嵌入式系统中 EEPROM 文件系统的设计与实现

彭晓锋
北京邮电大学电信工程学院，北京（100876）

摘　要：AT24CXX 系列 EEPROM 在嵌入式领域有着广泛的运用。本文参考微机文件系统的原理实现能兼容 AT24C08-AT24C1024 的简单文件系统，实验结果证明本系统高效可行。

关键词：EEPROM, 文件系统，嵌入式

1. 引言

随着大量嵌入式设备的出现，在嵌入式系统中用于存储数据的EEPROM因其简单、方便、可靠的性能和低廉的价格而被广为使用。当今社会嵌入式系统无所不在，各种嵌入式设备品种繁多，差别巨大。因此各公司也推出多种不同容量不同型号的EEPROM适应多样的市场应用。人们一方面希望能像管理大容量存储器（如硬盘，FLASH等）中数据一样简单便捷的操作EEPROM中的数据（包括打开、关闭、读写文件等），同时也希望这种文件系统能兼容不同容量、型号，具有较强的通用性。而对于采用两线IIC总线读写方式[1]的EEPROM来说，无法使用类似与FLASH所支持的TFFS之类的文件系统，本文参照上述思想，实现了一种能兼容AT24C08-AT24C1024类似于文件系统的用于管理EEPROM中数据的方法，并在实践项目中得到良好运用。

2. AT24CXX 系列 EEPROM 简介

AT24CXX系列是ATMEL公司生产的串行电可擦的可编程存储器，它采用8引脚封装，具有可掉电记忆，结构紧凑、存储容量大等特点，可以在2线总线上并接多片芯片，适用于具有大容量数据存储要求的嵌入式系统[2]。

ⅰ）封装及管脚说明

AT24C08-AT24C1024的封装如图1所示（对不同型号A0-A2相应改为NC,详见表1），各引脚的功能如下：

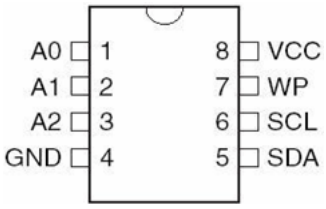

图1. AT24CXX系列EEPROM封装

⑴ A0、A1、A2：器件地址(device address)。IIC串行总线需连接多个EEPROM芯片时，可用A0、A1、A2来区分各芯片,悬空时为0。

⑵ SDA：I2C 串行数据。

⑶ SCL：I2C 串行时钟。一般在其上升沿将SDA上的数据写入存储器，而在下降沿从存储器读出数据并送往SDA。

⑷ WP：写保护。此引脚接地时，允许写操作；与VCC相连时，所有写操作被禁止。如果不连，该脚将在芯片内部下拉到地。

⑸ VCC：电源；GND：地；NC：悬空。

ⅱ）与处理器通信

AT24C系列的接口特性：一般A0-A2、WP接VCC或GND，SCL、SDA接处理器的IIC接口相应管脚，即可实现处理器对EEPROM的操作。

ⅲ）设备地址（device address）

对EEPROM读写数据前，需先发一个字节的device address以选择芯片进行读写。其中首部四比特的“1010”为固定值；A0- A2用于对多个EEPROM进行区分，注意对AT24C不同型号，

A0-A2可能用于指示片内物理地址，此时相应比特位值由访问地址决定；也可能为NC，此时值置0；最后一比特为读写操作位，1表示读操作，0表示写操作。器件地址格式见图2。

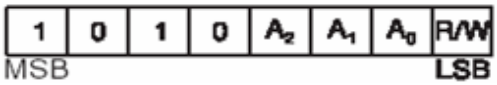

图2. EEPROM器件地址格式

ⅳ）写操作

AT24C系列的写操作有按字节写和按页写两种方式。

字节写时通常在向EEPROM发送设备地址字并收到应答信号后，发送read address选择待写数据的地址。EEPROM收到这个地址后返回一个ACK，然后接收一字节数据，再返回一个ACK，处理器收到此ACK后发停止状态结束写。

