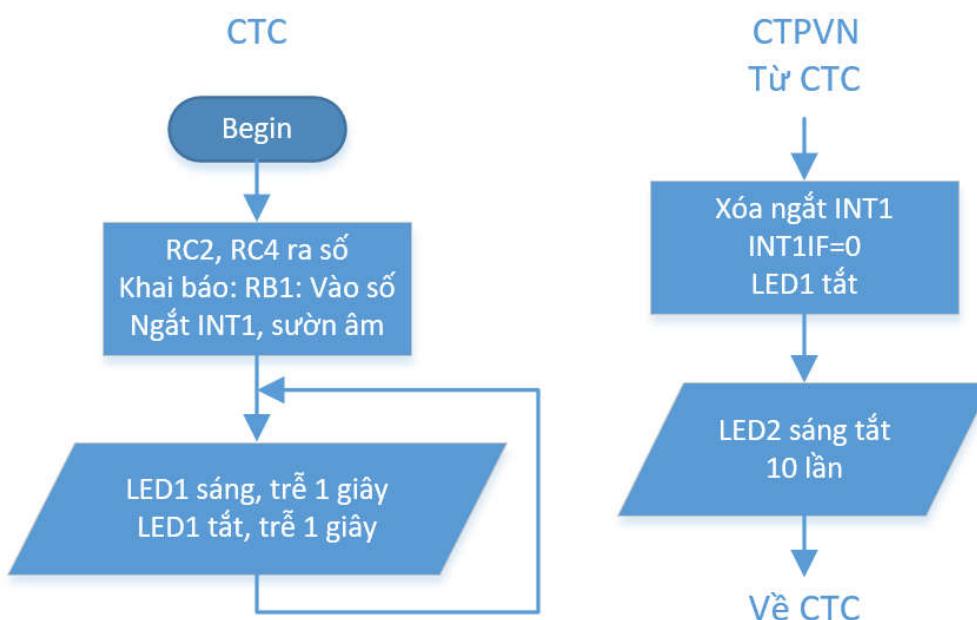
Phân tích:

Bài sử dụng 01 nút nhấn ngắt INT1.

Yêu cầu của đề bài:

- Khi bật nguồn hoặc reset **LED1** sáng/ tắt liên tục \Rightarrow **chương trình chính** dòng 35 chương trình mẫu.
- **Khi nhấn KEY** \Rightarrow chương trình ngắt (không phải viết chương trình if như bài phần IO) toàn bộ chương trình của nút nhấn trước giờ viết vào dòng 25 của chương trình mẫu.

Thuật toán



Khai báo:

ADCON1, TRIS và khai báo ngắt 1: **GIE = 1; INT1IE = 1, INT1IF = 0,**
INTEDG1 (xác định giá trị theo sườn ngắt) xem **các thanh ghi ngắt mục 2.1**

Chương trình

Tạo chương trình mẫu như mục 2.2.2, nhấn F10 biên dịch thành công:

- Sử dụng lệnh **#define** để khai báo 2 chân LED vào dòng số **10**
- Chương trình tiếp theo từ dòng 22 của chương trình mẫu.

Khai báo thêm biến **int i**, khai báo chân kết nối **LED1-2** bên dưới lệnh **config**

```
void ngat_uu_tien_cao(void)
{
    if(INTCON3bits.INT1IF == 1) // kiem tra co ngat INT1
    {
        // LED 1 tat
        for(i=0;i<10;i++) { // Lap 10 lan
            // LED2 sang, tre 1 giay
            // LED2 tat, tre 1 giay
        }
        INTCON3bits.INT1IF = 0; // xoa co ngat INT1
    }
}
void main()
{
    TRISB = 0x07;           // RB0-2 in, RB3-7 OUT
    TRISC = 0;              // PORTC OUT
    ADCON1 = 0x0F;          // IO vao ra so

    INTCONbits.GIE = 1;     // Cho phep MCU ngat
    INTCON3bits.INT1IE = 1; // cho phep ngat INT1
    INTCON2bits.INTEDG1 = 0; // ngat suon am
    INTCON3bits.INT1IF = 0; // xoa co ngat INT1
    while(1){
        LED1=1;             // LED 1 sang
        Delay10KTCYx(100); // tre 1.0 giay
        LED1=0;             // LED 1 tat
        Delay10KTCYx(100); // tre 1.0 giay
    }
}
```

Tương tự lập trình yêu cầu thay đổi sử dụng với ngắt **INT0** hoặc **INT2**.

Lưu ý: Các chân nối với các linh kiện được thay đổi theo từng đề thi/kiểm tra

Bài 2 a. Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Các điện trở, 08 LED; 01 nút nhấn thường mở (PB).

- LED1 ÷ LED8 được nối với các chân RC0 ÷ RC7; Nút nhấn PB1 được nối với lần lượt với các chân RB0/INT0.

b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- Bật nguồn hoặc Reset 8 LED tắt

- Sử dụng ngắt INT0, đếm số lần nhấn phím, hiển thị giá trị đếm trên PORTC dưới dạng mã nhị phân. Giả sử số lần nhấn không quá 255 lần.

Phân tích

- Thiết kế mạch nút nhấn nối vào ngắt **INT0** (RB0).
- Lập trình theo code mẫu phần trên và viết chương trình thêm vào như bên dưới. Lưu ý: lệnh PORTC = dem có thể viết ở chương trình ngắt sau khi tăng biến đếm hoặc trong chương trình chính while(1) như.
- Sinh viên tự vẽ lưu đồ, viết chương trình và chạy thử.
- Lập trình yêu cầu tương tự sử dụng các ngắt **INT1, INT2**

2.3.2 Bài tập dùng hơn 1 ngắt ngoài

Câu 3 a. Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Các điện trở, 05 LED; 02 nút nhấn thường mở (PB).

- LED1 ÷ LED5 được nối với các chân RB3 ÷ RB7; Nút nhấn PB1, PB2 được nối với lần lượt với các chân RB0/INT0, RB2/INT2.

b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

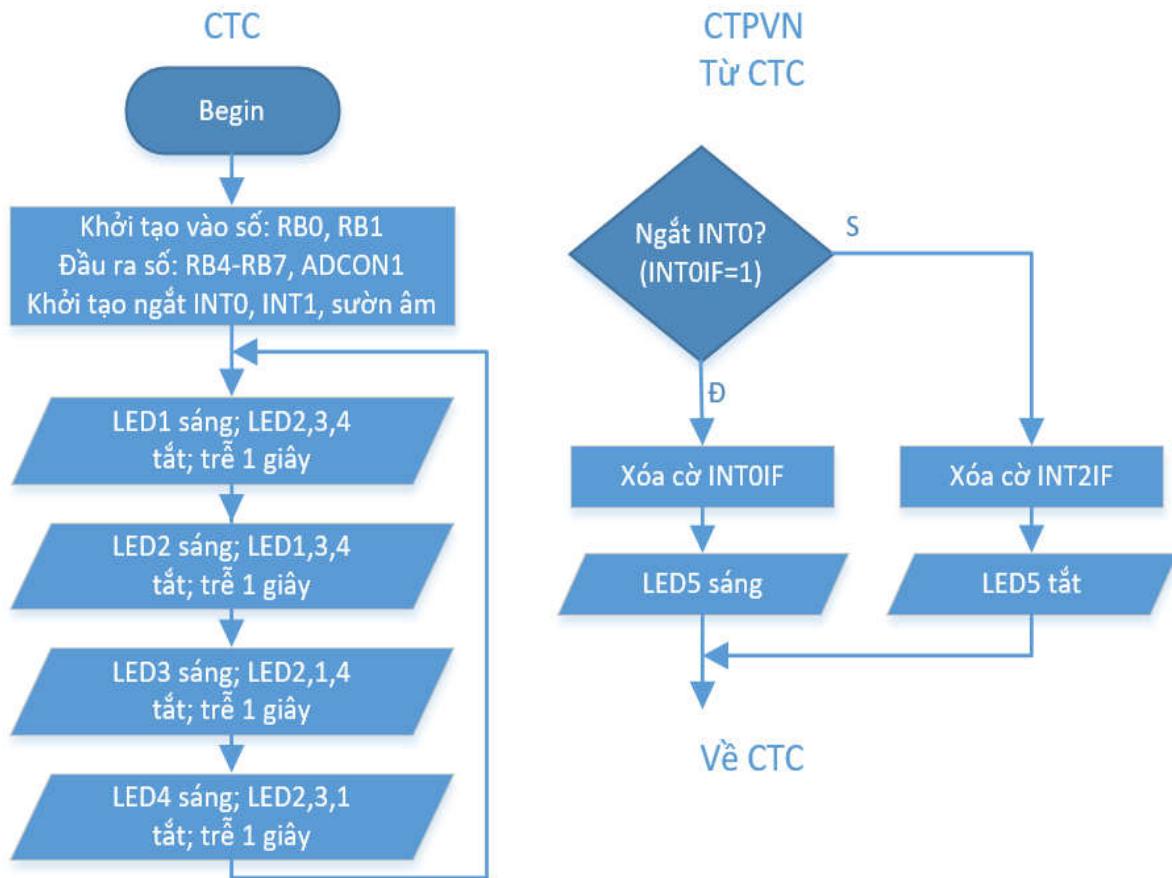
- LED1 ÷ LED4 sáng/tắt liên tục theo chu trình: (1) LED1 sáng, các LED khác tắt trong ~1 giây => (2) LED2 sáng, các LED khác tắt trong ~1 giây => (3) LED3 sáng, các LED khác tắt trong ~1 giây => (4) LED4 sáng, các LED khác tắt trong ~1 giây => (1) => ...

- LED 5 sáng khi nhấn PB1, tắt khi nhấn PB2. Ban đầu (khi chưa nhấn PB1 hoặc PB2), LED 5 tắt.

Phân tích:

- Chức năng không ngắt tương tự bài 1 phần I/O
- Viết bô xung chương trình ngắt

Lưu đồ thuật toán:



Hướng dẫn lập trình.

