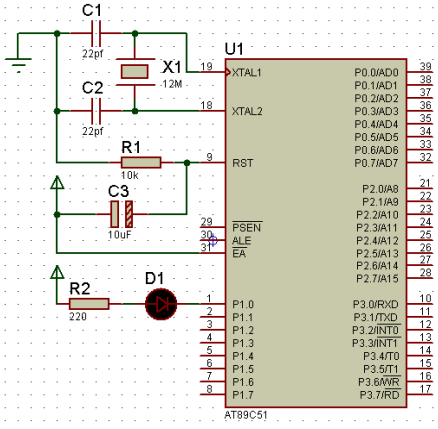
# 单片机原理与接口技术

华侨大学信息科学与工程学院 by: ylc



1

```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED=P1^0;

void DelayMS(uint x)
{
    uchar i;
    while(x--)
    {
        for(i=0;i<120;i++);
    }
}
void main()
{
    while(1)
    {
        LED=~LED;
        DelayMS(150);
    }
}
```






2

## SFR的定义

```
#ifndef __REG51_H_
#define __REG51_H_

/*  BYTE Register
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;
```

```
/*  BIT Register  */
/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;

/*  TCON  */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit ET0  = 0xA9;
sbit EX0  = 0xA8;
```

```
/*  IP   */
sbit PS   = 0xBC;
sbit PT1  = 0xBB;
sbit PX1  = 0xBA;
sbit PT0  = 0xB9;
sbit PX0  = 0xB8;

/*  P3  */
sbit RD   = 0xB7;
sbit WR   = 0xB6;
sbit T1   = 0xB5;
sbit T0   = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD  = 0xB1;
sbit RXD  = 0xB0;

/*  SCON  */
sbit SM0  = 0x9F;
sbit SM1  = 0x9E;
sbit SM2  = 0x9D;
sbit REN  = 0x9C;
sbit TB8  = 0x9B;
sbit RB8  = 0x9A;
sbit TI   = 0x99;
sbit RI   = 0x98;

#endif
```

《单片机原理与接口技术》 c51入门

3

## sfr

The **sfr** type defines a special function register (SFR). It is used as follows:

```
sfr name = address;
```

**Where**

*name* is the name of the SFR.

*address* is the address of the SFR.

SFRs are declared in the same fashion as other C variables. The only difference is that the type specified is **sfr** rather than **char** or **int**. For example:

```
sfr P0 = 0x80;    /* Port-0, address 80h */
sfr P1 = 0x90;    /* Port-1, address 90h */
sfr P2 = 0xA0;    /* Port-2, address 0A0h */
sfr P3 = 0xB0;    /* Port-3, address 0B0h */
```

**P0**, **P1**, **P2**, and **P3** are the SFR *name* declarations. Names for sfr variables are defined just like other C variable declarations. Any symbolic name may be used in an sfr declaration.

The *address* specification after the equal sign ('=') must be a numeric constant. Expressions with operators are not allowed. Classic 8051 devices support the SFR address range 0x80-0xFF. The NXP 80C51MX provides an additional extended SFR space with the address range 0x180-0x1FF.

**Note**

- **sfr** variables may not be declared inside a function. They must be declared outside of the function body.

4

**sbit**

The **sbit** type defines a bit within a special function register (SFR). It is used in one of the following ways:

```
sbit name = sfr-name ^ bit-position;
sbit name = sfr-address ^ bit-position;
sbit name = sbit-address;
```

**Where**

| | |
|---|---|
| *name* | is the name of the SFR bit. |
| *sfr-name* | is the name of a previously-defined SFR. |
| *bit-position* | is the position of the bit within the SFR. |
| *sfr-address* | is the address of an SFR. |
| *sbit-address* | is the address of the SFR bit. |

With typical 8051 applications, it is often necessary to access individual bits within an SFR. The **sbit** type provides access to bit-addressable SFRs and other bit-addressable objects. For example:

```
sbit EA = 0xAF;
```

This declaration defines **EA** as the SFR bit at address **0xAF**. On the 8051, this is the *enable all* bit in the interrupt enable register.

