支持 Yaffs2 文件系统的 U-Boot 的实现

程 伟

(安徽理工大学 电气与信息工程学院，安徽 淮南 232001)

摘 要：为了在大容量 NAND Flash 存储器中运行 Yaffs2 (yet another flash file system) 文件系统，分析了 Yaffs2 文件系统的结构、原理、性能和嵌入式系统中启动代码 U-Boot 的功能。在此基础上对 U-Boot 进行了改进。实现了在嵌入式系统的开发过程中用 U-Boot 向大容量 NAND Flash 中写入 Yaffs2 文件系统，并将其成功应用在正在开发的嵌入式手持设备中，从而可以方便地使用大容量的 NAND Flash 存储器。

关键词：大容量存储器；Yaffs2；嵌入式文件系统；U-Boot；设计开发

中图法分类号：TP311 **文献标识号**：A **文章编号**：1000-7024 (2012) 03-0936-05

Implementation of Yaffs2 file system support in U-Boot

CHENG Wei

(School of Electrical Engineering and Information, Anhui University of Science and Technology, Huai'nan 232001, China)

Abstract: To run Yaffs2 (yet another flash file system) file system in large-capacity NAND Flash memory, the structure, principle, performance of Yaffs2 file system and the function of bootloader code U-Boot for embedded system are analyzed. On the basis, U-Boot is improved. The function of Yaffs2 file system is realized, which is wrote into the large-capacity NAND Flash by U-Boot the U-Boot is successfully applied in development of embedded handheld device, so the large-capacity NAND Flash memory is used conveniently.

Key words: large-capacity NAND Flash; Yaffs2; embedded file system; U-Boot; design and development

0 引 言

在嵌入式产品中，随着系统功能复杂度的增加和对数据操作灵活性的提高，小容量的 Flash 设备已经不能满足需求，大容量 Flash 设备使用的越来越广泛。嵌入式文件系统可以对 Flash 存储器上庞大的数据进行存储和管理，而 Yaffs2 文件系统就是专门针对大容量 NAND Flash 存储器设计的嵌入式文件系统，在 NAND Flash 上运行性能稳定优异。由于 Yaffs2 文件系统自身的特点，在写入到 NAND Flash 存储器中时不能使用一般的工具和方法。U-Boot 作为目前被广泛使用的一种嵌入式系统中的启动引导程序，可以在开发过程中实现对文件的烧写和调试[1]。目前 U-Boot 自身还不支持烧写 Yaffs2 文件系统，由于 Yaffs2 文件系统的特殊结构，需要对 U-Boot 做一定的改进。本文给出了 U-Boot 的修改过程并进行了详细分析，改进后的 U-Boot 可直接把 Yaffs2 文件系统写入 NAND Flash 存储器中，方便了构建 Yaffs2 文件系统＋大容量 NAND Flash 形式的嵌入式系统结构[2]，使得嵌入式系统的设计开发更加便利。

1 NAND Flash 存储器

目前 Flash 存储器以其速度快、容量大、成本低等优点，正被广泛的应用于嵌入式系统中。常用的 Flash 存储设备类型有 NOR Flash 和 NAND Flash 两种。其中 NOR Flash 采用或非结构栅格存储矩阵实现，可片内运行程序，常用于系统启动代码和内核映像等存储[3]，但是工艺复杂，容量小、价格较贵，具有很低的写入和擦除速度。NAND Flash 在嵌入式系统中的地位与 PC 上的硬盘类似，用于保存系统运行所必须的操作系统、应用程序、用户数据等，容量大可达 1GB 以上且价格也相对便宜，其可擦除次数是 NOR Flash 的 10 倍，其寿命也远远超过 NOR Flash。NAND Flash 也不是完全可靠的，每块芯片出厂时都有一定比例的坏块存在，在使用前需要将坏块扫描出来，确保不再使用它们，否则会使产品有严重的故障隐患；而且发生位反转的概率要高于 NOR Flash，当位反转发生在关键

