

**电子线路设计实验报告**

——单片机播放《bad apple》动画（带延时功能）



班级姓名：自动化71班任泽华

学号：2171411498

指导老师：张良祖 刘美兰

目录

[一、 硬件原理图 2](#_Toc24060037)

[(一) 数码管 2](#_Toc24060038)

[(二) Lcd显示屏 3](#_Toc24060039)

[(三) 键盘 5](#_Toc24060040)

[(四) SD卡模拟器 5](#_Toc24060041)

[(五) 时钟电路 6](#_Toc24060042)

[(六) 单片机AT89C52 6](#_Toc24060043)

[二、 软件实现流程 7](#_Toc24060044)

[(一) 主函数部分main.c 7](#_Toc24060045)

[(二) SD卡读取数据部分SD.c 10](#_Toc24060046)

[(三) 头文件SD.h 10](#_Toc24060047)

[三、 实验总结 11](#_Toc24060048)

[四、 做的不太成功的另一个作品 13](#_Toc24060049)

[五、 附录 15](#_Toc24060050)

[(一) 原理图 15](#_Toc24060051)

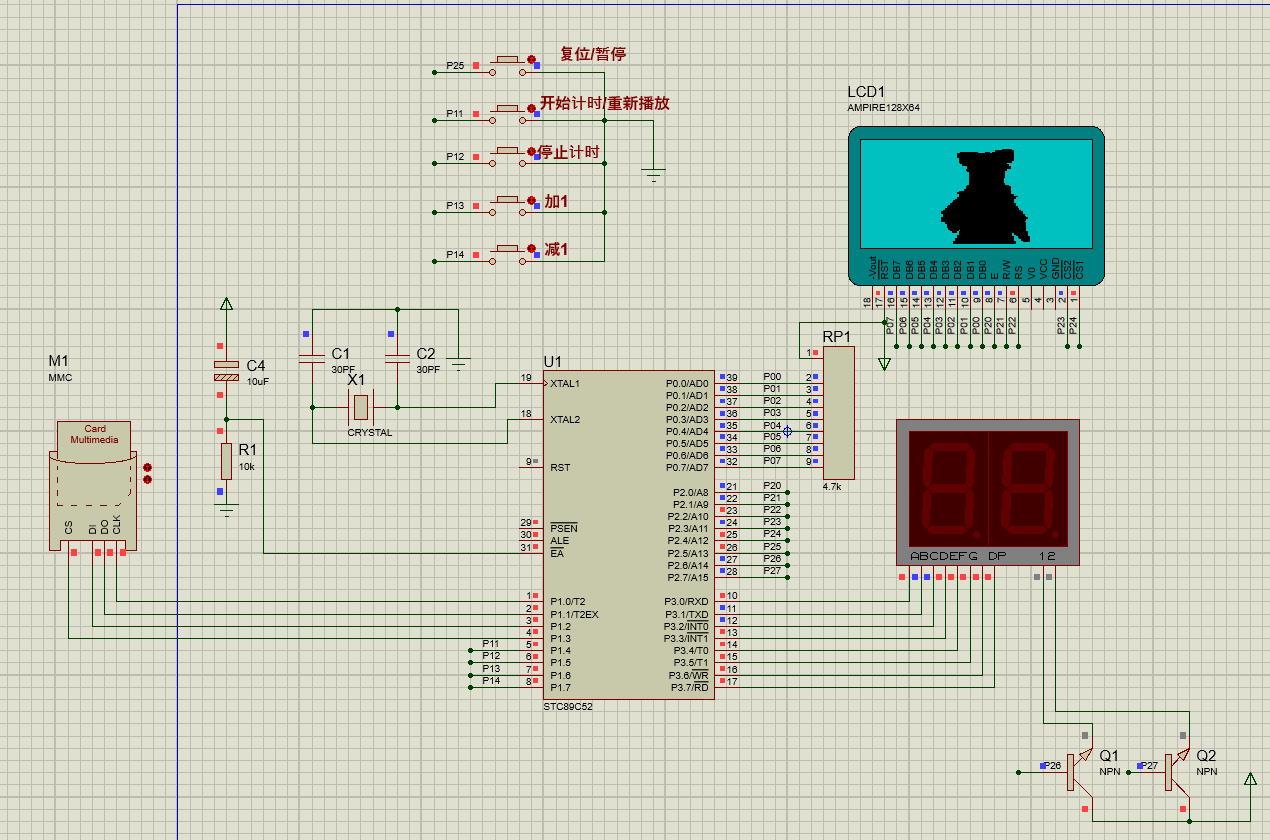
[(二) 程序代码 15](#_Toc24060052)

[1) Main函数 15](#_Toc24060053)

