# 华中科技大学电子与信息工程系
# 2012 年 TI 杯电子设计大赛项目总结报告

项目名称：电子书阅读器

团队成员：提高 0901 班李剑
提高 0901 班夏煜
通信 0901 班黄振清

指导教师: 曾喻江

2012 年 7 月 3 日

# 课题名称：电子书阅读器

【摘要】本次课程设计以 MSP430F149 开发板为基础，从 SD 卡读取 TXT 文件，经过 MSP430F149 MCU 处理，在 LCD12864 上显示，并通过开发板上的 key 模块按键实现对阅读器控制的电子书阅读器。

电子书要实现以下几个基本功能：

1. 从 SD 卡上读取文本文档，在 LCD 显示器上滚动显示

2. 通过按键实现手动翻页与自动翻页的模式转换

3. 自动翻页模式下通过按键改变翻页的速度

【关键字】读取，显示，翻页

【Abstract】his course design with MSP430F149 development board as the foundation, sd card from reading TXT, after MSP430F149 MCU processing, on display on LCD12864, and through the development of key module on board to realize the control of the key reader ebook reader.

To achieve the following basic ebook function:

1. The SD card to read text from the document, in the LCD monitor displayed on:

2. Through the button to scroll achieve manually and automatic scroll mode conversion

3. Auto flip mode button to scroll through the speed of change

【Key word】 read, showed that content

# 目录

# 1. 概述

图书电子化已成为主流趋势。近日美国加州州长施瓦辛格宣布加州取消课本，今年秋季起成为全球第一个教材全部电子化的地区。可以想象，如果未来电子书技术稳定，成本低廉，便于携带和交流，肯定会取代纸质图书，成为人们阅读的主要媒介。

目前，便携式的电子书阅读设备已经比较普遍，智能手机，PDA，MID 和各种笔记本电脑都可以很好的支持多种格式的电子书籍。但是上述设备的机能强大，如果仅作为电子书阅读器来使用有些大材小用，而且价格不菲。前年上市的 Amazon Kindle 是一台专注于电子书阅读的设备，可是其推广和销售都没有较大成功，原因有两点：1.采用不成熟的 E-ink 技术，成本高，性价比极低。2. 与 Amazon 捆绑，过分的版权设定，通用性差。而这，正是我们想要解决的问题。

本报告以下的内容将会按照以下结构来组织：在第二小节中我们将会介绍设计的目标；第三小节中，我们将会介绍组员分工情况；第四小节的内容是系统整体功能的介绍；第五小节里，每一个小模块的软硬件设计与性能将会被详细介绍；我们系统整机测试的过程与结果将会展现在第六小节。

# 2. 设计目标

- 从 SD 卡上读取文本文档，在 LCD 显示器上滚动显示：
- 通过按键实现手动翻页与自动翻页的模式转换
- 自动翻页模式下通过按键改变翻页的速度

# 3. 团队组成与任务分工

由于课程设计时间较短，为了提高效率，我们小组将整个项目划分为不同模块，同模块之间首先需要将相互之间的接口定义好，定义完成以后不同模块就能够相对地独立工作了,每人一部分同时进行，到最后再合并调试的方式进行。具体分分工如下：

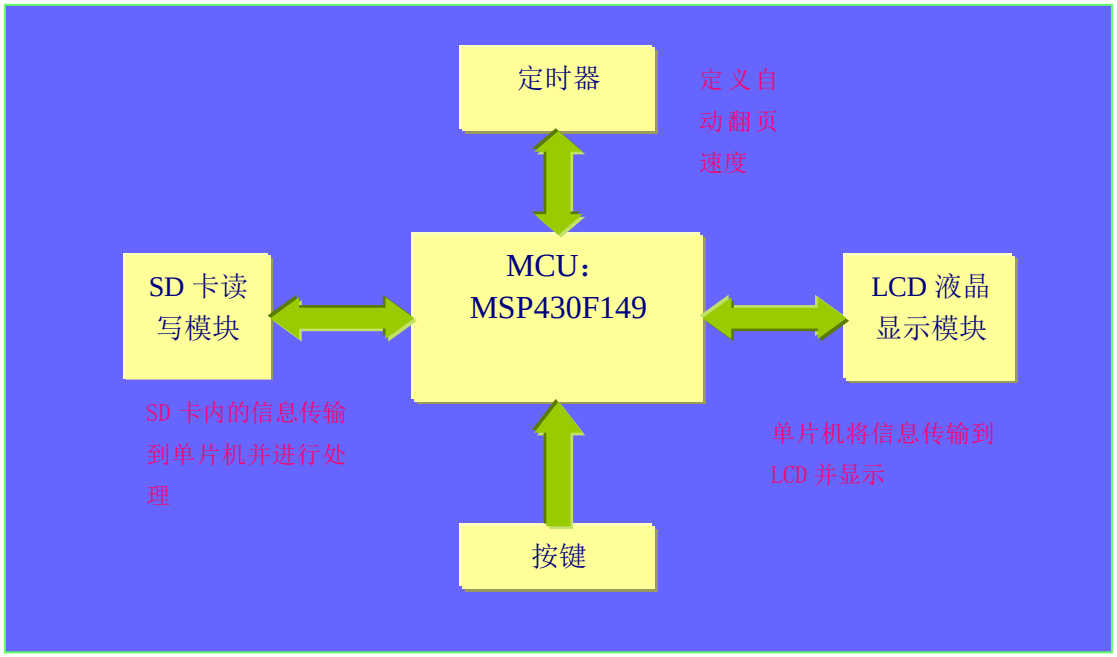
李剑 负责 SD 卡读取 TXT 文档的电路与程序的设计与调试，周工作报告的总结；

夏煜 负责 LCD12864 液晶显示字符显示模块，进行元器件采购；

黄振清 负责 TI 文档的制作，协助 SD 读取 TXT 文档的制作。

# 4. 系统的总体设计

## 4.1 总体设计框图




## 4.2 主要器件选择与方案论证

电子书阅读器的硬件设计可以大体分为三部分：存储部分，显示部分和控制部分。

当前移动设备的存储介质种类非常丰富。小型设备（如手机 mp3 GPS 设备等）上闪存式存储占据上风，而对存储容量要求较大的移动设备上，传统机械式硬盘仍占据主导地位。固态硬盘(SSD)近年来发展迅速，未来很可能取代两者成为兼顾性能，稳定性，移动性和性价比的主导存储设备。

显示设备当前市场上的技术主要有以下三种：1.LCD 液晶显示特别适合作为移动设备的显示模块。相关技术比较成熟，成本较低，是市场上的绝对主流。2. LED 发光二极管显示近年来得到了突破性的进展，其超低能耗，超长寿命的特点决定其将取代液晶显示成为下一代主流。不过目前制作成本仍然较高，并且发展存在很多不确定性，短期内不会取代 LCD 的主流地位。 3.E-INK 作为一种创新型的显示技术， “电子墨”技术被人们寄予厚望，并在 SONY reader, Amazon Kindle 等产品上得到了效果不凡的实际应用，不过受困于过高的成本和显示的稳定性，该技术成熟还需要较长的发展时间。