收稿日期：2011-03-25；**修订日期**：2011-05-26

作者简介：程伟 (1985-)，男，山东泰安人，硕士研究生，CCF 会员，研究方向为电路设计自动化、嵌入式系统设计。
E-mail: chengwei16@126.com

代码、数据上时，有可能导致系统崩溃，一般在读写时需要使用 ECC（error correction code）进行错误检验和恢复。

为了方便管理，NAND Flash 的存储空间使用了块（block）和页（page）两级存储体系。一般 128MB 以下容量的 NAND Flash 芯片，一页大小为（512 + 16）字节，其中的 512 字节就是一般的存储数据的区域，16 字节为空闲区（spare data）又称为 OOB（out of band）区；通常在 OOB 区存放坏块标记、前面 512 字节的 ECC 校验码等[4]。128 MB 以上大容量的 NAND Flash 芯片，一页大小通常为 2 KB，每页包含一个 2048 字节的数据区和 64 字节的 OOB 区[5]。目前大容量的 NAND Flash 作为嵌入式产品中的存储介质，其应用日益广泛，NAND Flash 在 Compact Flash、Secure Digital、PC Cards 和 MMC 存储卡市场上所占份额很大。

2 Yaffs2 文件系统分析

2.1 Yaffs 文件系统

嵌入式文件系统是嵌入式操作系统的一部分，它的任务是对逻辑文件进行管理，提供对逻辑文件操作的接口，如检索、修改、删除、复制等，以方便用户的使用[6]。嵌入式文件系统还具有兼容性好、支持自定义的实时文件系统、可裁剪可配置、支持多种存储设备等特点，一个适合嵌入式设备的文件系统，将使嵌入式设备上的文件管理更加方便快捷，大大提高嵌入式设备的性能[7]。支持在 Flash 上运行的常用嵌入式文件系统有 Cramfs、Jffs、Jffs2、Yaffs、Yaffs2 等，Cramfs 文件系统是只读文件系统。通常在 NOR Flash 上多选用 Jffs 及 Jffs2 文件系统，在 NAND Flash 上选用 Yaffs 或 Yaffs2 文件系统。

Yaffs 文件系统是一种类似于 Jffs/Jffs2、专门为 NAND Flash 设计的嵌入式文件系统，目前有 Yaffs 和 Yaffs2 两个版本，它是日志结构的文件系统[8]，开源具有很好的移植性，能够在 Linux、ucLinux 和 WinCE 下面运行。提供了损耗平衡和掉电保护，可以有效地避免意外掉电对文件系统一致性和完整性的影响[9]。

2.2 Yaffs2 文件系统

2.2.1 Yaffs2 文件系统简介

Yaffs 是效果很理想的 NAND Flash 上的文件系统，但它不支持数据压缩，而且它仅针对每页 512 字节（小页）大小的 NAND Flash 存储器；而很多大容量的 NAND Flash 使用大小为 2KB 的页（大页），Yaffs 并不能支持这种大页 NAND Flash。Yaffs2 正是为了支持 2KB 每页的大容量 NAND Flash 和严格的连续页写命令而开发出来的，Yaffs2 作为 Yaffs 的升级版，不但支持这两种页大小的 NAND Flash，而且还支持一些新型的和具有严格写入时序的 NAND Flash。NAND Flash 的基本擦除单位是 Block（块），而基本写入单位是 page（页）；Yaffs2 在分配存储空间的时候也是以页为单位的，不过在 Yaffs2 中通常被称为块（chunk），其实和大页 NAND Flash 的页（page）是一样的大小，在大多数情况下和页是一个意思。

Yaffs2 充分考虑了大页 NAND Flash 的结构特点，文件系统本身就包含了 OOB 区的数据（里面有坏块标记、ECC 校验码以及其它和 Yaffs2 相关的信息）。根据大页 NAND Flash 以页面为单位存取的特点，将文件组织成固定大小的数据段；利用大页 NAND Flash 提供的每个页面 64 字节的 OOB 空间来存放 ECC 和文件系统的组织信息，实现了错误检测和坏块处理[10]。写入 Yaffs2 时，不需要再计算 ECC 值，首先检查是否坏块（是则跳过），然后写入 2048 字节的数据，最后写入 64 字节的 OOB 数据，如此循环。