- Sử dụng chương trình mẫu phần ngắt (trang 32-33 tài liệu tra cứu PIC), tham khảo copy giống với mục 2.2.2.
- Do có 2 ngắt nên khi viết chương trình ngắt cần lệnh kiểm tra cờ ngắt.
- Khai báo **LED5** hoặc cả 5 LED tùy theo lập trình theo PORT hay bits

Khung chương trình 2 ngắt.

```

void ngat_uu_tien_cao(void)
{
    if(INTCONbits.INT0IF == 1) // kiem tra co ngat INTO
    {
        INTCONbits.INT0IF = 0; // Xoa ngat INTO
        // LED5 sang
    }
    if(INTCON3bits.INT2IF == 1) // kiem tra co ngat INT2
    {
        INTCON3bits.INT2IF = 0; // Xoa ngat INT2
        // LED5 tat
    }
}

```

```

void main() {
    TRISB = 0x07;           // RB0-2 in, RB3-7 OUT
    ADCON1 = 0x0F;          // IO vao ra so

    INTCONbits.GIE = 1;     // Cho phep MCU ngat
    INTCONbits.INT0IE = 1;   // cho phep ngat INT0
    INTCON2bits.INTEDG0 = 0; // ngat INT0 suon am
    INTCONbits.INT0IF = 0;   // xoa co ngat INT0
    INTCON3bits.INT2IE = 1;   // cho phep ngat INT2
    INTCON2bits.INTEDG2 = 0; // ngat INT2 suon am
    INTCON3bits.INT2IF = 0;   // xoa co ngat INT2
    while (1) {
        // LED1-LED4 sang tat theo chu tr?nh
    }
}

```

LẬP TRÌNH SỬ DỤNG 2 NGẮT KHÁC: INT0, INT1 HOẶC INT1, INT2

Bài 2. a. Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Các điện trở, 05 LED; 02 nút nhấn thường mở (PB1,2).

- LED1 ÷ LED5 được nối với các chân RB3 ÷ RB7; Nút nhấn PB1, PB2 được nối với lần lượt với các chân RB0/INT0, RB1/INT1.

b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- LED5 sáng/tắt liên tục; trong khi LED1 ÷ LED4 hiển thị số lần nhấn PB1 dưới dạng số nhị phân (LED sáng tương ứng với bit “1”; LED tắt tương ứng với bit “0”). Số lần nhấn PB1 không lớn hơn 15 lần.

- Khi nhấn PB2 số lần nhấn PB1 được RESET về 0.

Hướng dẫn:

- Khai báo LED và ngắt INT0, INT1
- Chương trình chính: LED5 nhấp nháy
- Chương trình phục vụ ngắt:
 - + Ngắt INT0 – đếm số lần nhấn PB1 hiển thị giá trị đếm
 - + Ngắt INT1 – xóa biến đếm, hiển thị giá trị đếm.
- Hiển thị giá trị LED dạng mã nhị phân theo cấu trúc

LED = dem >> x; // hiển thị bitx của biến đếm trên LED

Áp dụng thực tế:

LED1 = dem>>0; // hiển thị bit0 của biến đếm trên LED1

LED2 = dem>>1; // hiển thị bit1 của biến đếm trên LED2

3 Lập trình Timer

Timer có 2 chức năng:

- Định thời: tạo một khoảng thời gian theo yêu cầu.
- Đếm sự kiện: đếm xung từ bên ngoài vào chân TxCKI

3.1 Chế độ định thời

Lưu ý: Các chân nối với các linh kiện, các bộ timer và các tham số (tần số/chu kỳ, Fosc) được thay đổi theo từng đề thi/kiểm tra

PIC18F4520 có 04 bộ timer (trang 36-43 tài liệu tra cứu PIC)

	Timer0	Timer1	Timer3	Timer2
Chế độ	8/16 bit	16bit	16bit	8bit (N<256)
Nguồn xung	Fosc/4 (trong); T0CKI (ngoài)	Fosc/4 (trong); T13CKI (ngoài)	Fosc/4 (trong); T13CKI (ngoài)	Fosc/4 (trong)
Khởi tạo	T0CON OpenTimer0()	T1CON OpenTimer1()	T3CON	T2CON OpenTimer2()
Hệ số chia tần	1,2,4,...,256	1,2,4,8	1,2,4,8	Prescaler: 1,4,16 Postscaler: 1,2,3,..,16
Nạp giá trị ban đầu	TMR0H= TMR0L= WriteTimer0()	TMR1H= TMR1L= WriteTimer1()	TMR3H= TMR3L= WriteTimer3()	PR2= N
Điều kiện tràn	Giá trị của timer lớn hơn 255 (8bit)/ 65535(16bit)	Giá trị của timer lớn hơn 65535	Giá trị của timer lớn hơn 65535	Số lần TMR2=PR2 trùng với Postscaler đã đặt
Cờ tràn	TMR0IF	TMR1IF	TMR3IF	TMR2IF

Cách tính thời gian tràn:

Gọi x là hệ số chia tần; Fosc(Mhz) là tần số hoạt động của VĐK; 256-N (với chế độ 8bit) hoặc 65536-N (với chế độ 16bit) là giá trị ban đầu của timer, T là chu kỳ của xung đầu ra, K: hệ số chia tần của Timer, **Fosc**: tần số thạch anh, **F**: Tần số xung.

$$T \text{ thời gian tràn (uS)} = \frac{4 \cdot N \cdot K}{Fosc} = \frac{T}{2} \Rightarrow N = \frac{Fosc \cdot T}{4 \cdot 2 \cdot K} = \frac{Fosc}{4 \cdot 2 \cdot F \cdot K}$$

Bài 1. a. Thiết kế mạch điện như sau:

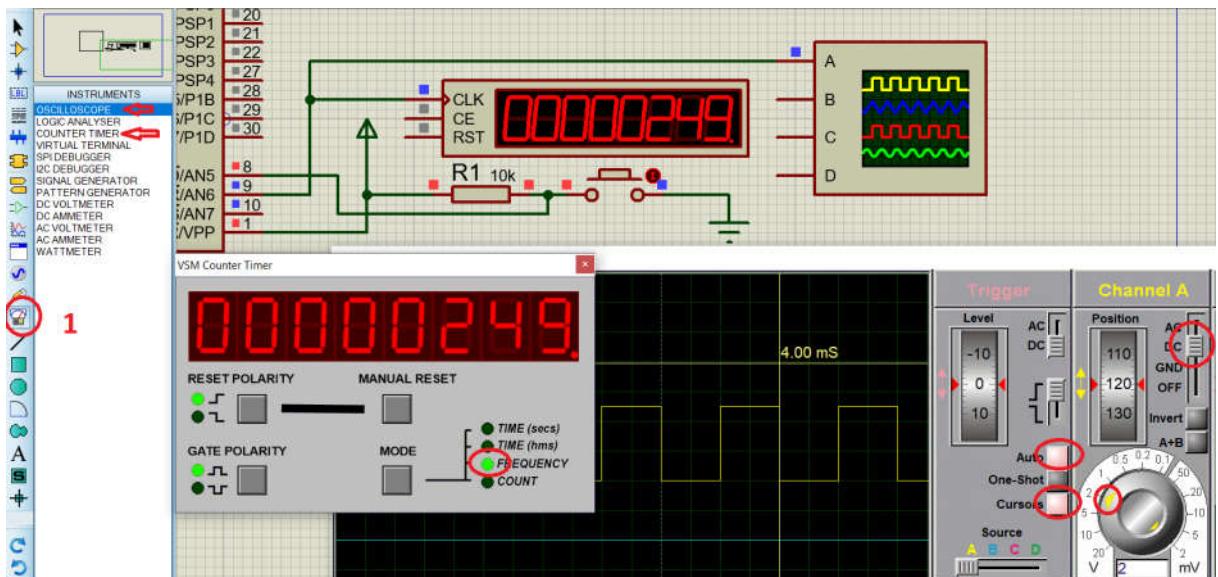
- Các linh kiện được sử dụng: Điện trở, nút nhấn thường mở (PB).
- Nút nhấn PB được nối với chân RE0; Chân RE1 được nối với Oscilloscope và Counter Timer.

b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- Khi PB ở trạng thái nhấn: Chân RE1 có mức logic 0;
- Khi PB ở trạng thái nhả: Sử dụng Timer0/1/2/3 tạo xung có tần số $\approx 250\text{Hz}$ trên chân RE1 (bỏ qua sai số do thời gian thực hiện lệnh của vi điều khiển). PIC18F4520 hoạt động ở tần số 8Mhz ($F_{osc}=8\text{Mhz}$).

Sơ đồ mạch:

Nút nhấn nối RE0, chân RE1 nối với **Oscilloscope**, **Counter Timer** để kiểm tra dạng xung và đo tần số. Lưu ý chọn các chế độ như hình bên dưới.



Nếu Oscilloscope bị mất màn hình, chạy mô phỏng rồi vào menu **Debug** => **Digital Oscilloscope** (ở bên dưới gần cuối).

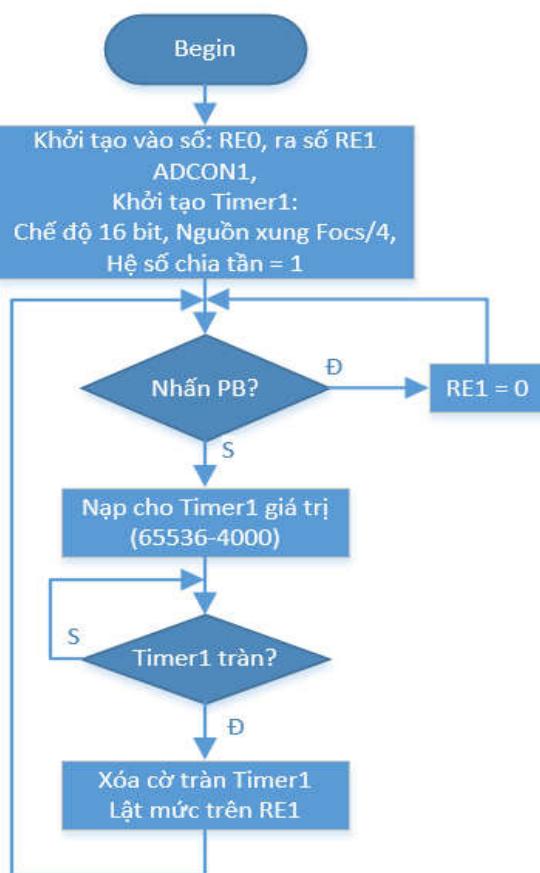
Hướng dẫn với Timer1:

$$N \cdot K = \frac{F_{osc}}{4 \cdot 2 \cdot F_{out}} = \frac{8M}{4 \cdot 2 \cdot 250} = 4000 < 2^{16} = 65536$$

Trong đó: (N: giá trị nạp cho Timer, $N < 2^{\text{bits}}$, K: hệ số chia tần – Prescaler)

Do $N \cdot K < 2^{16} = 65536 \Rightarrow$ Chọn hệ số chia tần **K = 1** $\Rightarrow N = 4000$

Tương tự với Timer0 và Timer3, lựa chọn giá trị hệ số chia tần đầu vào (Prescale) là 1, chọn xung CLK = $F_{osc}/4$. Giá trị nạp tương tự với Timer0.



