ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HÀNH MÔN VI XỬ LÝ – VI ĐIỀU KHIỂN

BÀI THỰC HÀNH SỐ 4: GIAO TIẾP VỚI 7-SEGMENT LED VÀ TIMER

Sinh viên thực hiện:

Trần Ngọc Ánh 22520077

Giảng viên hướng dẫn: Phạm Minh Quân

Mã lớp: CE103.O22

TP. HỒ CHÍ MINH, 6 THÁNG 5 NĂM 2024

I. Trình bày và vẽ lưu đồ giải thuật quét LED ứng dụng để hiển thị 7-segment led.

1. Giải thuật quét LED:

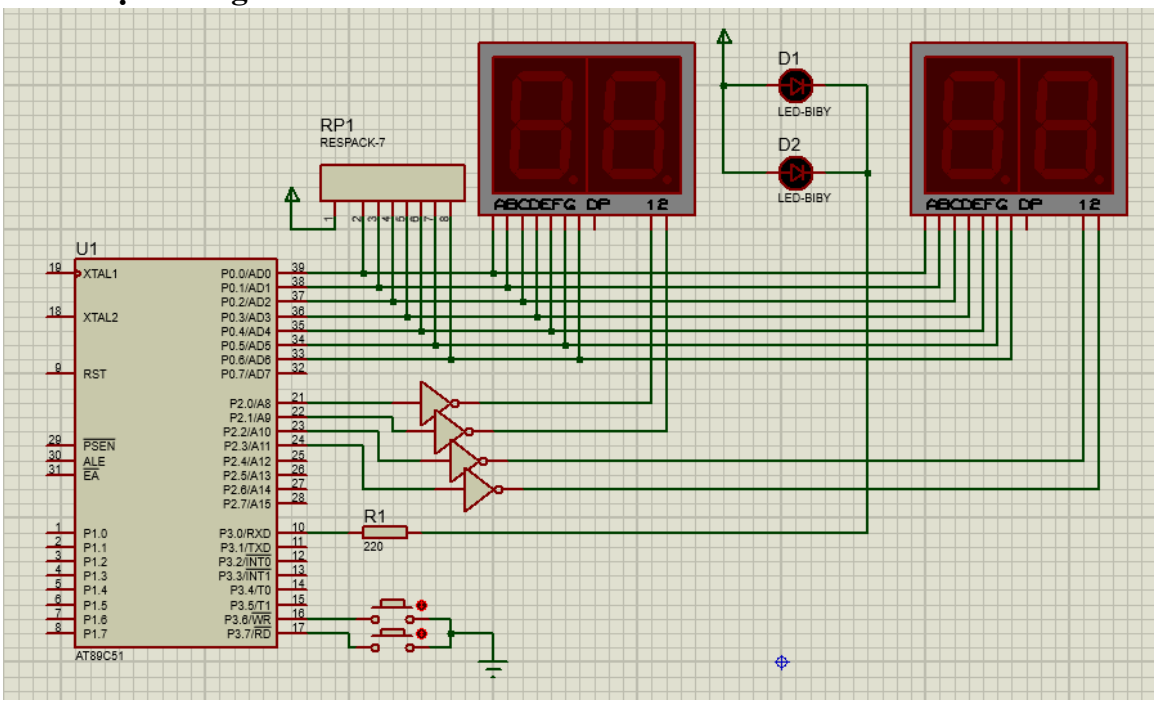
- Mục đích: Điều khiển lần lượt các LED trong một nhóm LED 7 đoạn để hiển thị các ký tự số hoặc chữ cái mong muốn.
- Nguyên tắc hoạt động:
  - Chia nhỏ nhóm LED 7 đoạn thành các cụm LED (thường 4 hoặc 8 LED).
  - Chọn 1 cụm LED để hiển thị.
  - Gán giá trị mã cho các chân điều khiển của cụm LED tương ứng với ký tự cần hiển thị.
  - Tăng giá trị chỉ mục để chọn sang cụm LED tiếp theo.
  - Lặp lại bước 3 và 4 cho đến khi hiển thị hết các ký tự mong muốn.