2.2.2 Yaffs2 文件系统数据存储结构

Yaffs2 文件系统中的文件、目录、链接、设备文件（以下统称文件）统一用文件头（Yaffs2 _ objectHeader 结构）描述，每个文件头存放在 NAND Flash 某页的数据区内，其中包括了这个文件的模式、类型、所有者 ID、创建时间、Parent Object ID 等信息，Yaffs2 文件系统分区内的所有文件用 object ID 来惟一标识。一些文件系统组织信息（YAFFS TAG）即元数据（如文件 ID、页 ID、有效字节数等）存放在 NAND Flash 每页的 OOB 区中，由于文件系统的基本组织信息保存在页面的空闲空间中，因此在文件系统加载时只需要扫描各个页面的空闲空间，即可建立起整个文件系统的结构，提高了文件系统的加载速度。同时由于支持的页变大，Yaffs2 的 OOB 区的数据结构与 Yaffs 的略有不同，比如 Yaffs2 的 OOB 区中增加了块分配序列号[11]。

2.2.3 Yaffs2 文件系统的优势

与 Yaffs 相比，Yaffs2 除了可以支持 2KB 每页的 NAND Flash 外，还做了一些改进[12]，可存储信息更多，也更灵活。如文件头的部分元数据，免去读文件头获取这些数据的时间；Yaffs 在更新文件 chunk 数据时，将标志位复制下来且写入序列号加 1 和新数据一起写入空白页，然后将原 chunk 的标志位字节写为 0，标记 chunk 无效；而 Yaffs2 在更新时不再重写其标志位，来标记 chunk 无效，加快了写入速度[13]。块分配序列号的采用使加载时还可区别 chunk 有效和无效，在垃圾收集的时候也会以此作为参考之一，判断该块是否适合回收，同时垃圾回收策略也有改进。Yaffs2 在内存空间占用、垃圾回收速度、读/写速度等方面相对于 Yaffs 也有较大改进，如表 1 所示。

表 1　Yaffs2 与 Yaffs 的性能比较

比较		Yaffs2	Yaffs
写操作	快 1～3 倍	1.5MB/s～4.5MB/s	1.5 MB/s
读操作	快 1～2 倍	7.6 MB/s～16.7 MB/s	7.6 MB/s
删除操作	快 4～34 倍	7.8 MB/s～62.5 MB/s	1.8 MB/s
垃圾回收	快 2～7 倍	2.1 MB/s～7.7 MB/s	1.1 MB/s
内存消耗	减少 25%～50%	—	—

表 1 中 Yaffs2 的最差性能表现在每页 512 字节的 NAND Flash 上，与 Yaffs 相似；较佳性能则表现在每页 2KB 的 NAND Flash 上。Yaffs2 还可以支持东芝和 SanDisk 公司的多层单元闪存（multi-level cell，MLC）部件[14]。

3 实现 U-Boot 写入 Yaffs2 文件系统

3.1 U-Boot 对文件系统的支持

在嵌入式系统中，Bootloader 是系统上电后执行的一小段程序，它主要完成了初始化硬件设备、准备好软件环境，最后调用操作系统内核，真正起到了引导和加载内核的作用。U-Boot 作为嵌入式系统中一种比较流行、功能强大的 Bootloader，支持多种嵌入式操作系统和多种处理器系列，性能可靠稳定，功能设置灵活，源代码开放调试方便，可移植性强可以在不同硬件平台上进行移植，通过修改可以方便的增加其功能。

在实际生产中，可以通过烧片器等手段将内核、文件系统映像烧入固态存储设备中，U-Boot 不需要具备烧写功能。但在嵌入式产品开发过程中，为了方便通常在 U-Boot 中增加烧写内核、文件系统映像文件等功能，完成向 Flash 中烧写文件。在目前的 U-Boot 官方版本中仅能够支持 Cramfs、Jffs1/2 文件系统的烧写，这些文件系统最初是针对 NOR Flash 的设计的。现在大容量的 NAND Flash 使用越来越多，同时由于很多使用 NAND Flash 的系统，在 Linux 下都用 Yaffs2 作为存储数据的文件系统，甚至是根文件系统，所以在实际开发过程中 U-Boot 能够实现烧写 Yaffs2 文件系统非常必要。

