君正®Libra开发板

软件手册

版本: 1.0

日期: 2006 年 4 月

北京君正集成电路有限公司

Ingenic Semiconductor Co. Ltd

君正 **Libra** 开发板

软件手册

Release history

Date	Revision	Change
Apr. 2006	1.0	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

北京君正集成电路有限公司

北京市海淀区上地东路 1 号
盈创动力 E 座 801C
Tel: 86-10-58851003
Fax: 86-10-58851005
Http: //www.ingenic.cn

内容

1 概述

Libra(天秤座)是嵌入式处理器 JZ4730 的开发板。您可以用它验证 JZ4730 中集成的功能部件。帮助您开发自己的产品。

1.1 Libra 的硬件配置

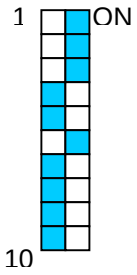
- CPU: JZ4730
- SDRAM: 64MB
- NOR-Flash: Intel TE28F128 (16MB)
- NAND-Flash: SAMSUNG K9F2808U0C (16MB)
- MMC/SD card slot: 1
- LCD: SAMSUNG LTP400WQ (480x272 TFT)
- Audio: AC97 (Philips UCB1400) / I2S (TI TSC2301)
- Camera: OV7111
- PS/2 Keyboard
- USB Host: 2
- USB device: 1
- Network: 10/100M Ethernet
- RS232: 2
- ISO7816: 2
- RTC: Philips PCF8563

1.2 Libra 的软件系统

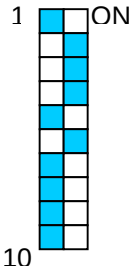
- Boot loader: U-Boot-1.1.3 / NAND-Boot
- OS Kernel: CELinux-040503
- Demo Application: Qtopia-free-1.6.1

1.3 运行 Libra

为 Libra 接上 12V 电源，打开开关即可。如果您想使用键盘，请您在开机之前接好 PS/2 键盘。鼠标只能通过 USB 口接入。Libra 的缺省配置是从 NAND 闪存启动的。系统包含一个简单的 LINUX 根文件系统、Qtopia™和使之运行的基本环境。通过拨动 SW1 开关，您可以选择从 NOR-Flash 启动。



Booting from NAND-Flash



Booting from NOR-Flash

有个媒体文件放在了/opt 目录下，您可以用 Qtopia 里的文件管理器(File Manager)打开它(双击)。

2 开发环境

我们推荐使用基于 LINUX 的软件开发环境。请公司的系统管理员配置运行一系列服务器程序：DHCP 服务、NFS 服务和 TFTP 服务。注意，DHCP 服务的使用应该加倍小心，不要和公司的其他设备冲突。

2.1 安装 软件开发工具链

在我们提供的光盘里，有专门用于进行软件开发的工具链。我们提供支持 LINUX 和 Windows 上 cygwin 的工具链。您只需要按照光盘上的说明，逐步安装即可。

```
$ cd /home
$ mkdir mipstools
$ cd mipstools
$ tar xfj mipseltools-jz-linux-yyyymmdd.tar.bz2
$ cd
$ vi /etc/profile
export PATH=$PATH:/home/mipstools/mipseltools-yyyymmdd/bin
```

2.2 安装 uudecode

请您确认您的开发环境里有 uudecode 工具。该工具在 shareutils 的 RPM 安装包里。如果您的开发环境是 cygwin，那么您也许需要下载一个 shareutils 源代码来自己编译生成 uudecode.

2.3 安装 JTAG 工具

请参考我们专门介绍 JTAG 工具的文档。

2.4 安装 NFS-ROOT

NFS 服务已经配置并且运行了，您可以从我们的光盘里找到一个 ROOT 包，把它解开放在 NFS 共享出去的目录里。

3 引导程序

在 Libra 中有两个引导程序可供产品开发人员使用：专门用作调试系统从 NOR 闪存启动的 U-Boot 和从 NAND 闪存启动的 NANDBoot。

U-Boot 是为嵌入式平台开发而提供的开放源代码的引导程序，其前身是专为 PowerPC 开发的 PPCBoot。U-Boot 功能强大，提供串行口下载、以太网下载等多种下载方式，同时提供 NOR 闪存和 NAND 闪存存取管理，以及 USB 驱动和文件系统等的支持。而且 U-Boot 简单易学，配置和编译过程比较简单，非常适合开发人员使用。