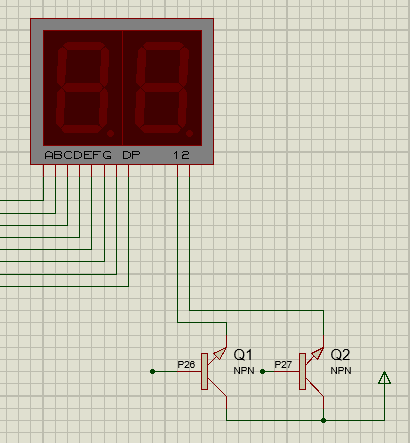
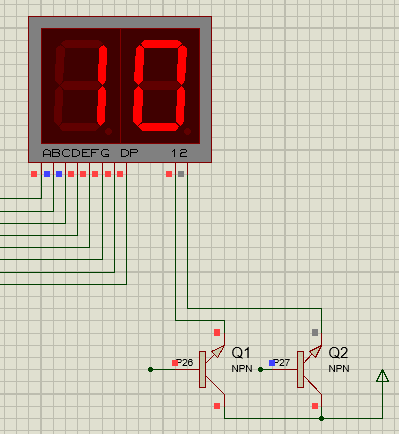
[2) 导入存储卡文件的SD.c 22](#_Toc24060054)

[3) 头文件SD.h 28](#_Toc24060055)

# 硬件原理图

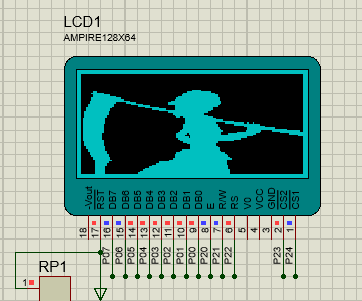
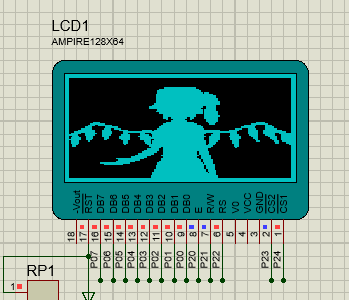
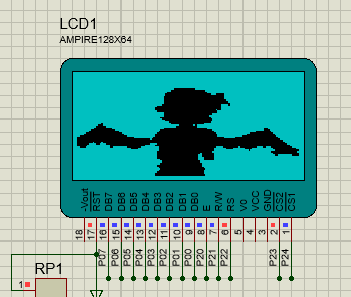
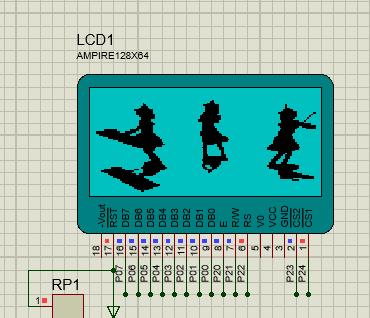


## 数码管

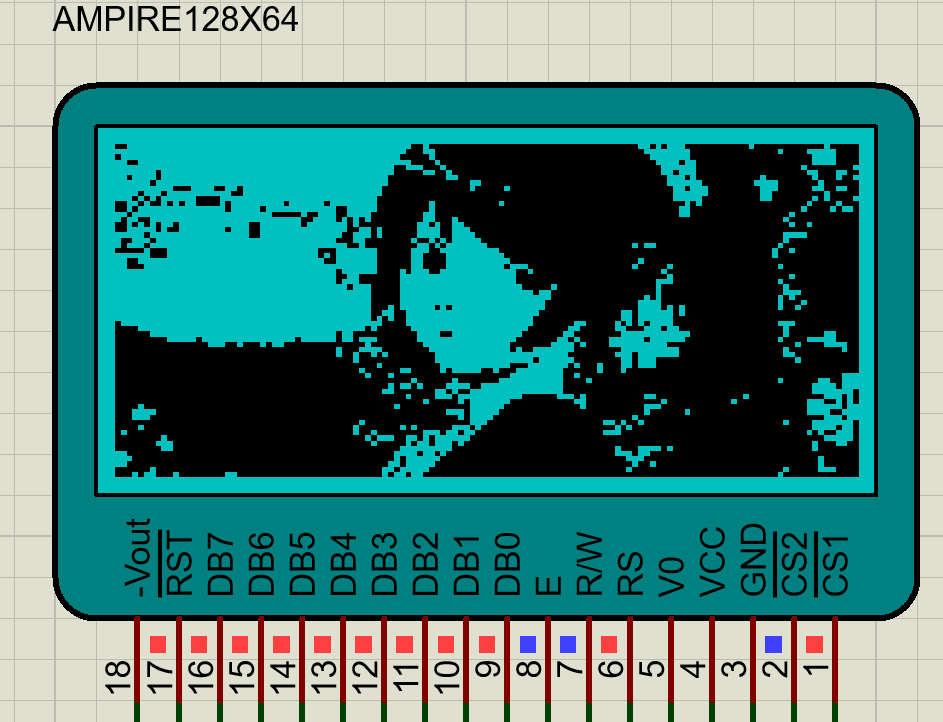
我采用的是7SEG-MPX2-CA型两位数字显示的数码管，可以显示0~99一共100个数字，由ABCDEFG DP八个输入控制，其电位分别由单片机的P3.0~P3.7接出，并行受到单片机的控制。1，2端分别接Q1,Q2两个晶体管和电源作为驱动。两个晶体管的控制信号分别由单片机的P2.6、P2.7口接出。在与老师的交流中我了解到，虽然仿真时不接驱动也可以实现，但是在实际操作中，外加驱动是必不可少的，单片机只能提供控制信号，而数码管的能量要靠驱动获得。