**Note**

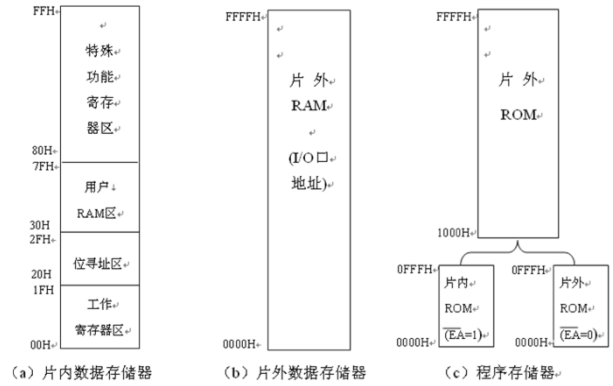
- Storage of objects accessed using **sbit** is assumed to be little endian (LSB first). This is the storage format of the **sfr16** type but it is opposite to the storage of **int** and **long** data types. Care must be taken when using **sbit** to access bits within standard data types.



【存储种类】 数据类型 【存储类型】 变量名

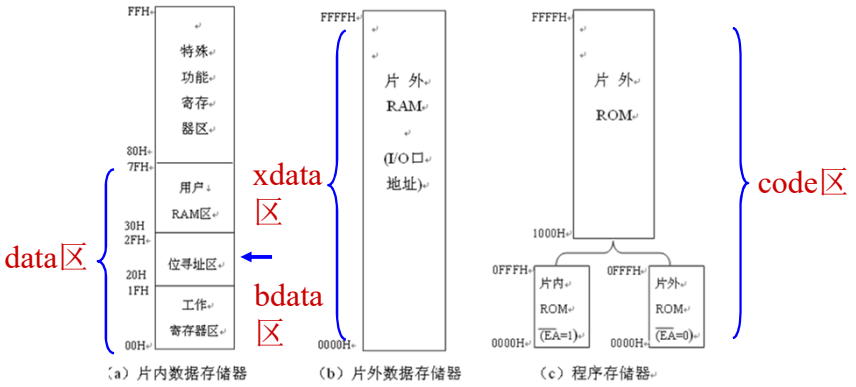
51系列单片机有三个逻辑存储空间：

片内数据存储器，片外数据存储器和程序存储器。




（a）片内数据存储器 （b）片外数据存储器 （c）程序存储器

## 建立C51存储类型与存储空间的对应关系






7

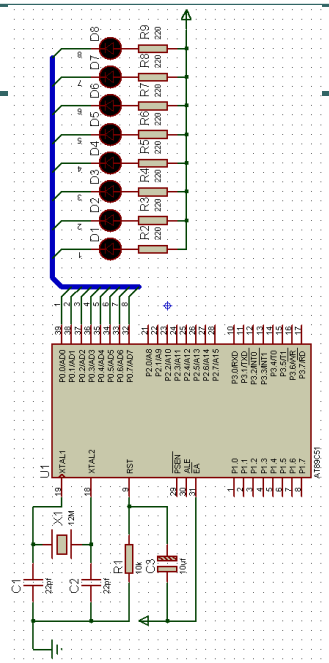
## C51的存储类型与存储空间对应关系表

| 存储类型 | 存储空间位置 | 字节地址 | 说明 |
|---|---|---|---|
| data | 片内低 128B 存储区 | 0H～7FH | 访问速度快，可作为常用变量或临时性变量存储器 |
| bdata | 片内可位寻址存储区 | 20H～2FH | 允许位与字节混合访问 |
| idata | 片内高 128B 存储区 | 80H～FFH | 只有 52 系列才有 |
| pdata | 片外页 RAM | 00H～FFH | 常用于外部设备访问 |
| xdata | 片外 64KB RAM | 0000H ～ FFFFH | 常用于存放不常用的变量或等待处理的数据 |
| code | 程序 ROM | 0000H ～ FFFFH | 常用于存放数据表格等固定信息 |



8

```c
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int

void DelayMS(uint x)
{
    uchar i;
    while(x--)
    {
        for(i=0;i<120;i++);
    }
}

void main()
{
    P0=0xfe;
    while(1)
    {
        P0=_crol_(P0,1);
        DelayMS(150);
    }
}
```



《单片机原理与接口技术》 c51入门

9

```c
#ifndef __INTRINS_H__
#define __INTRINS_H__

#pragma SAVE

#if defined (__CX2__)
#pragma FUNCTIONS(STATIC)
/* intrinsic functions are reentrant, but need static attribute */
#endif

extern void          _nop_      (void);
extern bit           _testbit_  (bit);
extern unsigned char _cror_     (unsigned char, unsigned char);
extern unsigned int  _iror_     (unsigned int,  unsigned char);
extern unsigned long _lror_     (unsigned long, unsigned char);
extern unsigned char _crol_     (unsigned char, unsigned char);
extern unsigned int  _irol_     (unsigned int,  unsigned char);
extern unsigned long _lrol_     (unsigned long, unsigned char);
extern unsigned char _chkfloat_(float);
#if defined (__CX2__)
extern int           abs        (int);
#endif
#if !defined (__CX2__)
extern void          _push_     (unsigned char _sfr);
extern void          _pop_      (unsigned char _sfr);
#endif

#pragma RESTORE

#endif
```

10

## 特殊库函数讲解

**_crol_**

**Summary**

```
#include <intrins.h>

unsigned char _crol_ (
  unsigned char c,        /* character to rotate left */
  unsigned char b);       /* bit positions to rotate */
```

**Description** The **_crol_** routine rotates the bit pattern for the character *c* left *b* bits. This routine is implemented as an intrinsic function.