NANDBoot 是我们自己开发的 NAND 闪存引导程序，它的二进制映像不足 4096 字节，放在 NAND 闪存物理第一个块里。使用它，可以极快地引导系统。

3.1 编译生成 U-Boot 的二进制映像

首先，从我们的技术网站的相关链接下载两个包：U-Boot 的源代码“u-boot-1.1.3.tar.bz2”和我们提供的补丁“u-boot-1.1.3-jz-yyyymmdd.patch.gz”。然后按下面的步骤给源码打上补丁：

```
$ tar xfj u-boot-1.1.3.tar.bz2
$ cd u-boot-1.1.3
$ zcat ../u-boot-1.1.3-jz-yyyymmdd.patch.gz | patch -p1
```

接下来，使用 make 命令配置 U-Boot：

```
$ make libra_config
$ make
```

用户可以根据需要来修改自己的配置，Libra 平台的配置文件在 u-boot-1.1.3/include/configs/libra.h 里。

3.1.1 使用 U-Boot

U-Boot 的二进制映像烧录到 NOR 闪存后，启动目标板，在串口输出的命令行里运行“help”命令，用户可以看到 U-Boot 可以支持的所有命令，运行“help command”可以查看具体命令的格式和使用方法。如果想从网络下载 Linux 核心并运行，可以参考下面的步骤：（这里假设您的服务器 IP 地址为 10.0.0.1，并且服务器已经启动了 BOOTP，TFTP 和 NFS 服务）

```
LIBRA # setenv bootfile uImage
LIBRA # setenv bootargs mem=64M console=ttyS0,115200 ip=bootp
nfsroot=10.0.0.1:/nfsroot/mipsroot rw
LIBRA # setenv ethaddr 00:12:34:56:78:90
LIBRA # bootp
LIBRA # setenv serverip 10.0.0.1
LIBRA # tftp
LIBRA # bootm
```

察看当前的环境变量，使用 printenv 命令。设置新的环境变量使用 setenv 和 askenv 命令。保存新设置的环境变量到 Flash 中，使用 saveenv 命令。

3.1.2 Linux™命令行参数

U-Boot 的命令行参数采用设置环境变量的方式传给 Linux 核心，用户只需要设置一个环境变量 bootargs 即可。编译生成的 U-Boot 二进制映像设置了缺省的环境变量，用户可以修改 Libra 的配置文件来修改这些设置。

3.1.2.1 指明内存容量

开发板上有 64M 字节的 SDRAM，所以命令行参数应该这样设置：

```
mem=64M
```

如果 SDRAM 的容量是 16M 字节，那么设置中的 64 应该改成 16。并非所有的内存都给用户程序使用。LINUX™的内存将被系统核心和 init RAMDISK 占用一部分。如果用户使用动态链接编译应用程序，那么系统内存还将被动态链接库占用一部分。

3.1.2.2 系统控制台

系统控制台的实现语法是"console=设备，设备控制参数"，一个系统中可以有多个控制台。最后一个声明的是主控制台。系统控制台用来显示输出来自系统核心的一些启动信息。如果您不想显示这些信息，可以使用"quiet"来屏蔽它们。下面是两个常用的控制台设备：

```
console=tty1
console=ttyS0,115200
```

第二个例子中的 115200 是指 ttyS0 的波特率。而 ttyS0 在 LINUX™指的是串口 0。

3.1.2.3 IEEE802.3 网络设备 MAC 地址

MAC 地址是全球唯一的，需要向专门的管理机构申请。我们可以通过命令行参数来向核心传递 MAC 地址的值，例子如下：

```
ethaddr=00:12:34:56:78:90
```

3.1.2.4 IP 地址

我们可以静态的指定 IP 地址，也可以通过 BOOTP 来获得动态的 IP 地址。LINUX 支持 ARP, BOOTP 和 DHCP 等 3 种方式。在一些资料里，DHCP 本质是 BOOTP 的扩展，理论上两者兼容。但是，在实际的应用中，两者是有区别的。

```
ip=bootp
ip=192.168.9.200:::::eth0
```

3.1.2.5 LINUX 根设备

任何 LINUX™系统都需要一个根文件系统。而这个根文件系统可以放在任意介质上，这个介质就是我们通常锁说的根设备。常用的根设备是 MTD,INITRD 和 NFSROOT。我们在附件光盘里提供了一些根文件系统包，您可以选择一个适合自己的作为自己的作为自己的工作根目录。但是，它们可能也不适合您。请您根据我们提供的资料或者来自互联网的免费资料自行制作。