控制技术的发展丰富多样，触控技术随着微软，苹果等巨头的引导成为新主流，不过传统的机械键盘仍占据绝大市场份额。

权衡成本，通用性，技术成熟性和制作的复杂程度之后，我们决定选用闪存技术的 SD 卡作为存储媒介，点阵式 LCD 作为显示器，用机械按键来进行操作控制。

## 4.2.1 SD 卡

SD 卡是一种基于半导体快闪记忆器的新一代记忆设备，它被广泛地于便携式装置上使用，有高记忆容量、快速数据传输率、极大的移动灵活性以及很好的安全性等优点。SD 卡共支持三种传输模式：SPI 模式（独立序列输入和序列输出），1 位 SD 模式 （独立指令和数据通道，独有的传输格式）， 4 位 SD 模式 （使用额外的针脚以及某些重新设置的针脚。支持四位宽的并行传输），在此我们选用 SPI 串行模式。

SD 卡端口示意图如下

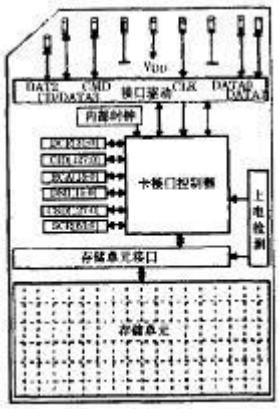


图 2 SD 卡的内部结构

连接电路如下

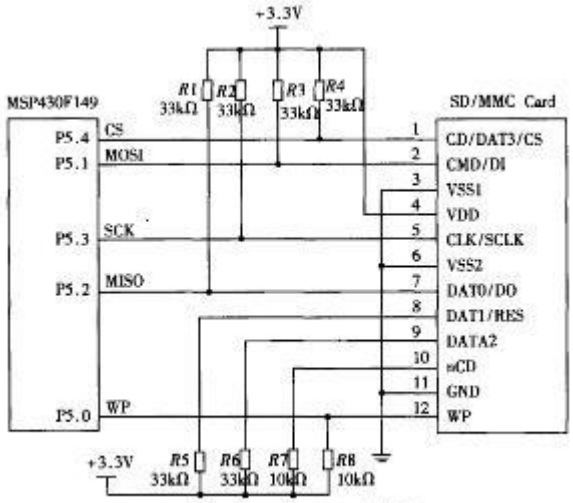



图 3 MSP430F149 与 SD 卡座的电路连接图

SD 卡的工作电压为 3.3V，直接用单片机的 5V 供电会烧坏 SD 卡，所以必须使用变压电路，我们所用的 MSP430F149 开发板上自带电源电路，可以满足以上需求。

## 4.2.2 LCD

我们选用的是 128X64 的图形点阵液晶显示器 12864ZW。它主要由行驱动器/列驱动器及 128X64 全点阵液晶显示器组成，可完成图形显示，也可以显示 8X4 个（16X16 点阵汉字），与外部 CPU 接口可采用串行或并行方式控制,这满足了电子书阅读器的显示需求。

（1）12864zw 的各种规格

| 项目 | 参考值 |
|---|---|
| LCM 尺寸（长×宽×厚） | 93.0×70.0×13.5 |
| 可视区域（长×宽） | 72.0×40.0 |
| 点间距（长×宽） | 0.52×0.52 |
| 点尺寸（长×宽 | 0.48×0.48 |
| 逻辑工作电压（Vdd | +5.0V 或+3.3V（出厂时设定+5.0V） |
| LCD 驱动电压（Vdd-V0） | +3.0 ~ +5.0V |
| 工作温度（Ta） | 0 ~ +50℃（常温）/ -20 ~ +70℃（宽温） |
| 储存温度(Tsto) | -10 ~ +60℃（常温）/ -30 ~ +80℃（宽温） |
| 工作电流（背光除外） | 3.0mA(max) |

（2）12864ZW 外部接口

| | | | |
|---|---|---|---|
| 1 | VSS | -- | 电源负端(0V) |
| 2 | VDD | -- | 电源正端(+3.3V 或+5.0V，出厂时设定+5.0V) |
| 3 | V0 | -- | LCD 驱动电压(可调) |
| 4 | RS(CS) | I | RS=0:<br>当MPU 进行读模块操作，指向地址计数器。<br>当MPU 进行写模块操作，指向指令寄存器。<br>RS=1:<br>无论MPU 读/写操作，均指向数据寄存器。<br>串口方式:<br>CS: 串行片选信号，高电平有效。 |
| 5 | R/W(SID) | I | 并口方式:<br>R/W=0 写操作。<br>R/W=1 读操作。<br>串口方式:<br>串行数据输入端 |
| 6 | E(SCLK) | I | 并口方式：使能信号，高电平有效。<br>串口方式：串行时钟信号。 |
| 7_14 | DB0~DB7 | I/O | MPU与模块之间并口的数据传送通道，<br>4 位总线模式下D0 ~ D3脚断开 |
| 15 | PSB | I | 串/并口控制选择端:<br>H: 并口控制;<br>L: 串口控制。 |
| 16 | NC | -- | 空脚 |
| 17 | RST | I | 复位脚（低电平有效） |
| 18 | VOUT | -- | 倍压输出脚。（VDD=+3.3V 时有效） |
| 19 | LEDA | -- | 背光电源正端(+3.3V 或+5.0V，出厂时设定+5.0V) |
| 20 | LEDK | -- | 背光电源负端(0V) |

### 4.2.3 按键

我们选用的是开发板上已经焊接好的按键。

## 4.3 主要元器件清单

MSP430F149 开发板一块
SD 卡及卡槽一套
12864ZW 液晶显示屏一块

# 5. 软硬件设计与实现

## 5.1 SD 卡 TXT 文档读取模块

### 5.1.1 综述

用单片机读取 SD 卡中的 txt 文档可以分为 3 步：
第一步：建立单片机与 SD 卡的 SPI 通信
第二步：建立 FAT32 文件系统，提供应用层需要的 API 函数
第三步：通过 FAT32 文件系统提供的 API 函数读写文档，管理文件

### 5.1.2 硬件电路设计

SD 卡与 F149 单片机的硬件连接图非常简单，有需要将 F149 的 SPI 控制端口 P3 与 SD 卡相应端口连接即可。当然 SD 卡要与 SD 卡座相连。

| | |
|---|---|
| P3.0 | CS |
| P3.1 | MOSI |
| P3.2 | MISO |
| P3.3 | SCK |
| Vcc | Vcc |
| GND | GND |

注：SD 卡座有 8 个接口，上面用到了 6 个，还有一个 GND 和 5V 电源可以不接。

### 5.1.3 单片机读取 SD 卡 txt 文档的软件设计

（1）F149 单片机与 SD 卡建立 SPI 通信

1） 读写字节

SPI 模式是一种简单的命令响应协议，单片机发出命令后，SD 卡针返回对应的响应。要实现单片机与SD的SPI通信模式，首先要能实现单片机与SD卡能互相收发一个字节的二进制数据，如果单片机不自带SPI控制器，就只能用IO端口模拟SPI控制器，网上这方面的资料很多，可以借鉴，具体实现也不是很简单，由于我们的F149单片机上集成了SD卡座，但是是与通用IO口P6相连，不具备SPI控制功能，所以想模拟SPI，尝试后发现困难重重，最后还是用了P3端口的SPI控制器。利用自带SPI 硬件接口，因此只需要读写SPI 数据寄存器的值