**Return Value** The **_crol_** routine returns the rotated value of *c*.

**See Also** _cror_, _irol_, _iror_, _lrol_, _lror_

**Example**

```
#include <intrins.h>

void test_crol (void) {
  char a;
  char b;

  a = 0xA5;
  b = _crol_(a,3); /* b now is 0x2D */
}
```





## 特殊库函数讲解

**_irol_**

**Summary**

```
#include <intrins.h>

unsigned int _irol_ (
  unsigned int i,         /* integer to rotate left */
  unsigned char b);       /* bit positions to rotate */
```

**Description** The **_irol_** routine rotates the bit pattern for the integer *i* left *b* bits. This routine is implemented as an intrinsic function.

**Return Value** The **_irol_** routine returns the rotated value of *i*.

**See Also** _crol_, _cror_, _iror_, _lrol_, _lror_

**Example**

```
#include <intrins.h>

void test_irol (void) {
  int a;
  int b;

  a = 0xA5A5;
  b = _irol_(a,3);      /* b now is 0x2D2D */
}
```

## _testbit_

**Summary**

```
#include <intrins.h>

bit _testbit_ (
  bit b);       /* bit to test and clear */
```

**Description** The **_testbit_** routine produces a **JBC** instruction in the generated program code to simultaneously test the bit *b* and clear it to 0. This routine may be used only on directly-addressable bit variables and is invalid on any type of expression. This routine is implemented as an intrinsic function.

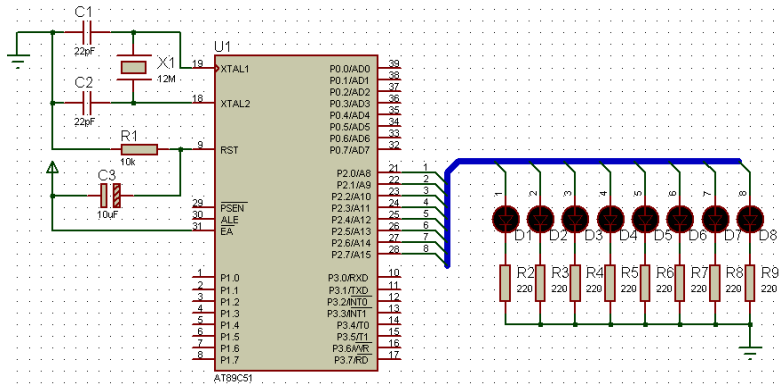
**Return Value** The **_testbit_** routine returns the value of *b*.

**Example**

```
#include <intrins.h>
#include <stdio.h> /* for printf */

void tst_testbit (void){
  bit test_flag;

  if (_testbit_ (test_flag))
    printf ("Bit was set\n");
  else
    printf ("Bit was clear\n");
}
```



## 流水灯

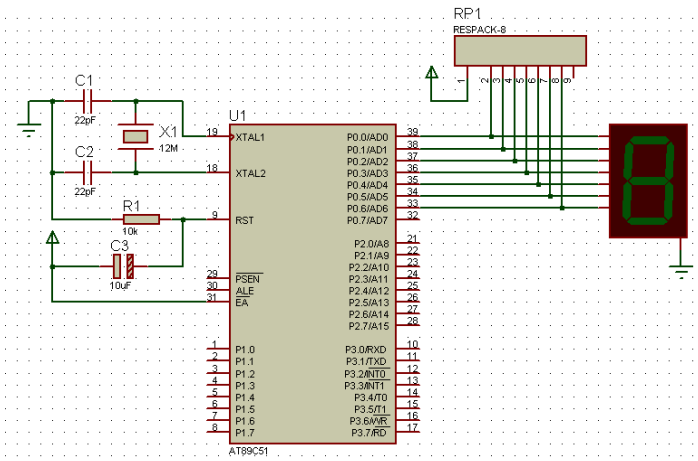


## 流水灯

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int

void DelayMS(uint x)
{
    uchar i;
    while(x--)
    {
        for(i=0;i<120;i++);
    }
}