## Lcd显示屏

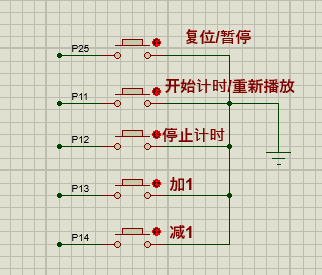
 

Lcd 显示屏由128x64个像素组成，型号为AMPIRE128x64。Lcd显示屏的8，7，6，2，1口分别接单片机的P2.0~P2.4口；9~16口分别接单片机的P1.0~P1.7口；17口接电源。由单片机产生并行信号把图片显示在屏幕上，经过设定显示的内容、延时等等，就可以实现动画效果。

在老师的例程中，我探究如何把图片转换为像素文件，在Image2Lcd软件中，将比例为128x64比例的图片导入，但是最开始，我的图片显示出现了问题，呈现一种多个图片重复叠加的效果，在尝试之后发现应当把扫描方式改为数据水平，字节垂直；下面勾选“字节内象素数据反向”。最后呈现的结果如下图所示：

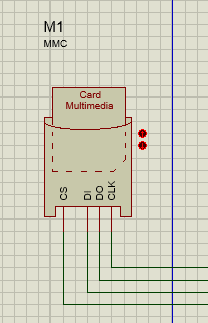


## 键盘



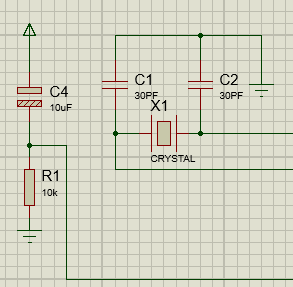
键盘用五个可复位按键开关组成，分别用于实现倒计时复位/视频暂停、倒计时开始/视频重新播放、停止计时、计时加一、计时减一的功能。其中第一个按键接单片机的P2.5口，第二~第五分别接单片机的P1.4~P1.7口，用于输入控制信号。

## SD卡模拟器

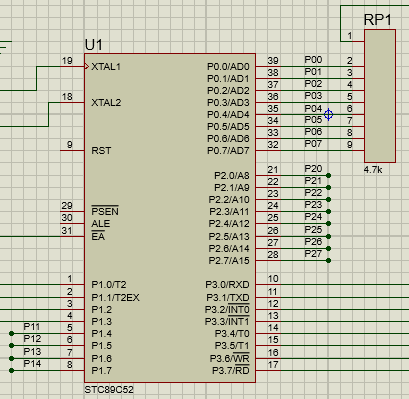
这是一个SD卡模拟器MMC，可以模拟外接的存储卡。其中的内容由SD.mmc提供。由于视频数据比较大而单片机的空间有限，只需要在播放时从外部读取每一帧的数据即可，该元件就是实现了这个功能。其CLK、DO、DI、CS口分别接单片机的P1.0~P1.3口，用于输入数据。

## 时钟电路



单片机必备的计时电路，由晶振、电容等等元件组成，生成时钟信号，分别接单片机的XTAL1、XTAL2、EA口。

## 单片机AT89C52



电路中最重要的器件，本实验中采用AT89C52单片机，其中引脚与外部的连接在前面已经介绍，在本例中的主要作用是依据键盘输入的信号，控制数码管和lcd显示屏输出内容。包括开始计时、暂停计时、计时数增减、暂停播放视频、从头播放视频等等。其中MMC、键盘、时钟是输入信号，lcd、数码管是输出设备。

# 软件实现流程

## 主函数部分main.c

开始

初始化定时器

初始化计数变量

While(1)（）

是

数码管显示、等待输入

否

暂停

否

重播

是

置零

否

是

播放动画

计时结束



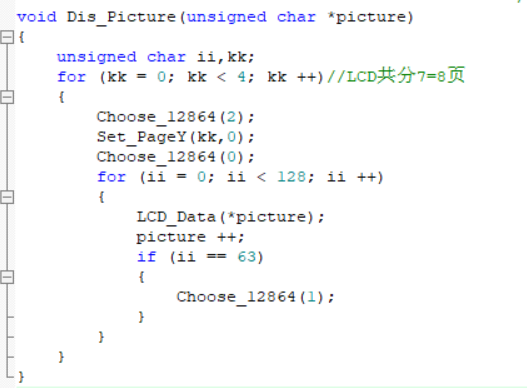
主要利用了计时当中key()函数中全局变量jishu、bz的变化，在播放动画时，若键盘有输入，jishu的值也会变化，此时通过条件语句判断，如果jishu不为0，跳出循环（暂停）。若bz为1，播放动画的帧序号变量SD\_ADDR置零，实现重复播放。

#### 播放动画部分

主要函数为dispicture、dispictureb、SdReadBlock

作用分别是：显示LCD上半部分、显示LCD下半部分、读取SD卡文件。

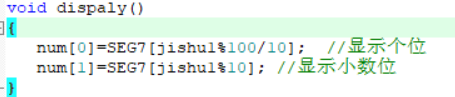
其中SdReadBlock函数在文件SD.c中，SD.c为从网上搜索的读取SD卡方法的例程，在研究清楚其具体作用后，我将它放入了自己的工程中，并进行了调用。

