PHẦN IV

Mục đích chung của phần này nhằm giúp người học vận dụng ngôn ngữ lập trình cấp cao C để lập trình cho vi điều khiển. Các ví dụ nhằm mục đích để người học có thể tiếp cận được cấu trúc phần cứng bên trong của dạng máy tính này. Phần này sẽ đề cập nhiều hơn về việc đa dạng hóa các thiết bị ngoại vi có thể kết nối với máy tính trên chip họ 8051 cũng như các vấn đề minh họa việc giao tiếp mở rộng bộ nhớ. Ngoài ra, việc truyền thông trao đổi dữ liệu giữa máy tính đa dụng và máy tính trên chip họ 8051 cũng được trình bày trong phần này.

Từ các ví dụ cơ bản được cung cấp trong phần này, người học có thể hiểu sâu cách sử dụng của từng mã lệnh, cấu trúc và sự ảnh hưởng của các thanh ghi, cũng như phương pháp lập trình điều khiển khi thực hiện các tác vụ từ máy tính trên chip 8051.

Sơ lược ngôn ngữ lập trình C dành cho vi điều khiển

Chức năng của trình biên dịch là nhằm để chuyển mã nguồn cấp cao C thành mã máy HEX cái mà phần cứng máy tính có thể hiểu và thực hiện lệnh như mong muốn thể hiện của người lập trình. Mã máy có định dạng HEX sẽ được tải xuống ROM của vi điều khiển. Kích thước của mã máy là rất quan trọng bởi nó bị giới hạn về tài nguyên phần cứng của vi điều khiển. Thông thường thì không gian bộ nhớ để lưu trữ mã khoảng 64K bytes.

Một số lý do để sử dụng C trong việc lập trình vi điều khiển đó là

- Việc lập trình sử dụng mã cấp cao C sẽ tiết kiệm thời gian hơn so với lập trình bằng ngôn ngữ cấp thấp như ASM. Tuy nhiên, sử dụng ngôn ngữ C sẽ dẫn đến mã HEX lớn hơn.
- Có thể sử dụng khái niệm hàm và các thư viện hàm
- Mã nguồn C có tính di động cao đối với nhiều kiến trúc vi điều khiển khác nhau. Như vậy, cùng một mã nguồn C có thể chạy trên một số vi điều khiển khác nhau mà không cần phải chỉnh sửa mã nguồn ban đầu.Việc hiểu rõ kiểu dữ liệu trong ngôn ngữ C sẽ giúp người lập trình có thể tối ưu mã nguồn và kích thước file HEX. Một số kiểu dữ liệu như là

Ký tự không dấu	Unsigned char
Ký tự có dấu	Signed char
Số nguyên không dấu	Unsigned int
số nguyên có dấu	Signed int
kiểu bit đơn	Sbit (single bit)
Kiểu bit và thanh ghi có chức năng đặc biệt	Bit and SFR

-

Kiểu ký tự không dấu (UNSIGNED CHAR)

Vì vi điều khiển họ 8051 được cấu tạo từ một bộ xử lý 8-bit, việc sử dụng một kiểu char không dấu là cần thiết và phù hợp. Với kiểu dữ liệu này, tầm giá trị trong khoảng 0-255 (tức 00-FF). Với kiểu dữ liệu này, có thể được sử dụng nhằm lưu các giá trị bộ đếm và các ký tự trong bảng mã ASCII. Trong trình biên dịch C, nó sẽ mặc định sử dụng kiểu dữ liệu có dấu signed nếu chúng ta không khai báo từ khóa unsigned

Ví dụ minh họa

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C để viết một chương trình xuất 00-FF tại cổng P1 của vi điều khiển	
`#include <reg51.h>` `void main(void)` `{` `unsigned char z;` `for (z=0;z<=255;z++)` `P1=z;` `}`	Trong ví dụ này, chúng ta nên lưu ý vì port P1 của vi điều khiển là 8-bit vì thế hãy sử dụng kiểu dữ liệu unsigned char thay vì kiểu int

