硬件课程设计报告

--课题：简易电子书

同组人姓名：张琢，赵冰洁，段经璞

学号：U200913945

U200913956

U200913933

指导老师：曾喻江

课设评价：

课设成绩：

1. 概述....4
1.1 课程设计的核心任务....4
1.2 课程设计的工作进程....4
2. 需求分析....4
2.1 功能分析....4
2.2 可行性分析....5
2.2.1 技术可行性分析....5
2.2.2 经济可行性....5
3. 概要设计....5
3.1 对 MSP430F149 单片机的使用....5
3.2 SD 卡及 FAT32 文件系统的读写....6
3.3 菜单显示....6
3.4 电子书显示....6
3.5 音乐播放功能....7
3.6 游戏功能....7
4. 详细设计....7
4.1 SD 卡及其结构....7
4.2 12864 液晶屏结构及用法....11
4.2.1 12864 液晶屏的硬件结构....11
4.2.2 12864 内部的数据缓存....12
4.2.3 12864 液晶屏的指令....13
4.2.4 12864 液晶屏驱动函数....14
4.3 FAT32 文件系统....14
4.3.1 FAT32 文件系统结构简介....14
4.3.2 FAT32 文件系统驱动函数....16
4.4 VS1003 音频解码芯片....17
4.5 整机结构....18
4.5.1 MSP430 开发板与 12864 液晶屏的连接....19
4.5.2 MSP430 开发板与 SD 卡的连接....19
4.5.3 MSP430 开发板与 VS1003 芯片的连接....20
4.6 菜单功能....20
4.7 电子书显示功能....21
4.7.1 实现显示功能的基本函数....21
4.7.2 向下翻页功能....22
4.7.3 向上翻页功能....22
4.7.4 显示阅读时间....23
4.7.5 按百分比跳转....23
4.7.6 书签功能的实现....23
4.7.7 英文单词的人性化显示....24
4.8 音乐播放功能....24
4.9 俄罗斯方块游戏功能....25
4.10 按键检测....27
5. 系统测试....28
5.1 菜单功能的测试....28

5.2 电子书阅读功能测试.....29
5.3 音乐播放功能测试.....31
5.4 游戏功能的测试.....31
6. 心得体会.....31
7. 参考文献.....32

1. 概述

1.1 课程设计的核心任务

随着嵌入式技术的飞速发展，各种电子产品出现了小型化，智能化，多功能化方向发展的趋势。在这种趋势下，许多小型电子产品层出不穷，并且逐渐开始替代传统的生活用品，并走近我们的生活。电子书阅读器的出现就再图书信息的承载和阅读方式上有了很大的革新。本项目的目的就是制作一个电子阅读器，可以让人们方便的进行随身阅读。同时在阅读疲劳时还可以利用其中的 mp3 播放器和内置的游戏进行放松娱乐。

1.2 课程设计的工作进程

本课设由段经璞，赵冰洁，张琢三人组队共同完成。大家相互之间进行分工合作，以取得快速的工作进展和优秀的最后结果。

2. 需求分析

2.1 功能分析

电子书，最出名的代表应该是亚马逊公司的 kindle 电子书阅读器。Kindle 使用电子墨水屏幕，可以提供接近纸质书的阅读体验。同时还可以进行音乐播放，以及无线上网等功能。并且 kindle 通过和亚马逊的网上书店整合，可以实现图书的快速查找，购买以及阅读。而现在，单纯的电子书几乎不存在，电子书的功能几乎都被整合在了各式各样掌上终端中。这样的掌上终端在进行电子书阅读的同时，可以播放音乐，无线浏览网页以及进行游戏。因此单纯的电子书可以说是完全没有市场的，它必须被整合其他的功能才能实现良好的效果。

通过以上的分析可以发现，电子书阅读器必须与其他功能进行整合才能被推入市场。当然，在整合其他功能的同时，我们更要把电子书阅读的功能完善好。根据分析，我们决定在课设中实现电子书，音乐播放以及俄罗斯方块游戏的功能。

电子书阅读部分，我们决定实现以下功能：

1. SD 卡及 FAT32 文件系统的读写
2.长/短文件名的显示
3.电子书的显示
4.按阅读进度百分比直接跳转
5.阅读时间显示
6.书签功能
7.英文单词的人性化显示

在音乐播放部分，我们决定实现以下功能：

1.歌曲名称的显示
2.歌曲音量的调节
3.歌曲的前进，后退，快进，快退及暂停的功能。

2.2 可行性分析

2.2.1 技术可行性分析

在电子书阅读部分，我们组使用 msp430f149 芯片作为主控芯片。使用 SD 卡作为存储设备。因为 msp430f149 芯片自带 spi 控制器，所以可以直接使用 spi 接口，利用 SD 卡的 spi 模式与 SD 卡进行数据交换。同时准备使用 12864 液晶屏作为我们电子书的显示部分。12864 具有内建的英文及中文字库，可以方便的显示各种文字信息，从而实现电子书的显示。

在音乐播放部分，我们组使用 vs1003 音频解码芯片来进行音频解码。Vs1003 自带 spi 接口，可以方便的利用 spi 总线与 msp430f149 主控芯片进行连接。在进行音乐播放时，只要把音频文件从 SD 卡中取出再直接发送到 vs1003 中即可。

在游戏部分，我们利用 12864 的绘图功能实现俄罗斯方块游戏。

2.2.2 经济可行性

因为我们小组不具备自己焊接电路板的时间和能力，因此我们组直接从市场上购买现成的开发板。最终购买的开发板的价格如下：

1、msp430f149 开发板：64 元。
2、12864 液晶屏：56 元。
3、vs1003 音频解码芯片：43 元。
4、1GB 大小的 SD 卡：20 元
5、SD 卡插座：6 元。

整体系统的总价格：179 元。

整体价格在可以接受的范围内。而且如果可以自己直接购买芯片并焊接电路板的话，就可以大幅度降低整体系统的价格。

3. 概要设计

3.1 对 MSP430F149 单片机的使用

MSP430F149 单片机是由 TI 公司开发的一款高性能低功耗单片机。实验中主要使用 F149

的 CPU 内核，SPI 控制器，IO 端口及 TimerA 定时器。

3.2 SD 卡及 FAT32 文件系统的读写

要想实现电子书阅读功能，就必须实现 SD 卡的读写，因为电子书及音乐数据都是以 SD 卡作为存储介质的。为了使用 SD 卡，就必须实现 SD 卡的初始化功能，读扇区功能及写扇区功能。同时，我们组在进行设计时，确定了使用 FAT32 文件系统作为 SD 卡上得文件系统。FAT32 文件系统是一种简单高效的文件系统，在实现 SD 卡读写扇区功能的基础上，我们需要基于 FAT32 文件系统的特性实现对 FAT32 文件系统的读写，这样才能获取 SD 卡上的文本文件及音乐文件数据。

3.3 菜单显示

一个完整可用的电子书系统一定要有良好的菜单系统的支持。我们组设计了两级菜单显示，其设计思想如下：

1、主菜单：当进入系统时，会显示主菜单，其中包含电子书，音乐，游戏三个选项，用户选择这三个选项中的一个就可以进入下一级菜单。

2、目录菜单：当在主菜单中选择电子书或音乐时，就会进入电子书或音乐的目录菜单。在电子书目录菜单中，会列出当前 SD 卡中所存储的所有电子书的文件名，用户选择项阅读的文件即可进行阅读。同样的，在音乐目录菜单中，会列出当前 SD 卡中所存储的所有音乐文件的文件名，用户选择项播放的音乐即可进行音乐播放。

3.4 电子书显示

电子书显示功能是一个电子书阅读器的核心，它以 12864 液晶显示屏作为基础，从 SD 卡中读取文件并显示在屏幕上。整个电子书的显示功能是为 12864 液晶屏量身设计的，它文本模式可以显示 4 行文字，每行显示 8 个汉字或 16 个英文字母。为了取得良好的阅读体验，我们组设计了以下的功能：

1、向上向下翻页功能：当人在阅读电子书时，如果翻来一页新的文字时，读者有可能会因暂时性忘记上一页最后一行的若干内容而造成对下一页起始内容的理解困难。因此，我们决定在向下翻页时，仅会翻三行内容，也就是说，我们会以当前屏幕的最后一行，为新一页的第一行。这样就可以保证阅读的流畅性。同理，在向上翻页时，我们会以当前屏幕的第一行，为后一页的最后一行。

2、按比例跳转功能：读者在读书的时候，有时会想直接跳转到这本书的后面去进行阅读。这时候就需要一个按书内容的百分比进行跳转的功能。本课设中实现了这个功能。

3、英文单词的人性化显示功能：为了保证在阅读英文文章时的流畅性，我们必须保证每一个词都不会因为换行而被截断。本课设就实现了这个功能，从而保证了阅读英文文章时的流畅性。

4、书签功能：这个功能对电子书至关重要，因为读者几乎无法连续的读完一本书，当读

者退出阅读时，他需要保存当前的阅读进度以便在下次方便的继续阅读。这也是本课设中所实现的关键功能。

5、阅读时间的显示：大家都知道，阅读时间过长对视力不好。因此一个完善的电子书阅读器必须可以显示当前电子书的阅读时间，以使读者可以清楚的知道自己何时需要休息。

3.5 音乐播放功能

在前面已经讨论过，一个成熟的电子书，一定要集成其他的附加功能才能提升整个系统的档次。音乐播放功能就是非常重要的一个功能。本课设中利用 vs1003 制作了一个 mp3 播放器，可以播放高品质的音乐。并且实现了调整音量，暂停，快进快退，向前向后选择歌曲的功能。这样，当阅读电子书感到疲劳时，就可以听听音乐放松一下心情。