3.2 具体实现方案

本次设计的嵌入式手持设备中用到的软硬件开发平台为处理器 ARM+DSP 结构，ARM 处理器采用三星公司的 S3C2440A，U-Boot 为 u-boot-2010.06 版本，基于嵌入式 Linux 操作系统。由于性能上的需要用到了三星大容量 NAND Flash 芯片，型号为 K9K4G08U0M 总容量 512M，每页大小为（2048+64）字节，开发过程中需要使用 U-Boot 烧写 Yaffs2 文件系统。

要实现对 Yaffs2 文件系统的支持，首先要修改 U-Boot 的 NAND Flash 读写命令，增加烧写 Yaffs2 文件系统的命令，其次重要的是实现 U-Boot 的内存技术设备（memory technology device，MTD）层的驱动支持，针对大页 NAND Flash 的特点增加处理 OOB 区域数据的功能。MTD 是 Linux 中对 ROM、NOR Flash、NAND Flash 等存储设备抽象出来的一个设备层，对 Flash 操作的接口提供了一系列的标准函数，将硬件驱动设计和系统程序设计分开，硬件驱动人员不用了解存储设备的组织方法，只需提供标准的函数调用。无论是 Jffs2 还是 Yaffs2[15]，还有当 NAND Flash 进行写入和擦出操作时，都需要 MTD 的支持。

3.3 在头文件中添加支持 Yaffs2 文件系统的相应的宏

在此之前还要针对具体的 NAND Flash 芯片，对 NAND Flash 的底层驱动代码进行必要的修改，使 U-Boot 支持大容量的 NAND Flash。

在 U-Boot 的源代码目录中，首先添加自己的开发板命名为 Key2440（可任意设置），然后再添加配置文件 include/configs/Key2440.h，并在文件中添加如下宏定义：

```
#define CONFIG_CMD_NAND_YAFFS2 1
/*支持 Yaffs2 文件系统*/
#define CONFIG_CMD_NAND_YAFFS2_SKIPFB 1
/*Yaffs2 文件系统特性决定跳过第一个可用的逻辑块*/
```

3.4 添加新的 U-Boot 命令支持烧写 Yaffs2 文件系统

U-Boot 的命令为用户提供了交互功能，并且已经实现了几十个常用的命令。如果开发板需要很特殊的操作，可以添加新的 U-Boot 命令。U-Boot 的每一个命令都是通过 U_Boot_CMD 宏定义的，这样每一个 U-Boot 命令有一个结构体来描述。在 u-boot-2010.06 版本中已经可以通过“nand write…”、“nand write.jffs2…”等命令来烧写内核，烧写 Cramfs、Jffs2 文件系统，因此可以仿照此结构添加烧写 Yaffs2 文件系统的命令。

3.4.1 修改 common/cmd_nand.c 文件

（1）先添加“nand write [.Yaffs2]”命令的使用说明，U-Boot 中的命令都在 common/cmd_nand.c 文件中定义。

```
U_BOOT_CMD（nand，CONFIG_SYS_MAXARGS，1，do_nand，
“nand write [.Yaffs2] addr off size — write the 'size' byte Yaffs2 image starting \n”  /*Yaffs2 的写命令*/
“at offset 'off' from memory address 'addr' (.Yaffs2 for 2048+64 NAND) \n”
```