void main()
{
    uchar i;
    P2=0x01;??
    while(1)
    {
        for(i=0;i<7;i++)
        {
            P2=_crol_(P2,1);
            DelayMS(150);
        }
        for(i=0;i<7;i++)
        {
            P2=_cror_(P2,1);
            DelayMS(150);
        }
    }
}
```



## 单个数码管显示控制



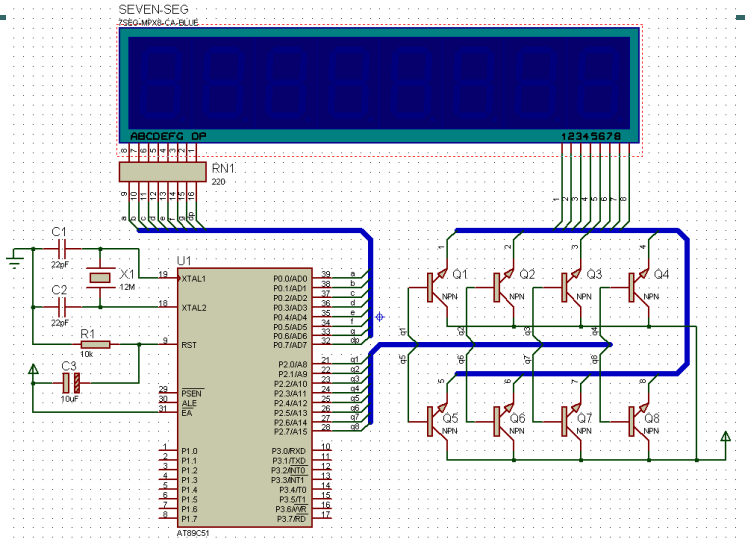
## 单个数码管显示控制

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff
void DelayMS(uint x)
{
    uchar t;
    while(x--){
        for(t=0;t<120;t++);
    }
}
void main()
{
    uchar i=0;
    P0=0x00;
    while(1)
    {
      for(i=0;i<10;i++){
        P0=~DSY_CODE[i];
        DelayMS(300);
      }
    }
}
```



17

## 数码管动态扫描

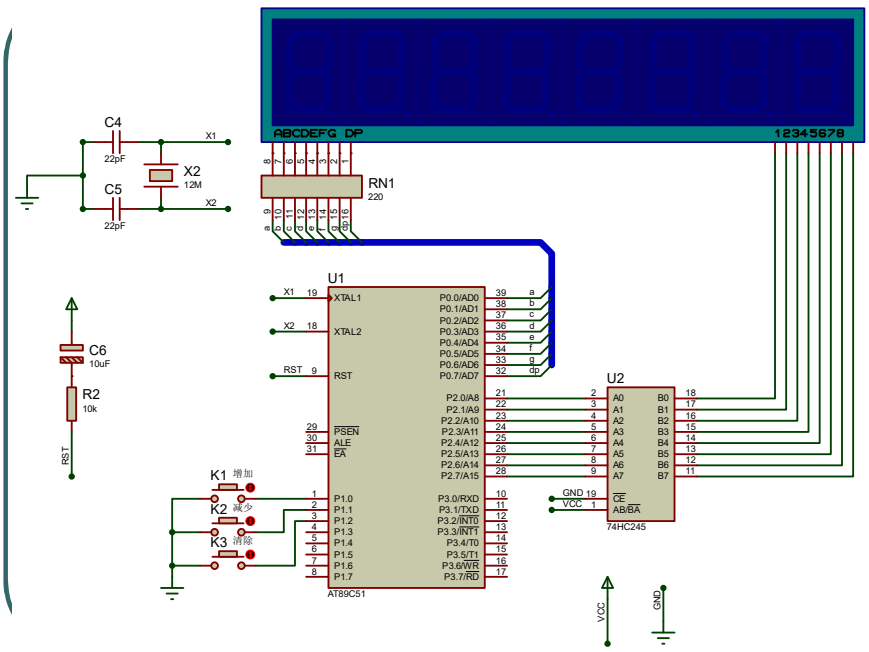





18

```c
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
void DelayMS(uint x)
{
    uchar t;
    while(x--) for(t=0;t<120;t++);
}
void main()
{
    uchar i,wei=0x80;
    while(1)
    {
        for(i=0;i<8;i++)
        {
            P2=0xff;
            P0=DSY_CODE[i];  //发送段码
            wei=_crol_(wei,1);
            P2=wei;          //发送位码
            DelayMS(2);
        }
    }
}
```