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để gửi các giá trị HEX của các ký tự ASCII như 0, 1, 2, 3, 4, 5, A, B, C… ra cổng P1	
`#include <reg51.h>` `void main(void)` `{unsigned char` `mynum[]="012345ABCD";` `unsigned char z;` `for (z=0;z<=10;z++)` `P1=mynum[z];` `}`	Lưu ý các ký tự ASCII đều có thể hiện chỉ với 8-bit

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit trên cổng P1 liên tục	
`//Toggle P1 forever` `#include <reg51.h>` `void main(void)` `{` `for (;;)` `{` `p1=0x55;` 01010101 `p1=0xAA;` 10101010 `}` `}`	Giải thích chương trình này, tại sao lại đặt giá trị 0x55 và 0xAA vào P1 để thực hiện toggle

Kiểu ký tự có dấu (SIGNED CHAR)

Cũng như kiểu ký tự không dấu, kiểu ký tự có dấu sử dụng 8-bit để lưu giá trị. Tuy nhiên, bit có trọng số lớn MSB được sử dụng để lưu miền giá trị + hoặc –. Như vậy, với kiểu dữ liệu có dấu này, tầm giá trị mà nó có thể lưu được là -128 đến 127. Chỉ sử dụng kiểu dữ liệu có dấu nếu muốn thể hiện giá trị số <0. Ví dụ như giá trị NHIỆT ĐỘ

Ví dụ minh họa

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để gửi giá trị số -4 đến +4 ra cổng P1	
`//Singed numbers` `#include <reg51.h>` `void main(void)` `{` `char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};` `unsigned char z;` `for (z=0;z<=8;z++)` `P1=mynum[z];` `}`	Quan sát ngõ ra của cổng P1 trên vi điều khiển

Kiểu số nguyên không dấu và có dấu (UNSIGNED/SIGNED INT)

Đối với kiểu dữ liệu này, 16-bit được sử dụng để thể hiện dữ liệu. Đối với số nguyên không dấu (unsigned int) tầm giá trị vào khoảng 0 đến 65535 (tức 0000 đến FFFF). Việc sử dụng kiểu dữ liệu này nhằm để:

- định nghĩa các biến 16-bit như là địa chỉ của bộ nhớ
- lưu trữ giá trị của bộ đếm nếu khi đếm tầm giá trị vượt quá 256
- đa phần vì thanh ghi và việc truy cập bộ nhớ trong khoảng 8-bit thế nên việc sử dụng các biến không đúng kích thước sẽ dẫn đến việc tạo HEX có kích thước lớn.

Cũng giống như kiểu ký tự không dấu, kiểu số nguyên có dấu sẽ sử dụng MSB để lưu trữ - hoặc +. Như vậy chỉ còn 15 bit để lưu trư độ lớn của giá trị. Tầm giá trị sẽ từ –32768 to +32767

Kiểu bit đơn (SBIT)

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle bit D0 của P1 (P1.0) 50.000 lần	
`#include <reg51.h>` `sbit MYBIT=P1^0;` `void main(void)` `{` `unsigned int z;` `for (z=0;z<=50000;z++)` `{` `MYBIT=0;` `MYBIT=1;` `}` `}`	từ khóa sbit cho phép truy cập đến các bit đơn của các thanh ghi SFR

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle chỉ 1 bit D4 của P2 các bit khác trên P2 không bị ảnh hưởng	
`//Toggling an individual bit`	Lưu ý:

`` #include <reg51.h> sbit mybit=P2^4; void main(void) { while (1) { mybit=1; //turn on P2.4 mybit=0; //turn off P2.4 } } ```	từ port P0 đến P3 của vi điều khiển họ 8051 đều là các cổng 8-bit có thể truy xuất theo bit. Sbit là kiểu dữ liệu để truy cập bit đơn Sử dụng định dạng Px^y trong đó • X là port 0-3 • Y là 0-7 tương ứng 8-bit