Có 2 phương pháp sử dụng khởi tạo giá trị cho Timer:

- Sử dụng thanh ghi TxCON
- Sử dụng hàm trong thư viện **timers.h**

Khi sử dụng hàm thì cần lưu ý, thêm thư viện `#include <timers.h>` vào bên dưới thêm thư viện `#include <p18f4520.h>`. Một số hàm sử dụng:

`OpenTimerx(); // Khởi tạo`

`WriteTimerx(a); // Nạp giá trị a vào Timerx`

Chi tiết các hàm xem trong file **timers.h** hoặc **trang 44 của Tài liệu tra cứu PIC.pdf**

Chương trình sử dụng thanh ghi, khai báo thư viện, PB – RE0, xung – RE1

```

void main() {
    TRISE = 0b01;      // RE1 Output, RE0 Input
    ADCON1 = 0x0F;     // PIN Digital mode
    T1CON = 0B00000001; // On TMR1, 16 bit, PS = 1;
    while(1) {
        if(PB==0){    // Press PB
            PORTEbites.RE1=0;
        }
        else {        // Release PB
            TMR1H = (65536-4000)/256;    // byte cao
            TMR1L = (65536-4000)%256;    // byte thap
            while(PIR1bits.TMR1IF==0);   // wait TMR1IF=1
            xung = ~xung;                // Toggle pin RE1
            PIR1bits.TMR1IF=0;           // clear TMR1IF
        }
    }
}

```

Với chương trình dùng hàm cần thêm thư viện `#include <timers.h>`

```

#include <p18f4520.h>
#include <timers.h>      // Add library for Timer function
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
#define PB      PORTEbits.RE0
#define xung    PORTEbits.RE1
void main() {
    TRISE = 0b01;    // RE1 Output, RE0 Input
    ADCON1 = 0x0F;   // PIN Digital mode
    OpenTimer1(TIMER_INT_OFF    // no interrupt
                & T1_PS_1_1    // Prescale = 1
                & T1_SOURCE_INT); // CLK = Fosc/4
    while(1) {
        if(PB==0){ // Press PB
            PORTEbits.RE1=0;
        }
        else { // Release PB
            WriteTimer1(65536-4000); // Write to Timer1
            while(PIR1bits.TMR1IF==0); // wait TMR1IF=1
            xung = ~xung;           // Toggle pin RE1
            PIR1bits.TMR1IF=0;      // clear TMR1IF
        }
    }
}

```

Đối với **Timer0**, **Timer3** thì làm tương tự **Timer1**, **Timer2** có sự khác biệt chút khi giá trị thanh ghi chỉ là 8 bit (tối đa 256) và nạp vào thanh ghi PR2. Ngoài ra **Timer2** có đếm số lần **TMR2 = PR2** rồi mới xảy ra ngắt nên công thức sẽ khác có thêm tham số **P**: số lần **TMR2 = PR2**. Nếu **P=1** sau 1 lần **TMR2 = PR2** thì Timer2 tràn (TMR2IF=1). Nếu **P=4** sau 4 lần **TMR2=PR2**, Timer2 sẽ tràn (TMR2IF=1), như sau:

$$T \text{ thời gian tràn (uS)} = \frac{4 \cdot K \cdot P \cdot N}{Fosc} = 2000 \mu\text{s} \Rightarrow \frac{4KPN}{8} = 2000 \\ \Rightarrow KPN = 4000, C \text{ ọn } K, P \text{ để } N < 256$$

Lựa chọn **K = 1, 4, 16** và **P = 1....16** để **N < 256**

$K \cdot P \cdot N = 4000 > 256 \Rightarrow K \cdot P \geq 4000/256 = 16 = K_{max}$. Nên chọn **K = 16, P = 1** $\Rightarrow N = 4000/K \cdot P = 4000/16 = 250 = PR2$

Chương trình sử dụng hàm như sau:

```
#include <p18f4520.h>
#include <timers.h>      // Add library for Timer function
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
#define PB      PORTEbits.RE0
#define xung    PORTEbits.RE1
void main() {
    TRISE = 0b01;      // RE1 Output, RE0 Input
    ADCON1 = 0x0F;     // PIN Digital mode
    OpenTimer2(TIMER_INT_OFF      // no interrupt
                & T2_PS_1_16    // Prescale = 16
                & T2_POST_1_1); // Postscaler = 1
    PR2 = 250;         // Write to PR2 of Timer2
    while(1) {
        if(PB==0){    // Press PB
            PORTEbits.RE1=0;
        }
        else {        // Release PB
            while(PIR1bits.TMR2IF==0); // wait TMR2IF=1
            xung = ~xung;           // Toggle pin RE1
            PIR1bits.TMR2IF=0;      // clear TMR2IF
        }
    }
}
```

Như vậy nếu so sánh 2 phương pháp sử dụng hàm và thanh ghi, thì phương pháp sử dụng hàm sẽ đơn giản hơn, không cần tra cứu chi tiết giá trị thanh ghi của từng Timer, nhưng làm theo phương pháp nào thì cần nhớ được công thức tính tần số đầu ra, đặc biệt là **Timer2** khác so với các **Timer** khác.

Câu 2.

a. Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Điện trở, nút nhấn thường mở (PB).

- Nút nhấn PB được nối với chân RC0; Chân RD1 được nối với

Oscilloscope và Counter Timer.

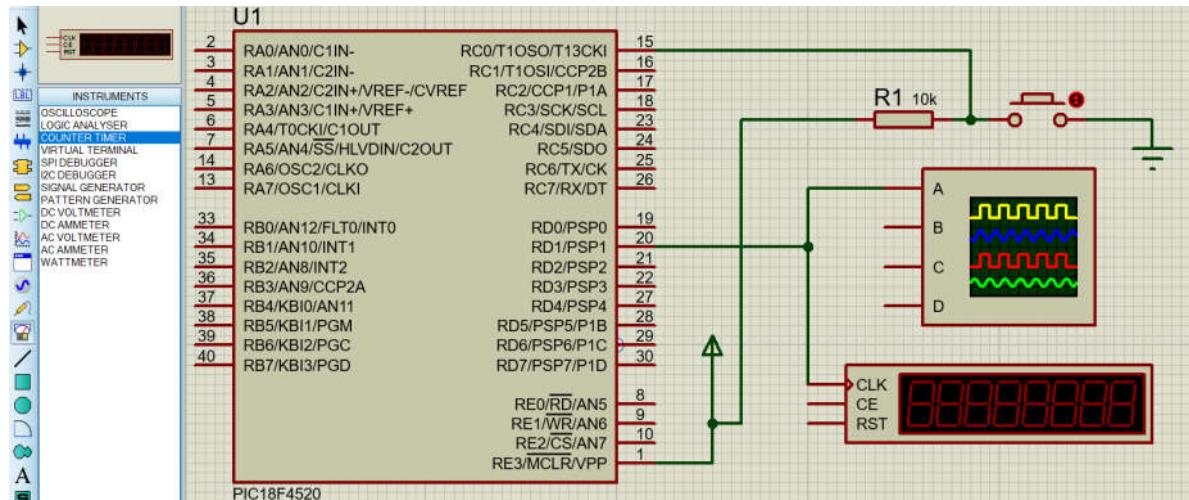
b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- Khi PB ở trạng thái nhấn: Sử dụng Timer0/1/2/3 tạo xung có tần số \approx 50Hz trên chân RD1;

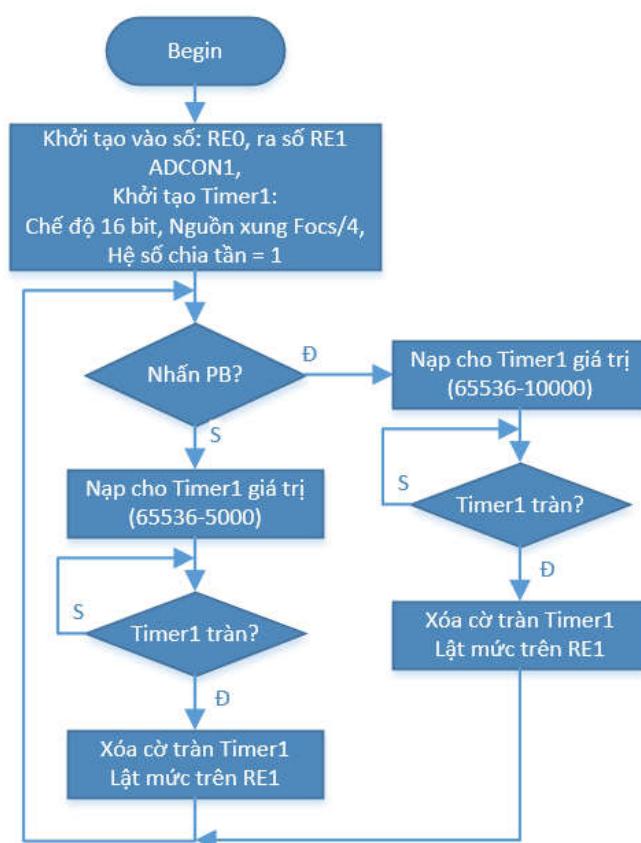
- Khi PB ở trạng thái nhả: Sử dụng Timer0/1/2/3 tạo xung có tần số \approx 100Hz trên chân RD1.

Bỏ qua sai số do thời gian thực hiện lệnh của vi điều khiển; PIC18F4520 hoạt động ở tần số 4Mhz ($F_{osc}=4Mhz$).

Tạo 01 nút nhấn trên RC0, chân RD1 nối với **Oscilloscope, Counter Timer**. Lưu ý chọn các chế độ như hình bên dưới.