就可以完成收发1Byte的工作。

**写一个字节：**

```
void WriteSpiByte(uchar byte)                    //write one byte to spi port
{
    TXBUF0 = byte;
    while (!(IFG1 & UTXIFG0));                   // USCI_A0 TX buffer ready?
}
```

**读一个字节：**

```
uchar ReadSpiByte()                              //read one byte from spi port
{
    unsigned char temp=0;
    while (!(IFG1 & UTXIFG0));                   // USCI_A0 TX buffer ready?
    TXBUF0 =0xff;
    while ((IFG1 & URXIFG0)==0);
    temp=RXBUF0;
    return temp;
}
```

2）发送命令

完成了SD卡与单片机收发字节数据的功能后，单片机就可以通过发命令来与SD卡真正实现交流。

首先介绍SD卡的命令格式：

| 第 1 字节 | | | 第 2～5 字节 | 第 6 字节 | |
|---|---|---|---|---|---|
| 0 | 1 | 命令号 | 参数 | CRC 校验 | 1 |

在SPI 模式中，命令都是以如上的6字节形式发送的，每帧命令都以“01”开头，然后是6位命令号和4字节的参数（高位在前，低位在后），最后是7位CRC 校验和1位停止位“1”。如CMD1，

他的命令号为1，第一个字节就是01000001，也就是0x41，同时也可以看出该模式下最多只能定义64个命令。SD 卡的每条命令都会返回对应的响应类型。在SPI 模式下，共有3种响应类型：R1、R2和R3，分别占1、2和3个字节。这里仅列出了R1响应的格式，如下表所示。当出现表中

所描述的状态时，相应的位置1。R2和R3的第1个字节格式与R1完全一样。

**表 2　R1 响应格式**

| 位 | 描　述 |
|---|---|
| 7 | 起始位，0 |
| 6 | 参数错误 |
| 5 | 地址错误 |
| 4 | 擦除顺序错误 |
| 3 | CRC 错误 |
| 2 | 非法命令 |
| 1 | 擦除复位 |
| 0 | 空闲状态 |

由于SD 卡的命令具有统一的格式，因此可以用一个通用的写命令函数来实现所有命令的发送。另外，考虑到多数命令的响应类型都是R1，这里的通用写命令函数所接收的响应，类型默认为R1。具体实现步骤如下：

先等待8个时钟周期，然后写入命令号，这是第一个字节，然后发送4字节的命令参数，最后是CRC校验和接受响应。代码如下：

```
byte SD_SendCommand_R1(byte cmd, dword arg) {
    byte i,r1;
    SPI_WriteByte(0xff);                    //等待几个时钟周期
    SPI_WriteByte((byte)(cmd|0x40));        //写入命令号
    SPI_WriteByte((byte)(arg>>24));         //4 字节命令参数
    SPI_WriteByte((byte)(arg>>16));
    SPI_WriteByte((byte)(arg>>8));
    SPI_WriteByte((byte)(arg));
    SPI_WriteByte((byte)(cmd == 0x00? 0x95 : 0xff));
                                            //CRC 校验和
    for(i = 0; i < 10; ++i) {               //接收响应
      r1 = SPI_ReadByte();
      if(r1 != 0xff) break;
    }
    return r1;
}
```

3）SD卡初始化

单片机能发送命令后就可以实现对SD卡的初始化了，SD 卡的初始化要遵循一定的步骤。首先将SPI 时钟降低到400 kHz，等待至少74个时钟周期。接着拉低片选信号，并发送CMD0命令，对SD 卡进行复位并使其进入SPI 模式，这里需要正确的CRC 校验，校验字节为0x95。若SD 卡进入空闲状态（即接收响应为0x01时），则发送CMD1命令，激活卡的初始化过程，此时响应为0x00。然后设置块的长度，一般为512字节。最后将片选拉高并将SPI 时钟设为最大值，以保证最大的读写速度。

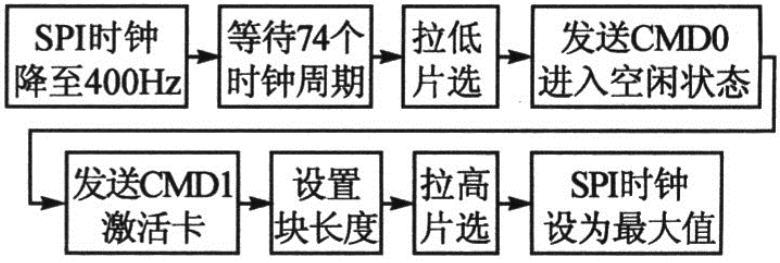



图 3　SD 卡初始化过程

后来发现初始化开始时并不需要降低时钟频率，只需要延迟1ms左右再拉低片选，发送

CMD0，不需要改变时钟频率，因为降低频率等待74周期的目的还是延迟。具体代码参照图3步骤写即可。

4）SD卡单块数据读写

SD卡初始化后就可以读写单块数据了，一般SD卡单块数据位512字节。

读单块数据：

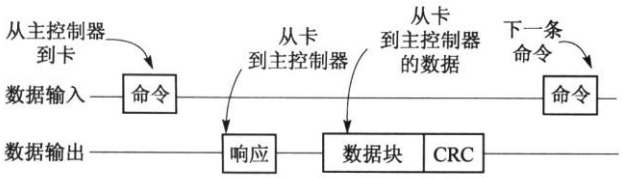



图 4　读单块数据操作

拉低片选CS后单片机发送读单块数据命令CMD17，然后等待SD 卡的响应。当收到数据块开始标志0xfe 后，开始从SD 卡读取512字节的数据，最后读取2字节的CRC 校验位。具体代码参照图4步骤写即可。

写单块数据：

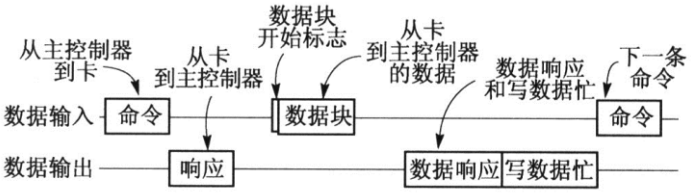



图 5　写单块数据操作

拉低片选后同样由单片机发送写单块数据命令CMD24，SD卡正确响应后发送数据块开始标志0xfe，接着发送512字节数据块和2字节CRC校验。具体代码残周图5的步骤进行即可。至此，SD卡与单片机的SPI通信就完成建立好了，接下来就可以进入上一层的文件系统的建立了。

（2）FAT32文件系统的建立

建立了单片机与SD卡的SPI通信只是建好了最底层的通信，此时单片机还不认识SD卡里面各式各样格式的文件，更别谈读取SD卡里面的txt文档了，要实现对这些文档的读取，就要建立文件系统实现对文档的管理。