页写时EEPROM可一次连续写入整页数据。其发地址过程与写字节时完全相同。不同的是：当写完一个数据字节后，处理器不发停止状态，而是在应答信号后继续写入数据，每一个字节接收完毕后，EEPROM都返回一个ACK，一直到写完整页。注意如果页写时写入数据超出该物理页边界，则超出数据将重新写入页首地址覆盖之前所写数据。

ⅴ）读操作

读操作有当前地址读、随机读、多字节连续读三种方式。其发地址过程与写操作相同，只把device address的最低位改为读。在当前地址读操作方式时无需发送read address，每次只将当前地址所存数据读出，片内读地址始终保持自加，直到读完整个EEPROM后又回到0地址。而随机读要先写read address，然后才能读。多字节连续读操作既可以是当前地址读，也可以是随机地址读，每次处理器接收到一字节数据都返回一个ACK，EEPROM接收到此ACK后会自动地址加1，接着输出下一个字节数据，直到处理器返回NO ACK时，读过程结束。

表1. 各型号EEPROM的参数

EEPROM 型号	容量 (bytes)	页大小[1] (bytes)	总页面数	地址位 (bits)	A0-A2使用情况
AT24C08	1k	16	64	8	A2 used for device addressing and A0 A1 used for memory page addressing.[2]
AT24C16	2k	16	128	8	No bit used for device address and A0 A1 A2 used for memory page addressing.
AT24C32	4k	32	128	16	A0 A1 A2 used for device address .
AT24C64	8k	32	256	16	A0 A1 A2 used for device address.
AT24C128	16k	64	256	16	A0 A1 used for device address, A2 is NC.
AT24C256	32k	64	512	16	A0 A1 used for device address, A2 is NC.
AT24C512	64k	128	512	16	A0 A1 used for device address, A2 is NC.
AT24C1024	128k	256	512	16	A1 used for device address ,A0 A2 is NC.

1. AT24C系列把地址空间物理分页，支持按页写模式，即在一次写操作中可连续写入一整页，由于每次写操作之后必须等待Write Cycle Time之后才能继续对EEPROM进行操作，所以

文件系统采用页写模式可大大提高写文件速度。

2. 对AT24C08，由于只用8bits的read address无法提供其容量为1024bytes的地址空间，所以使用device address中的A0 A1两位，提供2^{10}=1024bytes的寻址空间。

3. 文件系统设计原理

考虑到嵌入式系统资源的有限性和实时性要求，参考微机FAT文件系统的基本思想[3]，作者设计出一个简单易行，实用行很强的嵌入式文件系统。它采用二级树形目录组织，将整个EEPROM划分为三个部分，文件目录区；页面管理区；数据存储区。

（一）文件目录区

这部分主要存储EEPROM内的文件信息，包括文件名、文件大小、修改时间、文件存储首页页地址四项。

文件信息结构定义如下：

```
typedef struct
{
  char       FileName[13];      /*文件名:最大支持12位*/
  UINT8      FileDate[5];       /*生成日期:年月日时分*/
  UINT16     FirstPageAddr;     /*首页地址*/
  UINT32     FileSize;          /*文件大小*/
}EEpromFileInfo;                /*目录区文件信息格式*/
```

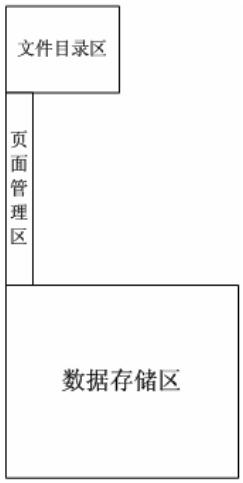

图3.系统存储结构

（二）页面管理区

页面管理区以1-2bytes为1单元，单元总数与数据存储区物理页面总数相等，每个单元管理数据存储区的一个物理页，页面管理区单元的值指示着数据存储区相应物理页的使用状况：值为0表示数据区对应的物理页未被使用；为1表示数据区对应页为某文件的末页；为其他值则是文件下一页数据所在的物理页地址，所以可以把页面管理区理解为一个类链表结构，每一单元都指向文件下页存储位置。注意单元的宽度（PageAddrWidth）为1或2由EEPROM的总物理页数决定：当总物理页数大于256时，需2byte的单元宽度才能完全指示所有物理页。