使用闪存上建立的 mtd 设备

```
root=/dev/mtdblock0 rw
```

使用 NFSROOT

```
nfsroot=192.168.9.20:/nfsroot/developroot rw
```

使用 INITRD 时，应该先把 INITRD 的映像文件下载到 SDRAM 里，然后使用下面的命令行参数：

```
initrd=0x00700000,0x008e8dff root=/dev/ram0 rw
```

当然，我们还支持把 INITRD 编译到核心里。这时，命令行参数就成了：

```
root=/dev/ram0 rw
```

3.2 NANDBoot

NANDBoot 的任务是完成系统初始化工作，把 LINUX 核心映像或 INITRD 映像从 NAND 闪存里加载到系统内存里。然后传入命令行参数并且运行 LINUX 核心。需要用户关心的是系统初始化工作和 LINUX 命令行参数。这些和用户的最终产品密切相关。

3.2.1 系统初始化

系统初始化分为 3 个主要步骤：

- 时钟和电源管理单元的初始化，时钟和电源管理单元控制着处理器的工作时钟和功能模块的开启关闭状态。依据您的设计，选择合适的工作时钟。
- GPIO 功能管脚初始化，片上设备的简单初始化。处理器的功能管脚都可以复用作 GPIO 管脚，而这些 GPIO 管脚既可以做 I/O 也可以做中断输入用。
- SDRAM 初始化。SDRAM 初始化过程较为复杂，依据相关硬件规范，一方面要设置正确的时序参数，另一方面要按要求的状态机变换过程初始化 SDRAM 芯片。

3.2.2 引导映像

我们提供了 Libra 的 NANDBoot 源代码。您可以自己修改它。有个需要注意的是引导映像。在源代码中，我们使用 58 来标示它的长度。而这个长度的单位是“块”。您可以从使用的 NAND 闪存的数据手册里找到“块”的尺寸参数。当然，这个参数就是我们所说的“删除尺寸”。另外两个参数是起始块号和加载地址。在我们提供的源代码里，我们只有一个引导映像，它就是 LINUX 核心。不要太去关心起始块号和加载地址。您需要关心的是您的核心的实际长度。

3.2.3 Linux™命令行参数

在我们提供的 NANDBoot 源代码里，您可以找到有关 Linux™命令行部分。NANDBoot 提供的最大命令行长度为 256 字节。您可以按照我们前面提到过的命令行细节来组织自己的命令行。

3.3 用 JTAG 工具烧录二进制映像

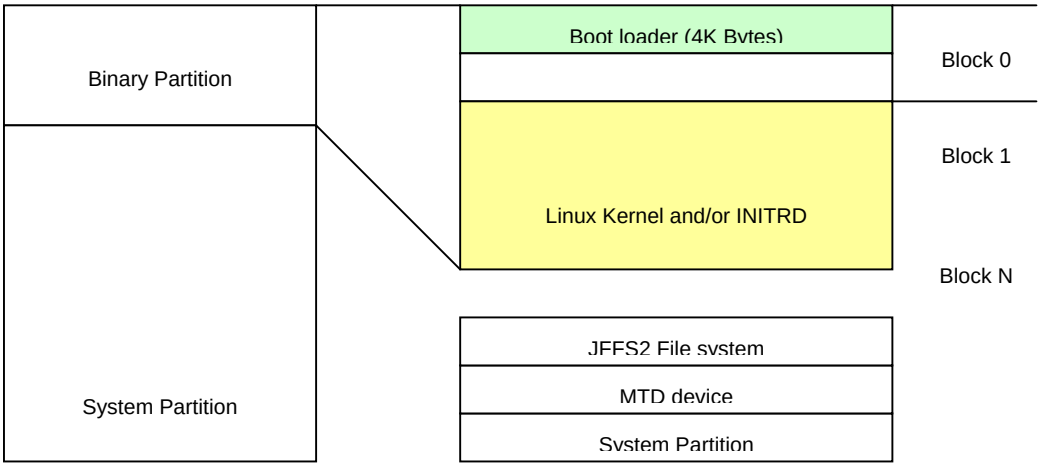
用户可以使用 JTAG 工具在线烧录 U-Boot 和 NANDBoot 的二进制映像到目标板的 NOR 闪存和 NAND 闪存里。具体使用 JTAG 工具的帮助可以参考 JTAG 工具的专门说明文件。