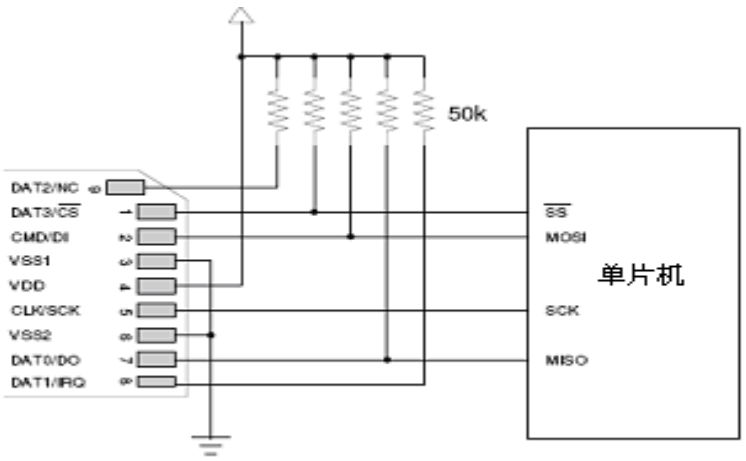
3.6 游戏功能

课设中利用 12864 的绘图模式制作了俄罗斯方块的游戏。这样读者可以在阅读疲劳时玩玩俄罗斯方块来放松心情。

4. 详细设计

4.1 SD 卡及其结构

SD 卡 SPI 模式下与单片机的连接图：



SD 卡支持两种总线方式：SD 方式与 SPI 方式。其中 SD 方式采用 6 线制，使用 CLK、CMD、DAT0~DAT3 进行数据通信。而 SPI 方式采用 4 线制，使用 CS、CLK、DataIn、DataOut 进

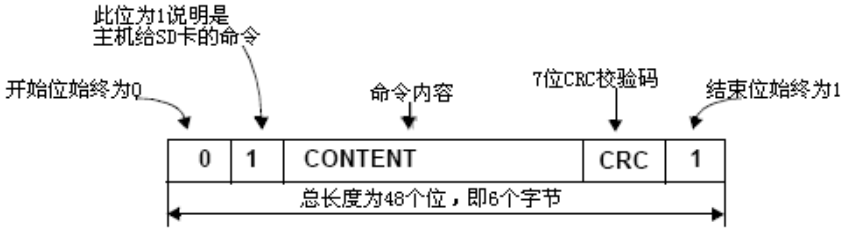
行数据通信。SD 方式时的数据传输速度与 SPI 方式要快，采用单片机对 SD 卡进行读写时一般都采用 SPI 模式。采用不同的初始化方式可以使 SD 卡工作于 SD 方式或 SPI 方式。这里只对其 SPI 方式进行介绍。

1、SPI 方式驱动 SD 卡的方法

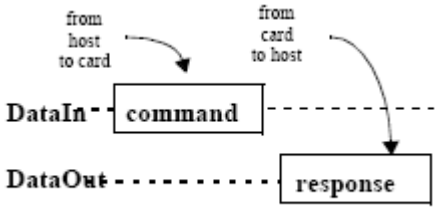
SD 卡的 SPI 通信接口使其可以通过 SPI 通道进行数据读写。从应用的角度来看，采用 SPI 接口的好处在于，很多单片机内部自带 SPI 控制器，不光给开发上带来方便，同时也见降低了开发成本。然而，它也有不好的地方，如失去了 SD 卡的性能优势，要解决这一问题，就要用 SD 方式，因为它提供更大的总线数据带宽。SPI 接口的选用是在上电初始时向其写入第一个命令时进行的。以下介绍 SD 卡的驱动方法，只实现简单的扇区读写。

1）命令与数据传输

SD 卡自身有完备的命令系统，以实现各项操作。命令格式如下：



命令的传输过程采用发送应答机制，过程如下：



每一个命令都有自己命令应答格式。在 SPI 模式中定义了三种应答格式，如下表所示：

字节	位	含义
1	7	开始位，始终为 0
	6	参数错误
	5	地址错误
	4	擦除序列错误
	3	CRC 错误
	2	非法命令
	1	擦除复位
	0	闲置状态

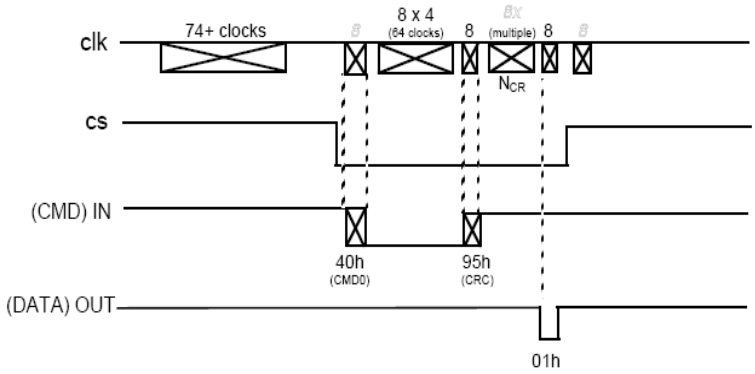
字节	位	含义
1	7	开始位，始终为 0
	6	参数错误
	5	地址错误
	4	擦除序列错误
	3	CRC 错误
	2	非法命令
	1	擦除复位
	0	闲置状态
2	7	溢出，CSD 覆盖
	6	擦除参数
	5	写保护非法
	4	卡 ECC 失败
	3	卡控制器错误
	2	未知错误
	1	写保护擦除跳过，锁 / 解锁失败
	0	锁卡

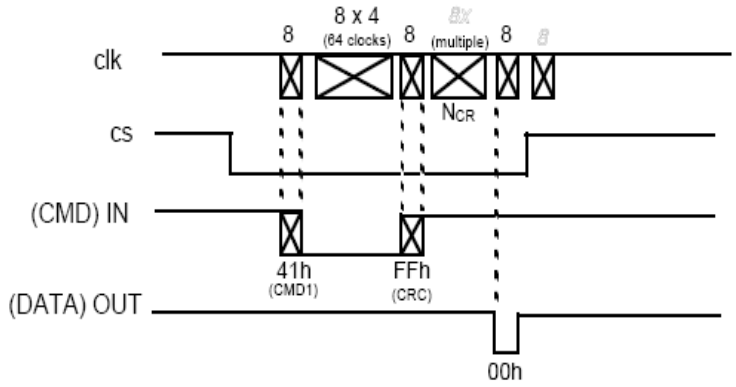
字节	位	含义
1	7	开始位，始终为 0
	6	参数错误
	5	地址错误
	4	擦除序列错误
	3	CRC 错误
	2	非法命令
	1	擦除复位
	0	闲置状态
2～5	全部	操作条件寄存器，高位在前

2）初始化

SD 卡的初始化是非常重要的，只有进行了正确的初始化，才能进行后面的各项操作。在初始化过程中，SPI 的时钟不能太快，否则会造初始化失败。在初始化成功后，应尽量提高 SPI 的速率。在刚开始要先发送至少 74 个时钟信号，这是必须的。在很多读者的实验中，很多是因为疏忽了这一点，而使初始化不成功。随后就是写入两个命令 CMD0 与 CMD1，使 SD 卡进入 SPI 模式

初始化时序图：



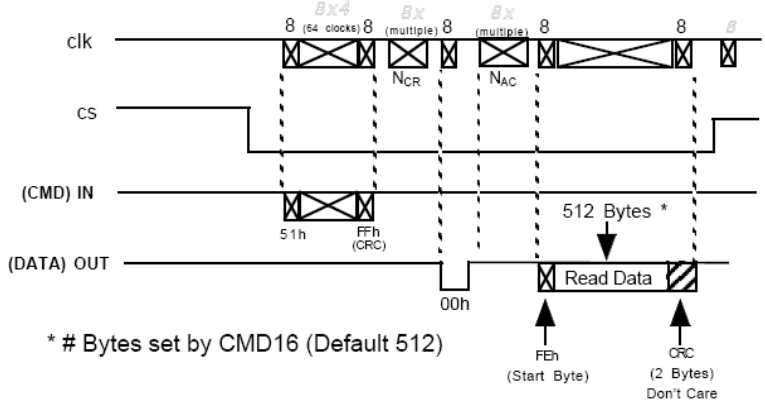


在项目中，初始化 SD 卡的函数为：unsigned char InitMmc()。

3）扇区读

扇区读是对 SD 卡驱动的目的之一。SD 卡的每一个扇区中有 512 个字节，一次扇区读操作将把某一个扇区内的 512 个字节全部读出。过程很简单，先写入命令，在得到相应的回应后，开始数据读取。

扇区读的时序：



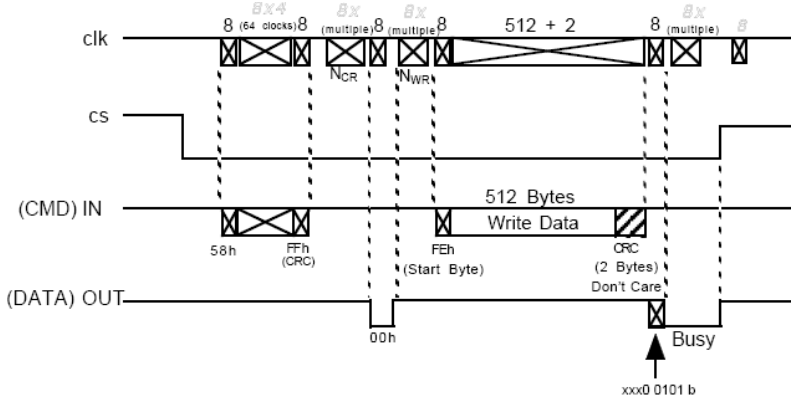
在项目中，读 SD 卡扇区的函数由

ReadMmcSector(unsigned long lba, unsigned int Bytes,unsigned char *buffer)，调用 WriteMmcSectorAddress(unsigned long sector)和 MMC_get_data(unsigned int Bytes,unsigned char *buffer) 实现。其中 ReadMmcSector 输入的参数为 SD 卡的物理扇区号 lba，要读出的字节数 Bytes，以及读出后的数据所存放的数组的首地址 buffer。