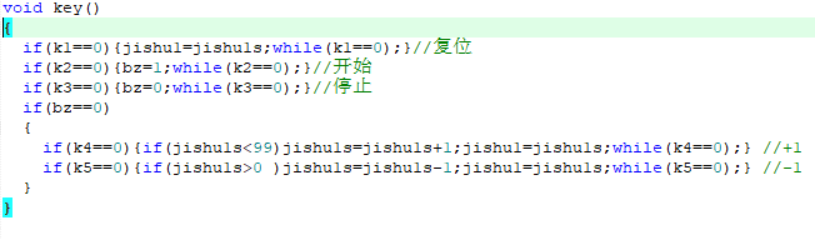
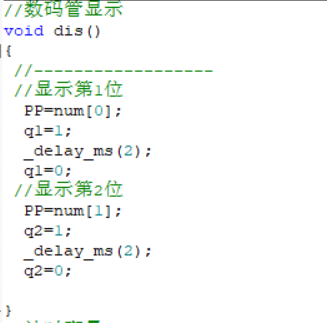


#### 倒计时部分

主要函数为display()、dis()、key()

分别的作用是：获取数码管显示的数字信息、进行数码管显示、读取按键内容。





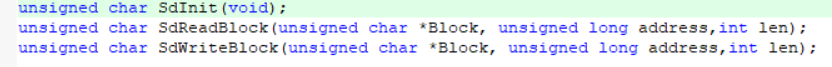
可以看出，key()函数的主要作用就是通过读入按键的信息，改变全局变量的值，在数码管例程中，它的作用是改变数码管的值，那么我也可以把它们应用到动画播放当中，从而实现动画播放当中的控制，包括暂停，重播等等。与倒计时配合，可以达到预期效果。

## SD卡读取数据部分SD.c

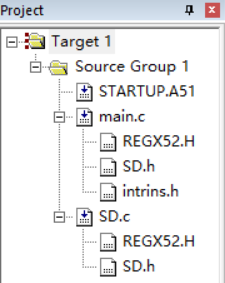
此部分为例程引用，是一个单独的部分，对于它的功能进行了探究，它的作用是从SD卡中读取数据，并且通过四根引线传入单片机当中。

## 头文件SD.h

起初考虑过把所有的内容都放到一个c文件当中，但是考虑到程序的可读性和修改的方便，同时为了区分我的工作内容和外部引用，我把它们分了开来。其实多个代码文件构成一个工程，共同实现一个功能，这是所有单片机开发者必须学会的操作。我借此练习了一下多个程序文件之间连接调用的方法。通过查阅资料，我了解到，正如学习C语言是要用到的头文件，我也可以自己定义一个头文件，这样就可以实现外部的函数引用。



刚开始我的程序报错了，原因是没有在外部引用的c文件中也包含该头文件，在修改以后就实现了外部库函数的调用。



视频见：“计时播放视频.mp4”工程文件见“lib3”文件夹

# 实验总结

通过这半个学期的电子线路设计实验，我着实学到了许多。忽然想到一句话：“纸上得来终觉浅，绝知此事要躬行”。在课堂上学到的东西很多，但是大部分如果不用的话很快就会忘记，但是这个实验中学到的许多知识却是记忆犹新。

有许多操作，许多电子线路、数字逻辑、单片机原理、接口技术等等的知识，虽然以前在课堂上学过，但是却不太记得。在调试实验电路时，我遇到问题，会从网上论坛、课本等等渠道去寻求解答。在实在解决不了问题时，我会和同学老师讨论。在这样的氛围下，我把之前不会的、不扎实的知识又重新学习巩固了一遍。

让我印象深刻的是第二次实验，我做了一个放烟花的电路，其中控制电路用到了晶体管。虽然模拟电子技术当中反反复复的学过，但是真正应用的时候，还是又这样那样的问题。遇到了二极管不亮的情况，返回去排查电路，应用模电知识，我发现是晶体管工作在了截至区的缘故。

还有就是在应用Image2Lcd软件时，刚开始显示的图片是重叠的。经过探索，我发现是扫描方式没有选对。这样经过探究错误、发现问题、解决问题的过程，让我对于像素表示图像的原理有了一定的认识。

最后一次实验也是我一直想做的，之前看到b站好多大佬用单片机播放bad apple动画，我都很羡慕，期望有一天能自己实现。现在在仿真软件上勉强实现了一下，还加入了一些小功能。不过其中很大一部分代码不是我的原创，是从各种例程当中改过来的。当然，实际的芯片和仿真软件也会有差别，所以说我要做的还有很多，在即将到来的嵌入式实验中，我可以进一步在实体芯片上再次尝试实现播放bad apple。

在应用单片机时，并行、串行、总线等概念时时出现，这些内容也在微机原理与嵌入式系统的课程中反复出现，我要学的还有很多。虽然现在能实现一个简单的功能，但是我能感觉到自己对于这些概念的理解还有欠缺，所以在之后要再次进行深入的学习。