## 按键控制数码管显示内容

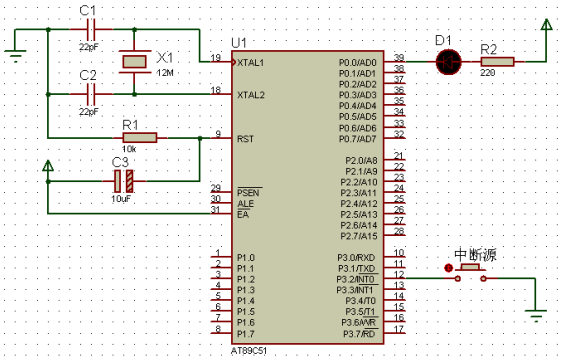
```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int

uchar code DispCode[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
uchar LEDBuf[]={0,0,0};
uchar KeyCode,KeyCounts=0;

void DelayMs(uint x)
{
    uchar i;
    while(x--){
        for(i=0;i<120;i++)
            ;
    }
}
```

```
void LEDShow()
{
    uchar i,j=0x01;
    LEDBuf[2]=KeyCounts/100;
    LEDBuf[1]=KeyCounts/10%10;
    LEDBuf[0]=KeyCounts%10;
    for(i=0;i<3;i++)
    {
        j=_cror_(j,1);
        P0=0xff;
        P0=DispCode[LEDBuf[i]];
        P2=j;
        DelayMs(2);
    }
}
```



```
void main()
{
    uchar i;
    //P0=0xff;
    P0=DispCode[0];
    P2=0xff;
    while(1)
    {
        LEDShow();
        KeyCode=P1;
        if(KeyCode!=0xff){
            DelayMs(10);
            KeyCode = P1;
            if(KeyCode != 0xff){
                while(1){
                    if(P1 == 0xff)
                        break;
                }
                switch(KeyCode)
                {
                    case 0xfe:  if(KeyCounts<255) KeyCounts++;
                                break;
                    case 0xfd:  if(KeyCounts>0) KeyCounts--;
                                break;
                    case 0xfb:  KeyCounts=0;
                }
            }
        }
    }
}
```

## 基于定时器的流水灯






23

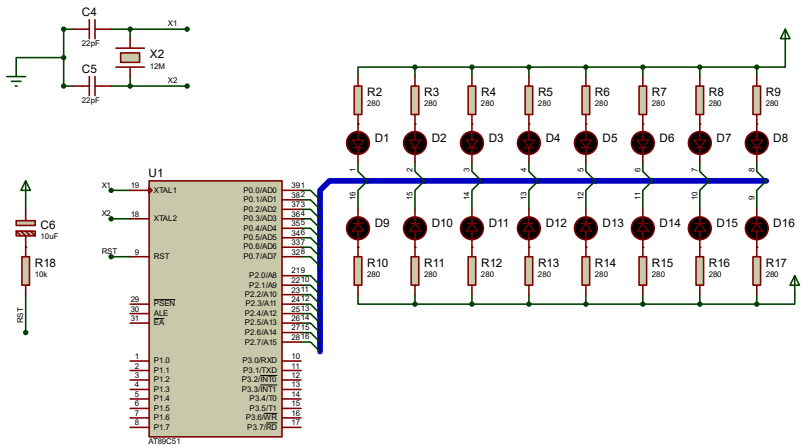
## 定时器控制单个LED灯闪烁

```
#include<reg51.h>
sbit LED=P0^0;

void main()
{
    LED=1;
    EA=1;
    EX0=1;
    IT0=1;
    while(1);
}

void EX_INT0() interrupt 0
{
    LED=~LED;
}
```