4）写扇区

扇区写是 SD 卡驱动的另一目的。每次扇区写操作将向 SD 卡的某个扇区中写入 512 个字节。过程与扇区读相似，只是数据的方向相反与写入命令不同而已。

扇区写的时序：



在项目中，写 SD 卡的函数为 SD_write_sector(unsigned long addr,unsigned char *Buffer)。输入参数为 SD 卡的物理扇区号以及写入 SD 卡指定扇区中的内容的数组的首地址。

4.2 12864 液晶屏结构及用法

4.2.1 12864 液晶屏的硬件结构

12864 是一种具有具有 4 位/8 位并行，2 线或 3 线串行多种接口方式。在本课设中使用了 8 位并行方式与 MSP430 单片机进行链接。该液晶屏内部含有国标一级，二级简体中文字库的点阵图形液晶显示模块，其显示分辨率为 128x64。其汉字的分辨率为 16x16。其 ASCii 字符的分辨率为 16x8。也就是说 12864 液晶屏总共可以显示四行字符，每一行可以显示 8 个汉字或 16 个英文字母。其 8 位串行状态下引脚接口如下表所示：

管脚号	管脚名称	电平	管脚功能描述
1	VSS	0V	电源地
2	VCC	3.0+5V	电源正
3	V0	-	对比度（亮度）调整
4	RS(CS)	H/L	RS=“H”，表示 DB7——DB0 为显示数据 RS=“L”，表示 DB7——DB0 为显示指令数据
5	R/W(SID)	H/L	R/W=“H”，E=“H”，数据被读到 DB7——DB0 R/W=“L”，E=“H→L”， DB7——DB0 的数据被写到 IR 或 DR
6	E(SCLK)	H/L	使能信号
7	DB0	H/L	三态数据线
8	DB1	H/L	三态数据线
9	DB2	H/L	三态数据线
10	DB3	H/L	三态数据线
11	DB4	H/L	三态数据线
12	DB5	H/L	三态数据线
13	DB6	H/L	三态数据线
14	DB7	H/L	三态数据线
15	PSB	H/L	H：8 位或 4 位并口方式，L：串口方式（见注释 1）
16	NC	-	空脚
17	/RESET	H/L	复位端，低电平有效（见注释 2）
18	VOUT	-	LCD 驱动电压输出端
19	A	VDD	背光源正端（+5V）（见注释 3）
20	K	VSS	背光源负端（见注释 3）

4.2.2 12864 内部的数据缓存

下面仅介绍课设中使用倒的两个数据缓存：

1、显示 RAM（DDRAM）

显示 RAM 提供 64x2 字节的空间，最多可以显示 4 行每行 8 个汉字，或 4 行每行 16 个 ASCii 字符。只要直接将汉字或 ASCii 的编码写入 DDRAM 中，屏幕中就会出现相应的文字。屏幕上第一行到第四行所对应的 DDRAM 中的地址分别为 0x80-0x87，0x90-0x97，0x88-0x8f，0x98-0x9f。其中，每一个地址对应屏幕上分辨率为 16x16 的一个区域。在其中可以显示两个 ASCii 字符，或一个汉字。要将文字写入显存并显示在屏幕中指定的位置上，就必须先向 12864 芯片写入指定位置的地址，然后再连续的写入两字节的数据。然后这两字节的数据所对应的文字就会显示在地址指定位置的 16x16 的点阵区域内。因为汉字编码为两个字节，而 ASCii 编码为 1 一个字节。所以屏幕上指定位置的 16x16 的点阵区域可以显示两个英文字符或一个汉字。

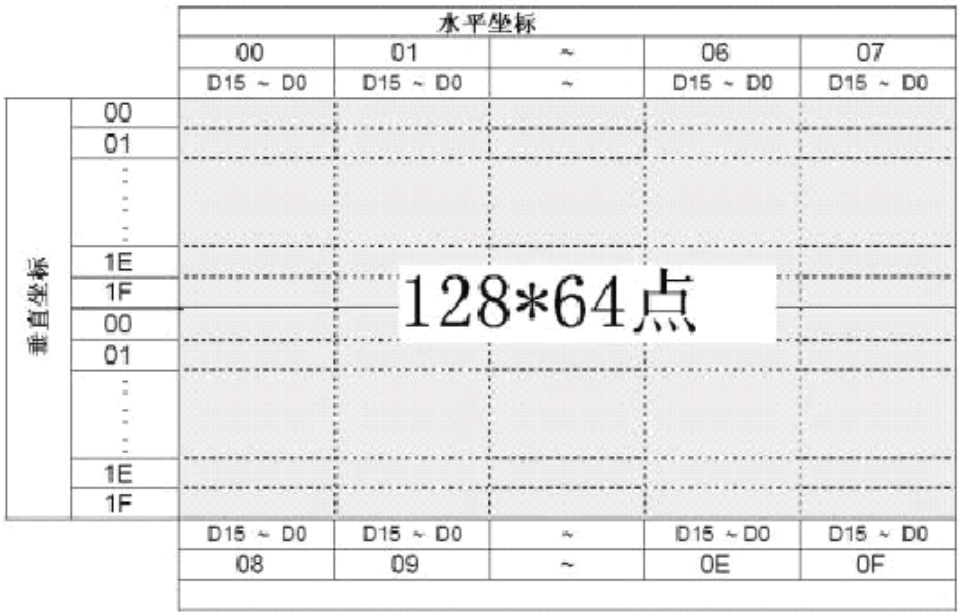
12864 屏幕上的位置以及其在 DDRAM 中所对应的地址如下图所示。注意，其中每一个位置均表示一个 16x16 的区域。

80H	81H	82H	83H	84H	85H	86H	87H
90H	91H	92H	93H	94H	95H	96H	97H
88H	89H	8AH	8BH	8CH	8DH	8EH	8FH
98H	99H	9AH	9BH	9CH	9DH	9EH	9FH

2、绘图 RAM（GDRAM）

绘图 RAM 中每一位的值用来控制 12864 液晶屏上每一个像素点的亮灭。值为 1，对应的

像素点就被点亮，值为 0，对应的像素点就被熄灭。在设定绘图 RAM 的值时，先写入垂直地址，再写入水平地址。之后向绘图 RAM 里连续写入两字节的数据。这样可以完成 12864 液晶屏上连续 16 个像素点控制。绘图 RAM 的地址所控制的像素点在屏幕上的对应位置如下图所示：



4.2.3 12864 液晶屏的指令

以下仅仅列出程序中所使用的指令：

功能设定	0	0	0	0	1	DL	X	RE	X	X	DL=0/1：4/8 位数据 RE=1：扩充指令操作 RE=0：基本指令操作
显示状态开/关	0	0	0	0	0	0	1	D	C	B	D=1：整体显示 ON C=1：游标 ON B=1:游标位置反白允许
清除显示	0	0	0	0	0	0	0	0	0	1	将DDRAM填满"20H",并且设定DDRAM的地址计数器(AC)到"00H"
设定DDRAM地址	0	0	1	0	AC5	AC4	AC3	AC2	AC1	AC0	设定 DDRAM 地址（显示位址） 第一行：80H－87H 第二行：90H－97H
写数据到 RAM	1	0	数据								将数据 D7——D0 写入到内部的 RAM (DDRAM/CGRAM/IRAM/GRAM)
读出 RAM 的值	1	1	数据								从内部 RAM 读取数据 D7——D0 (DDRAM/CGRAM/IRAM/GRAM)

扩充功能设定	0	0	0	0	1	CL	X	RE	G	0	CL=0/1：4/8 位数据 RE=1：扩充指令操作 RE=0：基本指令操作 G=1/0：绘图开关
设定绘图 RAM 地址	0	0	1	0 AC6	0 AC5	0 AC4	AC3 AC3	AC2 AC2	AC1 AC1	AC0 AC0	设定绘图 RAM 先设定垂直(列)地址 AC6AC5…AC0 再设定水平(行)地址 AC3AC2AC1AC0 将以上 16 位地址连续写入即可

4.2.4 12864 液晶屏驱动函数

1、写数据函数：void LCD_write_data(unsigned char data)
2、读数据函数：unsigned char LCD_read_data()
3、写命令函数：void LCD_write_com(unsigned char com)
4、文本显示模式初始化函数：void LCD_init(void)
其初始化过程为，先发送两次命令 0x30，设置显示模式为基本指令集。然后发送命令 0x0c，打开液晶屏屏幕显示。最后发送命令 0x01，清屏。
5、将一个长度为 17 的字符串显示在屏幕指定行的函数：void DisplayCgrom(uchar addr,uchar *hz)
6、绘图模式初始化函数：void Graph_Init(void)
初始化好绘图模式之后，就可以利用绘图模式来进行俄罗斯方块游戏。
7、点亮屏幕上指定位置像素的函数：void Put_Pixel(unsigned char row, unsigned char column)
该函数先将指定位置像素所在的 GDRAM 中的连续 16 个像素点的值读出来，改变需要改变的那个像素点的值，在将这 16 个像素点的值写回 GDRAM 的原位置中。
8、清楚屏幕上指定位置像素点的函数：void Clear_Pixel(unsigned char row, unsigned char column)
该函数的实现机制和上一个函数相同。