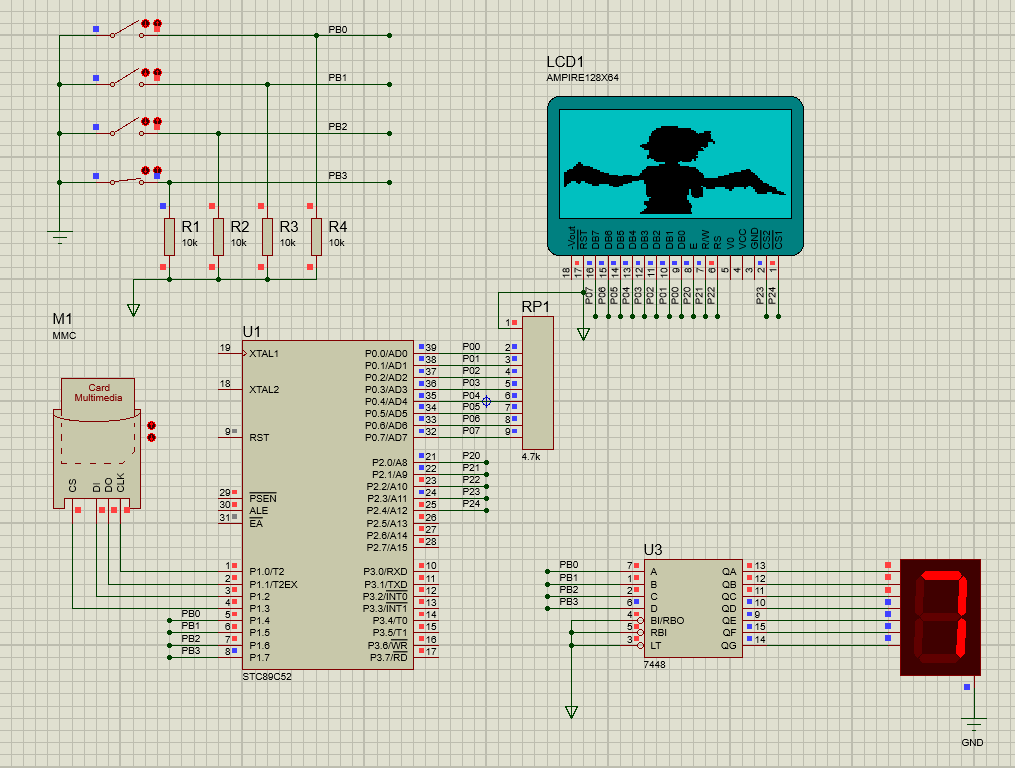
非常感谢老师的悉心指导！

# 做的不太成功的另一个作品

该部分实现结果不太理想，故没有放到正文中，不过还是把它写在报告里面。

视频文件见“another\_播放速度控制.mp4”工程文件见“another”文件夹

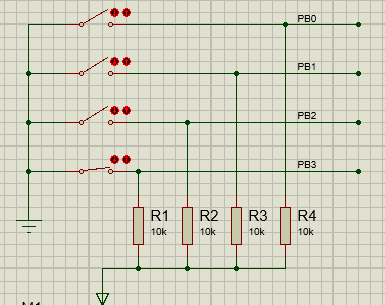
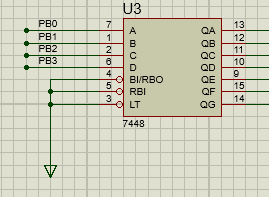
原理图：



最大的不同就是加入了一个BCD码转七段数码管的元件7448.通过输入对应延时量的BCD码，改变动画播放延时的大小，从而控制动画播放的速度。

一些不同的器件：

键盘：

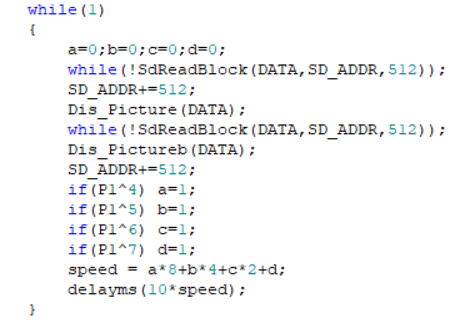
 

PB0~PB3分别接单片机的P1.4~P1.7和7448的7、1、2、6口，输入的是需要延时的BCD码。

7448：

只是显示键盘输入BCD码的十进制表示，其实与单片机没有直接联系，本来我想通过单片机输出信号控制它，但是在播放动画的同时控制它会报错，查阅资料后发现加入一个锁存器会解决这一问题。