2. Lưu đồ giải thuật:

Khởi tạo các biến và cài đặt chân I/O của AT89C51
|
Vòng lặp vô hạn:
| Lặp qua các số từ 0 đến 9:
| | Hiển thị số lên màn hình 7 đoạn:
| | | Gửi tín hiệu cho các LED tương ứng dựa trên mẫu hiển thị của số hiện tại.
| | |
| | Ngắt khoảng thời gian ngắn để hiển thị số.
| |
Kết thúc vòng lặp và quay lại bước đầu tiên.

II. Sử dụng Timer của vi điều khiển 8051 thiết kế 1 mạch đồng hồ với format 24h với thời gian ban đầu được được cài đặt trong source code.

1. Schematic mạch đồng hồ với format 24h:



2. **Source code *(keil C):***

```
#include <REGX51.H>
#define hour P3_6
#define min P3_7
#define led1 P2_0
#define led2 P2_1
#define led3 P2_2
#define led4 P2_3
#define on 0
#define off 1

char so[] = {0xc0,, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};
void delay_ms(int ms)
{
	while(ms--)
	{
		TMOD = 0x01;
		TH0 = 0xfc;
		TL0 = 0x18;
		TR0 = 1;
		TF0 = 0;
		while(!TF0);
		TR0 = 0;
	}
}

void hienThi(unsigned char gio, usigned char phut)
{
	unsigned char chuc0, chuc1, donvi0, donvi1, i;
	chuc0 = gio / 10;
	donvi0 = gio % 10;
	chuc1 = phut / 10;
	donvi1 = phut % 10;
	for (i = 0; i < x; i++)
	{
		led1 = on;
		P0 = so[chuc0];
		delay_ms(10);
		led1 = off;
		led2 = on;
		P0 = so[donvi0];
```

```
                delay_ms(10);
                led2 = off;
                led3 = on;
                P0 = so[chuc1];
                delay_ms(10);
                led3 = off;
                led4 = on;
                P0 = so[donvi1];
                delay_ms(10);
                led4 = off;
        }
}

void main()
{
        unsigned char gio = 0, phut = 0, giay = 0;
        P3_0 = 0;
        while(1) {
                P3_0 = ~P3_0;
                giay++;
                hienThi(gio, phut);   // delay 1s
                if (giay == 59)
                        phut++;
                if (phut == 59)
                        gio++;
                if (hour == 0)
                        gio++;
                if (min == 0)
                        phut++;
                if (phut > 59)
                        phut = 0;
                if (gio > 23)
                        gio = 0;
        }
}
```

3. **Video kết quả mô phỏng:** https://drive.google.com/drive/folders/1lbAfrqscZS7J0JuOBITmnL_NGR-RbEq6?usp=sharing