（三）数据存储区

数据存储区用于存储文件数据，它以页为单位划分，并与 EEPROM 的物理页完全重合，因此在操作文件时可以进行按页读写，极大的提高写文件速率。

综上所述，我们可以以读文件操作为例说明其基本原理（如图 4 所示）：读文件时，先到文件目录区找到该文件，并按文件信息中的首页页地址读取文件数据，然后在页面管理区读取下页数据所在的物理页地址，继续文件数据的读写，如此反复进行，直到页面管理区单元的值为 1 则表示已读到文件末页。

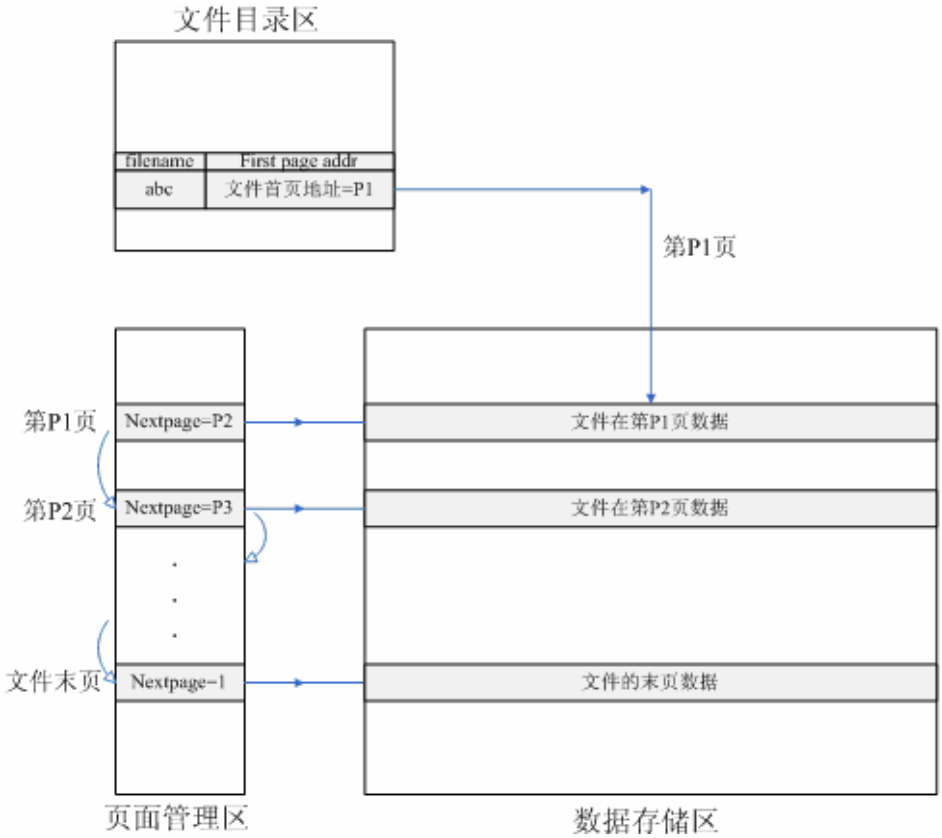

图4. 文件系统读文件操作原理

4. 系统程序实现

文件系统的程序实现包括两部分：底层的IIC读写驱动程序[4]和之上的文件系统函数。

（一）IIC读写驱动程序

由于本系统兼容AT24C08-AT24C共八种EEPROM，它们的操作时序都不尽相同，有的型号read address只有1byte（如AT24C08），有的却是2bytes（如AT24C32），有的read address不够编址空间，需借助device address中的A0-A2（如AT24C16）。因此它们使用的IIC读写驱动也不能完全一样，在这里使用宏定义，首先定义EEPROM型号，再根据此型号定义EEPROM的各参数，例：

```
#define   AT24C1024                          /*根据实际使用情况定义EEPROM型号*/
#ifdef   AT24C1024
  #define TotalPhyPages 512                 /*总物理页面数*/
  #define PageSize 256                      /*页大小*/
  #define PageAddrWidth 2                   /*页面管理区  单元宽度：2bytes*/
  #define EEpromCapacity 131072             /*EEPROM容量大小*/
#endif
#ifdef   AT24C1024
  A0=(UINT8)((EEpromAddr&0x10000)>>15);
  DeviceAddr=0xa0|A0;       /*由读写地址EEpromAddr计算device address中的A0值*/
  ReadAddr=(UINT16)(EEpromAddr&0xffff); /*计算ReadAddr*/
  MSB_flag=1;                                /*read address为2bytes：MSB+LSB*/
#endif
```

在得到变量DeviceAddr、ReadAddr、MSB_flag的值后很容易写出IIC读写程序。