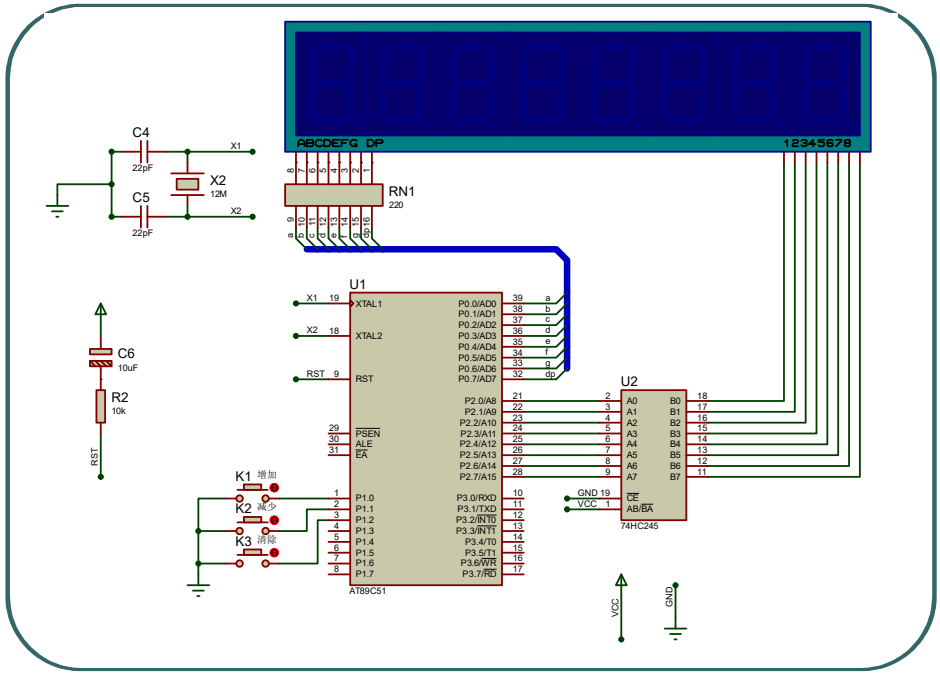
24

## 基于定时器的流水灯






```c
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
void main()
{
    uchar T_Count=0;
    P0=0xfe;
    P2=0xfe;
    TMOD=0x01;
    TH0=(65536-40000)/256;
    TL0=(65536-40000)%256;
    TR0=1;
    while(1)
    {
        if(TF0==1)
        {
            TF0=0;
            TH0=(65536-40000)/256;
            TL0=(65536-40000)%256;
            if(++T_Count==5)
            {
                P0=_crol_(P0,1);
                P2=_crol_(P2,1);
                T_Count=0;
            }
        }
    }
}
```




《单片机原理与接口技术》 c51入门

27

## 基于定时器的数码管动态扫描

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int

uchar code DispCode[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
uchar LEDBuf[8];
uchar count = 0;
uchar KeyCode =0;
uchar KeyCounts = 0;
uchar index = 0x80;

void DelayMs(uint x)
{
    uchar i;
    while(x--){
        for(i=0;i<120;i++)
            ;
    }
}
```

《单片机原理与接口技术》 c51入门

28

## 基于定时器的数码管动态扫描

```
void Time0Init(void)
{
    TMOD = 0x01;
    TH0= (65536-5000)/255;
    TL0 = (65536-50000)%255;
    ET0= 1;
    EA = 1;
    TR0 = 1;
}
```

```
void LEDShow() interrupt 1
{
    TH0= (65536-5000)/255;
    TL0 = (65536-5000)%255;
    LEDBuf[2]=KeyCounts/100;
    LEDBuf[1]=KeyCounts/10%10;
    LEDBuf[0]=KeyCounts%10;
    P2 = 0x00;
    P2 = index;
    P0 = DispCode[ LEDBuf[count] ];
    index = _cror_(index,1);
    count++;
    if(count==8){
        count = 0;
        index = 0x80;
    }
}
```



```
void main()
{
    uchar i;
    //P0=0xff;
    Time0Init();
    P0=DispCode[0];
    P2=0xff;
    while(1)
    {
        KeyCode=P1;
        if(KeyCode!=0xff){
            DelayMs(10);
            KeyCode = P1;
            if(KeyCode != 0xff){
                while(1){
                    if(P1 == 0xff)
                        break;
                }
                switch(KeyCode)
                {
                    case 0xfe:  if(KeyCounts<255) KeyCounts++;
                                break;
                    case 0xfd:  if(KeyCounts>0) KeyCounts--;
                                break;
                    case 0xfb:  KeyCounts=0;
                }
            }
        }
    }
}
```

```c
#include <reg52.h>

void ConfigUART(unsigned int baud);

void main()
{
    ConfigUART(9600);  //配置波特率为9600

    while (1)
    {
        while (!RI);      //等待接收完成
        RI = 0;           //清零接收中断标志位
        SBUF = SBUF + 1; //接收到的数据+1后，发送回去
        while (!TI);      //等待发送完成
        TI = 0;           //清零发送中断标志位
    }
}
/* 串口配置函数，baud-通信波特率 */
void ConfigUART(unsigned int baud)
{
    SCON  = 0x50;  //配置串口为模式1
    TMOD &= 0x0F;  //清零T1的控制位
    TMOD |= 0x20;  //配置T1为模式2
    TH1 = 256 - (11059200/12/32)/baud;  //计算T1重载值
    TL1 = TH1;     //初值等于重载值
    ET1 = 0;       //禁止T1中断
    TR1 = 1;       //启动T1
}
```



31

