君正 **USB Boot** 工具用户手册

Revision: 1.4
Date: Feb 2009

北京君正集成电路有限公司
Ingenic Semiconductor Co. Ltd

君正 **USB Boot** 工具用户手册

Release history

Date	Revision	Change
2009.02.17	1.4	Upgrade to support JZ4750 and multi-planes NAND operations.
2008.10.12	1.0	Created.

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

北京市海淀区东北旺西路 **8** 号中关村软件园信息中心 **A** 座 **108** 室
Tel: 86-10-82826661
Fax: 86-10-82825845
Http: //www.ingenic.cn

内容

1 概述

USB Boot 工具是北京君正集成电路有限公司研发的具有特色的开发、生产工具。可用于产品研发阶段构建开发环境，也可用于产品出厂时的大批量烧录和产品售后升级。USB Boot 工具将尽最大的努力不断升级完善，为用户提供更稳定、更便捷的功能，来进一步提高君正处理器平台的易用性。

USB Boot 1.4 工具有以下特点：

- 基于 USB 来传输数据，具有较高的数据传输速度和烧录速度；
- 工作条件简单；
- 一台主机可以同时操作多个设备；
- 支持君正处理器芯片 Jz4740 和 Jz4750；
- 稳定支持页大小为 512/2048/4096 的 Nand Flash，并且支持 two plane 方式的烧录；
- 支持两种基本的开发板测试功能；

使用者请务必留意本文档的说明，尽量按照约定使用，从而减少因为误操作带来的麻烦，提高用户开发效率，也减少君正公司对重复问题支持的精力和时间。

2 名称约定

以下是本文使用的一些名称和术语的含义，只列举本文单独使用的名称，其他专业术语与原意相同。

主机：运行着 windows XP/2K 的兼容微型计算机，要求带有 USB 接口。

设备：搭载了支持 USB Boot 功能的君正处理器的开发板、产品板或者最小系统。

数据区：这是从驱动程序角度划分的区域，对应与 Nand Flash 物理结构中的一个区域。该区域的大小由 Nand Flash 的结构决定。驱动程序用这个区域存放实际数据。

OOB 区：与数据区相对应，可以理解为紧跟在数据区后面的一个小的区域，大小与数据区有固定的比例关系。用来存放由实际数据产生的 ECC 信息和文件系统的额外信息。

第一阶段：主机方程序对设备 PLL，SDRAM 和串口进行初始化的阶段，对应的代码成为第一阶段代码，在 cache 中运行一次然后返回。

第二阶段：能够响应用户命令，完成烧录、测试等实际功能的阶段，对应的代码成为第二阶段代码，运行在 SDRAM 中，是常驻的服务程序。

Boot 操作：指使设备进入可以交互状态的操作。实际就是主机方程序让第二阶段代码在设备上正常运行的整个过程。

复位状态：设备从 usb device 复位后的状态。

Boot 状态：设备在复位状态进行 Boot 操作之后的状态。

文件类型：指执行烧录命令时，待烧录文件的数据组织结构。这个结构与 Nand Flash 的存储结构相对应，分为无 OOB、有 OOB 无 ECC 和有 OOB 有 ECC 三种。

3 使用准备

3.1 基本准备

USB Boot 1.4 工具可以工作在 Windows XP/2000 平台上，主机 USB Host 接口支持 2.0 和 1.1。USB Boot 1.4 工具包括以下文件。

- usb_boot.exe，这是运行在主机上的 win32 命令行应用程序，用户通过该程序获得所需要的功能。
- fw.bin，这是设备方第一阶段的可执行代码。进行 Boot 操作的初期，主机方的程序会把该文件里的代码通过 USB 发送到设备上执行。
- usb_boot.bin，这是设备方第二阶段代码，这部分代码实现烧录等具体功能。Boot 操作的后期，主机方的程序会把该文件里的代码通过 USB 发送到设备上执行。
- Usb_Boot_Driver.sys，Windows 驱动程序的执行代码，只有安装了这个驱动之后，USB Boot 的功能才能正常使用。
- Usb_Boot_Driver.inf，Windows 驱动程序配置信息，与 Usb_Boot_Driver.sys 配套。
- USBBoot_PAVO.cfg 和 USBBoot_APUS.cfg，这是主机方的程序运行依赖的配置文件，分别对应 PAVO 板和 APUS 板，用户可以随时修改。使用时要将 USBBoot_PAVO.cfg 或者 SBBoot_APUS.cfg 重新命令为 USBBoot.cfg。