4.3 FAT32 文件系统

4.3.1 FAT32 文件系统结构简介

课设中选用了 FAT32 文件系统来存储文件。下面简介 FAT32 文件系统
1）主引导扇区（MBR）：主引导扇区位于整个 SD 卡物理扇区的第 0 号扇区。从中可以读取引导代码以及文件系统的起始位置。在课设中，仅从 MBR 中读取整个 SD 卡大小以及文件系统的其实扇区位置。

2）隐藏扇区（Hidden Sectors）：从 MBR 一直到文件系统的起始扇区之间的扇区被称为隐藏扇区。

3）保留扇区(Reserved Sectors)：文件系统的起始扇区及其后的若干扇区被称为保留扇区。

保留扇区中的第一个扇区，也就是文件系统的起始扇区是整个 SD 卡中最重要的部分。这个扇区又被称为 DBR。DBR 记录了文件系统的基本信息，其中包括：每扇区字节数，每簇扇区数，保留扇区数，FAT 表个数，文件系统总扇区数，每个 FAT 表的扇区数，根目录的其实簇号（通常为 2）及其他一些附加信息。课设中，系统初始化时，就需要读取 DBR 扇区，并在内存中记录其中的信息。

4）FAT 表：位于保留扇区后的是 FAT 表区，它由两个完全相同的 FAT 表组成。FAT 表有两个重要的作用，分别是描述簇的分配状态以及表明文件或目录的下一簇的簇号。在 FAT 表中，每个簇由 4 个字节的 FAT 表项来进行代表。FAT 表从第 0 号簇开始进行划分，每四个字节代表一个簇。其中第 0 号簇和第 1 号通常情况下不使用。如果某个簇未被分配使用，则它所对应的 FAT 表内的 FAT 表项值为 0。当某个簇已经被分配使用时，则它所对应的 FAT 表项内的 FAT 表项值也就是在该文件中，该簇的下一个簇的簇号。若这个簇号为 0x0fffffff，则表明当前簇为该文件的最后一个簇。

5）根目录：FAT 表后面就是数据区。数据区的开始是整个文件系统的根目录。它由目录项组成，用来记录每个文件的文件名，大小，起始簇号，类型，创建时间等信息。

6）目录项：每个目录项占 32 个字节，其结构如下图所示：

偏移字节	字节数	含义
00~00	1	文件名的第一个ASCII码字符，在某些情况下座位该目录项的分配状态值：如果是0x00，表示该目录项未被使用过；如果是0xE5，说明该目录项曾经被使用过，但是现在已被删除
01~0A	10	文件名的第2至第11个ASCII码
0B~0B	1	文件属性：0x01-只读；0x02-隐藏；0x04-系统文件；0x08-卷标；0x0F-为此值时表示该目录项为长文件名目录项；0x10-目录；0x20-存档
0C~0C	1	保留未使用
0D~0D	1	建立时间（十分之一秒）
0E~0F	2	建立时间（时、分、秒）
10~11	2	建立日期
12~13	2	最后访问日期
14~15	2	文件内容起始簇号（4字节）的高两个字节（FAT12/16文件系统将此两个字节设置为0，因为他们只使用2个字节描述起始簇号，不使用高字节）
16~17	2	最后修改时间
18~19	2	最后修改日期
1A~1B	2	文件内容起始簇号的低两个字节
1C~1F	4	文件内容大小字节数（子目录不适用大小值，设置为0）

7）长文件名目录项：FAT32 系统在为文件分配一个普通目录项时，还会为那些文件名长度超过 8 字节的文件分配长文件名目录项，长文件名目录项专门用来记录它所对应的文件的文件名。长文件名目录项通常顺序排列在短文件名目录项的上面，其结构如下图所示：

偏移字节	字节数	含义
00~00	1	如果目录项使用中则为序列号，如果未分配过则为0x00；如果曾经使用过但已经被删除，则为0xE5
01~1A	10	文件名的第1~5个字符（Unicode），未使用部分先填充两个字节的“00”，然后用0xFF填充
0B~0B	1	长目录项属性标志0x0F
0C~0C	1	保留未使用
0D~0D	1	校验和
0E~19	12	文件名的第6~11个字符（Unicode），未使用部分先填充两个字节的“00”，然后用0xFF填充
1A~1B	2	保留
1C~1F	4	文件名的第12~13个字符（Unicode），未使用部分先填充两个字节的“00”，然后用0xFF填充

4.3.2 FAT32 文件系统驱动函数

1、unsigned char strcmp(char* string1, char* string2)

这个函数主要是比较两个字符串是否是相同的，相同就返回 1，不相同就返回 0；简单来说就是比较后缀的。

2、void ReadMBR()

这个函数主要是读出文件系统的起始扇区号

3、void ReadDBR()

DBR 中保存的是文件系统的基本信息，每簇扇区数，隐藏扇区数（起始扇区数前面的那些没有读的,）FAT 表的个数，FAT 表扇区数，保留扇区数等

4 、 void ReadSectorFromCluster(unsigned long ClusterNumber,unsigned char SectorOffset)

这个函数作用的是找到扇区的位置，然后读出 512 字节

5、void GetFileNumber(void)

这里面是包含这种文件，只要不是被删除的文件或者是长文件名（特征是他的第 11 字节是 0X0F）则所找到的文件都会是 filenumber+1，同时各种文件类型各自加 1.。同时比较文件名是不是 convert，或者是 dirtable，若果是 convert，那么长文件名转码文件是存在的，置一，以长文件名显示；若是 dirtable，那么是存在书签的，则在下次读取文本信息的时候就会调用书签信息，实现书签功能。

7、unsigned long GetAudioFile(unsigned long AudioFileIndex)

8、unsigned long GetTextFile(unsigned long TextFileIndex)

这两个函数的原理是有些相同的，第一，先在根目录中找每一个目录项，如果说是音频文件且他的音频索引是 audiofileindex，那么这就是要找到的音频文件。同理第八个函数就是所要找到的文本文件。

对于第七个函数，首先，要记住来两个信息，1，记录当前目录项在整个根目录所有目录的偏置信息，2，记录当前目录项在整个音频文件目录项中的偏置位置。然后返回当前音频文件的起始簇号。

对于第八个函数，只用记录当前目录项在真整个根目录所有目录项中的偏置信息，然后返回当前文本文件的起始簇号。

9、unsigned long FindNextCluster(unsigned long ThisCluster)

10、 unsigned long FindPreviousCluster(unsigned long ThisCluster)

第九第十就是找一个文件中当前读的簇的下一个簇或是上一个簇，然后返回簇号

11、void GetDirItem(unsigned long DirItemIndex)

在根目录项中找到DIRitemindex 这个目录项,将这 32 字节值保存在全局数组DIRitem[32]中。这个函数主要是给 7 8 这两个函数调用用的，7,8 用这个函数找到所需要的目录项

12、void ReadSectorFromFile(unsigned long ThisCluster,unsigned long offset)

Long offset 是扇区在整个文件的偏置 根据这个偏置信息找到相应的扇区，读取相应的信息

13、void WriteSectorToFile(unsigned long ThisCluster,unsigned long offset)

这个就是 12 函数的一个写操作。。

15、void InitFAT()

初试化 FAT 就是调用 read MBR 和 read DBR 这两个函数

4.4 VS1003 音频解码芯片

VS1003 音频解码芯片是一块高性能的音频解码芯片。其支持的文件格式有 mp3，wav 等。它可以通过 SPI 总线与单片机进行数据通信。它内置了两个从机系统。第一个就是命令从机系统，在使用时，只要单片机将 VS1003 的 XDCS 信号线拉低，就选择了 VS1003 的命令从机系统，从而可以给 VS1003 发送各种命令，如进行软复位，调节音量等。第二个就是数据从机系统。在使用时，只要单片机将 VS1003 的 XCS 信号线拉低，就选择了 VS1003 的数据从机系统，然后就可以给 VS1003 发送音频数据。在发送时，VS1003 要求单片机连续不停地发送 32 个字节的数据。当 VS1003 接受到这 32 个数据后，它就会将 DREQ 信号线置为低电平， 以表示 VS1003 处于忙碌状态。这是单片机就要循环检测 VS1003 的 DREQ 的电平值，当 DREQ 被置为高电平时，表明 VS1003 做好了继续接收数据或命令的准备。

VS1003 的驱动如下：

1、void write_vs1003_commad(unsigned char addr,unsigned char hdat,unsigned char ldat)

该函数将 vs1003 的 xdcs 信号拉低，之后发送一个字节的写命令标志位，一个字节的地址，表明单片机所要入的命令寄存器，以及两个字节的命令值。

2、void InitVs1003(void)

该函数按照一定的次序初始化 vs1003。Vs1003 在初始化后就可以进行音频解码输出。

3、void WriteDataToVs1003()