（二）文件系统函数

文件系统提供常用的文件操作函数[6]：格式化、文件的打开/关闭、读/写、新建/删除、文件的定位。

a）格式化：即给文件系统的初始化过程，只要将文件目录区和页面管理区全填0（标注未用）即可。文件目录区的大小按系统最大存储文件个数分配，由此需先考虑文件系统最大支持的文件数目：filecounts，本系统采用实参的方式传递给格式化函数，由此可得到文件目录区的大小sizeof(EEpromFileInfo)*filecounts，再将EEPROM剩下的空间分成长度相等的页面管理区和数据存储区。

```
BEGING
  文件系统本身占用空间FileSystemPages=文件目录区+页面管理区
  pageAddr=0
  while pageAddr<FileSystemPages
  {
    第pageAddr页整页写0
    pageAddr++
  }
END
```

b）文件的打开/关闭：打开文件时先在文件目录区中找到该文件名匹配的文件信息，将此文件信息赋给系统管理给该文件的一个文件描述符，文件描述符内保存有文件的当前读写位置，函数返回一个指向被打开文件文件描述符的指针，主要代码如下。关闭文件时应先该更新文件在文件目录区中的文件信息，再释放管理该文件的文件描述符。

```
BEGIN
 依次读取文件目录区中文件名NAME
 if NAME等于要打开文件名
    读取文件信息到指向该文件文件描述符
    文件读写指针指向文件的第一个字节
 if  文件目录区中无此文件
    返回ERROR
END
```

c）文件的读/写：首先根据要操作的字节数判断是否要跨页操作，如需跨页则应在页面管理区中读取存储文件数据的下页地址，再由页地址计算其实际物理地址，注意操作后更改文件的读写指针，尤其是跨页读写时情况更为复杂，注意各种临界情况，保证文件系统的健壮性。当写入数据时要判断是覆盖写还是增加新内容，要适时申请新物理页并更新目录区内文件信息。下段程序实现文件的跨页读，当然在读过程中还要判断是否超出文件范围，限于篇幅，不一一写出，写操作与读类似。

```
BEGIN
  pages=0
  整页读取页数=待读字节数/PageSize
  while pages<整页读取页数
  {
```

```
    整页读取数据
    pages++
   }
  剩余不足一页的数据大小=待读字节数%PageSize
  读取剩余字节
END
```

d）文件的新建/删除：新建文件时首先在文件目录去申请空间，新建的文件大小初始化为0，读取当前系统时间，再把这些文件信息存入文件目录区即可，删除文件时则把文件在文件目录区的文件信息删除(填0)，并在页面管理区把该文件占用的页面全部回收即可。