Lưu đồ thuật toán



Tính giá trị cho Timer0, 1, 3

Cần tính 2 giá trị và nạp khác nhau khi nhấn phím hay nhả phím. Sử dụng công thức tính

$$N = \frac{F_{osc}}{4 \cdot 2 \cdot F \cdot K} = \frac{4M}{4 \cdot 2 \cdot F \cdot K} \Rightarrow$$

$$NFK = 500000$$

Chọn K = 1:

- Với tần số 50Hz: N= 10000
- Với tần số 100Hz: N= 5000

Tương tự như bài trên, các em tự hoàn thiện chương trình bài này.

- Thay vì khi nhấn phím **RE0= 0** thì viết chương trình như phần nhả phím và chỉnh các thông số nạp cho Timer

3.2 Chế độ đếm sự kiện

Vì sử dụng hàm trong thư viện timers.h sẽ đơn giản hơn việc dùng thanh ghi. Nên trong phần này sẽ hướng dẫn làm theo phương pháp sử dụng hàm.

Cần lưu ý, đếm sự kiện chỉ có **Timer0, 1, 3** còn **Timer2** thì không.

Sự khác biệt đối với chế độ định thời là lựa chọn nguồn xung

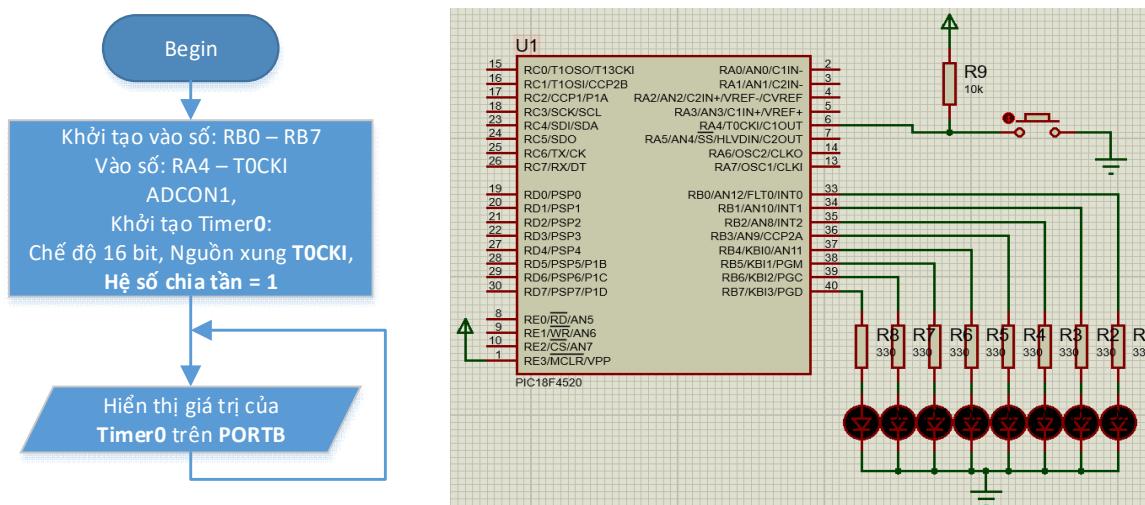
Tx_SOURCE_INT => Tx_SOURCE_EXT và thêm lựa chọn sườn xung

T0_EDGE_FALL hoặc **T0_EDGE_RISE**. Hệ số chia tần luôn là 1:

Tx_PS_1_1.

3.2.1 Hiển thị LED đơn

Bài 1: Sử dụng TIMER 0/1/3 chế độ đếm sự kiện, hiển thị giá trị đếm trên PORTB dưới dạng mã nhị phân.



3.2.1.1 Sử dụng **TIMER0** chế độ thanh ghi.

```

// Dem so lan nhan phim Su dung Timer0,
// Hien thi 8 LED don PORTB, ma nhi phan
#include <p18f4520.h>
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
void main(){
    ADCON1 = 0x0F;           // Cac chan che do Digital
    TRISA = 0xFF;            // RA4 INPUT
    TRISB = 0;                // PORTB OUT - LED
    T0CON = 0b10111000; // T0 ON, 16 BIT, PS = 1, TOCKI, suon am
    while(1){
        PORTB = TMR0L; // Hien thi gia tri TMR0L => PORTB
    }
}

```

Chương trình trên hiển thị 16 bit (16LED) nên cần 2 PORTB và PORTD. Giá trị hiển thị trên PORTD là 8 bit cao (TMRxH)

3.2.1.2 Sử dụng hàm trong thư viện *timers.h*

Xem các hàm trang 44 trong Tài liệu tra cứu PIC

```
// Dem so lan nhan phim Su dung Timer0,
// Hien thi 8 LED don PORTB, ma nhan phan
#include <p18f4520.h>
#include <timers.h>      // Thu vien su dung cac ham Timer
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
void main() {
    ADCON1 = 0x0F;          // Cac chan che do Digital
    TRISA = 0xFF;           // RA4 INPUT
    TRISB = 0;               // PORTB OUT - LED
    OpenTimer0( TIMER_INT_OFF &           // Khong ngat
                T0_SOURCE_EXT &           // XUNG T0CKI
                T0_EDGE_FALL &           // Suon am
                T0_16BIT &              // 16 bit
                T0_PS_1_1 );            // PS = 1
    while(1) {
        PORTB = ReadTimer0();       // Hien thi gia tri TMRO => PORTB
    }
}
```

Hiển thị 8 bit cao: **PORTD = ReadTimer0()>>8;**

Dùng đếm sử dụng lệnh **T0CON = 0;**

3.2.1.3 Chế độ đếm Timer1

Tương tự thay đổi cho **TIMER1**, nút nhấn nối **RC0/T13CKI** thay cho **RA4**. Khởi tạo Timer1: **T1CON** hoặc **OpenTimer1(...)**; trang 45 tài liệu tra cứu PIC.

```
T1CON = 0b00000011; // T13CKI, TIMERON
OpenTimer1( TIMER_INT_OFF &           // Khong ngat
            T1_SOURCE_EXT &           // XUNG T13CKI
            T1_OSC1EN_OFF &           // T1OSC OFF
            T1_PS_1_1 );             // PS = 1
```

Sử dụng hàm đọc giá trị **ReadTimer1()**; hoặc thanh ghi **TMR1H**, **TMR1L**. Dùng đếm sử dụng lệnh **T1CON = 0;**

3.2.1.4 Chế độ đếm Timer3

- **Timer3** thì nên sử dụng thanh ghi nút nhấn nối **RC0/T13CKI** thay cho **RA4** **T3CON = 0b00000011; // T13CKI, TIMERON**

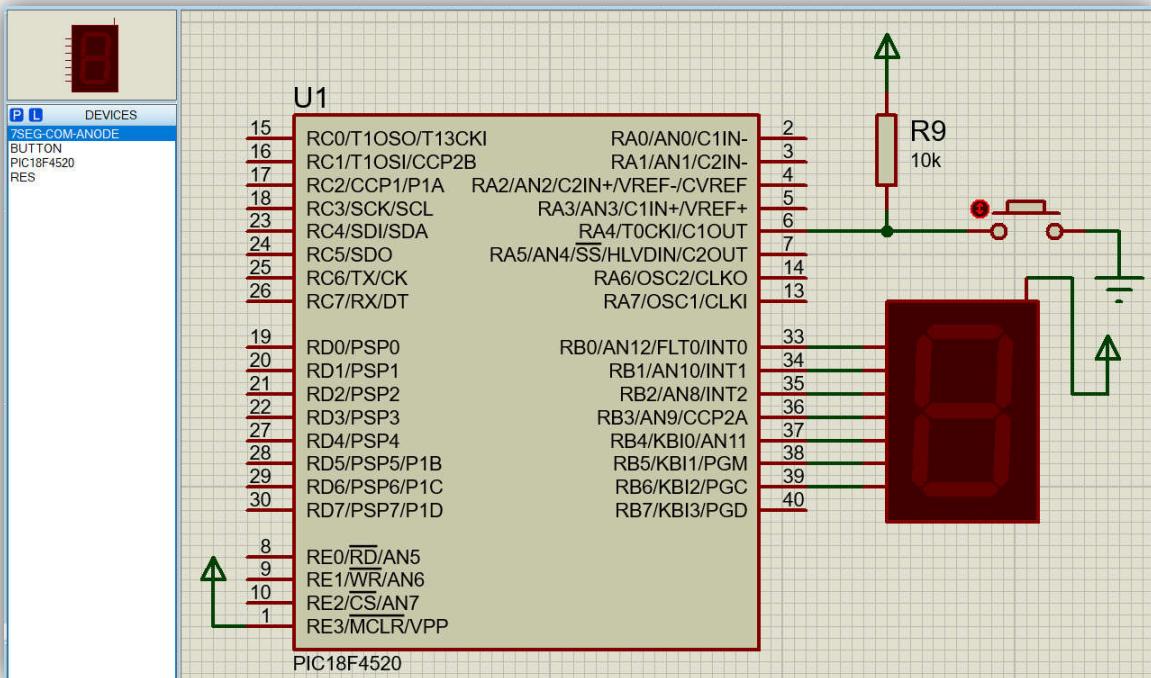
Sử dụng hàm đọc giá trị **ReadTimer3()**; hoặc thanh ghi **TMR3H**, **TMR3L**. Dùng đếm sử dụng lệnh **T3CON = 0;**

3.2.2 Hiển thị LED 7 thanh

Bài 2: Sử dụng TIMER 0/1/3 chế độ đếm sự kiện, hiển thị hàng đơn vị giá trị đếm trên LED 7 thanh.

Thuật toán giống bài 1 chỉ khác phần hiển thị.