4 NAND 闪存的使用

由于 JZ4730 芯片内部提供 NAND 闪存的引导机制，使得用户可以把自己的引导程序放在 NAND 闪存的最开始 4K 字节空间里。系统在启动时会先执行这 4K 字节引导程序，由这 4K 字节引导程序完成系统初始化，加载系统映像，完成最后的引导。如何划分 NAND 闪存的空间？我们如何来使用这些空间？本章将展示一些 NAND 闪存引导的细节。分为两部份：二进制分区和系统分区。二进制分区专门用作 NAND 闪存引导。而系统分区则是用来做文件系统使用的。我们提供 MTD 设备来建立 JFFS2 文件系统。

4.1 NAND 闪存的分区

NAND 闪存被划分为两部分：一部分是二进制分区，而另一部分为系统分区。二进制分区用来存放 NANDBoot，LINUX 系统核心和/或 INITRD 映像。系统核心用来作 LINUX MTD 设备。我们在上面建立好了一个 JFFS2 文件系统。下面的图表显示了一些分区细节。



4.2 利用 JTAG 烧录 NAND 闪存

如前所述，我们的 JTAG 工具提供了对 NAND 闪存的烧录功能。这个功能对整个 NAND 闪存是有效的，所以您在使用的时候要非常小心，要确保该操作仅仅在二进制分区范围内，不要误写系统分区，否则会损坏已经写入系统分区里的 JFFS2 文件系统。

4.3 利用 nandutil 烧录 NAND 闪存

我们也提供在目标板上运行的 NAND 闪存烧录程序"nandutil"。"nandutil" 具有和 JTAG 烧录工具类似的功能。当您把系统分区用作 MTD 设备时，请遵循 MTD 设备的常规操作流程。

4.3.1 烧录 NANDBoot

```
# nandutil -b loader.bin
```

4.3.2 烧录 CE-Linux™系统核心

```
# nandutil -b zImage -s 1
```

4.3.3 如果系统有 INITRD，并且放在 NAND 闪存的块 52 开始的地方

```
# nandutil -b initrd.bin -s 52
```

4.3.4 从 block 0 开始连续擦除 1024 个 block

```
# nandutil -e 1024
```

如果系统的 NAND 闪存是从旧货市场买来的，普通的擦除可能会失败。原因是我们避开了 NAND 闪存出厂时标注的坏块，而使用过的 NAND 闪存可能已经把好的块也给标注成坏块了。您可以使用附加参数-f 来强行擦除所有的块，包括已经标注成坏块的块。这里有个问题：强行擦除操作也会擦掉 NAND 闪存出厂时标注的坏块标志，这使得真正的坏块有机会混入好的块里，给将来的使用埋下隐患。

4.4 MTD 设备

MTD (Memory Technology Device) 是 LINUX™的标准设备。这也是嵌入式应用开发工程师非常熟悉的设备。上面可以创建 JFFS2 文件系统。在基于 NAND 闪存的 MTD 设备上创建 JFFS2 文件系统的步骤如下：

格式化 MTD 分区：

```
# flash_eraseall -j /dev/mtd2
```

mount 文件系统并安装目标程序：

```
# mount -t jffs2 /dev/mtdblock2 /mnt
# cd /mnt
# cp /home/destdir/* . -a
# sync
# cd /
# umount /mnt
```

如果您的 NAND 闪存以前写入过数据，那么请用我们提供的 nandutil 进行强行擦除。如下：

```
# nandutil -e 1024 -f
```

5 CE-Linux™系统核心

CE-Linux™是面向消费电子类应用的嵌入式 Linux™。它在启动速度、核心尺寸和电源功耗管理支持方面都有明显的改善。

5.1 基本的核心开发流程

5.1.1 解压缩源代码，并且打上补丁

```
$ tar xfj celinux-040503.tar.bz2
$ cd celinux-040503
$ zcat ../celinux-jz-yyyymmdd.patch.gz | patch -p1
```

5.1.2 配置核心

打补丁后的 CE-Linux 具有一个缺省的配置文件称为：".config"。按照下面的方法，我们可以把这个文件填上我们需要的配置项，不需要用户到配置菜单里自己选择了。

```
$ make defconfig-libra
$ make oldconfig
```