FAT32文件系统的SD卡整体布局：


由整体布局可以看出，要建立FAT32，必须先读取MBR和DBR这两个非常重要的引导扇区的数据，只有读取了这两个扇区的数据才能进行FAT32文件系统的初始化。

1）读取MBR

MBR由446个字节的引导代码、64字节的主分区（4个）表及两个字节的签名值“55 AA”组成。

64字节的主分区表使我们要读取的重点数据，其数据结构如下所示：

| 偏移（十六进制） | 字节数 | 描述 |
|---|---|---|
| 00~00 | 1 | 可引导标志，0x00不可引导，0x80可引导 |
| 01~03 | 3 | 分区起始CHS地址 |
| 04~04 | 1 | 分区类型 |

| | | |
|---|---|---|
| 05~07 | 3 | 分区结束CHS地址 |
| 08~0B | 4 | 分区起始LBA地址（Little-endian顺序） |
| 0C~0F | 4 | 分区大小扇区数（Little-endian顺序） |

软件实现过程是：先判断最后两个字节是不是55aa,是的话就判断第446个字节的值，如果为0x80(可引导)或者0x00(不可引导)就读取64字节的数据，这些数据对FAT32文件系统非常重要，前面的445字节数据可以不管，对SD卡来说没什么用。具体代码如下所示：

```
void ReadMBR()
{
    ReadMmcSector(0,512,buffer);
    if(buffer[0x1fe]==0x55 && buffer[0x1ff]==0xaa)
    {
        if( buffer[0x01be]==0x80 || buffer[0x01be]==0x00 )
        {
            StartingSector=(unsigned long)buffer[0x1C6]
                |((unsigned long)buffer[0x1C7])<<8
                |((unsigned long)buffer[0x1C8])<<16
                |((unsigned long)buffer[0x1C9])<<24;
            TotalSectors=(unsigned long)buffer[0x1CA]
                |((unsigned long)buffer[0x1CB])<<8
                |((unsigned long)buffer[0x1CC])<<16
                |((unsigned long)buffer[0x1CB])<<24;
        }
    }
```

2）读取DBR

DBR引导扇区是FAT32文件系统的第一个扇区,它包含FAT32文件系统的基本信息：

【1】 每扇区字节数

【2】 每簇扇区数

【3】 保留扇区数

【4】 FAT表个数

【5】 文件系统大小（扇区数）

【6】 每个FAT表大小（扇区数）

【7】 根目录起始簇号

这些基本信息是实现对文件操作的基础，DBR512字节的数据都非常重要，故每个字节数据都要读取。具体代码如下

```
void ReadDBR()
{
    unsigned int i=0;
    if(StartingSector)
        ReadMmcSector(StartingSector,512,buffer);
    else
    {
        for(i=0;i<1000;i++)
```

```
        {
            ReadMmcSector(StartingSector,512,buffer);
            if(buffer[0x0]==0xEB & buffer[0x1fe]==0x55 &
buffer[0x01ff]==0xAA)
                break;
        }
    }
    if(buffer[0x0]==0xEB & buffer[0x1fe]==0x55 & buffer[0x01ff]==0xAA)
    {
        BytesPerSector=(unsigned int)buffer[0x0B]
            |((unsigned int)buffer[0x0C])<<8;
        SectorsPerCluster=buffer[0x0D];
        ReservedSector=(unsigned int)buffer[0x0e]
            |((unsigned int)buffer[0x0f])<<8;
        NumberofFAT =buffer[0x10];
        HiddenSector=(unsigned long)buffer[0x1c]
             |((unsigned long)buffer[0x1d])<<8
             |((unsigned long)buffer[0x1e])<<16
             |((unsigned long)buffer[0x1f])<<24;
        LargeSector=(unsigned long)buffer[0x20]
             |((unsigned long)buffer[0x21])<<8
             |((unsigned long)buffer[0x22])<<16
             |((unsigned long)buffer[0x23])<<24;
        SectorsPerFAT=(unsigned long)buffer[0x24]
             |((unsigned long)buffer[0x25])<<8
             |((unsigned long)buffer[0x26])<<16
             |((unsigned long)buffer[0x27])<<24;
        RootClusterNumber=(unsigned long)buffer[0x2c]
             |((unsigned long)buffer[0x2d])<<8
             |((unsigned long)buffer[0x2e])<<16
             |((unsigned long)buffer[0x2f])<<24;
        ClusterStartSector=HiddenSector
             +(unsigned long)ReservedSector
             +(unsigned long)NumberofFAT*SectorsPerFAT;
    }
}
```

读完MBR和DBR后就要初始化FAT表，

①定位根目录

在FAT文件系统中，寻找第一簇（即2号簇）的位置也就是寻找数据区的开始位置，这并不是一件容易的事，因为它不是位于文件系统开始处，而是位于数据区。在数据区前面是保留区域和FAT区域，这两个区域都不使用FAT表进行管理。因此，数据区以前的区域只能使用扇区地址（逻辑卷地址），而无法使用簇地址。

要想定位一个FAT32文件系统的数据起始处，可以通过引导扇区的相关参数计算出来。

1.从引导扇区的偏移0x0E~0x0F字节处得到保留扇区。
2.从偏移0x10字节处得到FAT表的个数。
3.从偏移0x24~0x27字节处得到每个FAT表的大小扇区数。
4.利用如下公式计算：
保留扇区数 + 每个FAT表大小扇区数 × FAT表个数 = 数据区起始扇区号

②读取文件的重要参数

```
BytesPerSector        扇区字节数
SectorsPerCluster     每簇扇区数
ReservedSector        保留扇区数，包括引导扇区
NumberofFAT           一般为两个，FAT1和FAT2
HiddenSector          隐藏扇区数，一般对SD卡来说没有隐藏扇区，为0
SectorsPerFAT         每个FAT表的扇区数
RootClusterNumber     根目录的簇号，一般为2
ClusterStartSector=0; 2号簇的起始扇区号，一般也为2
FileStartCluster      当前TXT文件的起始簇号
CurrentCluster;       当前簇
CurrentOffset;        当前扇区偏移
DirItem[32];          目录项缓存
```

③读取txt文档的相关函数介绍

```
void ReadSectorFromCluster(unsigned long ClusterNumber,unsigned char SectorOffset)
```

从簇号读取扇区号，即当前簇号ClusterNumber-2，再乘以每簇的扇区数，在加上根目录的起始扇区，就可以得到当前簇的扇区数。

```
unsigned long FindNextCluster(unsigned long ThisCluster)
```

由文件的当前簇号得到下一个簇号，查看FAT表即可

```
void GetDirItem(unsigned long DirItemIndex)
```

得到目录项缓存内容，由目录项索引号得到

```
void ReadSectorFromFile(unsigned long ThisCluster,unsigned long offset)
```

读取当前文件的扇区号

（3）读取txt文件的具体过程

现在我们从根目录中读取TESTF149.TXT的文档：

步骤1：从卷0号扇区读取引导扇区，根据引导扇区中的信息定位FAT表、数据区和根目录的位置。

步骤2：在根目录下寻找名字为“TESTF149.TXT”且具有文件属性的目录项。

步骤3：由“TESTF149.TXT”的目录项中获取它的起始簇号,读取该簇的内容。

步骤4：到FAT表中找到该文件的簇链，确定下一簇的簇号，读取该簇的内容，直至结束标志FFFFFFFF。

至此整个从SD卡读取文档的过程就完成了，总的来说分为3个层次，第一层为搭建底层SPI通信，第二次为建立文件系统，第三层为通过文件系统提供的API进行读取文件的操作。