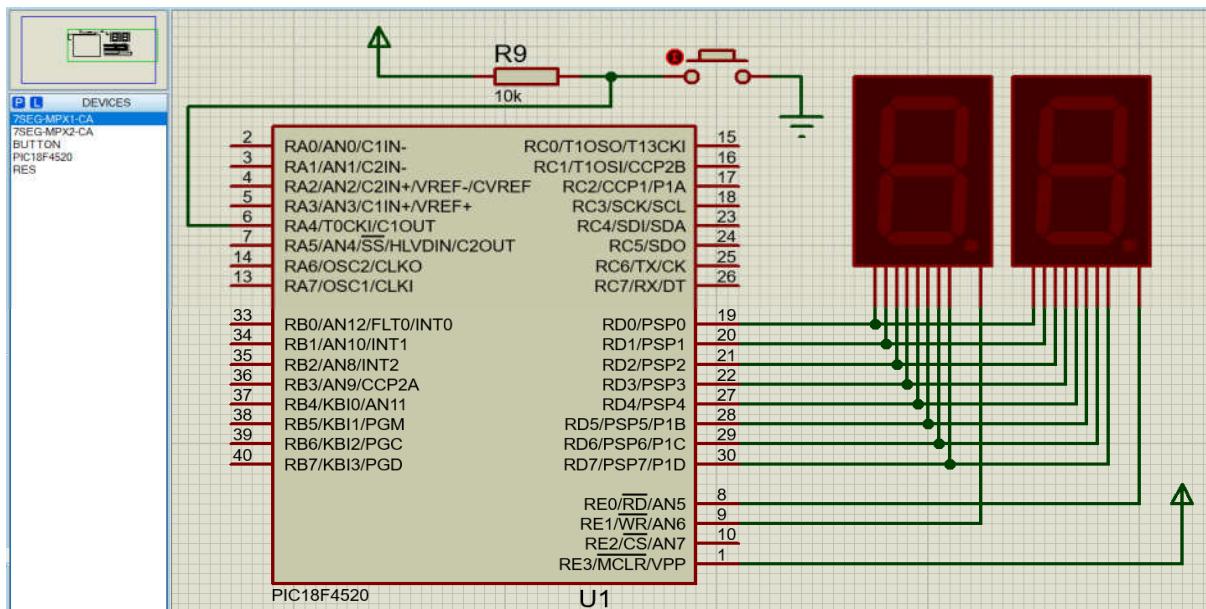
Hiển thị trên 01 LED 7 thanh. Mã LED trang 73 Tài liệu tra cứu PIC



```
// Dem so lan nhan phim Su dung Timer0,
// Hien thi 8 LED don PORTB, so don vi LED 7 SEG
#include <p18f4520.h>
#include <timers.h>
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
int count;
const char ma_led[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
void main(){
    ADCON1 = 0x0F;           // Cac chan che do Digital
    TRISA = 0xFF;            // RA4 INPUT
    TRISB = 0;                // PORTB OUT - LED
    OpenTimer0( TIMER_INT_OFF &          // Khong ngat
                T0_SOURCE_EXT &        // XUNG T0CKI
                T0_EDGE_FALL &        // Suon am
                T0_16BIT &             // 16 bit
                T0_PS_1_1 );           // PS = 1
    while(1){
        count = ReadTimer0();           // doc gia tri Timer0
        PORTB = ma_led[count%10];      // Hien thi hang don vi LED 7SEG
    }
}
```

Đọc giá trị 16 bit theo thanh ghi: **count = TMR0H*256 + TMR0L**

Bài 3: Sử dụng TIMER 0/1/3 chế độ đếm sự kiện, hiển thị trên 02 LED 7 thanh.



Chương trình đếm sự kiện sử dụng hàm Timer0 hiển thị 02 LED 7 thanh

```
// Dem so lan nhan phim Su dung Timer0,
// Hien thi 02 LED 7 thanh
#include <p18f4520.h>
#include <delays.h>
#include <timers.h>      //Thu vien su dung cac ham Timer
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
#define LED1    PORTEbits.RE0    // Don vi
#define LED2    PORTEbits.RE1    // Chuc

const char ma_led[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int count,i;
void main(){
    ADCON1 = 0x0F;          // Cac chan che do Digital
    TRISA = 0xFF;           // RA4 INPUT
    TRISD = 0;               // PORTD data 7 SEG
    TRISE = 0;               // PORTE Control 7 SEG
    OpenTimer0( TIMER_INT_OFF &           // Khong ngat
                T0_SOURCE_EXT &           // XUNG T0CKI
                T0_EDGE_FALL &           // Suon am
                T0_16BIT &              // 16 bit
                T0_PS_1_1 );            // PS = 1
    while(1){
        count = ReadTimer0();           // doc gia tri Timer0
        LED1 = 1; LED2 = 0;             // Hang don vi PORTE = 0x01;
        PORTD = ma_led[count%10];     // Hien thi hang don vi LED 7SEG
        Delay1KTCYx(1);
        PORTD = 0xFF;                 // Tat LED
        LED1 = 0; LED2 = 1;             // Hang chuc PORTE = 0x02;
        PORTD = ma_led[(count/10)%10]; // Hien thi hang don vi LED 7SEG
        Delay1KTCYx(1);
        PORTD = 0xFF;                 // Tat LED
    }
}
```

Khi muốn dừng đếm thì xóa giá trị TxCON = 0; (x = 0, 1, 3)

4 Lập trình UART

Sử dụng hàm trong thư viện #include <uart.h> trang 66 Tài liệu tra cứu PIC
x= ReadUSART(); // đọc dữ liệu lưu vào biến x
putrsUSART(" HaUI"); // truyền chuỗi ký tự "HaUI"
WriteUSART(x); // ghi 1 ký tự ra USART
while(BusyUSART()); // Cho USART truyền xong
WriteUSART(13); // xuống dòng hoặc '\n'

Tính tốc độ baud, hệ số $spbrg = n = \frac{Fosc}{16 \text{ baud}} - 1$

Ví dụ $Fosc = 4\text{Mhz}$, $\text{baud} = 9600 \Rightarrow n = \frac{Fosc}{16 \text{ baud}} - 1 = \frac{4M}{16 \cdot 9600} - 1 = 25$

Bài 1: a. Thiết kế mạch điện như sau: (câu 11 trong bộ câu hỏi các câu phân 2)

Mạch điện gồm các điện trở, các LED và VIRTUAL TERMINAL

- USART của PIC18F4520 nối với VIRTUAL TERMINAL.

- LED1-LED8 lần lượt được nối với các chân RD0-RD7 tương ứng.

b. Viết công thức, tính giá trị của đối số spbrg trong hàm OpenUSART (hoặc giá trị ghi vào cặp thanh ghi SPBRGH: SPBRG) và viết chương trình theo yêu cầu như sau:

Thiết lập USART với các thông số: Không sử dụng ngắt; chế độ cận đồng bộ; truyền/nhận 8 bit; nhận liên tục các byte; tốc độ baud cao; tốc độ baud: 19200. PIC18F4520 hoạt động ở tần số 11.0592 Mhz.

- Truyền qua USART 02 byte có giá trị 0x40 và 0x61;

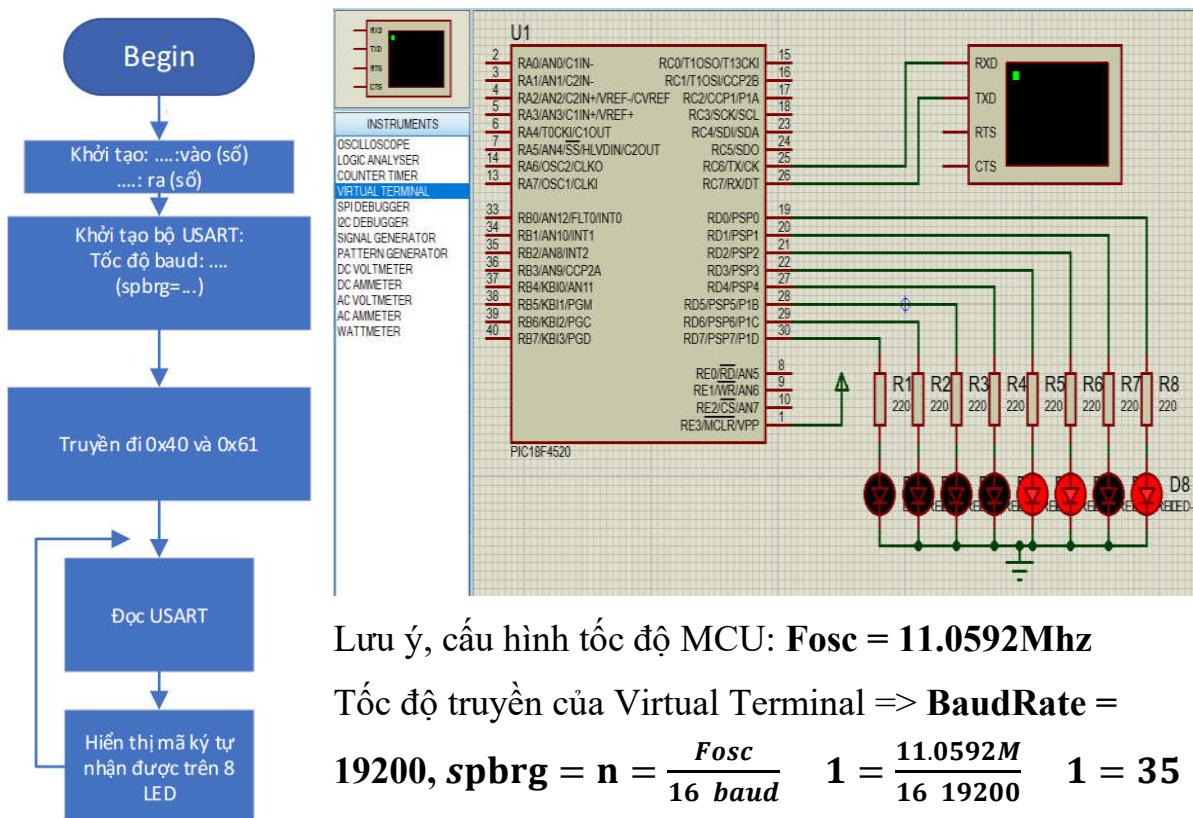
- Liên tục nhận về các ký tự từ VIRTUAL TERMINAL và hiển thị mã của ký tự nhận được dưới dạng số nhị phân trên các LED1-LED8 (LED sáng tương ứng bit "1", LED tắt tương ứng bit "0").

Phân tích:

- Nối chân **RC7 – RX, RC6 – TX** với **Virtual Terminal**.
- **8 LED** nối **PORTD**
- Tính giá trị **spbrg** theo **Fosc** và **baud**
- **Fosc = 11.0592Mhz, baud = 19200 =>**

$spbrg = n = \frac{Fosc}{16 \text{ baud}} - 1 = \frac{11.0592M}{16 \cdot 19200} - 1 = 35$

Thuật toán và sơ đồ mạch.