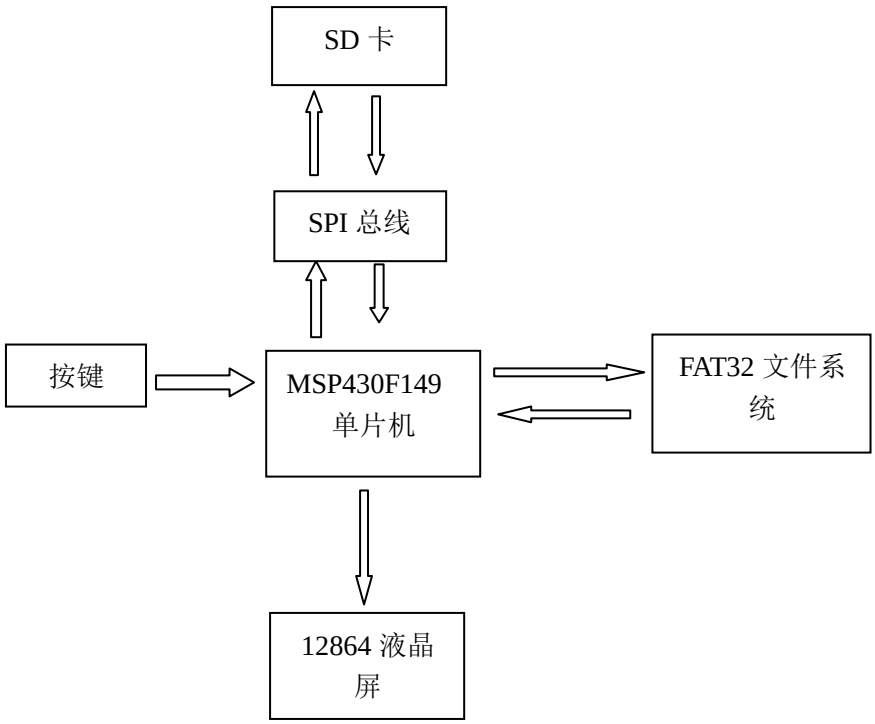
该函数将从 SD 卡内读仅 buffer 缓冲区内的一个扇区的数据连续发给 vs1003。在发送时，没发 32 字节的数据，就去循环检测 DREQ 的电平值。等到 DREQ 的电平值变高，就发送下 32 字节的数据。直到将一个扇区的数据发完为止。

4、void SetVolume()

该函数可以设定 VS1003 播放音乐时的音量。

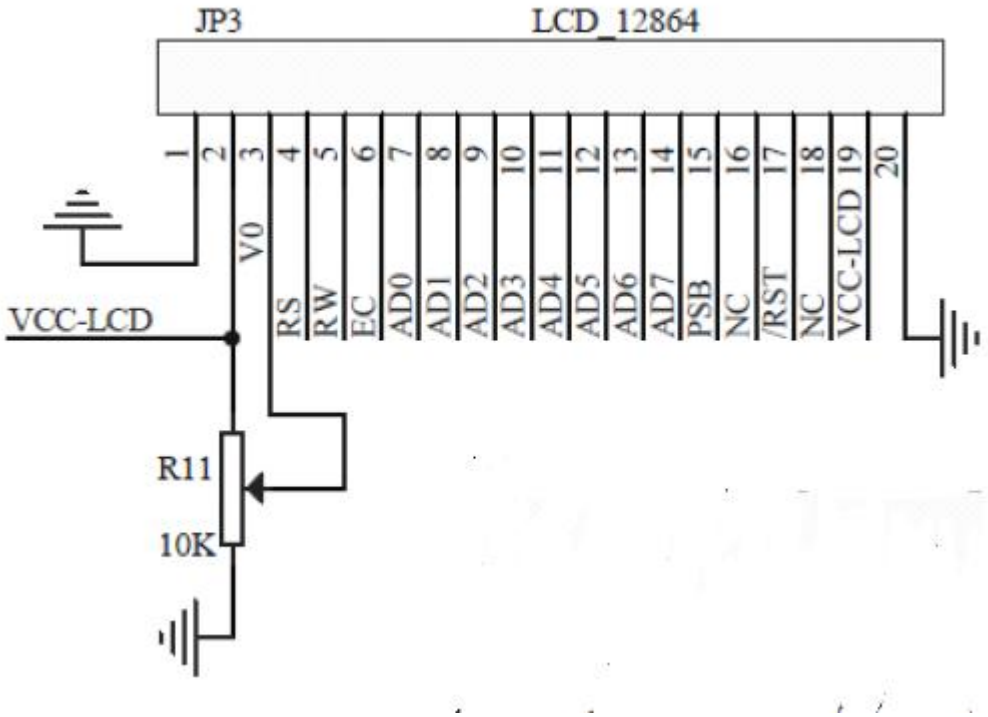
4.5 整机结构

课设的整机结构由 MSP430F149 开发板与 12864 液晶屏，VS1003 解码芯片，SD 卡及 SD 卡插座组成。整机结构图如下所示：



4.5.1 MSP430 开发板与 12864 液晶屏的连接

液晶接口原理图图下图所示：

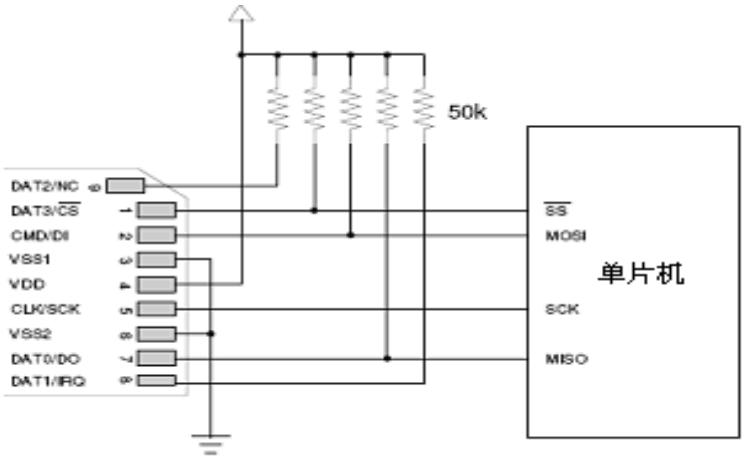


液晶模块端口分配资源如下表所示：

功能	RS	RW	EC	D0	D1	D2	D3	D4	D5	D6	D7	PSB	RST
端口	P55	P56	P57	P40	P41	P42	P43	P44	P45	P46	P47	P50	P51

4.5.2 MSP430 开发板与 SD 卡的连接

在 spi 模式下，SD 卡与单片机的连接原理图如下图所示：



MSP430 有两个 USART 端口，分别为 P3.1-P3.3 以及 P5.1-P5.3。课设中选用 P3.1-P3.3 端口来连接 SD 卡。其中 P3.1 为 SIMO 端，用来与 SD 卡在 SPI 模式下的 MOSI 端相连。P3.2 为 SOMI 端，用来与 SD 卡在 SPI 模式下的 MISO 端相连。而 P3.3 为 UCLK 端，用来与 SD 卡在 SPI 模式下的 SCK 端相连。使用 P3.0 为片选信号，并直接使用开发板上的 3.3V 电源向 SD 卡供电。

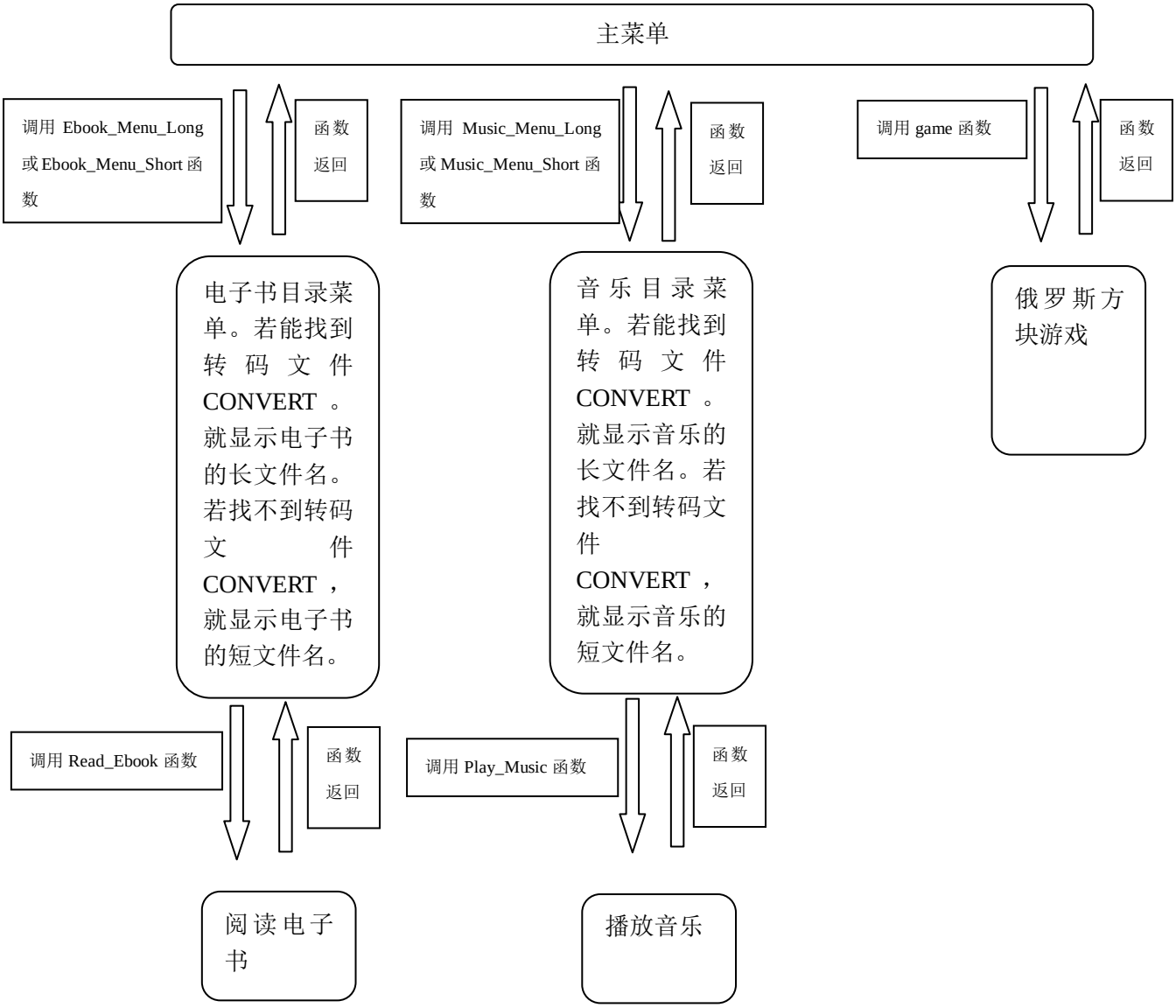
4.5.3 MSP430 开发板与 VS1003 芯片的连接

课设中使用 P2.4 与 vs1003 的 xcs 管脚相连，使用 P2.5 与 vs1003 的 dreq 管脚相连，使用 P2.6 与 xrest 管脚相连，使用 P2.7 与 xdcs 管脚相连。因为在课设中，SD 卡和 vs1003 共用一套 SPI 总线，因此 P3.1-P3.3 分别与 vs1003 的 MOSI，MISO，SCK 端相连。Vs1003 芯片直接使用开发板上得 5V 电源供电。