e）文件的定位:文件的定位可以实现文件的随机读写。程序实现比较简单，只需将该文件文件描述符中的读写指针移到实参指定位置即可。

以上列出了常用的文件操作函数，此外对于标准输入输出设备重定向在串口超级终端的嵌入式系统还可自行编写函数show_EEprom_info()实现在超级终端输入命令即可观察EEPROM 内所有文件信息；对于同时连接了EEPROM和FLASH的嵌入式系统可以使用函数EEPROM_TO_FLASH()、FLASH_TO_ EEPROM()实现EEPROM和FLASH文件的互拷，由于FLASH有支持FTP协议的文件系统，所以可以通过从EEPROM拷贝到FLASH再通过FTP软件下载FLASH中文件的方法实现在PC上查看EEPROM中的文件，或将PC上的文件下载到EEPROM。这三个函数非常实用并且在程序实现上很简单，在上述基本操作函数的基础上容易编写。

5. 性能分析

本系统实现了EEPROM中数据的文件化存储和管理，使用者只需进行文件操作，而不是通常采用的顺序存储，所以管理数据时只要知道其所在文件中的位置而不必精确的知道每字节数据存储的物理地址，大大提高了编程效率和数据管理的复杂度。同时本系统兼容AT24C08- AT24C1024共八种不同容量的EEPROM，能适应各种不同的应用场合。通过测试，本系统在所有型号EEPROM上都正常运行。作者的测试环境以ARM7芯片S3C4510为控制核心[5]，I/O管脚模拟的IIC总线与EEPROM通信，SCL时钟频率设置为400k，测试结果如下：

a）文件操作函数测试

以AT24C128为例，格式化EEPROM之后，先新建多个文件，再对这些文件进行循环的读写操作，并使这些文件完全写满EEPROM所有物理页；然后删除前8个文件；此时再使用函数show_EEprom_info()查询EEPROM内文件信息，打印信息如图5所示。所有显示数据与是实验前计算值相等。