（2）修改函数 do_nand，添加对“write [.Yaffs2]”命令的支持，do_nand 函数主要是处理与 NAND Flash 操作相关的命令，负责 NAND 命令族的区分和执行[16]。

```
} else if (s! =NULL &&! strcmp (s," .Yaffs2"))
{ if (read)
{ printf (" nand read. Yaffs [2] is not provide temporarily !" ); }
    else { nand->rw_oob = 1; /*写 OOB 数据区*/
    #if defined ( CONFIG_CMD_NAND_YAFFS2_SKIPFB)
        nand->skipfirstblk = 1; /*跳过第一个可用块*/
    #else  nand->skipfirstblk = 0;
    #endif
ret = nand_write_skip_bad (nand, off, &size, (u_char *) addr);
```

/ * 此处又调用了 nand _ write _ skip _ bad 函数 * /

3.4.2 修改 include/linux/mtd/mtd. h 文件

在修改 do _ nand 时用到了 rw _ oob 和 skipfirstblk 两个结构体数据成员，它们是 mtd _ info 结构体中新加的项，mtd _ info 是用于描述 MTD 原始设备的数据结构，其中定义了大量关于 MTD 的数据和操作函数，所以要修改 mtd _ info 结构添加对两个成员的支持。

```
struct mtd _ in fo { u _ int32 _ t writesize;
                     u _ char rw _ oob;
                      u _ char skipfirstblk};
```

3.5 修改 U-Boot 的 MTD 层的相关驱动的代码

3.5.1 修改 drivers/mtd/nand/nand _ util. c 文件

在文件中添加 NAND Flash 的 OOB 区数据的相关操作信息。

```
/ * 大页 NAND Flash 中 ECC 校验码信息，OOB 区为 64 字节 * /
static struct nand _ ecclayout Yaffs2 _ ecclayout = {
. useecc = MTD _ NANDECC _ PLACE ,
/ * ECC 的放置模式 * /
. eccbytes = 24,              / * ECC 字节数 * /
. eccpos = {  40, 41, …… 62, 63  }
/ * ECC 校验码在 OOB 区中的位置 * /
. oobfree = { {2, 38} }     / * 还可被自由使用 OOB 区域的开始位置和长度 * / };
```

刚才在添加“nand write [. Yaffs2]”命令的代码中，调用了 nand _ write _ skip _ bad 函数，还要对其进行相应修改增加两部分程序，一部分是为了计算正常数据区的长度；另一部分是为了在写入一段数据后，数据指针能正确指向下一段数据。

(1) int nand _ write _ skip _ bad (nand _ info _ t * nand, loff _ t offset, size _ t * length, u _ char * buffer) {

```
#if defined (CONFIG _ CMD _ NAND _ YAFFS2)
  / * 得到正常数据区的长度 * /
if (nand->rw _ oob==1)
{ size _ t oobsize = nand->oobsize;
     size _ t datasize = nand->writesize;
     datapages = * length/ (datasize+oobsize);
     * length = datapages * datasize;
     left _ to _ write = * length;
#if ! defined (CONFIG _ MTD _ NAND _ YAFFS2)
  if (len _ incl _ bad == * length) {
    rval=nand _ write (nand, offset, length, +buffer);
     return rval;    }
(2) / * 使数据指针指向下一段数据 * /
  if (nand->rw _ oob==1)
       p _ buffer+=write _ size+ (write _ size/nand->
writesiz * nand->+ oobsize);
  else  p _ buffer += write _ size;
```

3.5.2 修改 drivers/mtd/nand/nand _ base. c 文件

在上步中 nand _ write _ skip _ bad 函数又对 nand _ write 函数进行了访问，所以还要对 nand _ write 函数进行修改，添加对 Yaffs2 的支持。主要是添加两部分代码，一部分把正常数据与 OOB 区数据进行分离；另一部分将写页时的模式设置为 MTD _ OOB _ RAW，使写页时不再进行 ECC 值的计算。因为根据 Yaffs2 文件系统的特性，ECC 的校验值已经包含在了 Yaffs2 文件系统自带的 OOB 区中，不能重写入。在此模式下，写入正常数据后会把 OOB 区缓存的数据写入 NAND Flash 的 OOB 区中。