Lưu ý, cấu hình tốc độ MCU: **Fosc = 11.0592Mhz**

Tốc độ truyền của Virtual Terminal => **BaudRate =**

$$19200, \text{ spbrg} = n = \frac{Fosc}{16 \text{ baud}} = \frac{11.0592M}{16 \cdot 19200} = 1 = 35$$

Chương trình

```
// USART, cau 11
#include <p18f4520.h>
#include <uart.h>
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
void main() {
    char x;
    TRISC = 0x80; // 0b10000000;
    TRISD = 0;
    OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNC_MODE &
              USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH, 35);
    // baud 19200, Fosc = 11.0592Mhz; SPBRG = (Fosc/16/baud)-1
    // =(11.0592M/16/19200)-1=35
    WriteUSART(0x40); // gửi UART 0x40
    while(BusyUSART()); // chờ USART truyền xong
    WriteUSART(0x61); // gửi UART 0x61
    while(BusyUSART()); // chờ USART truyền xong
    while(1) {
        x = ReadUSART(); // nhận USART lưu vào x
        PORTD = x; // Hiển thị dạng nhị phân trên PORTD
    }
}
```

Câu 12 a. Thiết kế mạch điện như sau:

Mạch điện gồm các điện trở, LED và VIRTUAL TERMINAL.

- USART của PIC18F4520 nối với VIRTUAL TERMINAL.

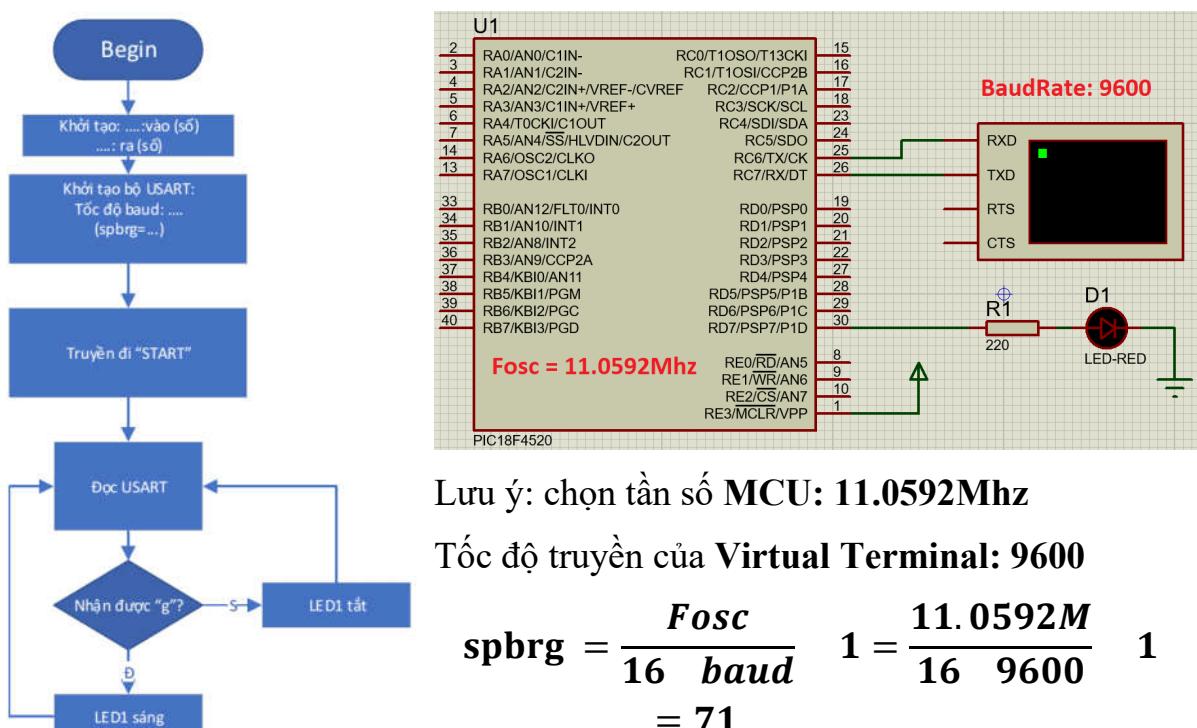
- LED1 nối với các chân RD7 .

b. Viết công thức, tính giá trị của đối số spbrg trong hàm OpenUSART (hoặc giá trị ghi vào cặp thanh ghi SPBRGH: SPBRG) và viết chương trình theo yêu cầu như sau:

Thiết lập USART với các thông số: Không sử dụng ngắn; chế độ cận đồng bộ; truyền/nhận 8 bit; nhận liên tục các byte; tốc độ baud cao; tốc độ baud: 9600. PIC18F4520 hoạt động ở tần số 11.0592 Mhz.

- Truyền qua USART các ký tự START;
- Liên tục nhận về các ký tự từ VIRTUAL TERMINAL và điều khiển LED1 sáng khi nhận được ký tự “g”, tắt khi nhận được các ký tự khác.

Thuật toán và sơ đồ mạch



Chương trình

```
// USART, cau 12
#include <p18f4520.h>
#include <uart.h>
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config MCLRE=ON
#define LED1 PORTDbits.RD7
void main() {
    char x;
    TRISC = 0x80; // 0b10000000;
    TRISD = 0;
    OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &
              USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH, 71);
    // baud 9600, Fosc = 11.0592Mhz; SPBRG = (Fosc/16/baud)-1
    // =(11.0592M/16/9600)-1=71
    putsUSART("START"); // chuoi ky tu "START"
    while(1) {
        x = ReadUSART(); // nhan USART luu vao x
        if(x=='g') LED1 = 1; // LED sang
        else LED1 = 0; // LED1 tat
    }
}
```

Khi muốn hiển thị **số đếm** trên **Virtual Terminal** thì cần khai báo thư viện và mảng ở trên chương trình chính như sau:

```
#include <stdio.h> // Thu vien dung ham sprintf
char txt[40]; // Khai bao mang
```

Số phần tử mảng cần lớn hơn số ký tự cần hiển thị, trong ví dụ khởi tạo mảng có 40 phần tử. Sử dụng 2 lệnh sau để chuyển kiểu và hiển thị biến đếm.

```
sprintf(txt, "So dem: %d", dem);
putsUSART(txt);
```

Trong đó các kiểu biến và cách sử dụng chuyển kiểu:

	Ký hiệu chuyển	Kiểu biến
Số tự nhiên	%d	int, char, unsigned char, unsinged int
Ký tự	%c	char
Số thực	%f	float, double

5 Biến đổi tương tự số ADC

Trong PIC18F4520 có 01 bộ biến đổi ADC **10 bit** với **13 kênh** đầu vào khác nhau (**AN0 – AN12**).

Có 2 cách thức để viết chương trình, dùng thanh ghi và sử dụng hàm.

- Thanh ghi điều khiển ADC gồm: ADCON0, ADCON1, ADCON2
- Sử dụng hàm trong thư viện adc.h chi tiết **trang 65 tài liệu tra cứu PIC**

Hàm	Mô tả
BusyADC	Hàm báo bận.
CloseADC	Cắt hoạt động chuyển động A/D.
ConvertADC	Bắt đầu quá trình chuyển đổi A/D.
OpenADC	Cấu hình cho bộ chuyển đổi A/D.
ReadADC	Đọc giá trị trả về của chuyển đổi A/D.
SetChanADC	Chọn kênh đầu vào cho bộ A/D.

Trong đó hàm **OpenADC** chi tiết và mẫu ở **trang 57** tuy nhiên cần thêm vào mẫu lệnh **ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS** thì đầy đủ. Hoặc để **ADCON1** bên dưới lệnh **OpenADC** thì không cần để **ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS** trong hàm OpenADC.

Lưu ý khi sử dụng giá trị **ADCON1** được cấu hình các chân đầu vào tương tự sẽ là chữ **A** trong bảng **Bit 3-0 PCFG<3:0> trang 53**.

Các ví dụ trước không sử dụng ADC thì **ADCON1 = 0x0F**, trong phần này khi khác. Ví dụ sử dụng **AN0** thì **ADCON1** theo bảng tối đa là **0x0E** để **AN0** là đầu vào tương tự (**A**). Đồng thời giá trị của **ADCON1** cũng được đặt vào cuối lệnh **OpenADC** thay số **15** trong chương trình mẫu.

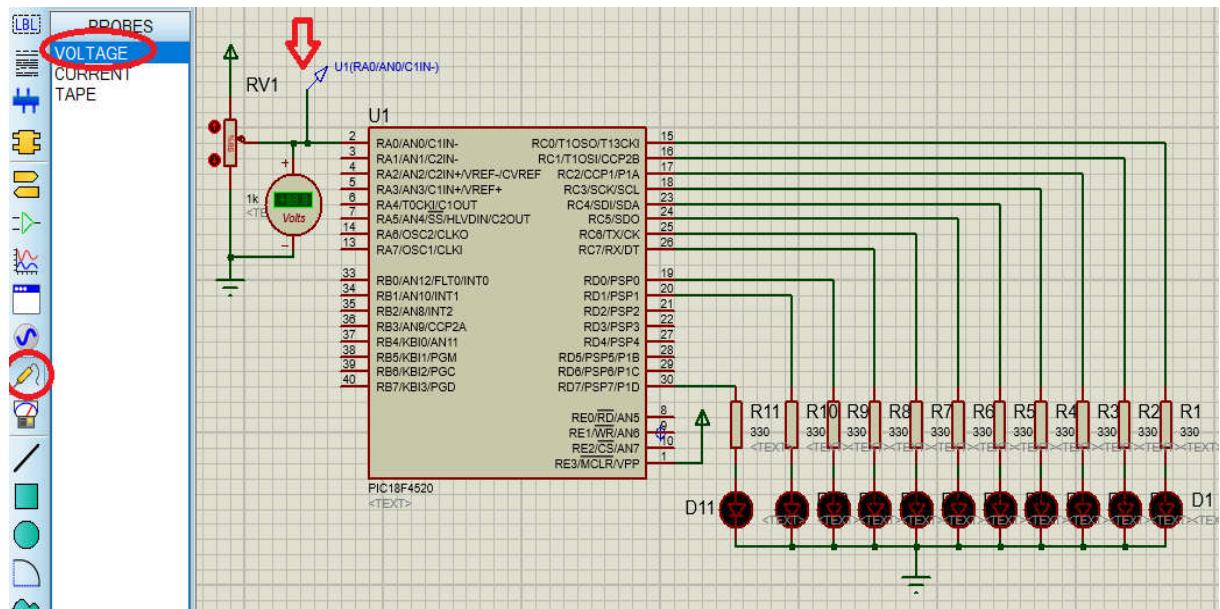
Ví dụ chọn kênh **AN6 => ADCON1 = 0x08** hoặc **0b1000**. Lúc này mặc định các chân **AN0-5** đều là đầu vào tương tự, như vậy tổng có thể sử dụng tất cả 7 đầu vào tương tự **AN0-6**.