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle chỉ 1 bit D5 của P1 50000 lần	
`` sbit MYBIT=0x95; void main(void) { unsigned int z; for (z=0;z<50000;z++) { MYBIT=1; MYBIT=0; } } ```	Chúng ta có thể truy cập bit đơn của bất kỳ SFR nào nếu chỉ rõ địa chỉ bit Không cần thiết sử dụng #include <reg51.h>. điều này cho phép chúng ta truy cập đến bất cứ byte nào của bộ nhớ truy xuất ngẫu nhiên SFR tại 80-FF

TẠO TRỄ

Có 2 cách để tạo trễ thời gian

- sử dụng một vòng lặp đơn giản
- sử dụng bộ định thời timer

Có 3 yếu tố ảnh hưởng đến độ chính xác của trễ là

1. Thiết kế của máy tính 8051
   - số lượng chu kỳ máy
   - số lượng chu kỳ xung nhịp cho một chu kỳ máy
2. Tần số của thạch anh được kết nối đến chân ngõ vào x1 và x2
3. Lựa chọn trình biên dịch

Trình biên dịch sẽ chuyển các câu lệnh c và hàm thành các lệnh hợp ngữ. Như vậy, trình biên dịch khác nhau sẽ tạo nên những mã nguồn khác nhau.

Ví dụ minh họa

<table>
<tr><td colspan="2">sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit của P1 liên tục sử dụng khái niệm thời gian trì hoãn</td></tr>
<tr><td><pre>
#include <reg51.h>
void main(void)
{
unsigned int x;
for (;;) //repeat forever
{
p1=0x55;
for (x=0;x<40000;x++);
//delay size
//unknown
p1=0xAA;
for (x=0;x<40000;x++);
}
}
</pre></td><td>sử dụng máy dao động ký để quan sát thời giant hay đổi trạng thái tại các chân của cổng P1</td></tr>
</table>

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit của P1 liên tục sử dụng thời gian trì hoãn là 250m giây	
`#include <reg51.h>` `void MSDelay(unsigned int);` `void main(void)` `{` `while (1) //repeat forever` `{` `p1=0x55;` `MSDelay(250);` `p1=0xAA;` `MSDelay(250);` `}` `}` `void MSDelay(unsigned int itime)` `{` `unsigned int i,j;` `for (i=0;i<itime;i++)` `for (j=0;j<1275;j++);` `}`	sử dụng máy dao động ký để quan sát thời giant hay đổi trạng thái tại các chân của cổng P1 nhằm so sánh với thời gian dự kiến là 250ms

sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit của P0, 1 và 2 liên tục sử dụng thời gian trì hoãn là 250m giây. Sử dụng từ khóa sfr để khai báo địa chỉ cổng	
`//Accessing Ports as SFRs using sfr data type` `sfr P0=0x80;` `sfr P1=0x90;` `sfr P2=0xA0;` `void MSDelay(unsigned int);` `void main(void)` `{` `while (1)` `{` `P0=0x55;`	một cách khác để truy cập đến bộ nhớ SFR tại 80-FF là sử dụng kiểu dữ liệu sfr

```
P1=0x55;
P2=0x55;
MSDelay(250);
P0=0xAA;
P1=0xAA;
P2=0xAA;
MSDelay(250);
}
}
```

BÀI THỰC HÀNH SỐ 1

Mục đích:

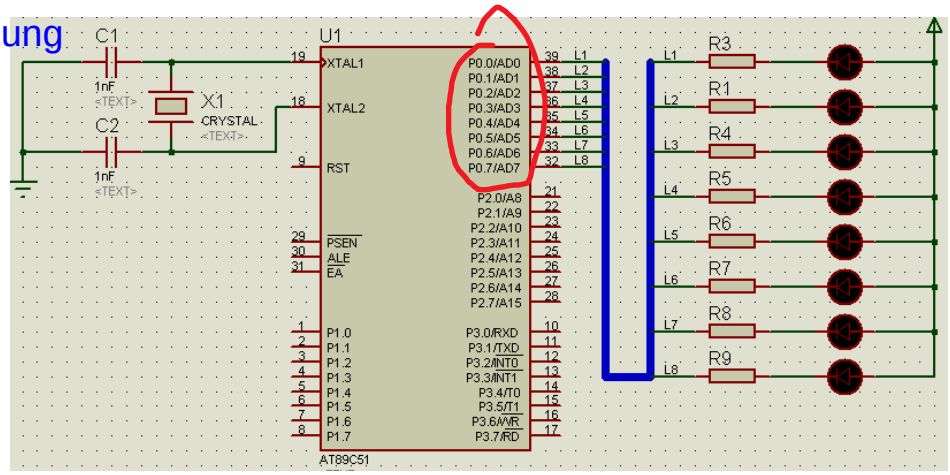
- Giới thiệu về giao tiếp vi điều khiển và ngoại vi LED đơn ở các dạng khác nhau
- Giới thiệu một số hiệu ứng đơn giản có thể thực hiện được với LED đơn

Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu về giao tiếp vi điều khiển và ngoại vi LED đơn
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để giao tiếp với LED đơn

Vấn đề 1 Thiết kế sơ đồ nguyên lý giao tiếp LED đơn theo phương pháp tích cực LED mức thấp như hình vẽ sau. Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau

chua hoan toan dung



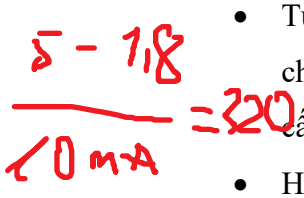
Hình 1: Sơ đồ nguyên lý kết nối VĐK và LED đơn tích cực mức thấp

Sử dụng phần mềm Keil C viết chương trình điều khiển 8 LED lần lượt sáng nhấp nháy, so le nhau

```
#include <at89x51.h>
void delay(int interval){
      int i,j;
      for(i=0;i<255;i++){

      for(j=0;j<interval;j++);
      }
}
void main(){
```

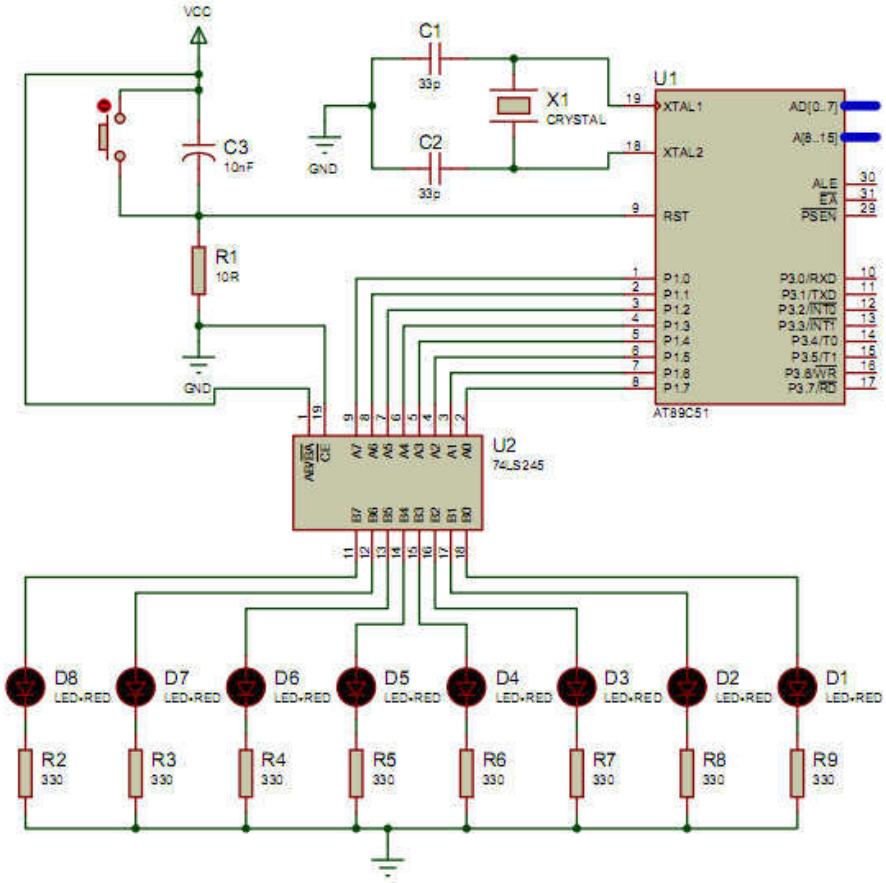
```
while(1){
            P0=0x55;

      delay(100);
            P0=0xAA;

      delay(100);
      }
}
```