nand _ write 函数在 drivers/mtd/nand/nand _ base. c 文件中。

(1) static int nand _ write (struct mtd _ info * mtd, loff _ t to, s ize _ t len, size _ t * retlen, const uint8 _ t * buf)

```
{  if (mtd->rw _ oob==1)
{  size _ t oobsize = mtd->oobsize;
/ * mtd->oobsize 是 U-Boot 初始化 NAND Flash 时得到的 OOB 区的大小，本系统中用的是三星 K9K4G08U0M 芯片 OOB 区为 64 字节 * /
    size _ t datasize = mtd->writesize;   / * NAND Flash 页大小 2KB * /
    uint8 _ t oobtemp [oobsize]; / * 临时 OOB 区 * /
    datapages = len/ (datasize);       / * 需用页数/ *
for (i=0; i< (datapages); i++)
/ * 把正常数据与 OOB 区数据进行分离 * /
{memcpy ( (void * ) oobtemp, (void * ) (buf+datasize * (i+1)), oobsize);
memmove ( (void * ) (buf+datasize * (i+1)), (void * ) (buf +datasize * (i+1) +oobsize), (datapages- (i+1)) * (datasize) + (datapages-1) * oobsize);
……..    }}
```

(2) / * 设置模式为 MTD _ OOB _ RAW，使写页时不再进行 ECC 值的计算 * /

```
chip->ops. mode = MTD _ OOB _ RAW / * 模式设置 * /
```

上述对 U-Boot 代码的修改其实主要是通过修改 U-Boot 的 NAND Flash 读写命令，添加读写 Yaffs2 文件系统的命令，并对支持该命令的函数和被调用过的函数进行相应修改，针对大页 NAND Flash 和 Yaffs2 文件系统的特点增加处理 OOB 区数据的功能，至此 U-Boot 已经实现了对 Yaffs2 文件系统的支持。

4 U-Boot 的编译与测试

对 U-Boot 重新编译，生成新的 U-Boot. bin 文件并用 J-Link 工具将其烧入 NAND Flash 后启动本系统，在串口工

具中能够看到提示信息，输入 nand info 命令可查看到 NAND Flash 的信息，说明 U-Boot 识别出了 NAND Flash。在 U-Boot 的命令行中输入 nand help 命令，可以看到刚添加的命令 nand write [. Yaffs2] addr off size，这样就可以用它来下载 Yaffs2 文件系统了。

使用 mkyaffs2image 工具制作 Yaffs2 文件系统，手工输入烧写命令 Key2440＞nand write. Yaffs2 0x30000000 0x50000 0x3625170，烧写完毕后会提示：

NAND write：device 0 offset 0x50000，size 0x3625170

skip the first good block 0x3838600

Bad block at 0x1040000 in erase block from 0x1040000 will be skipped

Bad block at 0x1060000 in erase block from 0x1060000 will be skipped

……..

Writing data at 0x0x3838600－100% complete

52348128 bytes written：OK

说明 Yaffs2 文件系统已成功下载到 NAND Flash 中。

5 结束语

Yaffs2 文件系统与大容量 NAND Flash 的结合，能够加快文件系统的加载速度，实现错误检测、坏块处理以及可靠地掉电保护。编译和测试后的结果表明改进后的 U-Boot 可以将 Yaffs2 文件系统写入到大容量 NAND Flash 中，完善了 U-Boot 的文件烧写功能。改进后的 U-Boot 已经成功移植到正在开发的工业用嵌入式手持设备中，这样可以在大容量 NAND Flash 中使用 Yaffs2 文件系统，为设计开发人员带来了方便，同时产品的性能也得到了提升，在嵌入式系统的开发过程中具有实用价值。

参考文献：

[1] WU Yuxiang，ZHOU Jianxiang，GUO Jianxun. Porting and function expansion of U-boot based on S3C2410 [J]. Computer Engineering and Design，2010，31 (14)：729-732 (in Chinese). [吴玉香，周建香，郭建勋. U-Boot 在 S3C2410 上的移植及功能扩展 [J]. 计算机工程与设计，2010，31 (14)：729-732.]

[2] Lech Józwiak，Nadia Nedjah，Miguel Figueroa. Modern development methods and tools for embedded reconfigurable systems：a survey [J]. Integration the VLSI Journal，2010，43 (1)：1-33.