```c
#include <reg52.h>

void ConfigUART(unsigned int baud);
void main()
{
    EA = 1;   //使能总中断
    ConfigUART(9600);  //配置波特率为9600
    while (1);
}
/* 串口配置函数，baud-通信波特率 */
void ConfigUART(unsigned int baud)
{
    SCON  = 0x50;  //配置串口为模式1
    TMOD &= 0x0F;  //清零T1的控制位
    TMOD |= 0x20;  //配置T1为模式2
    TH1 = 256 - (11059200/12/32)/baud;
    TL1 = TH1;     //初值等于重载值
    ET1 = 0;       //禁止T1中断
    ES  = 1;       //使能串口中断
    TR1 = 1;       //启动T1
}
```

```c
/* UART中断服务函数 */
void InterruptUART() interrupt 4
{
    unsigned char temp;
    if (RI)  //接收到字节
    {
        RI = 0;   //手动清零接收
        temp = SBUF;
        SBUF = temp + 1;
    }
    if (TI)  //字节发送完毕
    {
        TI = 0;   //手动清零发送
    }
}
```



32

```c
#include <reg52.h>
sbit ADDR3 = P1^3;
sbit ENLED = P1^4;

unsigned char code LedChar[] = {  //数码管显示字符转换表
    0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,
    0x80, 0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E
};

unsigned char LedBuff[7] = {
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
};
unsigned char T0RH = 0;
unsigned char T0RL = 0;
unsigned char RxdByte = 0;

void ConfigTimer0(unsigned int ms);
void ConfigUART(unsigned int baud);

void main()
{
    EA = 1;         //使能总中断
    ENLED = 0;
    ADDR3 = 1;
    ConfigTimer0(1);   //配置T0定时1ms
    ConfigUART(9600);  //配置波特率为9600

    while (1)
    {   //将接收字节在数码管上以十六进制形式显示出来
        LedBuff[0] = LedChar[RxdByte & 0x0F];
        LedBuff[1] = LedChar[RxdByte >> 4];
    }
}
```



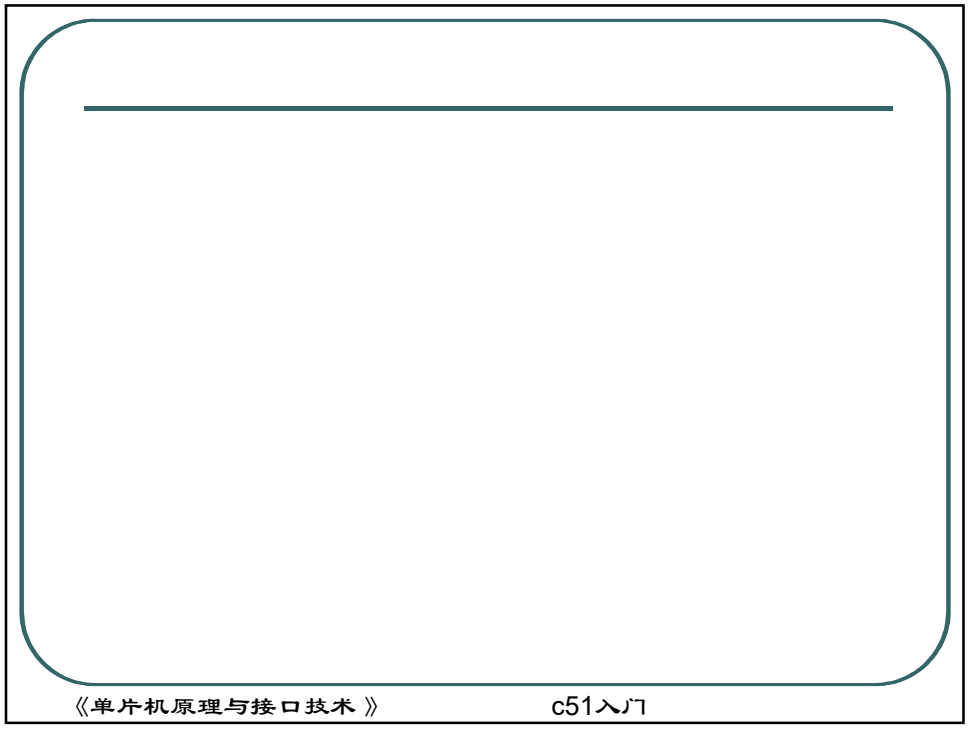
33