4.6 菜单功能

课设中总共有两级菜单，分为主菜单和目录菜单。其实现的机制就是函数之间的调用及返回。每一级菜单实际上都是一个循环运行的函数。每当从上一级菜单切换到下一级菜单时，就会在上一级菜单中发生一次函数调用，调用下一级菜单的运行函数，从而进入下一级菜单进行运行。而当从下一级菜单返回上一级菜单时，会发生函数的返回，从而从下一级菜单的运行函数返回上一级菜单的运行函数中运行。进入菜单函数时，首先要初始化菜单界面。而从下一级菜单返回上一级菜单时，要恢复上一级菜单的运行界面。

菜单功能的程序流程图如下所示：



4.7 电子书显示功能

4.7.1 实现显示功能的基本函数

整个显示的功能是构建在两个函数的基础上的，它们分别是：uint Fill_Line_Buffer(void)，和 void DisplayCgrom(uchar addr,uchar *hz)函数。

uint Fill_Line_Buffer(void)：该函数负责将文本文件中的内容填充到一个长为 17 的全局数组 unsigned char line_buffer[17]中去。每次向 line_buffer 中填充 16 个字节的内容，并向 line_buffer 的最后一个字节填充一个空字符。然后将这个 line_buffer 中的内容，利用 DisplayCgrom 显示到 12864 液晶屏指定的行号上。在上文中提到过，12864 液晶屏对汉字的显示位置有要求。汉字必须在向 12864 液晶屏输入相应的地址后再连续写入两字节的汉字编

码才能显示在屏幕上。因此若在向 12864 液晶屏内输入地址后，先写入一个字节的 ASCii 码后，发现从文本文件中读出来的是汉字编码，就必须立即输入一个字节的空格字符，然后等待 12864 发生地址自动偏移后，再连续写入两个字节的汉字字符。因此从文本文件中读出并写进 line_buffer 内的字节数不一定为 16。该函数就需要记录从文本中读出的字节数，并将其作为返回值返回。通过保存该值，程序就可以灵活的翻行翻页。

void DisplayCgrom(uchar addr,uchar *hz)：该函数就是负责将被填充好的 line_buffer 写到屏幕指定的行号上。

4.7.2 向下翻页功能

在课设中，我们的向上向下翻页功能并不是一次把一个全新的页面显示出来，而是会将当前页面的最后页面作为新页面的第一行，并将该行后面的三行内容显示出来。这个功能的实现就是借助了 Fill_Line_Buffer 的返回值。其执行过程如下：

1、每次显示一个页面时，都会执行一组这样的函数序列：

```
bytenum_of_each_line[0] = Fill_Line_Buffer();
DisplayCgrom(0x80,line_buffer);
bytenum_of_each_line[1] = Fill_Line_Buffer();
DisplayCgrom(0x90,line_buffer);
bytenum_of_each_line[2] = Fill_Line_Buffer();
DisplayCgrom(0x88,line_buffer);
bytenum_of_each_line[3] = Fill_Line_Buffer();
DisplayCgrom(0x98,line_buffer);
```

也就是说，当前屏幕中每一行所显示的字节数被保存了 bytenum_of_each_line 这个数组中。

2、当要向下翻页时，首先将指示缓冲区内下一个要被填充字节的指针 SD_buffer_position 的值减 bytenum_of_each_line[1]+bytenum_of_each_line[2]+bytenum_of_each_line[3]。若在减的过程中发生了扇区的切换，则从 SD 卡中把正在阅读的文件的当前的扇区的前一扇区读出来，将 SD_buffer_position 指向缓冲区最后一字节，并继续减去未减完的值。

3、再次执行显示一个页面的函数序列，就可以实现向下翻 3 行的功能。

4.7.3 向上翻页功能

向上翻页功能的实现比较复杂。首先我们向上翻页时也只是向上翻 3 行，和向下翻页相似。同时，为了保证出现在屏幕上的每一行的内容永远保持不变，就必须在向上翻页时寻找定位点。定位点实际上就是文本文件中的换行符 0x0D 和 0x0A，或者是文章的起点。定位点后面的字符肯定处在一行的最开始位置。有了定位点信息，就可以唯一的确定一行，然后再通过计算，就可以得到每一个行的内容。而且可以保证，每一行的内容在阅读的过程中均保持不变。其实现过程如下：

1、首先记录当前页面的第一个字节的位置。

2、然后从当前页面的第一个字节起向前找到定位点的位置，并让 SD_buffer_position 指向该位置。

3、然后不断利用 Pseudo_Fill_Line_Buffer 函数去假填行缓冲区。Pseudo_Fill_Line_Buffer 函

数的执行过程与 Fill_Line_Buffer 相同，只是它不填充行缓冲区 line_buffer，它用一个变量来伪记录行缓冲区的位置。
4、当某一次 Pseudo_Fill_Line_Buffer 函数执行完后，SD_buffer_position 的位置一定会回到第一步中记录的位置。这时记下 Pseudo_Fill_Line_Buffer 的返回值。
5、让 SD_buffer_positon 减去该返回值，并再执行两次 1-4 过程。然后从当前的 SD_buffer_position 处执行一次显示页面的函数序列。就可以实现想上翻三行的功能。

4.7.4 显示阅读时间

我们在课设中打开了 TimerA 的定时器中断，其中断发生周期为 10ms。设置了一个全局变量，循环加到一百归零。然后利用该变量改变秒钟。进而改变分钟和小时数。在阅读电子书时，只要按 S4 键，就可以再屏幕的后一行看到当前的阅读时间。

4.7.5 按百分比跳转

在阅读时，按 S4 键，就会打开一个新的页面。页面中会显示当前的阅读进度。这个进度是用当前簇偏置，当前簇内扇区偏置，及当前扇区内字节偏置和文件总长度计算出来的。然后按 S2 和 S3，就可以循环加这个阅读进度的十位和个位。当加到想要跳转的进度的时候，按 S4 就可以直接跳转到该进度进行执行。按百分比跳转的实现方法和向上翻页的实现方法相似。也是先找到跳转进度所指示的字节，然后向前找到定位点，之后用一个类似于 Pseudo_Fill_Line_Buffer 的方法来寻找跳转进度所指示的字节。找到该字节后，将 SD_buffer_position 的值减去组后一次伪填充所填充的字节数。然后执行显示一页的函数序列。

4.7.6 书签功能的实现

要实现书签功能，首先要在 SD 卡中创建一个文本文件，并将其命名为 DIRTABLE。只要系统在初始化时检测到该文件，就可以执行书签功能。
当从阅读电子书的状态退出进入目录菜单时，系统会提示读者是否要保存书签，如果要保存书签的话，就会保存当前阅读的进度。
当在此打开这本电子书时，系统会提示读者是否从书签处继续阅读，如果读者选择是，他就会立刻恢复上次阅读的阅读进度信息。
在保存阅读进度前，首先会将 SD_buffer_position 减去 1-4 行所填充的字节数，然后再将当前 SD_buffer_position 所对应的簇内扇区偏移，簇号，簇便宜，及扇区内 SD_buffer_position 相对于 SD_buffer 的偏移记录在 DIRTABLE 文件中。
阅读进度信息由 32 个字节组成，其内容如下：
1、第 1-8 字节记录当前阅读的电子书的短文件名目录项的前 8 字节，及其短文件名。
2、第 9-12 字节记录当前阅读的簇的簇内扇区偏移。
3、第 13-14 字节记录当前所阅读的扇区内的 SD_buffer_position 相对于 SD_buffer 的偏移。