5.1 Bài tập

Bài tập 1: a.Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Các điện trở; 11 LED đơn; 01 biến trở.
 - 08 LED (LED1-LED8) nối lần lượt với các chân RC0-RC7; 02 LED (LED9-LED10) nối lần lượt với các chân RD0-RD1.
 - Sử dụng biến trở để tạo điện áp 0 đến 5V trên chân AN0.
- b.Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:
- Đọc giá trị số biến đổi được từ điện áp trên kênh AN0 và hiển thị dưới dạng số nhị phân trên LED1-LED10. LED sáng tương ứng bit “1”, LED tắt tương ứng bit “0”; bit có trọng số lớn nhất (D9) tương ứng LED10, bit có trọng số nhỏ nhất (D0) tương ứng LED1.

Sơ đồ mạch mô phỏng. Biến trở **POT-HG**, đo điện áp Probe => Voltage hoặc DC Voltage meter

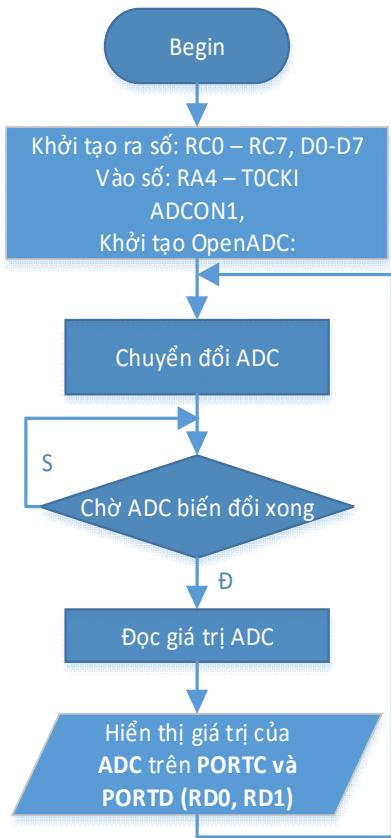


$$\text{Công thức tính giá trị sau khi biến đổi ADC : } D_{OUT} = \frac{V_{in} - V_{REF-}}{V_{REF+} - V_{REF-}} (2^{bit} - 1)$$

$$\text{Nếu } V_{REF+} = VCC = 5V, V_{REF-} = VSS = 0V \Rightarrow D_{OUT} = \frac{V_{in}}{5} \cdot 1023$$

$$\text{VD: Nếu } V_{ref+} = 5V, V_{ref-} = 2V, V_{in} = 3V \Rightarrow D_{OUT} = \frac{3-2}{5-2} \cdot 1023 = 341$$

Lưu đồ thuật toán và chương trình.



Thêm thư viện adc.h

Khởi tạo ADCON1 sao cho AN0 là đầu vào tương tự. Khởi tạo các chân vào ra khác.

Quy trình biến đổi ADC

- Chọn kênh –
SetChanADC(ADC_CHx); // **x=0-12**
- Biến đổi ADC – ConvertADC();
- Chờ biến đổi xong while(BusyADC());
- Đọc kết quả ADC – ReadADC();

Khi muốn chuyển kênh khác thì thay vào ADC_CH0 trong hàm OpenADC theo kênh tương ứng, và lựa chọn ADCON1 (PCFG3: PCFG0) sao cho kênh (chân VDK) tương ứng là đầu vào tương tự. Giá trị PCFG cũng được đặt vào cuối của hàm OpenADC.

```
// Doc AN0 Hien thi 10 LED don, 8bit thap PORTC, 2 bit RD0, RD1
// khai bao thu vien
#include<p18f4520.h>
#include<delays.h>
#include<adc.h>           // Thu vien ADC
// Cau hinh he thong
#pragma config OSC = HS
#pragma config MCLRE = ON
#pragma config LVP = OFF
// chuong trinh chinh
void main(){
    unsigned int adc;
    TRISC =TRISD =0; // thiet lap huong du lieu
    OpenADC( ADC_FOSC_32 & ADC_RIGHT_JUST
              & ADC_12_TAD, ADC_CH0 & ADC_INT_OFF
              & ADC_VREFPLUS_VDD& ADC_VREFMINUS_VSS, 0b1110 );
    ADCON1 = 0b1110;      // VREF+ = VDD, VREF- = VSS, AN0
    while(1){
        ConvertADC();      // Bien doi ADC
        while(BusyADC()); // Cho bien doi xong ADC
        adc = ReadADC();   // Doc ket qua ADC
        PORTC = adc;       // Hien thi Byte thap
        PORTDbits.RD0 = adc>>8; // hien thi RD0 = bit8
        PORTDbits.RD1 = adc>>9; // hien thi RD1 = bit9
    }
}
```

5.2 Các dạng bài ADC.

Dự theo công thức tính $D_{OUT} = \frac{V_{in}-V_{REF-}}{V_{REF+}-V_{REF-}} (2^{bit} - 1)$

Với $V_{REF+} = VCC = 5V$, $V_{REF-} = VSS = 0V \Rightarrow D_{OUT} = \frac{V_{in}}{5} \cdot 1023$

Nếu $V_{in} = 2V \Rightarrow D_{OUT} = \frac{2}{5} \cdot 1023 = \frac{2}{5} \cdot 1023 = 409$

Dạng bài 1. So sánh với điện áp để tắt bật LED. Ví dụ so sánh giá điện áp tại chân AN0 nếu $\geq 2V$ thì bật LED11, ngược lại LED11 tắt. Giả sử LED11 nối RD7.

Chương trình được viết sau hiển thị 10 LED (PORTC và RD0, RD1)

```
if (adc>=409) PORTDbits.RD7 = 1; // >=2V ~409, LED11 Sang  
else PORTDbits.RD7 = 0; // LED11 tat
```

Dạng bài 2. So sánh với dải điện áp để tắt bật LED. Ví dụ so sánh giá điện áp tại chân AN0 nếu từ 1-2V thì bật LED11, ngoài dải thì LED11 tắt. Giả sử LED11 nối RD7.

Chương trình được viết sau hiển thị 10 LED (PORTC và RD0, RD1)

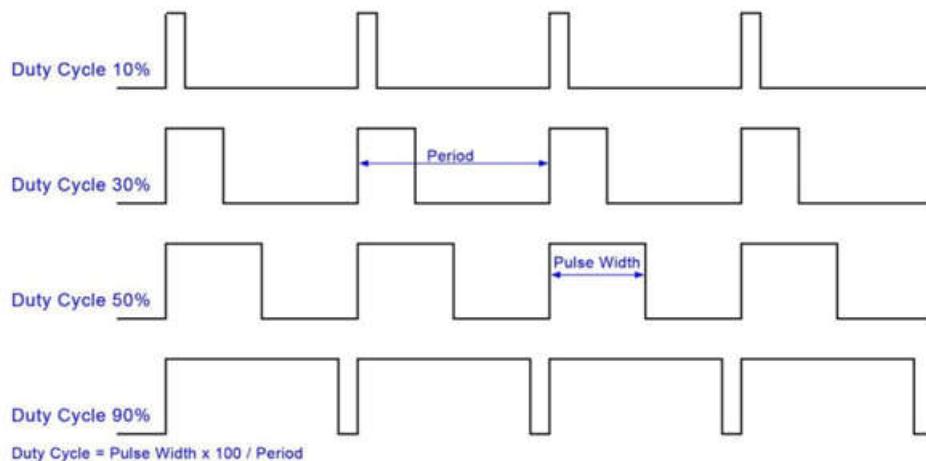
```
if ((adc>=204) & (adc<=409)) PORTDbits.RD7 = 1; // 1-2V 204-409, LED11 Sang  
else PORTDbits.RD7 = 0; // LED11 tat
```

Dạng bài 3. Lựa chọn nhiều kênh đầu vào, đặt giá trị ADCON1 sao cho các chân ANx cần dùng là đầu vào tương tự. Sử dụng nút nhấn RB7, nếu nhấn phím chọn AN0, nhả phím chọn AN1. Chương trình viết bô xung như sau:

```
while (1) {  
    if (PORTBbits.RB7==0) SetChanADC (ADC_CH0); //Nhan RB7 => AN0  
    else SetChanADC (ADC_CH1); //Nha RB7 => AN1  
    ConvertADC(); // Bien doi ADC
```

6 Lập trình PWM

Xung PWM được sử dụng rộng rãi trong thực tế, sử dụng phổ biến để điều chỉnh tốc độ động cơ, nhiệt, điện áp ... Xung PWM có chu kỳ (T) hoặc tần số F không đổi, độ rộng xung Ton – Pulse Width thay đổi.



Bộ tạo xung PWM trong PIC18F4520 gồm 2 bộ: CCP1 và CCP2.

- PWM1 tạo xung ra trên chân RC2/CCP1
- PWM2 tạo xung ra trên chân RC1/CCP2 hoặc RB3/CCP2

6.1 Các công thức PWM.

Tính giá trị **PR2** = period <256 đưa vào lệnh **OpenPWMMx(PR2)**;

Tính giá trị **dutycycle** đưa vào lệnh **SetDCPWMMx(dutycycle)**; tạo độ rộng xung

$$Ta\ có\ T = \frac{1}{F} = Ton + Toff; \ dutyCycle = \frac{Ton}{T} = \frac{Ton}{Ton + Toff} = \%dutyMax$$

Cách 1. Cách tính theo chu kỳ.

$$period = \frac{PWM_period}{4 \ Tosc \ K} \quad 1 = \frac{1000\mu s}{4 \ \frac{1}{8}\mu s \ K} \quad 1 = \frac{2000}{K} \quad 1$$

Giá trị lựa chọn giá trị chia tần của Timer2: $K = 1, 4, 16$ sao cho period <256, thay vào công thức trên => **prescaler = K = 16, PR2 = period = 124**

$$dutycycle = \frac{PWM_Duty_Cycle}{Tosc \ K} = \frac{PWM_Duty_Cycle}{\frac{1}{8}\mu s \ 16} = \frac{PWM_Duty_Cycle}{2}$$

Do **PWM_Duty_Cycle** có đơn vị **uS** nên bỏ qua đơn vị được hệ số như trên.