### 5.1.4 模块测试结果

将该模块程序下载到实验板后用CCS的在线调试功能单步执行程序，当执行完InitFAT()后，建立观测变量buffer，其值为引导扇区512字节的值，这些值正确与否直接关系到初始化是否成功，观测可得前466字节为0，最后两字节为55aa，中间46字节全为0，初步可以判断初始化成功，文件系统已建立，此时在SD卡中的根目录下最前面位置建立一个TESTF19.TXT的文本文档，写入内容aaaaa，一共5个字节，此时我们可以预测该文档的起始簇为3号簇，然后由函数ReadSectorFromCluster(unsigned long ClusterNumber,unsigned char SectorOffset)得到扇区数，即可观测到对应扇区的数据，结果确实为aaaaa，说明读取模块功能正常。

## 5.2 LCD12864 字符显示模块

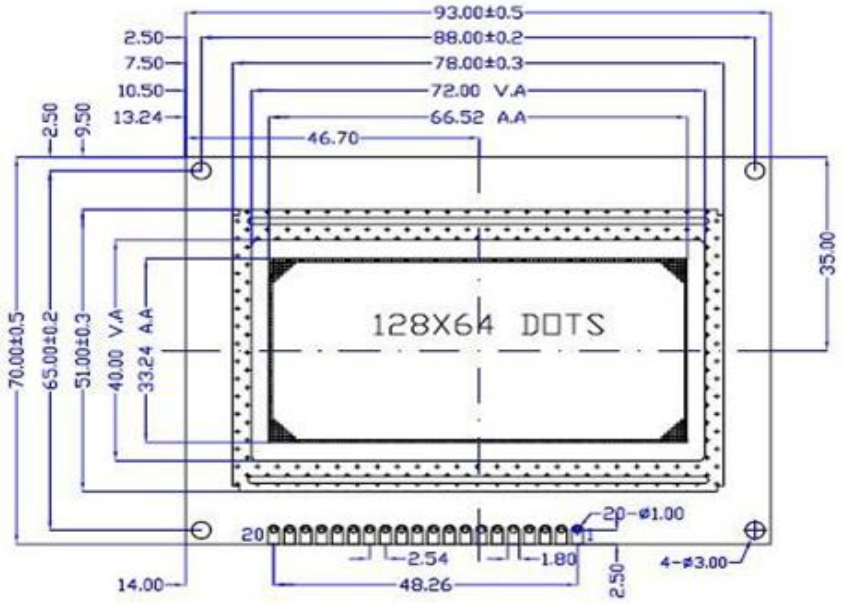
### 5.2.1 综述

本模块选用了 MSP430F149 16 位超低功耗 CPU，60KFLASH 存储器，2KRAM,LCD12864 可显示字符的蓝屏，LCD12864 低电源电压，显示分辨率(128*64),内置汉字字库，提供 8192 个 16×16 点阵汉字，内置 128 个 16×8 点阵字符，2MHZ 时钟频率,