除了以上资源，使用 USB Boot 1.4 工具还需要准备 USB 电缆和配置文件。

3.2 配置文件

配置文件非常重要，主机方的程序通过配置文件获取大部分关键信息，如果配置文件设置错误，必然导致之后的操作产生错误。

用户请格外注意，配置文件必须随时根据设备的变化修改，在执行操作之前，必须保证配置文件中的信息与设备开发板的具体情况一致。

配置文件的**模板**如下：

```
;-----------------------------------------------------------
; USB boot sample configuration file for Jz4750 APUS board
;-----------------------------------------------------------

[PLL]
EXTCLK          24          ;Define the external crystal in MHz
CPUSPEED        336         ;Define the PLL output frequency
```

```
PHMDIV          3       ;Define the frequency divider ratio of
PLL=CCLK:PCLK=HCLK=MCLK
BOUDRATE        57600   ;Define the uart boudrate
USEUART         3       ;Use which uart, 0/1 for jz4740,0/1/2/3 for jz4750

[SDRAM]
BUSWIDTH        16      ;The bus width of the SDRAM in bits (16|32)
BANKS           4       ;The bank number (2|4)
ROWADDR         12      ;Row address width in bits (11-13)
COLADDR         9       ;Column address width in bits (8-12)
ISMOBILE        0       ;Define whether SDRAM is mobile SDRAM, this only valid for
Jz4750 ,1:yes 0:no
ISBUSSHARE      1       ;Define whether SDRAM bus share with NAND 1:shared
0:unshared

[NAND]
BUSWIDTH        8       ;The width of the NAND flash chip in bits (8|16|32)
ROWCYCLES       3       ;The row address cycles (2|3)
PAGESIZE        4096        ;The page size of the NAND chip in bytes(512|2048|4096)
PAGEPERBLOCK    128         ;The page number per block
FORCEERASE      1       ;The force to erase flag (0|1)
OOBSIZE         128     ;oob size in byte
ECCPOS          24      ;Specify the ECC offset inside the oob data (0-[oobsize-1])
BADBLACKPOS     0       ;Specify the badblock flag offset inside the oob (0-[oobsize-1])
BADBLACKPAGE    127         ;Specify the page number of badblock flag inside a
block(0-[PAGEPERBLOCK-1])
PLANENUM        1       ;The planes number of target nand flash
BCHBIT          8       ;Specify the hardware BCH algorithm for 4750 (4|8)
WPPIN           0       ;Specify the write protect pin number
BLOCKPERCHIP    0       ;Specify the block number per chip,0 means ignore

[END]

;The program will calculate the total SDRAM size by : size = 2^(ROWADDR + COLADDR) * BANKS *
(BUSWIDTH / 4)
;The CPUSPEED has restriction as: ( CPUSPEED % EXTCLK == 0 ) && ( CPUSPEED % 12 == 0 )
;For jz4750, the program just init BANK0(DSC0).
;Beware all variables must be set correct!
```

USB Boot 1.4 的配置文件包括三个段：PLL、SDRAM 和 NAND，分别定义与 PLL，SDRAM 和 NAND Flash 相关的信息，段的顺序不能颠倒。其中 PLL 段和 SDRAM 段会影响 Boot 过程，NAND 段影响烧录。