4、第 15-18 字节记录当前所阅读的簇的簇号。

5、第 19-22 字节记录当前所阅读的簇是整个文件的第几个簇。

6、第 23-32 字节保留但不使用。

假设当前阅读的电子书的短文件名目录项在根目录中的索引号是 X。则阅读进度信息就会被存储在 DIRTABLE 的 32*X 到 32*X+31 这 32 个字节。

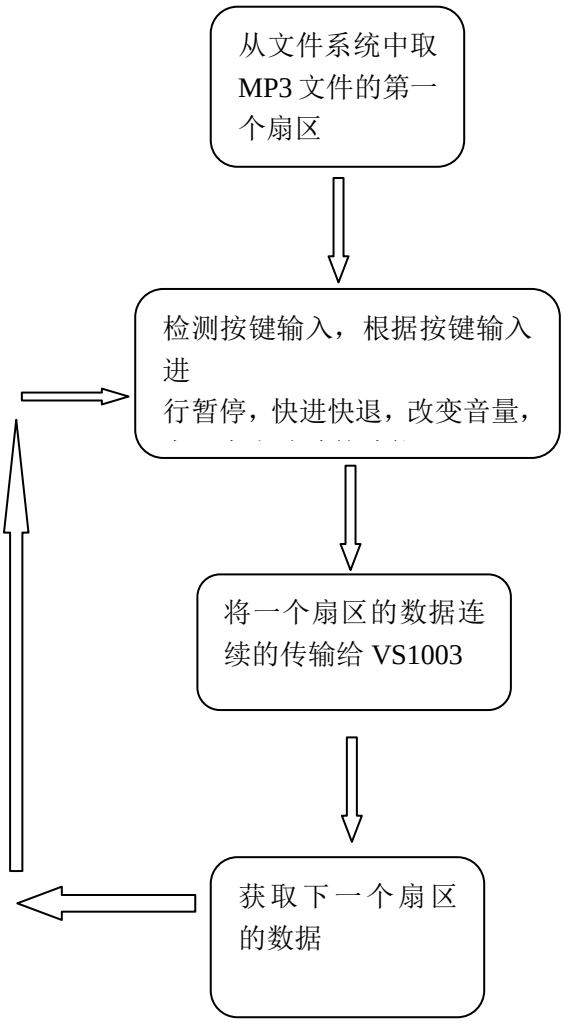
当再次进入电子书阅读时，若用户选择从书签处继续阅读，则系统就在 DIRTABLE 中寻找到这本电子书的书签，然后恢复其中所保存的进度信息，之后就可以从上次退出阅读的页面继续阅读了。

4.7.7 英文单词的人性化显示

如果在填充行缓冲区时，行缓冲区已经被填充到了第 16 字节，然后发现即将填入的字节是一个 ASCii 字符，同时即将填入字节的下一个字节也是 ASCii 字符。这时就会调用 AdjustWord 函数，以防止长度小于 16 个英文字母的单词被两行截断。

4.8 音乐播放功能

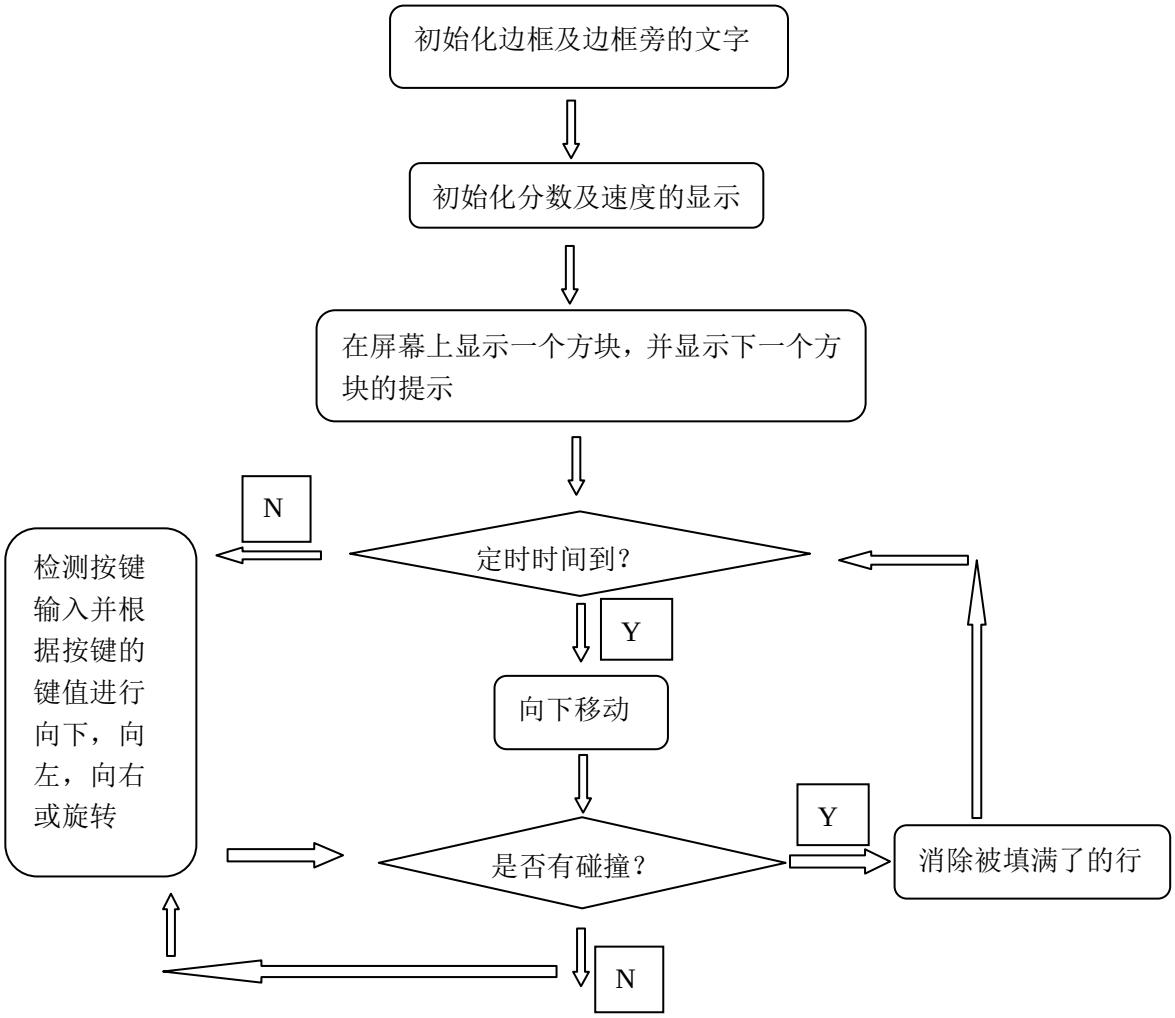
音乐播放功能实际上就是在音乐菜单目录函数中调用函数 Play_Music()。该函数的流程如下图所示：



4.9 俄罗斯方块游戏功能

本课设利用 12864 液晶屏的绘图功能实现了俄罗斯方块游戏的功能。该游戏时先初始化绘图模式，然后通过在主菜单函数 Main_Menu 中调用 game()函数来实现的。

俄罗斯方块的基本流程如下图所示：



俄罗斯方块功能的代码分析如下：

1、typedef struct

{

unsigned char * box;

unsigned char cube;

unsigned char state;

signed char row;

signed char column;

}block

这是方块的数据结构的定义，它又记录方块的形状的数组 box，方块形状的编号 cube，方块的旋转方向 state，以及方块形状数组第 0 行，第 0 列在整个游戏位图中的行数 row 和列数 column 组成。

2、unsigned char cubeMap[MAXROW][2]

这是游戏的位图，整个游戏界面就保存在这个味图中。游戏界面为 20 行 10 列。因此 MAXROW 被定义为 20，同时表示一行的两个字节值用到了第一字节和第二字节的高 2 位。

3、unsigned char cube[]

这是保存游戏每一个方块形状位图的数组。每一个方块的形状位图由连续四个字节的低四位组成。每一个方块具有 4 个方向，同一个方块的四个方向的位图被连续保存。因此只要给出方块编号 cube，再给出方块的旋转方向 state，就可以通过 4*cube+state 找出该方块的形状位图。

4、unsigned char asii[]

这是保存游戏中显示的英文字母和数字的位图。每一个英文字母或数字由连续的 5 个字节组成位图、

5、unsigned char downok = 0;

unsigned int score=0;

unsigned char speed=1;

unsigned long downtime = 100;

unsigned char next;

这是 5 个全局变量。Downok 表示着方块在下落过程中触底，无法继续下落。Score 表示当前游戏的得分，speed 表示当前游戏速度。Downtime 表示方块隔 100 个时钟中断，即 1 秒钟就会强制一格。Next 表示下一个方块的形状编号。

6、void Put_Double_Pixel(unsigned char x, unsigned char y)

因为在游戏中，每一个方块的每一个小格用两行两列的点阵来表示，因此这个函数的作用就是在游戏界面相对位置的第 x 行，第 y 列显示一个方格。

7、void Clear_Double_Pixel(unsigned char x, unsigned char y)

这个函数和上一个函数作用相似，只是它的作用是消除一个方格。

8、void showChar(unsigned char num, unsigned char x, unsigned char y)

将 ascii 数组中保存的字符位图显示在屏幕第 x 行，第 y 列开始的位置上。

9、void Game_Init()

该函数先打印游戏界面的边框，然后显示 next，speed，score。

10、void Show_Speed_Score()

这个函数会将当前的得分和速度显示在屏幕上。

11、void createCube(unsigned char num)

当 downok 被置 1 时，调用这个函数来重新创建一个方块。

12、void showCubeMap(void)

当消掉行时，调用这个函数来在屏幕上刷新位图中的内容。

13、void writeCubeToMap(void)

每当向左，向右，向下移动方块或旋转方块后，最后都通过该函数将移动后的方块显示在屏幕上，及写入位图中。

14、void clearCubeFromMap(void)

每当向左，向右，向下移动方块或旋转方块前，最后都通过该函数将移动后的方块在屏幕上清除，及在位图中清除。

15、unsigned char checkBorder(void)

检测当前的方块是否碰到边框。如果碰到左边框，就不能向左移动，如果碰到右边框，就不能向右移动。如果触底，就触发 downok。

16、unsigned char checkClask(void)

检测当前的方块是否碰到其他方块.