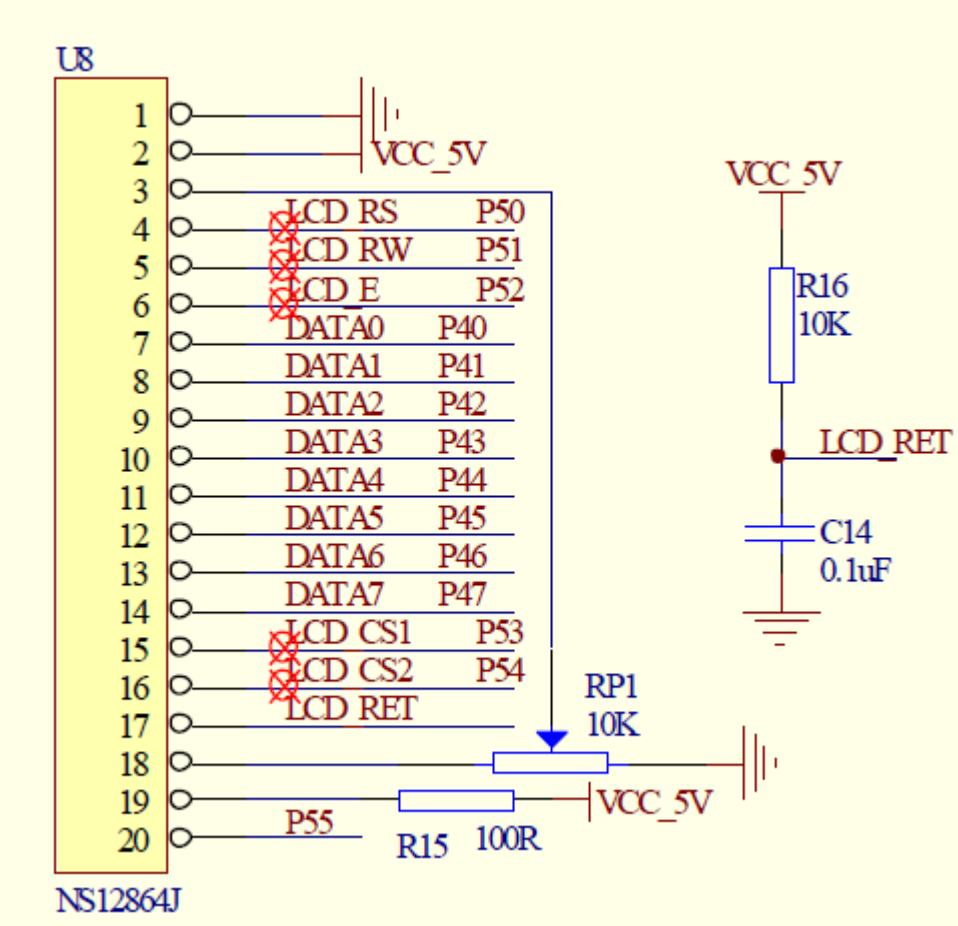
### 5.2.2 硬件电路设计

MSP430 的供电电压为 3.3v，LCD12864 供电电压为 3.3~5v，可以直接利用 MSP430 的供电电源为 LCD12864 供电。

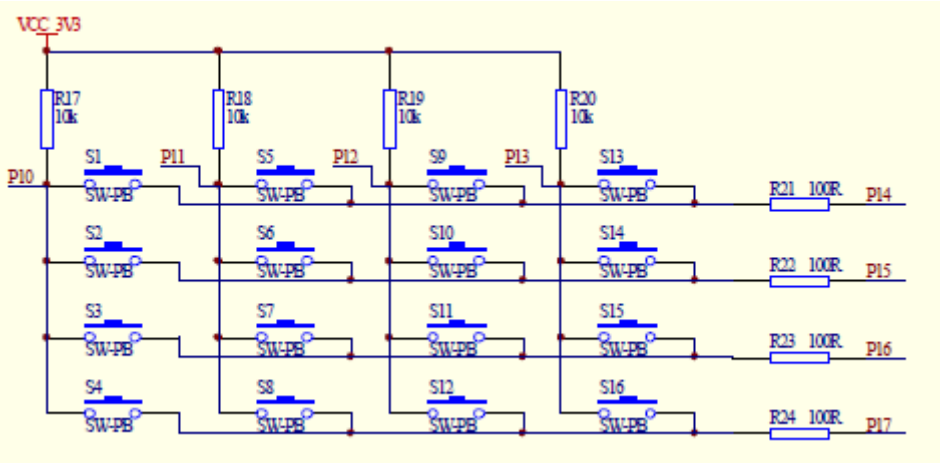
LCD12864 的外形尺寸图




硬件电路连接图




4X4 按键中断模块



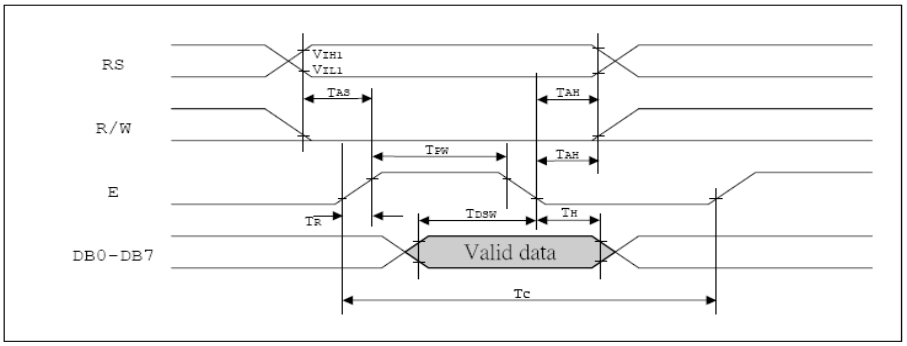

我们的识别思路是这样的，初使化时我们先让 P1 口的低四位输出低电平，高四位输出高电平，即让 P1 口输出 0xF0。扫描键盘的时候，我们读 P1 口，看 P1 是 否还为 0xF0,如果仍为 0xF0，则表示没有按键按下；如果不 0xF0，我们先等待 10ms 左右，再读 P1 口，再次确认是否为 0xF0，这是为了防止是 抖动干扰造成错误识别，如果不是那就说明是真的有按键按下了，我们就可以读键码来识别到底是哪一个键按下了。

识别的过程是这样的，初使化时我们让 P1 口的低四位输出低电平，高四位输出高电平，确认了真的有按键按下时，我们首先读 P1 口的高四位，然后 P1 口输出 0x0F，即让 P1 口的低四位输出高电平，高四位输出低电平，然后读 P1 口的低四位，最后我们把高四位读到

的值与低四位读到的值做或运算就得到了该按键的 键码。就可以知道是哪个键按下了。

以 0 键为例，初使化时 P1 输出 0xF0,当 0 键按下时，我们读高四位的状态应为 1110,即 P1 为 0xE0,然后让 P1 输出 0x0F,读低四位产状态应为 0111，即 P1 为 0x07,让两次读数相与得 0xE7。

### 5.2.3 MPU 写数据到液晶显示模块的并行接口时序图




当 PSB 脚（串/并口选择）接高电平时，模块将进入并口模式，在并口模式下可由指令 DL FLAG 来选择 8-位或 4-位接口，主控制系统将配合( RS、RW、E、DB0..DB7 )来达成传输动作。从一个完整的流程来看，当设定地址指令后(CGRAM、DDRAM)若要读取数据时需先 DUMMY READ 一次，才会读取到正确数据第二次读取时则不需 DUMMY READ 除非又下设定地址指令才需再次 DUMMY READ。在 4-位传输模式中，每一个八位的指令或数据都将被分为两个字节作：较高 4（DB7~DB4）的资料将会被放在第一个字节（DB7~DB4）部分，而较低 4 位（DB3~DB0）的资料则会被放在第二个字节的（DB7~DB4）部分至于相关的另四位则在 4-位传输模式中 DB3~DB0 接口未使用。当模块在接受指令前，微处理顺必须先确认模块内部处于非忙碌状态，即读取 BF 标志时 BF 需为 0，方可接受新的指令；如果在送出一个指令前并不检查 BF 标志，那么在前一个指令和这个指令中间必须延迟一段较长的时间，即是等待前一个指令确实执行完成，指令执行的时间请参考指令表中的个别指令说明。

显示数据RAM 提供64×2 个字节的空间，最多可以控制4 行16个中文字型（16×16 点阵），或4行×32个字符（8×16）。当写入显示资料RAM时，可以分别显示CGROM，HCGROM 与CGRAM 的字型；此中文字库系列模块可以显示三种字型，分别是半宽的 HCGROM 字型、自定义CGRAM 字型及中文CGROM 字型，三种字型的选择，由DDRAM 中写入的编码选择，在0000H ~ 0006H 的编码中将选择CGRAM 的自定字型，02H ~ 7FH 的编码中将选择半宽英数字的字型，至于A1 以上的编码将自动的结合下一个字节，组成两个字节的编码达成中文字型的编码简繁体中文BIG5（A140 ~ D75F），简体中文GB(A1A0 ~ F7FF)，详细各种字型编码如下：

1）显示半宽字型：将8 位资料写入DDRAM 中，范围为02H ~ 7FH 的编码。

2）显示CGRAM 字型：将16 位资料写入DDRAM 中，总共有0000H，0002H，0004H，0006H 四种编码。

3）显示中文字形：将16 位资料写入DDRAM 中，范围为A140H ~ D75FH 的繁体中文编码(BIG5) ，A1A0H ~ F7FFH 的简体中文编码(GB)。将16 位资料写入 DDRAM 方式为透过连续写入两个字节的资料来完成，先写入高字节（D15 ~D8）再写入低字节（D7 ~ D0）。

## 5.2.4 LCD12864 外部接口模块

| | | | |
|---|---|---|---|
| 1 | VSS | -- | 电源负端(0V) |
| 2 | VDD | -- | 电源正端(+3.3V 或+5.0V，出厂时设定+5.0V) |
| 3 | V0 | -- | LCD 驱动电压(可调) |
| 4 | RS(CS) | I | RS=0:<br>当MPU 进行读模块操作，指向地址计数器。<br>当MPU 进行写模块操作，指向指令寄存器。<br>RS=1:<br>无论MPU 读/写操作，均指向数据寄存器。<br>串口方式:<br>CS: 串行片选信号，高电平有效。 |
| 5 | R/W(SID) | I | 并口方式:<br>R/W=0 写操作。<br>R/W=1 读操作。<br>串口方式:<br>串行数据输入端 |
| 6 | E(SCLK) | I | 并口方式: 使能信号，高电平有效。<br>串口方式: 串行时钟信号。 |
| 7_14 | DB0~DB7 | I/O | MPU与模块之间并口的数据传送通道，<br>4 位总线模式下D0 ~ D3脚断开 |
| 15 | PSB | I | 串/并口控制选择端:<br>H: 并口控制;<br>L: 串口控制。 |
| 16 | NC | -- | 空脚 |
| 17 | RST | I | 复位脚（低电平有效） |
| 18 | VOUT | -- | 倍压输出脚。（VDD＝+3.3V 时有效） |
| 19 | LEDA | -- | 背光电源正端(+3.3V 或+5.0V，出厂时设定+5.0V) |
| 20 | LEDK | -- | 背光电源负端(0V) |