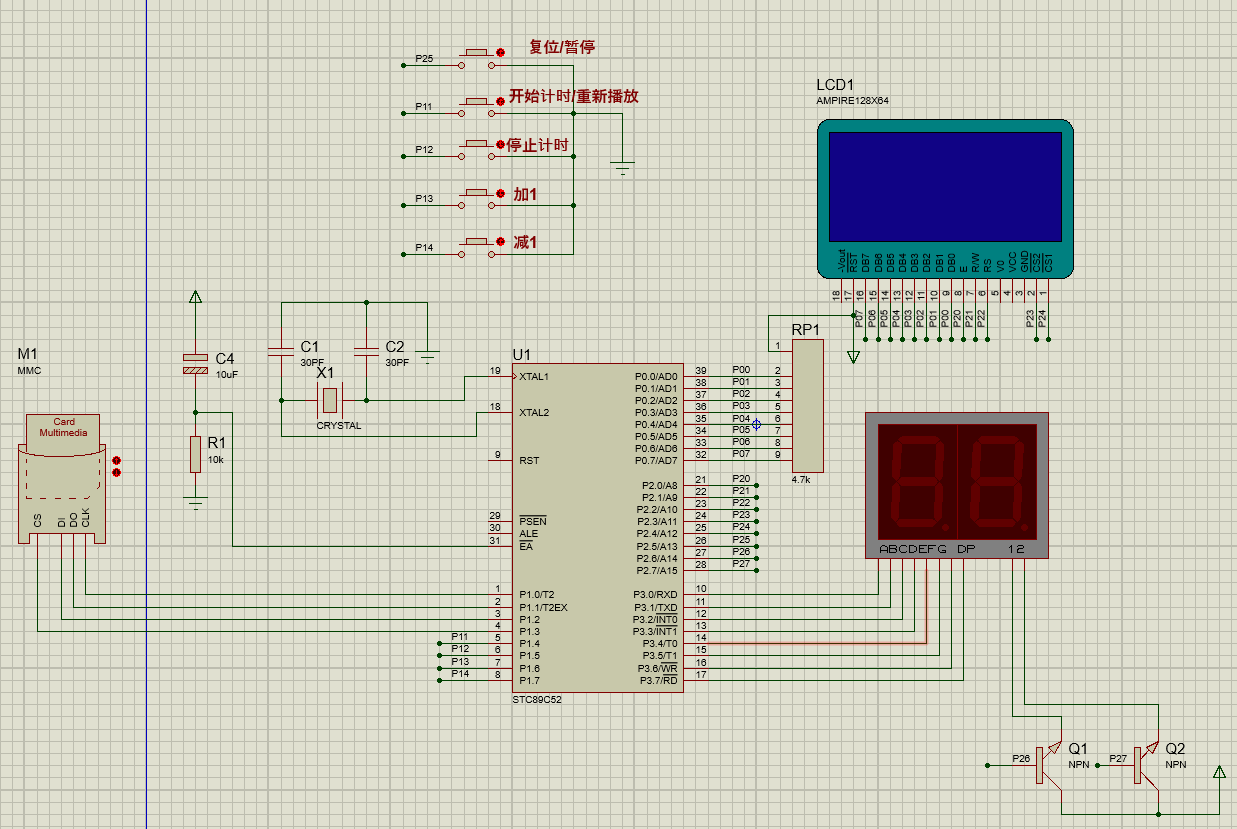
部分代码：



相交正式作品，这个比较简单，没有定时的部分，最后延时函数传入的参数是四位BCD码转换成的十进制数，从而实现了对延时的控制，进一步改变了播放速度。但是在仿真中，速度改变并不明显。

# 附录

## 原理图



## 程序代码

### Main函数

#include <REGX52.H>

#include "SD.h"

#include <intrins.h>

//数码管段码端口

#define PP P3

//共阳极数码管段码

unsigned char code SEG7[]={/\*0,1,2,3,4,5,6,7,8,9,\*/

0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,};

//数码管显示缓冲

unsigned char num[]={1,2};

//数码管位码端口

sbit q1=P2^6;

sbit q2=P2^7;

//复位开始停止加1减1按键

sbit k1=P2^5;

sbit k2=P1^4;

sbit k3=P1^5;

sbit k4=P1^6;

sbit k5=P1^7;

//----------------------------------------------------

unsigned long SD\_ADDR=0;

unsigned int count;

unsigned char xdata DATA[512];

sbit E = P2^0;

sbit RW = P2^1;

sbit RS = P2^2;

sbit CS2 = P2^3;

sbit CS1 = P2^4; //端口定义

#define DataPort P0

/\*12864判忙 \*/

bit Chek\_Busy(void)

{

DataPort = 0xff;

RW = 1;

RS = 0;

E = 1;

E = 0;

return (bit)(DataPort & 0x80);

}

/\*------------------------------------------------

选屏

i:0是左屏,1是右屏,2全屏

------------------------------------------------\*/

void Choose\_12864(unsigned char i)

{

switch (i)

{

case 0: CS1 = 0;CS2 = 1;break;

case 1: CS1 = 1;CS2 = 0;break;

case 2: CS1 = 0;CS2 = 0;break;

default: break;

}

}

/\*------------------------------------------------

写命令

------------------------------------------------\*/

void LCD\_Cmd(unsigned char cmd)

{

while(Chek\_Busy());

RW = 0;

RS = 0;

DataPort = cmd;

E = 1;

E = 0;

}

/\*------------------------------------------------

读LCD

------------------------------------------------\*/

unsigned char LCD\_Read()

{

unsigned char read\_data;

while(Chek\_Busy());

RW = 1;//需进行一次空读

RS = 1;