3.2.1 PLL 段

PLL 段定义频率相关信息：

- EXTCLK：指定当前电路板 CPU 主晶体的频率，对于 Jz4740 应该是 12MHz，对于 Jz4750 可以是 12，13，19.2，24，26，27MHz 中的一个。注意，如果主晶体的频率是 19.2MHz，请把 EXTCLK 的值设置为 19。
- CPUSPEED：指定 Boot 之后 CCLK 的工作频率，注意这个频率必须能被 12 和 EXTCLK 整除，否则无法正确升频。
- PHMDIV：指定 Boot 之后 PCLK，HCLK，MCLK 与 CCLK 工作频率的比率。3 则表示 1：3：3。例如，EXTCLK=12，CPUSPEED=336，PHMDIV=3，那么 PCLK=HCLK=MCLK=112MHz=1：3=CCLK= CPUSPEED=336MHz。
- BOUDRATE：指定 Boot 之后串口的波特率，如果没有特殊，请将该值设置为 57600。
- USEUART：指定使用哪个串口作为输出。Jz4740 有两个串口，应该是 0 或者 1；Jz4750 有四个串口，可以 0，1，2，3。

3.2.2 SDRAM 段

SDRAM 段定义当前开发板 SDRAM 的情况，注意，该段必须正确设置，否则不能成功 Boot。该段的定义将动态反映到 fw.bin 中，如果 SDRAM 的大小改变了，不需要重新编译 fw.bin，只需要修改配置文件，然后重新 Boot 即可。

- BUSWIDTH：定义 SDRAM 工作的数据宽度，32bit 或者是 16bit。如果实际硬件的 SDRAM 是 32bit 方式，BUSWIDTH 设为 32 和 16 都能正常工作；如果实际硬件的 SDRAM 是 16bit 方式，BUSWIDTH 必须设为 16。
- BANKS：定义 SDRAM 的 bank 个数，取值为 2 或者 4，这个值请参考所使用 SDRAM 的手册。对于 4bank 的 SDRAM，BANKS 可以设成 4 或者 2，都能正常工作；对于 4bank 的 SDRAM，BANKS 必须设成 2。
- ROWADDR：定义 SDRAM 的 row 地址位数，请参考所使用 SDRAM 的手册。
- COLADDR：定义 SDRAM 的 col 地址位数，请参考所使用 SDRAM 的手册。
- COLADDR：定义 SDRAM 的 col 地址位数，请参考所使用 SDRAM 的手册。
- ISBUSSHARE 该项只对 jz4750 有效，指定当前 CPU 是否工作在 bus share 模式。

USB Boot 1.4 主机方程序将根据 BUSWIDTH，BANKS ，BANKS，COLADDR 这四个值计算当前设备的 SDRAM 的总大小，然后根据这个值确定第二阶段代码的执行地址。如果计算出来的值超过了设备实际 SDRAM 的大小，将导致第二阶段代码无法执行。计算的公式如下：

SDRAM size = 2^(ROWADDR + COLADDR) * BANKS * (BUSWIDTH / 8)