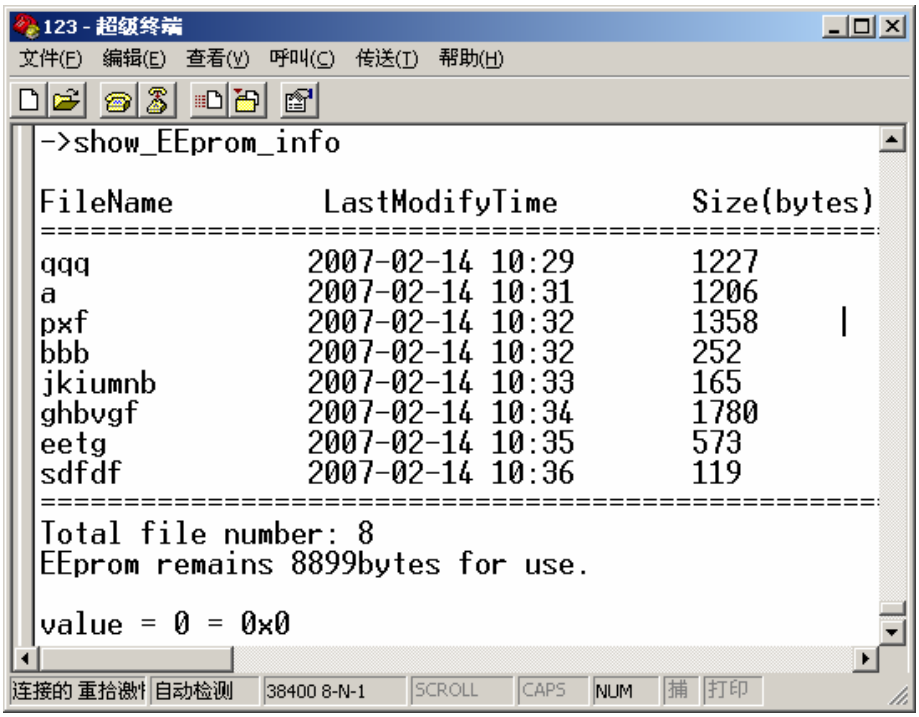

图5. 函数show_EEprom_info()显示EEPROM中文件信息

b）系统消耗及碎片情况

各型号EEPROM的系统消耗及碎片情况如表2所示。

由于文件目录区的大小按系统最大存储文件个数分配，以下数据均以系统最大存储文件数为10计算得到。由表中数据可知EEPROM容量越大，文件系统的效率越高。

表2. 系统消耗及碎片情况

EEPROM型号	文件系统本身占用空间(bytes)	最坏情况文件碎片大小(bytes)
AT24C08	288（约占总容量的28.1%）	150（约占总容量的14.6%）
AT24C16	352（约占总容量的17.1%）	150（约占总容量的7.3%）
AT24C32	384（约占总容量的9.3%）	310（约占总容量的7.5%）
AT24C64	512（约占总容量的6.2%）	310（约占总容量的3.7%）
AT24C128	512（约占总容量的3.1%）	630（约占总容量的3.8%）
AT24C256	1280（约占总容量的3.9%）	630（约占总容量的1.9%）
AT24C512	1280（约占总容量的1.9%）	1270（约占总容量的1.9%）
AT24C1024	1280（约占总容量的0.9%）	2550（约占总容量的1.9%）

c）读写速度

ARM的IIC时钟频率设为400kbps，以读写1k bytes为例：

写文件时$T_{write}=T_{data_write}+T_{Self\text{-}timed\ Write\ Cycle}=(1024/(400/8)+1024/PageSize*5)$ ms

读文件时$T_{read}=1024/(400/8)=20.5$ms

注：此处只考虑读写数据时间和写等待时间，忽略了指令运行时间。

各型号EEPROM的读写速度如表3所示。

表3. 文件系统读写速度

EEPROM型号	读1k bytes需时T_{read}	写1k bytes需时T_{write}
AT24C08	20.5ms	0.34s
AT24C16	20.5ms	0.34s
AT24C32	20.5ms	0.18s
AT24C64	20.5ms	0.18s
AT24C128	20.5ms	0.10s
AT24C256	20.5ms	0.10s
AT24C512	20.5ms	60.5ms
AT24C1024	20.5ms	40.5ms

由上表数据可知，EEPROM的读速度与容量大小无关，但写速度与容量大小成正比，这是因为容量越大则页大小（PageSize）越大，由于采用了页写模式，而写过程中主要时延在写完一页后的写等待（$T_{Self\text{-}timed\ Write\ Cycle}$）上，因此大容量的EEPROM写等待次数少，写速度快。

6. 总结

本文详述了一个适用于嵌入式系统的简单文件系统，提供普通的文件操作函数，实现数据的高效存取，可以避免程序员花费大量的精力在EEPROM的数据存储上。实验结果证明本系统运行可靠，高效实用。

参考文献

[1]. IIC协议标准
[2]. AT24CXX SERIAL EEPROMs DATA SHEET
[3]. 坦尼鲍姆，现代操作系统(第2版)，机械工业出版社，2005
[4]. 周立功，ARM嵌入式系统软件开发实例，北京航空航天大学出版社
[5]. 李驹光等，ARM应用系统开发详解--基于S3C4510B的系统设计，清华大学出版社，2005
[6]. 周启平，杨编 编著，VxWorks程序员速查手册，机械工业出版社，2005

A embedded file system based on AT24CXX series EEPROMs

Peng Xiaofeng
School of Telecommuniction Engineering，Beijing University of Posts and Telecommunications，Beijing (100876)

Abstract
AT24CXX series EEPROMs are widely used in embedded field. Considering the FAT file system in PC, this paper realized a simple file system which is compatible with AT24C08-AT24C1024, the experiment result shows that the system works well with a high performance.
Keywords：EEPROM, file system, embeded system.

作者简介：彭晓锋(1982-)，男，研究生，研究方向为数字视音频通信。