Theo yêu cầu đề bài, tính các dutycycle như sau :

- $Ton=20\%T = 20\%1000\mu s = 200 \Rightarrow dutycycle = \frac{PWM_Duty_Cycle}{2} = \frac{200}{2} = 100$
- $Ton=40\%T \Rightarrow dutycycle = 200 \quad Ton=80\%T \Rightarrow dutycycle = 400$

Cách 2. Cách tính theo tần số.

$$period = \frac{Fosc}{4 \ F pwm \ K} \quad 1 = \frac{8M}{4 \ 1K \ K} \quad 1 = \frac{2000}{K} \quad 1$$

Giá trị lựa chọn giá trị chia tần của Timer2: $K = 1, 4, 16$ sao cho period <256, thay vào công thức trên => **prescaler = K = 16, PR2 = period = 124**

$$DutyMax = \frac{Fosc}{F pwm \ prescaler} = \frac{8M}{1k \ 16} = 500 \text{ (tối đa là } 1023 = 10\text{bit)}$$

- $dutycycle = Ton = 20\%T = 20\% \ DutyMax = 20\% \ 500 = 100$
- $Ton=40\%T \Rightarrow dutycycle = 200, Ton=80\%T \Rightarrow dutycycle = 400$

Các hàm trong thư viện **pwm.h** xem trang 50 Tài liệu tra cứu PIC

Lưu ý: Để tạo xung trên chân RB3/CCP2A cần viết thêm câu lệnh:

#pragma config CCP2MX = PORTBE

6.2 Bài tập

Thiết kế mạch điện như sau:

- Các linh kiện được sử dụng: Các điện trở; 01 nút nhấn thường mở (PB).
- Chân RB3/CCP2A (hoặc chân CCP1 hoặc chân CCP2B) nối với một kênh của máy hiện sóng (oscilloscope); Nút nhấn PB được nối với chân RB0; PIC18F4520 hoạt động ở tần số 8Mhz.

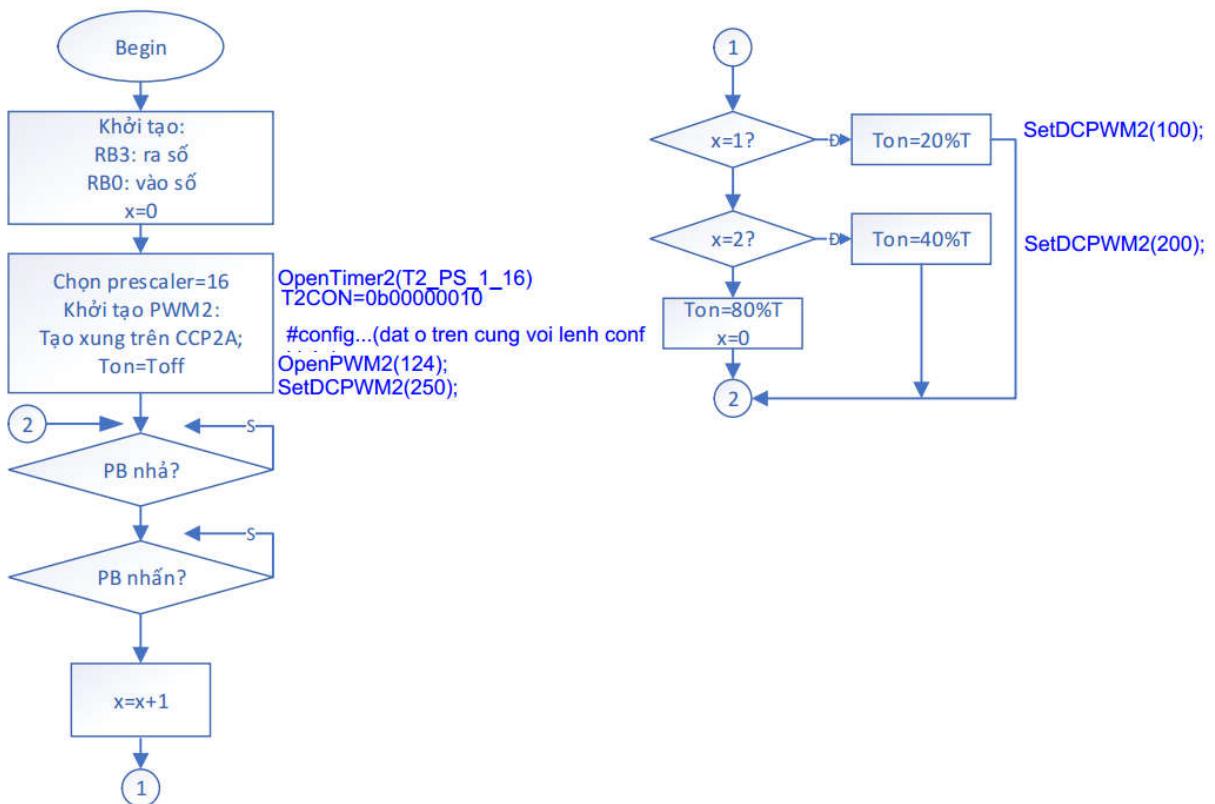
b. Vẽ lưu đồ thuật toán và viết chương trình theo yêu cầu như sau:

- Tạo xung trên chân CCP2A có chu kỳ $T=1000\mu s$; Trạng thái ban đầu $T_{on}=T_{off}$, với T_{on} và T_{off} lần lượt là độ rộng của nửa chu kỳ dương và âm.

- Đếm số lần nhấn PB và tạo xung PWM theo yêu cầu sau:

Số lần nhấn =1, $T_{on}=20\%T$; Số lần nhấn =2, $T_{on}=40\%T$; Số lần nhấn =3, $T_{on}=80\%T$; Số lần nhấn =4, $T_{on}=20\%T$;...

Thuật toán



Phản tính giá trị **PR2** và **dutyCycle** được tính ở trên.

Hệ số chia tần **K = 16, PR2 = 124**,

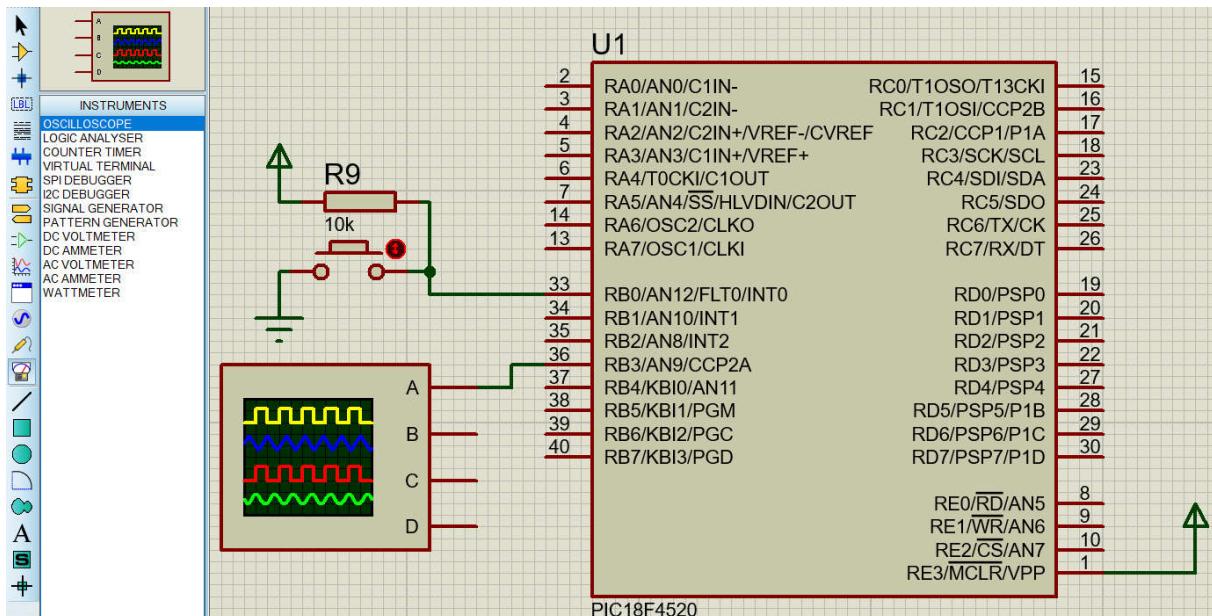
- **dutycycle = Ton = 20%T = 20% DutyMax = 20% 500 = 100**
- $T_{on}=40\%T \Rightarrow \text{dutycycle} = 200$
- $T_{on}=80\%T \Rightarrow \text{dutycycle} = 400$

Hệ số chia tần của Timer 2:

Nếu **K=1 => T2CONbits.T2CKPS0 = 0; T2CONbits.T2CKPS1 = 0;**

Nếu **K = 4 => T2CKPS0 = 1; Nếu K = 16 => T2CKPS1 = 1;**

Sơ đồ mạch



Chương trình

```
//khai báo các thư vi?n, ví d?:  
#include <P18f4520.h>  
#include <delays.h>  
#include <pwm.h>  
#include <timers.h>  
//C?u h?nh cho vi di?u khi?n, ví d?:  
#pragma config OSC = HS //ch? đ? dao đ?ng HS  
#pragma config MCLRE = ON //s? d?ng RE3 làm ch?n reset  
#pragma config WDT = OFF //kh?ng d?ng Watchdog timer  
#pragma config CCP2MX = PORTBE  
void main (void)  
{  
    TRISB=0b00000001; ADCON1=0x0F;TRISC=0b00000000;  
    OpenTimer2(T2_PS_1_16); //chon prescaler=16  
    // hoac T2CONbits.T2CKPS1 = 1;  
    OpenPWM2(124); //Tao xung co chu ky 1000uS (Fosc=8Mhz)  
    SetDCPWM2(250); //Thiet lap do rong cua Ton=50%T=500uS  
    while(1)  
    {  
        while(PORTBbits.RB0==0);  
        while(PORTBbits.RB0==1);  
        x++;  
        if(x==1) SetDCPWM2(100); //Ton=20%T  
        else if(x==2) SetDCPWM2(200); //Ton=40%T  
        else {SetDCPWM2(400);x=0;} //Ton=80%T  
    }  
}
```