```c
void ConfigTimer0(unsigned int ms)
{
    unsigned long tmp;  //临时变量

    tmp = 11059200 / 12;       //定时器
    tmp = (tmp * ms) / 1000;   //计算所
    tmp = 65536 - tmp;         //计算定
    tmp = tmp + 13;            //补偿中
    T0RH = (unsigned char)(tmp>>8);  /
    T0RL = (unsigned char)tmp;
    TMOD &= 0xF0;   //清零T0的控制位
    TMOD |= 0x01;   //配置T0为模式1
    TH0 = T0RH;     //加载T0重载值
    TL0 = T0RL;
    ET0 = 1;        //使能T0中断
    TR0 = 1;        //启动T0
}
/* 串口配置函数，baud-通信波特率 */
void ConfigUART(unsigned int baud)
{
    SCON  = 0x50;  //配置串口为模式1
    TMOD &= 0x0F;  //清零T1的控制位
    TMOD |= 0x20;  //配置T1为模式2
    TH1 = 256 - (11059200/12/32)/baud;
    TL1 = TH1;     //初值等于重载值
    ET1 = 0;       //禁止T1中断
    ES  = 1;       //使能串口中断
    TR1 = 1;       //启动T1
}
```

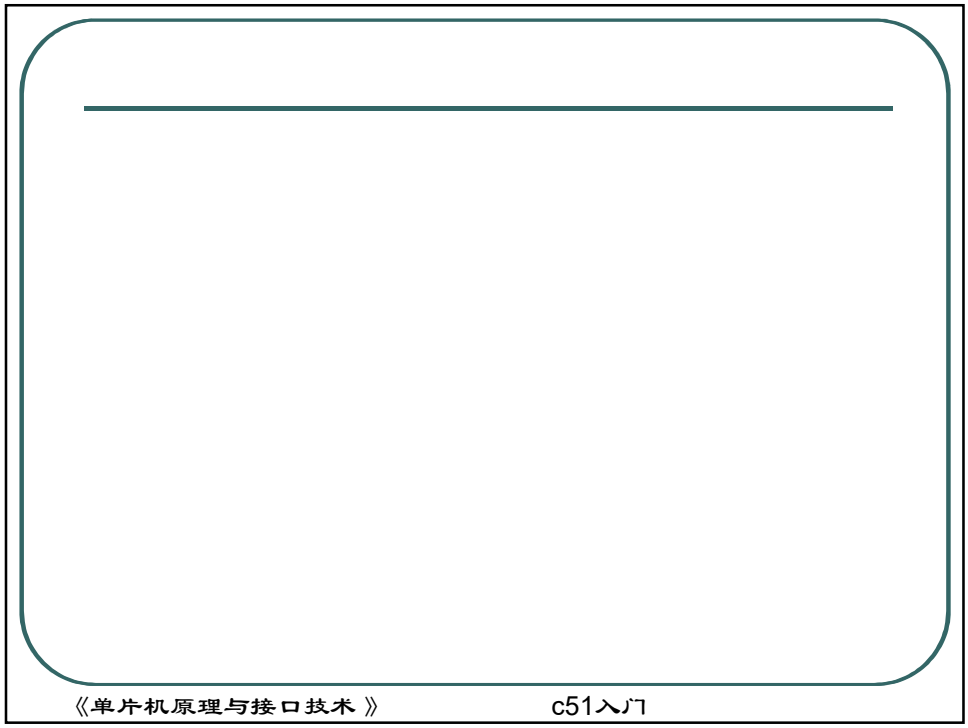
```c
void LedScan()
{
    static unsigned char i = 0;  //动

    P0 = 0xFF;              //关闭所有
    P1 = (P1 & 0xF8) | i;   //位选索引
    P0 = LedBuff[i];        //缓冲区中
    if (i < 6)              //索引递增
        i++;
    else
        i = 0;
}
/* T0中断服务函数，完成LED扫描 */
void InterruptTimer0() interrupt 1
{
    TH0 = T0RH;  //重新加载重载值
    TL0 = T0RL;
    LedScan();   //LED扫描显示
}
/* UART中断服务函数 */
void InterruptUART() interrupt 4
{
    if (RI)  //接收到字节
    {
        RI = 0;  //手动清零接收中断标
        RxdByte = SBUF;
        SBUF = RxdByte;
    }
    if (TI) { TI = 0;}
}
```



34




35




36