## 5.2.5 LCD 液晶显示和 4X4 按键中断的软件模块

软件设计流程如下

main（） case zidong（） case 向前翻页
case 向后翻页
case 跳转到手动 shoudong（）
case shoudong（） case 自动翻页 case 降低翻页速度（）
case 加快翻页速度（）
case 跳转到手动
case 跳转到手动

* 在 main（）函数中将 MSP430 和 lcd 连接的模块进行初始化。

P4SEL = 0; //将 P4 设置为 I/O 口

```
        P4DIR |= 0xFF; //  将 P4 设置为输出方向
      // 将 P5.0 P5.1 P5.2  设置为输出方向
      P5DIR |= BIT0;
      P5DIR |= BIT1;
      P5DIR |= BIT2;
      P5DIR |= BIT3;
      P5DIR |= BIT5;
      P5SEL = 0;  将 P5 设置为 I/O 口
```

* void Lcd_WriteCmd(uchar cmdcode)和 void Lcd_WriteData(uchar dispdata)实现将数据写入 lcd 并在 lcd 上进行显示。

```
CLR_LCD_RS;
CLR_LCD_RW;
SET_LCD_EN;
CLR_LCD_EN;
LCM_Data_Out = dispdata;
```

*  void Lcd_Init()实现将屏幕显示清除。

*  void hanzi_Disp(uchar x,uchar y,uchar  *s)实现字符显示的判定条件，因为屏幕最多显示 32 个字，所以一次控制只显示 32 个字符，符号也算作一个字来显示。

*  unsigned char GetKey(void)对 4X4 键盘进行扫描查询，查询到按键进入相应的触发程序。

```
P1OUT = 0xef;
temp = P1IN & 0x0f;
     if(temp == 0x0e)
KeyValue = 1
```

根据 P1OUT = 0xef;5 到 8 位为 1110，低电平触发，所以为第一列四个按键，s1，s2，s3，s4，(temp == 0x0e)，1 到 4 位为 1110，为第一排按键触发，所以是 s1 低电平触发。

4 排 4 列刚好控制 16 个按键，电子书模块用到的按键只有 7 个，分别是 s1（自动翻页模式）s2 手动翻页模式 s3 向前翻页，s4 向后翻页，s5 增加翻页速度，s6 降低翻页速度，s7 自动翻页

* void zidong(void)电子书翻页自动模块，可以调整翻页速度和进行自动翻页，每页显示 32 个字符

* void shoudong(void)电子书手动翻页模块，可以实现向前和向后翻页。

*void delayms(unsigned int t)由于自动翻页可以通过控制延时来实现对翻页速度的控制，MSP430F149 默认的时钟频率为 8MHz，每条指令需要四个周期时钟，因而处理四条指令是 2us，

```
void delayms(unsigned int t)         //约延时（Ms）
{
   unsigned int i;
   //unsigned int j;
  while(t--)
  {
       for(i=0;i<250;i++);
       {
       _NOP();_NOP();_NOP();_NOP();
  }
```

```
 }
}
```

可以延时（0.5*t ）ms

*uchar addr_tab[]实现屏幕上字符显示的位置。

### 5.2.6 模块测试结果

自动翻页模式中，翻页速度的控制每次调整为 500*0.5ms 也就是 0.25 秒，在人的视觉上还是可以分辨的，只不过浮动比较小，但调整也较为精确。手动翻页是瞬间触发的，效果还是基本能满足翻页需求。

# 6. 系统测试与结果

## 6.1 主要仪器仪表

直流稳压源、PC 机、万用表

## 6.2 调试电路的方法与技巧

调试前要作好仪器仪表的准备工作：

① 根据调试内容选用合格的仪器仪表；

② 检查仪器仪表有无故障，量程和精度应能满足调试要求，并熟练掌握仪器仪表的正确使用；

③ 将仪器仪表放置整齐，经常读取信号的仪器应放置于便于观察的位置。

检查的主要方法有两种：

(1)直观检查。按照电路原理图认真检查安装的线路，看是否有接错或漏接的线，包括错线、少线和多线，特别注意检查电源、地线是否正确。信号线、元器件引脚之间有无短接，连接处有无接触不良，二极管，三极管、集成电路、电解电容等引脚有无接错。也可用手轻拉导线并观察连接处有无接触不良。一般按顺序逐一对应检查，为防遗漏，可将已查过的线在图上做出标记，同时检查元器件引脚的使用端是否与图纸相符合。

(2)借助于万用表“R*I”档或数字万用表带声响的通断测试挡进行测试。注意观察连线两端连接元器件引脚的位置是否与原理图相符合，而且尽可能直接测元器件引脚，这样可同时发现引脚与连线接触不良的故障。

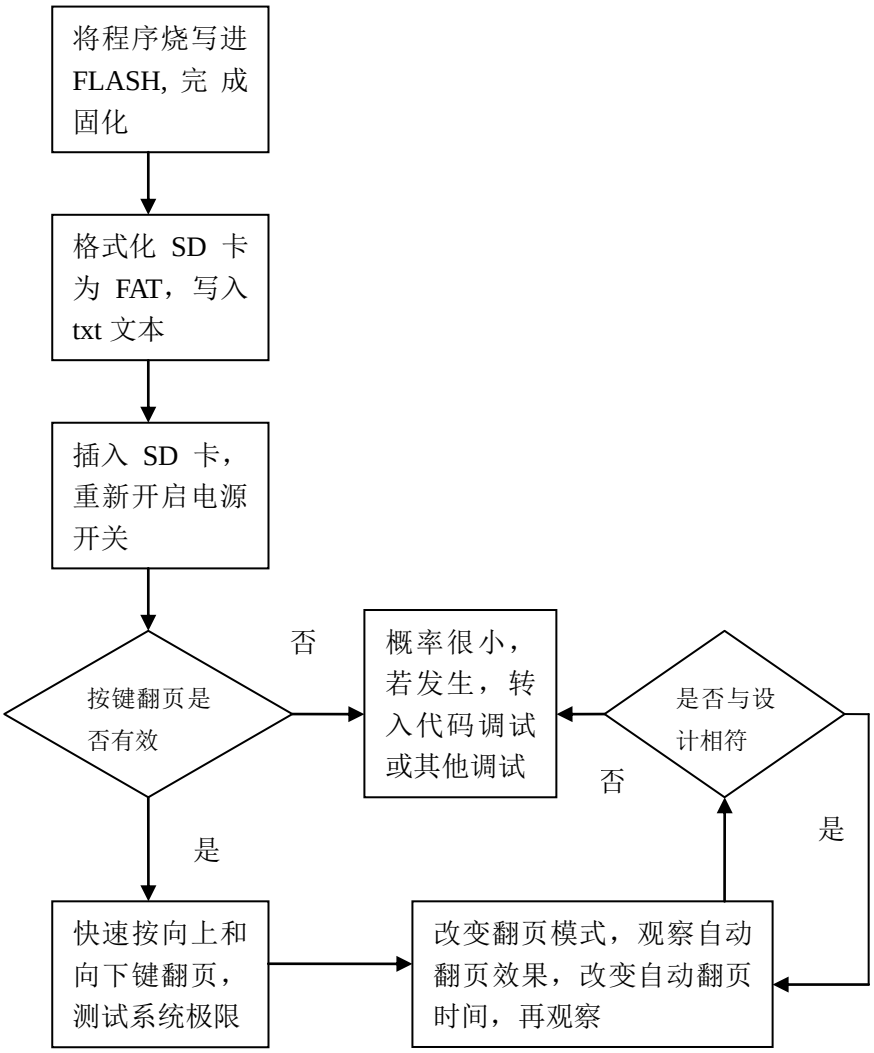
另外要特别注意检查电源，包括电源供电(包括极性)、信号源连线是否正确检查直流极性是否正确，信号线连接是否正确；电源端对地是否存在短路，在通电前，断开一根电源线，用万用表检查电源端对地是否存在短路等。