Yêu cầu thực hành:



- Từ lý thuyết mạch điện và điện tử cơ bản, hãy phân tích và lựa chọn giá trị cho điện trở kết nối LED ở hình trên nếu điện áp cung cấp V_{DD} là 5V.
- Hãy giải thích chức năng của hàm delay trong mã nguồn trễ. Hãy giải thích phương pháp tạp độ trễ trong hàm delay(). Dùng bộ dao động trong công cụ mô phỏng Proteus để khảo sát thời gian trễ.

Vấn đề 2 Thiết kế giao tiếp LED đơn theo phương pháp tích cực LED mức cao như hình vẽ sau. Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau



Hình 2: Sơ đồ nguyên lý kết nối VĐK và LED đơn tích cực mức cao

Sử dụng phần mềm Keil C viết chương trình điều khiển 8 LED lần lượt sáng từ trái sang phải

```
#include <at89x51.h>
#include <stdio.h>

#define LED0 P1_0
#define LED1 P1_1
```

```
case 5:
LED4=sang;
LED1=LED2=LED3=LED0=LED5=LE
D6=LED7=tat;
break;
```

```
#define LED2 P1_2
#define LED3 P1_3
#define LED4 P1_4
#define LED5 P1_5
#define LED6 P1_6
#define LED7 P1_7

#define sang 1
#define tat 0
//---------------
void delay(unsigned int
ms)
{
unsigned int i,j;
for (i=0;i<ms;i++)
for (j=0;j<120;j++)
{}
}
//--------------
void
display_LED(unsigned
char number)
{

switch (number)
{

case 1:
LED0=sang;
LED1=LED2=LED3=LED4=LED5
=LED6=LED7=tat;
break;
case 2:
LED1=sang;
LED0=LED2=LED3=LED4=LED5
=LED6=LED7=tat;
break;
case 3:
LED2=sang;
case 6:
LED5=sang;
LED1=LED2=LED3=LED4=LED0=LE
D6=LED7=tat;
break;
case 7:
LED6=sang;
LED1=LED2=LED3=LED4=LED5=LE
D0=LED7=tat;
break;
case 8:
LED7=sang;
LED1=LED2=LED3=LED4=LED5=LE
D6=LED0=tat;
break;
}
}
main ()
{
unsigned char m;
while(1)
{
for (m=0;m<9;m++)
{
display_LED(m);delay(500);}
}
}
```

LED1=LED0=LED3=LED4=LED5 *=LED6=LED7=tat;* *break;* *case 4:* *LED3=sang;* *LED1=LED2=LED0=LED4=LED5* *=LED6=LED7=tat;* *break;*	

Yêu cầu thực hành:

1. Hãy so sánh phương pháp kết nối và giao tiếp điều khiển LED ở vấn đề 1 và 2. Hãy nêu ưu và nhược điểm của 2 phương pháp này.
2. Viết chương trình tạo hiệu ứng "sáng đuổi" (các LED sáng lần lượt từ LED 1 tới LED 8)
3. Viết chương trình tạo hiệu ứng cho các LED sáng từ hai đầu (từ LED 1 tới LED 4, từ LED 8 về LED 5) sau đó quay đầu (từ LED 4 về LED 1 và LED 5 về LED 8), quá trình lặp đi lặp lại liên tục

1. vd1: gọn hơn nhưng ko thao tác điều khiển dc
vd2: cho phép thao tác vs led, đệm dữ liệu 2 chiều nhưng rườm rà

http://robocon.vn/detail/ic25-ic-giai-ma-74ls245.html

vấn đề 2: tốn kém chi phí, kích thước mạch in tăng lên nhưng bảo vệ dc ngoại vi bên ngoài

vấn đề 1: **ko lấy nguồn cung cấp từ khối điều khiển nhưng**
ko có khối mở rộng