[3] LI Qingcheng，SUN Mingda. Design of NAND flash memory-based embedded file system [J]. Application Research of Computers，2006，23 (4)：231-233. (in Chinese). [李庆诚，孙明达. 基于 NAND 型闪存的嵌入式文件系统设计 [J]. 计算机应用研究，2006，23 (4)：231-233.]

[4] WEI Dongshan. Complete guide to embedded Linux application development [M]. Beijing：Posts & Telecom Press，2008：283-287 (in Chinese). [韦东山. 嵌入式 Linux 应用开发完全手册 [M]. 北京：人民邮电出版社，2008：283-287.]

[5] Micron Company. Small block vs. large block NAND flash devices [EB/OL]. [2007-01-20]. http：//www.micron.com.

[6] LI Jun. Detailed embedded Linux device driver development [M]. Beijing：Posts & Telecom Press，2008：308-310 (in Chinese). [李俊. 嵌入式 Linux 设备驱动开发详解 [M]. 北京：人民邮电出版社，2008：308-310.]

[7] CHUNG Taesun，PARK Dong joo，PARK Sangwon，et al. A survey of flash translation layer [J]. Journal of Systems Architecture，2009，55 (5-6)：332-343.

[8] ZHANG Jian，WANG Jin. Approach to protect data on tax-controlled system with ucLinux [J]. Computer Engineering and Design，2006，27 (16)：3055-3057 (in Chinese). [张健，王锦. 基于 ucLinux 税控系统的数据保护方案设计 [J]. 计算机工程与设计，2006，27 (16)：3055-3057.]

[9] SUN Feng，ZHANG Fuxing. Resarch and improvement of YAFFS file system [J]. Computer Engineering，2008，34 (5)：257-259 (in Chinese). [孙丰，张福新. YAFFS 文件系统的研究与改进 [J]. 计算机工程，2008，34 (5)：257-259.]

[10] WEI Feng，LU Zaiqi，LIU Wei. Realization of YAFFS2 in the embedded system [J]. Modem Electronics Technique，2010，33 (8)：30-34 (in Chinese). [韦峰，卢再奇，刘伟. YAFFS2 在嵌入式系统中的实现 [J]. 现代电子技术，2010，33 (8)：30-34.]

[11] LONG Yachun，HUANG Pu，WU Sheng. Create a YAFFS2 bases on super-large NAND flash in Linux [J]. Journal of Beijing Electronic Science and Technology Institute，2007，15 (2)：80-84 (in Chinese). [龙亚春，黄璞，吴胜. 超大容量 NAND Flash 文件系统—YAFFS2 在 Linux 下的实现 [J]. 北京电子科技学院学报，2007，15 (2)：80-84.]

[12] Wookey. YAFFS2 specification and development notes [EB/OL]. [2005-05-23]. http：//www.aleph1.co.uk/node/38.

[13] CAI Yong，PENG Fushi. Research on NAND flash system YAFFS [J]. Journal of Zhengzhou University of Light Industry (Natural Science)，2007，22 (6)：54-58 (in Chinese). [蔡勇，彭福石. NAND 闪存文件系统 YAFFS 的研究 [J]. 郑州轻工业学院学报（自然科学版），2007，22 (6)：54-58.]

[14] LEE Kiyong，KIM Hyojun，WOO Kyounggu，et al. Design and implementation of MLC NAND flash-based DBMS for mobile devices [J]. Journal of Systems and Software，2009，82 (9)：1447-1458.

[15] LU Chihyuan，Hsieh Kuangyeu，LIU Rich. Future challenges of flash memory technologies [J]. Microelectronic Engineering，2009，86 (3)：283-286.

[16] The U-boot transplantation mini2440 detailed manual [R]. Guangzhou：Friendly Arm Technology Co Ltd，2010：75-79 (in Chinese). [mini2440 之 U-boot 移植详细手册 [R]. 广州：友善之臂科技有限公司，2010：75-79.]