III. Vẫn với thiết kế đồng hồ trên, sử dụng vòng lặp để tạo delay thay vì Timer. Nêu ưu nhược điểm của 2 cách.

1. **Source code và nguyên lí hoạt động:**

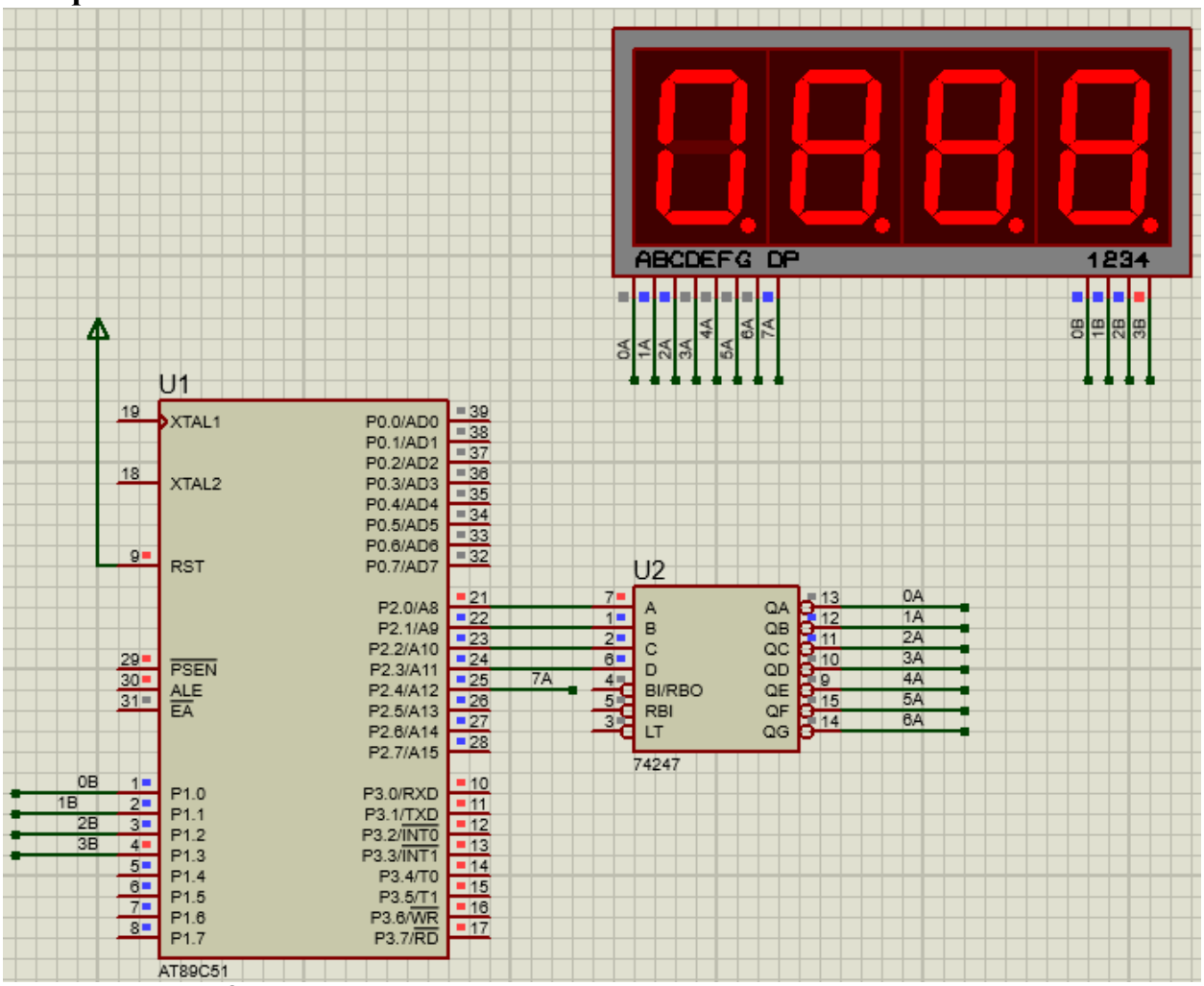
Source code	Nguyên lí hoạt động

`org 0000h`
`jmp Start`
 	
`milisec equ r0`
`sec equ r1`
 	
`DELAY:`
`MOV TMOD, #01h`
`MOV TH0, #high(-2500)`
`MOV TL0, #low(-2500)`
`SETB TR0`
`JNB TF0, $`
`CLR TR0`
`CLR TF0`
`RET`
`org 0100h`
 	
`Start:`
`MOV IE, #10000101B`
`MOV TMOD, #01H`
`MOV sec, #0`
`MOV milisec, #0`
 	
`main:`
`CALL display`
`INC milisec`
 	
`CJNE milisec, #100, main`
`MOV milisec, #0`
 	
`INC sec`
`CJNE sec, #100, main`
 	
`MOV sec, #0`
`JMP main`
 	
`loop:`
	1. Khởi tạo:
	▪ Code khởi tạo Timer 0 ở chế độ 16 bit và đặt giá trị khởi tạo cho TL0 và TH0 là 0xFFFF.
	▪ Code cũng cài đặt ngắt Timer 0.
	