17、void checkMap(void)

检测位图中是否有被填满的行，如果有就消掉该行，并且加响应的分，并根据当前分数调整速度。

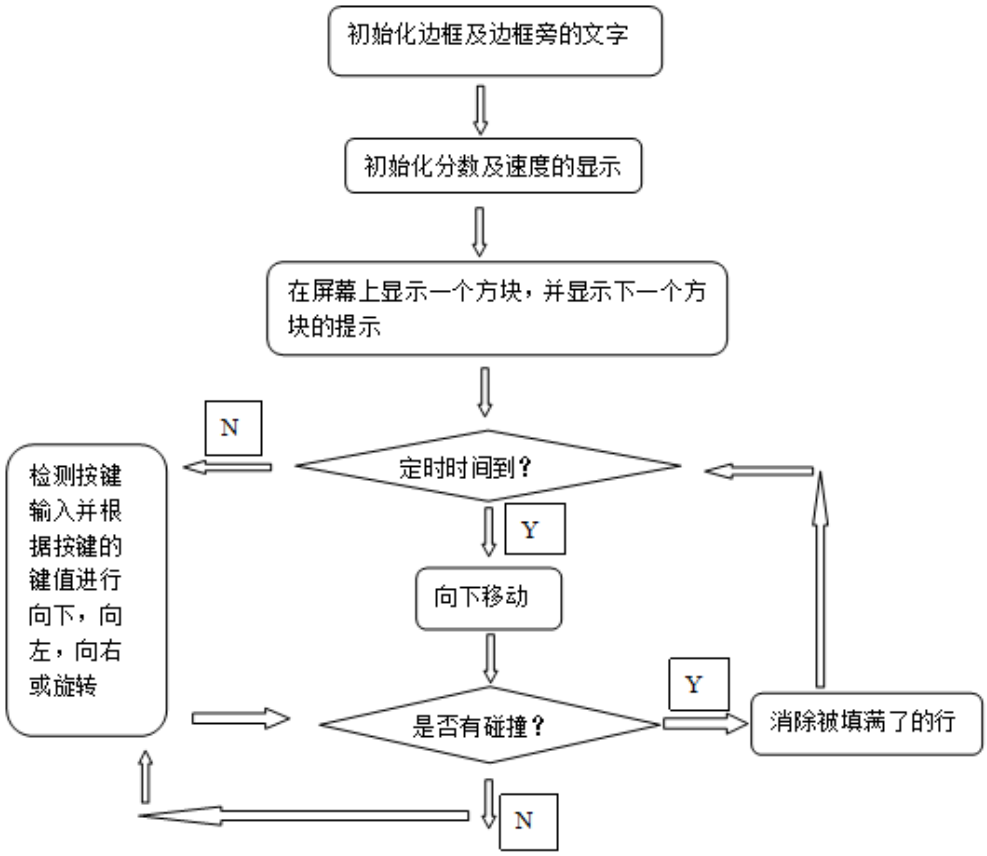
18、void moveRight(void)　void moveLeft(void)

void moveDown(void)　　void cubeRotation(void)

这四个函数分别表示向左，向右，向下移动方块及旋转方块。在这些函数中，先移动方块，再检测是否碰到边框或者碰到其他方块。若没碰到，就退出函数。若碰到了，就返回原来的位置。对于 moveDown。若在该函数中检测到了碰到边框或碰到其他方块，就会置位 downok 标识。

19、void game(void)

该函数的流程就如一开始给出的程序流程图。



4.10 按键检测

本课设中利用定时器中断检测按键。在定时器中断函数，添加了一个有限状态机，可以判

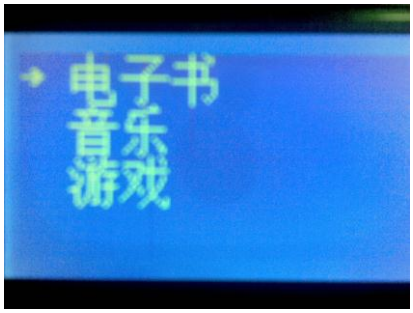
别短按键和长按键。当发生短按键时，置位短按键标志 spress_flag，并给出键值 key。当发生长按键时，置位长按键标志位 lpress_flag。并给出按键值 key。这样做使得按键检测非常灵敏，并且可以利用长短按键的组合产生各种功能，节省按键数量。

5. 系统测试

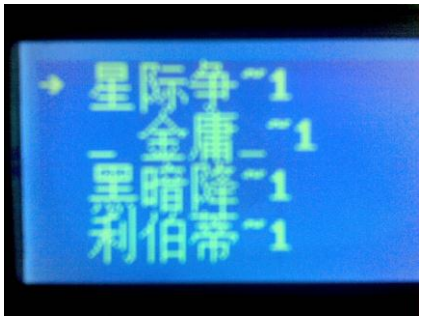
5.1 菜单功能的测试

来回进入退出各级菜单，来回切换流畅，进入下一级菜单时可正常初始化下一级菜单的内容，返回上一级菜单时可以恢复上一级菜单的内容。菜单功能的截图如下所示：

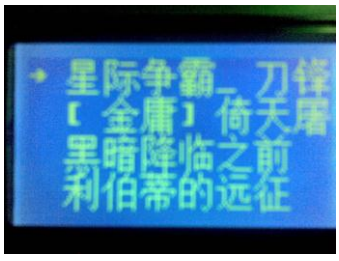
1、主菜单：



2、短文件名电子书目录菜单



3、长文件名电子书目录菜单



4、长文件名音乐目录菜单：

5.2 电子书阅读功能测试

测试时，在 SD 卡中放了以下文件：

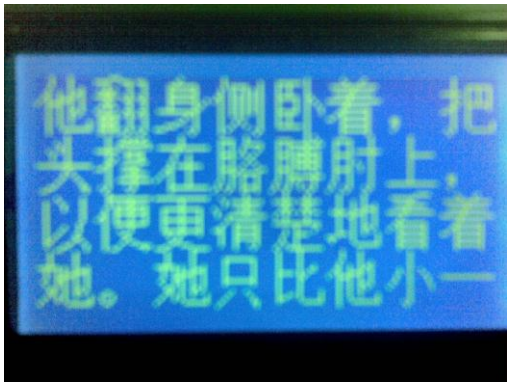
[金庸]倚天屠龙记.txt	2012/7/3 20:08	文本文档	2,013 KB
cnnarticle.txt	2012/7/2 18:52	文本文档	8 KB
CONVERT.txt	2009/3/7 14:51	文本文档	41 KB
DIRTABLE.txt	2012/7/2 20:17	文本文档	18 KB
longwordtest.txt	2012/7/3 20:11	文本文档	1 KB
黑暗降临之前.txt	2012/2/10 16:46	文本文档	195 KB
利伯蒂的远征.txt	2012/2/10 16:45	文本文档	249 KB
新修版_天龙八部.txt	2012/7/3 20:09	文本文档	2,465 KB
星际争霸_刀锋女王(完整版).txt	2012/2/10 16:44	文本文档	280 KB
这是个空文件.txt	2012/7/3 20:12	文本文档	0 KB

可以看到，这些文件的文件长度还是相当长的。其中 CONVERT 和 DIRTABLE 使用来实现长文件名显示和书签功能的。而 cnnarticle 和 longwordtest 是用来检测英文单词人性化显示的。这是个空文件是为了检测电子书在读到空文件时的响应的。

在测试时，所有电子书都能正常打开，快速向下向上翻页，按任意百分比进行跳转，以及保存书签和从书签处继续阅读。

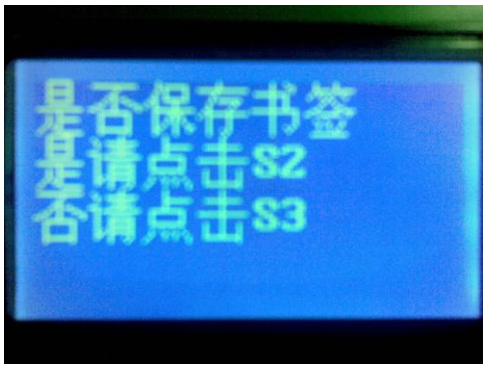
下面给出电子书阅读时的功能截图：

1、文本的显示：



2、跳转界面及阅读时间的显示：

3、书签功能：



4、英文单词的人性化显示：

5.3 音乐播放功能测试

测试时在 SD 卡里放了三首歌，分别是 Airplanes，Flexin，以及 Viva la vida。这三首歌均可以清晰流畅的播放，并且可以快进快退，暂停，向上向下，及调整音量。下面给出音乐播放时的截图：

5.4 游戏功能的测试

游戏功能可以正常初始化，方块可以正常下落，左移右移及旋转。可以正常的清楚被填满的行。当方块触顶时，会游戏结束而退出游戏。

下面给出游戏过程中的截图：

6. 心得体会

我们组的课设可以说做了很久，而且从接到这个题目开始，就一直在很认真的做，从来没有马虎。从一开始的查阅资料，到后来的购买器件，再到编码及调试。我们都付出了大量的心血。当然最后做出的东西也让我们感到满意。诚然我们的课设还有很多的小问题没有解决，它还没有做到完美。但时间有限，精力有限，做到现在这个程度我们已经非常非常满意了。

通过这次课设，我们真的深深体会到做好一个产品的不易。从前期的设计，到中期的编码，再到后期的调试，如果没有缜密的思考以及计划，就会胡乱不堪，毛病不断。这次课设真的让我们成长了很多，学习到了很多！

7. 参考文献

【1】MSP430x1xx User Guide
【2】12864 液晶屏中文资料
【3】FAT32 文件系统详解
【4】SD 卡在单片机上的应用
【5】基于 MSP430 单片机的菜单式 MP3 的实现