E = 1;

E = 0;

RW = 1;

RS = 1;

E = 1;

read\_data = DataPort;

E = 0;

return (read\_data);

}

/\*------------------------------------------------

写数据

------------------------------------------------\*/

void LCD\_Data(unsigned char dat)

{

while(Chek\_Busy());

RW = 0;

RS = 1;

DataPort = dat;

E = 1;

E = 0;

}

/\*------------------------------------------------

设置地址

PAGE:0-7;

Y\_Address:0-63

------------------------------------------------\*/

void Set\_PageY(unsigned char PAGE,unsigned char Y\_Address)

{

LCD\_Cmd(0xB8 + PAGE);

LCD\_Cmd(0x40 + Y\_Address);

}

/\*------------------------------------------------

清屏

------------------------------------------------\*/

void LCD\_Clear(void)

{

unsigned char page,row;

Choose\_12864(2);

for (page = 0xb8; page < 0xc0; page ++)

{

LCD\_Cmd(page);

LCD\_Cmd(0x40);

for (row = 0; row < 64; row ++)

{

LCD\_Data(0x00);//对12864所有地址全部写零

}

}

}

/\*------------------------------------------------

初始化

------------------------------------------------\*/

void LCD\_Init(void)

{

CS2 = 0;

CS1 = 0;

LCD\_Cmd(0x3F);//开显示

}

/\*-------------------------------------------------

显示一幅12864图片

-------------------------------------------------\*/

void Dis\_Picture(unsigned char \*picture)

{

unsigned char ii,kk;

for (kk = 0; kk < 4; kk ++)//LCD共分7=8页

{

Choose\_12864(2);

Set\_PageY(kk,0);

Choose\_12864(0);

for (ii = 0; ii < 128; ii ++)

{

LCD\_Data(\*picture);

picture ++;

if (ii == 63)

{

Choose\_12864(1);

}

}

}

}

void Dis\_Pictureb(unsigned char \*picture)

{

unsigned char ii,kk;

for (kk = 4; kk < 8; kk ++)//LCD共分7=8页

{

Choose\_12864(2);

Set\_PageY(kk,0);

Choose\_12864(0);

for (ii = 0; ii < 128; ii ++)

{

LCD\_Data(\*picture);

picture ++;

if (ii == 63)

{

Choose\_12864(1);

}

}

}

}

void delayus(unsigned char t)

{

while(--t);

}

void delayms(unsigned char t)

{

while(t--)

{

delayus(245);

delayus(245);

}

}

//--------------------------------------------------------------------

//倒计时模块

//延时函数ms

void \_delay\_ms(unsigned int t)

{

unsigned int i,j;

for(i=0;i<t;i++)

for(j=0;j<120;j++);

}

//数码管显示

void dis()

{

//------------------

//显示第1位

PP=num[0];

q1=1;

\_delay\_ms(2);

q1=0;

//显示第2位

PP=num[1];

q2=1;

\_delay\_ms(2);

q2=0;

}

//计时变量

unsigned int jishu1s=10;

unsigned int jishu1=10;

unsigned int jishu2;

//开始停止变量

unsigned int bz;

//显示控制

void dispaly()

{

num[0]=SEG7[jishu1%100/10]; //显示个位

num[1]=SEG7[jishu1%10]; //显示小数位

}

//按键处理

void key()

{

if(k1==0){jishu1=jishu1s;while(k1==0);}//复位

if(k2==0){bz=1;while(k2==0);}//开始

if(k3==0){bz=0;while(k3==0);}//停止

if(bz==0)

{

if(k4==0){if(jishu1s<99)jishu1s=jishu1s+1;jishu1=jishu1s;while(k4==0);} //+1

if(k5==0){if(jishu1s>0 )jishu1s=jishu1s-1;jishu1=jishu1s;while(k5==0);} //-1

}

}

//----------------------------

void main(void)

{

\_delay\_ms(10);

//定时器0配置方式1 16

TMOD=0x01;

//使能定时器0

ET0=1;

//定时器的初值

TH0=(65536-50000)/256;

TL0=(65536-50000)%256;

//停止开启定时器0

TR0=1;

EA=1;

LCD\_Init();

LCD\_Clear();

SdInit();

DATA[0]=255;;

DATA[1]=1;

DATA[2]=2;

DATA[3]=3;

DATA[511]=0xf0;

while(1)

{

dispaly();//显示控制

dis(); //显示

key();//按键

if(jishu1==0)

{

while(1)

{

while(!SdReadBlock(DATA,SD\_ADDR,512));

SD\_ADDR+=512;

Dis\_Picture(DATA);

while(!SdReadBlock(DATA,SD\_ADDR,512));

Dis\_Pictureb(DATA);

SD\_ADDR+=512;

delayms(100);

key();//按键

if(jishu1!=0) break;

if(bz==1) SD\_ADDR=0;

}

}

}

}

//----T0 计数

void Time0() interrupt 1

{

//重装初值50ms;

TH0=(65536-50000)/256;

TL0=(65536-50000)%256;

//开始

if(bz==1)

{

jishu2=jishu2+1;

//到达1s

if(jishu2==20)