例如：

BUSWIDTH=32，BANKS=4，ROWADDR=13，COLADDR=9；则 SDRAM size=64MB；

BUSWIDTH=16，BANKS=2，ROWADDR=13，COLADDR=9；则 SDRAM size=16MB。

3.2.3 NAND 段

NAND 段定义与 Nand Flash 相关的信息，这些信息影响烧录过程，请确保这些设置与设备的 Nand Flash 相符合，否则会导致烧录错误。

- BUSWIDTH：定义 Nand 访问的数据宽度，可以是 8bit 或者 16bit 方式。
- ROWCYCLES：定义命令周期数，取值为 2 或者 3，该值请根据 Nand Flash 手册设定。
- PAGESIZE：定义 Nand Flash 的页大小，取值为 512，2048 或者 4096，该值请根据 Nand Flash 手册设定。
- PAGEPERBLOCK：定义 Nand Flash 每一块包含多少个页，取值为 32，64，128，该值请根据 Nand Flash 手册设定。
- FORCEERASE：指定是否强制擦除坏块，如果为 1，则忽略坏块标记擦除所有的块；如果为 0，则只擦除没有坏块标记的块。注意，如果该值为 1，则会丢失在使用过程中产生的坏块记录信息。
- OOBSIZE：定义 Nand Flash 的 OOB 区的大小，取值为 16，64，128。
- ECCPOS：定义 ECC 校验信息存放位置在 OOB 区中的偏移，该值的设置由未来读取 Nand Flash 数据的程序决定，如果不一致，将导致烧完之后不能正常使用。例如，Jz4740 Linux MTD 驱动把 ECC 信息放在 OOB 区中偏移 28 的地方，那么烧录文件系统(yaffs2，vfat)的时候必须把这个值设置为 28。Jz4750 Linux MTD 驱动把 ECC 信息放在 OOB 区中偏移 24 的地方，那么烧录文件系统(yaffs2，vfat)的时候必须把这个值设置为 24。细节参考 linux/drivers/mtd/nand_base.c。
- BADBLACKPOS：定义坏块标记在 OOB 区中的偏移，例如，BADBLACKPOS=0 则表示 OOB 区中的第 0 个字节是坏块标记，如果该字节不等于 0xff，则表明这块是坏块。
- BADBLACKPAGE：定义坏块标记在块中的哪一页，例如，BADBLACKPAGE=0 则表示每块的第一页有坏块信息；BADBLACKPAGE=PAGEPERBLOCK 则表示最后页；如果 BADBLACKPAGE>PAGEPERBLOCK，则每块的头尾两页都包含坏块信息。
- PLANENUM ：定义是否使用 two-plane 的方式来烧录。请参考后文。
- BCHBIT：定义 BCH 算法的纠错能力，该值只对 Jz4750 有效，可选值为 4 或者 8。4 表示使用能纠 4 个 bit 错误的 BCH 算法，8 则能纠 8 个 bit。这个值的设定要跟操作系统驱动的设定一致。
- WPPIN：定义写保护管脚 GPIO 的编号。这个数值用来确定一个 GPIO，GPA0 的编号为 0，GPB0 的编号为 32，依此类推。如果这个值非 0，当访问 Nand Flash 的时候，程序会让对应的 GPIO 管脚输出低电平，以关闭写保护，访问结束，对应的管脚输出高。
- BLOCKPERCHIP：这个值暂时不使用，忽略。

3.3 从 USB 启动

使用 USB Boot 1.4 工具，开发板需要以 USB Device 的方式启动。启动最初期的代码已经固化在支持 USB Boot 的芯片内部（bootrom），使用时通过跳线设置就可以选择不同的启动方式。Bootrom 中的代码根据复位时 Boot_sel[0:1]的电位状况选择启动方式，当 Boot_sel[0:1] = 01 时，按下复位开关，处理器以 USB Device 的身份启动。操作时具体如何设置 Boot_sel[0:1]请参考具体开发板的原理图，以君正公司提供的 PAVO v1.3 开发板为例，按住 sw5 不放再按 reset 就可以从 USB Device 启动。

Jz4750 APUS v1.1.2，按下 sw6 不放再按 reset 就可以从 USB Device 启动。

注意只有支持 USB Boot 的芯片才能从 USB Device 启动，Jz4730 不支持 USB Boot 功能，也无法使用本文介绍的所有功能。

后文所说的复位均与此同。

3.4 主机安装驱动程序

以 USB Device 的方式启动的设备第一次连接主机（复位后）需要安装主机方的驱动程序，以后连接不再需要安装驱动。主机方的驱动程序可在工具软件发布包中找到，即 Usb_Boot_Driver.inf 和 Usb_Boot_Driver.sys 这两个文件。安装驱动的方法与其他设备类似，安装成功之后应该能在 windows 的设备管理器中查看到 Usb_Boot 设备。

4 USB Boot 工具命令

经过以上准备，可以开始使用 USB Boot 1.4 工具。USB Boot 1.4 的命令分为三类：帮助信息，烧录功能和测试功能。其中每一类命令执行所需要的的条件不同，下面将详细介绍。