## 6.3 调试故障、产生原因及排除方法

在用查询方法进入自动翻页的 case 后，由于自动翻页的循环模式，在循环中加入查询

跳转到手动模式和翻页速度模式的调整，由于翻页速度的影响，导致扫描频率太低，所以只有长按按键才会触发中断，所以在翻页显示后加上了一个扫描按键的循环，这样显示完一页之后有个 0.3 秒左右的按键控制时间，对于人的反应来说已经足够。但这样会导致翻页速度最低不会低于 0.3 秒，但是 0.3 秒每页的阅读速度对于阅读者来说是不可能的，所以这个设计在实际中还是可行的。在按键控制中，我们试图用中断的方式来进行，但是中断的方式，目前只能调整 4 个按键，所以还是放弃了。

## 6.4 系统总体测试流程




## 6.5 测试结果

测试结果显示，系统运行结果与设想几乎相符，而且具有很强的稳健性，表现在按键翻页十分灵敏，定时翻页时间和时钟时间相当精准，各种控制都能实时生效。

不足的地方是 LCD 显示比较呆板，这与设计中忽略这部分有关，这是测试结果显现出来的不令人满意的地方。测试过程中除开发板外，用到了 SD 卡，对于这样的成本开销

相对能令人接受。

# 7．项目总结与心得体会

在本次课程设计中，我们遇到了很多问题。在人员分工上开始的时候并不明确，导致开始的时候效率比较低，到后来才慢慢习惯分工合作同步进行的工作方式。但时间太短，导致很多预期的目标没有实现，最后只实现了基本功能。在开发过程调试过程中，我们也遇到了许多问题，但所幸都在查资料以及老师和同学的帮助下解决了，在不断的发现问题和解决问题中我们学到了很多，首先，我们适应了分工合作的工作方式，相信以后如果有机会再进行开发设计时，我们一定能很快的进入到工作状态；其次，我们也学到了许多知识，在开发过程中勤于思考，善于总结，在发现和解决问题中学到知识；再次，我们明白了一个人的力量终归是有限的，在我们自己不能解决问题是，要学会寻求帮助，单靠个人的力量是很难成功的。

在课设中，前期我们的分工并不明确，进度不大，只是确定了这个的硬件部分，并完成了焊接；到后来在老师的提醒下才开始分工工作，我们三个每人负责不同的部分，夏煜负责 LCD 及翻页部分的程序编写调试；李剑负责 SD 卡读写部分的程序编写调试；我负责各种资料的整理及汇总。但由于前期工作没有做好，在程序的编写调试过程中我们遇到了很多问题都感觉无从下手，大大的拖后了整个课设的进度，到最后也只完成了基本功能。但我们也收获了很多，在我看来最重要的是我们适应了分工合作的工作方式，在分工合作中我们不是单纯的各做各的，各部分之间分交流也是必不可少的，这样才能更好的调节整个项目的进度，调高大家的效率，更快更好的完成整个项目。

到最后我们的系统完满地实现了各种基本功能，在各种测试条件下保持了性能的稳定，阅读界面也比较友好，控制需求也基本满足，虽然扩展功能未能成功实现，但这还应该是一套比较适合的而且很实用的电子书阅读器。

对于我们小组而言这次硬件设计的过程是十分曲折的，中间的所走过的弯路，所经历的挫败、失望、彷徨直到坚持中突然的“起死回生”，在过去两个星期的每一天，融入在我们每一行代码中，每一根导线里，我们无心也无法去过多思考这零零总总的琐碎细节。然而，今天当我们回望这一切，欣慰夹杂着感慨便涌上心头。

一次硬件设计留给我们的除了知识与技能之外，更为重要的是一份态度，一种品质。

# 8．致谢

在本次课程设计中我们遇到了无数的困难和障碍，也得到了许多同学和老师的帮助。

在这里我们首先要诚恳的感谢我们的指导老师曾喻江，从开始选题到最后课设的完成，曾老师一直都给予悉心的指导，为我们提供无私的指导和帮助，不厌其烦的帮助我们解决一些我们没有头绪的问题，将他解决问题的经验不断地教会给我们。另外，在校图书馆查找资料的时候，图书馆的老师也给我提供了很多方面的支持与帮助。在此向帮助和指导过我们的各位老师表示最中心的感谢！

感谢在这次课设中借鉴的各种代码的作者。在这次课设中我们借鉴了许多不同的代码，如果没有各种代码提供帮助和启发，我们将很难完成这次课设。

感谢我们的同学和朋友，在我们课程设计过程中给予我了很多资料，而且还在调试过程

中给了我们很多无私的帮助。

最后，感谢电信系和 TI 公司举办并赞助这次竞赛。另外，特别要提的是 TI 公司的免费申请样片服务，这让我们增加了接近和了解 TI 芯片和 TI 公司的机会，也给我们提供了一个可以发挥我们想象力的平台。

由于我的语文水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正！

# 9. 参考文献

[1]电子阅读器系统设计，Derek Phillips，电子系统设计
[2]电子书阅读器，仝可 施迪夫 刘正文
[3]MSP430F149 开发板套件用户手册，斯玛特精品电子工作室
[4]CCSv5 教程，曹红光
[5]SPI 模式下对 SD 卡的读写控制，尚怡君 葛明涛，光盘技术 2009 年第 8 期
[6]12864ZW 使用说明书，百度文库
[7] LCD 驱动电路、驱动程序设计及典型应用，孙俊喜，人民邮电出版社
[8]点阵 LCD 驱动显控原理与实践，张新强，北京航空航天大学出版社

# 10. 附录

附件列表（附件参见各文件夹目录）

1) 硬件课程设计项目答辩 PPT
2) 硬件课程设计作品使用手册
3) 硬件课程设计作品设计及制作过程视频展示
4）作品在公共媒体上发表、展示及评分资料链接

http://ei.hust.edu.cn/teacher/zengyj/2012/EI1300292/Dianzishu02/index.htm