5.1.3 编译核心

```
$ make dep
$ make zImage 或者 make uImage
```

"make zImage" 最后会生成 3 个核心映像：

- 带有调试信息的 ELF 格式的非压缩核心。它放在：
  `celinux-040503/vmlinux`
- 压缩的 ELF 格式的核心。它放在：
  `celinux-040503/arch/mips/zboot/images/vmlinuz.elf`
- 压缩的二进制格式的核心。它放在：
  `celinux-040503/arch/mips/zboot/images/zImage`

这三个核心映像各有各的用途：第一个用于反汇编和调试核心用；第二个用于支持 ELF 格式的引导程序；第三个用于 NANDBoot。请用户加倍注意！

"make uImage" 最后也会生成 3 个核心映像：

- 带有调试信息的 ELF 格式的非压缩核心。它放在：
  `celinux-040503/vmlinux`
- 压缩的 uImage 核心。它放在：
  `celinux-040503/arch/mips/uboot/uImage`
- 非压缩的 uImage 核心。它放在：
  `celinux-040503/arch/mips/uboot/uImage.uncompressed`

5.2 Libra 的 CE-Linux™设备驱动程序

5.2.1 实时时钟 RTC

我们使用的 RTC 时钟是挂接在 I2C 总线上的 PCF8563。用户可以使用"hwclock"来设置它或者获取里面的值。

```
Character devices
   [●]y [ ]- [ ]n Philips PCF8563 Real Time Clock (I2C Bus)
```

5.2.2 智能卡驱动

我们的智能卡驱动程序被设计成为一种数据通信通道，它不提供 ISO7816 协议的翻译和控制。但是它提供给用户一些控制接口：您可以设置额外保护时间，速率因子和选择字符传输或块传输。请您参考我们提供的应用示例。我们建议您在写和智能卡相关的应用之前，请仔细阅读有关 ISO7816-3 的标准规范文本。

智能卡的设备文件有两个，它们的文件 I-node 属性是：字符设备，MAJOR(238), MINOR(32 和 33)。

```
Character devices
  [●]y [ ]- [ ]n JzSOC char devices support
  JzSOC char devices support
    [●]y [ ]- [ ]n JzSOC On-chip SmartCard Controller support
```

如果您的目标板的根文件系统中没有这些设备文件，请您用 mknod 创建。下面的例子是创建 ISO7816 设备文件节点：

```
# mknod /dev/scc0 c 238 32
# mknod /dev/scc1 c 238 33
```

5.2.3 MTD 设备驱动

我们提供 NAND 闪存和 NOR 闪存的 MTD 驱动。您可以在这两个 MTD 分区上创建 JFFS2 文件系统。NOR 闪存是"/dev/mtd1", NAND 闪存是"/dev/mtd2"。特别的，"/dev/mtd0"表示整个 NOR 闪存。请您不要在 mtd0 上作任何操作，以免损毁我们已经写入的 U-Boot。

```
Memory Technology Devices (MTD)
   [●]y [ ]m [ ]n Memory Technology Device (MTD) support
   [●]y [ ]m [ ]n  MTD partitioning support
   [●]y [ ]m [ ]n  Direct char device access to MTD devices
   [●]y [ ]m [ ]n  Caching block device access to MTD devices
   RAM/ROM/Flash chip drivers
      [●]y [ ]m [ ]n Detect flash chips by Common Flash Interface
(CFI) Probe
```

```
            [●]y [ ]m [ ]n Flash chip driver advanced configuration
options
            [●]y [ ]m [ ]n  Specific CFI Flash geometry selection
            [●]y [ ]m [ ]n  Support 16-bit buswidth
            [●]y [ ]m [ ]n  Support 1-chip flash interleave
            [●]y [ ]m [ ]n Support for Intel/Sharp flash chips
         Mapping drivers for chip access
            [●]y [ ]m [ ]n JZ4730 LIBRA MTD support
         NAND Flash Device Drivers
            [●]y [ ]m [ ]n NAND Device Support
            [●]y [ ]m [ ]n NAND Flash device on JzSOC board
```

在 MTD 设备上使用 JFFS2 文件系统是许多嵌入式系统的经典选择。您只需要使用下面的配置：

```
      File systems
         [●]y[ ]m[ ]n Journalling Flash File System v2 (JFFS2) support
         [0        ]  JFFS2 debugging verbosity (0 = quiet, 2 = noisy)
         [●]y[ ]-[ ]n JFFS2 support for NAND chips
```