{

jishu2=0;

if(jishu1>0)jishu1=jishu1-1;//减1s

if(jishu1==0)bz=0;//结束

}

}

}

### 导入存储卡文件的SD.c

#include <REGX52.H>

#include "SD.h"

sbit ACC0=ACC^0;

sbit ACC1=ACC^1;

sbit ACC2=ACC^2;

sbit ACC3=ACC^3;

sbit ACC4=ACC^4;

sbit ACC5=ACC^5;

sbit ACC6=ACC^6;

sbit ACC7=ACC^7;

//定义SD卡需要的4根信号线

sbit SD\_CLK = P1^0;

sbit SD\_DI = P1^2;

sbit SD\_DO = P1^1;

sbit SD\_CS = P1^3;

//写一字节到SD卡,模拟SPI总线方式

void SdWrite(unsigned char DATA)

{

ACC=DATA;

SD\_CLK=0;

SD\_DI=ACC7;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC6;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC5;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC4;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC3;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC2;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC1;

SD\_CLK=1;

SD\_CLK=0;

SD\_DI=ACC0;

SD\_CLK=1;

SD\_DI=1;//在空闲状态下DI需为高电平

}

//从SD卡读一字节,模拟SPI总线方式

unsigned char SdRead()

{

SD\_CLK=0;

SD\_CLK=1;

ACC7=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC6=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC5=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC4=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC3=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC2=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC1=SD\_DO;

SD\_CLK=0;

SD\_CLK=1;

ACC0=SD\_DO;

return ACC;

}

//检测SD卡的响应

unsigned char SdResponse()

{

unsigned char i=0,response;

while(i<=8)

{

response = SdRead();

if(response==0x00)

break;

if(response==0x01)

break;

i++;

}

return response;

}

//发命令到SD卡

void SdCommand(unsigned char command, unsigned long argument, unsigned char CRC)

{

SdWrite(command|0x40);

SdWrite(((unsigned char \*)&argument)[0]);

SdWrite(((unsigned char \*)&argument)[1]);

SdWrite(((unsigned char \*)&argument)[2]);

SdWrite(((unsigned char \*)&argument)[3]);

SdWrite(CRC);

}

//初始化SD卡

unsigned char SdInit(void)

{

int delay=0, trials=0;

unsigned char i;

unsigned char response=0x01;

SD\_CS=1;

for(i=0;i<=9;i++)

SdWrite(0xff);

SD\_CS=0;

//Send Command 0 to put MMC in SPI mode

SdCommand(0x00,0,0x95);

response=SdResponse();

if(response!=0x01)

{

return 0;

}

while(response==0x01)

{

SD\_CS=1;

SdWrite(0xff);

SD\_CS=0;

SdCommand(0x01,0x00ffc000,0xff);

response=SdResponse();

}

SD\_CS=1;

SdWrite(0xff);

return 1;

}

//往SD卡指定地址写数据,一次最多512字节

unsigned char SdWriteBlock(unsigned char \*Block, unsigned long address,int len)

{

unsigned int count;

unsigned char dataResp;

//Block size is 512 bytes exactly

//First Lower SS

SD\_CS=0;

//Then send write command

SdCommand(0x18,address,0xff);

if(SdResponse()==00)

{

SdWrite(0xff);

SdWrite(0xff);

SdWrite(0xff);

//command was a success - now send data

//start with DATA TOKEN = 0xFE

SdWrite(0xfe);

//now send data

for(count=0;count<len;count++) SdWrite(\*Block++);

for(;count<512;count++) SdWrite(0);

//data block sent - now send checksum

SdWrite(0xff); //两字节CRC校验, 为0XFFFF 表示不考虑CRC

SdWrite(0xff);

//Now read in the DATA RESPONSE token

dataResp=SdRead();

//Following the DATA RESPONSE token

//are a number of BUSY bytes

//a zero byte indicates the MMC is busy

while(SdRead()==0);

dataResp=dataResp&0x0f; //mask the high byte of the DATA RESPONSE token

SD\_CS=1;

SdWrite(0xff);

if(dataResp==0x0b)

{

return 0;

}

if(dataResp==0x05)

return 1;

return 0;

}

return 0;

}

//从SD卡指定地址读取数据,一次最多512字节

unsigned char SdReadBlock(unsigned char \*Block, unsigned long address,int len)

{

unsigned int count;

//Block size is 512 bytes exactly

//First Lower SS

SD\_CS=0;

//Then send write command

SdCommand(0x11,address,0xff);

if(SdResponse()==00)

{

//command was a success - now send data

//start with DATA TOKEN = 0xFE

while(SdRead()!=0xfe);

for(count=0;count<len;count++) \*Block++=SdRead();

for(;count<512;count++) SdRead();

//data block sent - now send checksum

SdRead();

SdRead();

//Now read in the DATA RESPONSE token

SD\_CS=1;

SdRead();

return 1;

}

return 0;

}

### 头文件SD.h

unsigned char SdInit(void);

unsigned char SdReadBlock(unsigned char \*Block, unsigned long address,int len);

unsigned char SdWriteBlock(unsigned char \*Block, unsigned long address,int len);