	2. Vòng lặp chính:
	▪ Vòng lặp chính lặp đi lặp lại liên tục.
	▪ Trong mỗi vòng lặp, code gọi hàm display để hiển thị thời gian hiện tại lên màn hình LED 7 đoạn.
	▪ Sau đó, code tăng giá trị mili giây và kiểm tra xem liệu nó có đạt đến 100 hay không. Nếu có, code đặt lại giá trị mili giây về 0 và tăng giá trị giây.
	▪ Tiếp theo, code kiểm tra xem giá trị giây có đạt đến 100 hay không. Nếu có, code đặt lại giá trị giây về 0 và tăng giá trị giờ.
	▪ Cuối cùng, code nhảy lại đầu vòng lặp chính.
	
	3. Hiển thị thời gian:
	▪ Hàm display chuyển đổi giá trị mili giây, giây và giờ thành mã hiển thị cho màn hình LED 7 đoạn.
	▪ Hàm sử dụng Timer 0 để tạo độ trễ giữa các chữ số hiển thị.
	
	4. Cài đặt thời gian ban đầu: Code cài đặt thời gian ban đầu bằng cách gán các giá trị cho biến sec và milisec trong hàm Start.

```
CALL display
JB P3.2, loop

JNB P3.2, $

JB P3.2, main

RETI

display:
MOV A, milisec
MOV B, #10
DIV AB
MOV 010H, A
MOV 011H, B

MOV P1, #00000000B

SETB P1.3
MOV P2, 011H
ACALL DELAY
CLR P1.3

SETB P1.2
MOV P2, 010H
ACALL DELAY
CLR P1.2

MOV A, sec
MOV B, #10
DIV AB
MOV 012H, A
MOV 014H, B

MOV P1, #00000000B

SETB P1.1
MOV P2, 014H
ACALL DELAY
CLR P1.1
```

```
SETB P1.0
MOV P2, 012H
ACALL DELAY
CLR P1.0
RET

END
```

2. **Kết quả:**



3. **Ưu và nhược điểm của hai cách:**

a. Sử dụng vòng lặp để tạo delay:

- Ưu điểm:
  - Dễ Triển Khai: Việc sử dụng vòng lặp để tạo delay là cách tiếp cận đơn giản và dễ hiểu, đặc biệt đối với người mới bắt đầu với vi điều khiển.
  - Không Yêu Cầu Kiến Thức Sâu Về Ngắt (Interrupt): Không cần phải xử lý ngắt hoặc kiến thức phức tạp về ngắt, giảm độ phức tạp của mã nguồn.

- Nhược điểm:
  - Không Chính Xác: Sử dụng vòng lặp delay có thể dẫn đến sai số trong việc đo thời gian, đặc biệt là khi có các tác động bên ngoài như các ngắt khác hoặc các tác động từ các phần khác của chương trình.
  - Không Linh Hoạt: Khó điều chỉnh thời gian delay mà không làm thay đổi các phần khác của chương trình.

b. Sử dụng Timer để tạo delay:

- Ưu điểm:
  - Chính xác hơn: Sử dụng timer cho phép chính xác hóa thời gian, đảm bảo rằng đồng hồ hoạt động đúng với thời gian cần thiết.
  - Linh hoạt và điều chỉnh được: Có thể dễ dàng điều chỉnh thời gian sử dụng timer mà không làm ảnh hưởng đến các phần khác của chương trình.
- Nhược điểm:
  - Yêu cầu kiến thức phức tạp hơn: Sử dụng timer yêu cầu kiến thức về xử lý ngắt và việc lập trình có thể phức tạp hơn so với sử dụng vòng lặp delay.
  - Tài nguyên hơn: Sử dụng timer có thể tiêu tốn nhiều tài nguyên hơn của vi điều khiển so với việc sử dụng vòng lặp delay.