4.1 复位状态与 Boot 状态

Boot 命令是改变设备状态的唯一方法。

设备从 USB 启动并连接到主机之后之后处于复位状态，主机方程序成功执行 Boot 命令之后，设备处于 Boot 状态。处于 Boot 状态的设备如果要返回复位状态，请将设备复位。

Boot 命令的用法如下：

格式：boot dev

其中 dev 为接入主机的设备的编号。如果当前只有一个设备连接，dev 为 0。

烧录类功能必须工作在 Boot 状态，而测试类的功能必须工作在复位状态。如果要使用烧录、下载等功能，设备连接主机之后请首先执行 Boot 命令，然后才能烧录。如果要使用 SDRAM，GPIO 这两个测试功能，请不要执行 Boot 命令。

4.2 帮助信息类命令

该类命令任何时候都可以执行，不依赖与设备的状态。

4.2.1 help 命令

help 命令将打印 USB Boot 1.4 工具目前支持的所有命令列表，如果需要查询某个命令的用法格式，输入该命令并回车即可。

4.2.2 version 命令

version 命令将打印当前 USB_Boot 工具主机方程序的版本。版本编号的约定是，版本数越大越新，后缀为 a 的表明是测试版本，可能有不稳定的地方；后缀为 b 的版本是较稳定的版本，已经经过许多测试。例如：

```
USBBoot :> version
```

```
USB Boot Software current version: 1.4a
USBBoot :>
```

4.2.3 exit 命令

执行该命令从 USB Boot 1.4 工具退出。

4.2.4 list 命令

该命令将探测当前有多少个 USB_Boot 设备连接，可以用该命令验证设备是否成功从 USB Device 启动。例如：

```
USBBoot :> list
Device number can connect :0
USBBoot :> list
Device number can connect :2
```

4.3 烧录功能类命令

设备必须处在 Boot 状态才能使用这类命令。

4.3.1 nprog 命令

nprog 命令用于烧录，是最常用的命令。该命令能将三种类型的文件烧录到 Nand Flash 的指定位置中，烧录过程中带有校验。

格式：nprog sp filename dev 0 filetype

sp：镜像烧录位置的起始页号。
filename：镜像文件名和路径。
dev：接入主机的设备的编号。
filetype：说明要烧录文件的类型。-n 表示无 OOB；-o 表示有 OOB 无 ECC；-e 表示有 OOB 也有 ECC。

文件类型的分类：

1） 无OOB：镜像文件中不包括OOB信息，文件内容的组织方式为连续的数据。编译生成的bootloader 和 kernel 属于这种类型。
2） 有 OOB 无 ECC：镜像文件中包括了 OOB 信息，但是没有包含 ECC 信息，文件内容的组织方式为数据和 OOB 间隔连续。某些文件系统镜像属于这种类型，如 yaffs2，用 mkyaffs2image 生成

的镜像就是这种类型。

3） 有 OOB 也有 ECC：镜像文件中包括了 OOB 信息，也包含 ECC 信息，文件内容的组织方式为数据和 OOB 间隔连续。从 Nand Flash 母片生成的文件属于这种类型。

注意事项：

1） nprog 校验的方法是将数据从 Nand Flash 中读回来逐字节对比，如果检验没有报错，则说明烧录没有问题。
2） 如果烧录过程没有报错，但是系统启动之后报错了，很大的可能是烧录方法与操作系统的读取方法不匹配，请仔细检查 ECC，PLANE 等设置。
3） Nand Flash 最开始的 16KB 的烧录方法是固定的，不受配置文件影响。对 Jz4740，最开始的 16KB 代码使用 RS ECC，ECC 存放的偏移为 6。对于 Jz4750，最开始的 16KB 代码使用 BCH 8-bit ECC，ECC 偏移为 3。
4） nprog 命令中包含了擦除的操作，烧录之前不需要先进行擦除。nprog 擦除的范围根据烧录文件的大小确定，如果需要擦除比镜像文件更大的空间，就需要先手动擦除。

烧录的实例参考后文。

4.3.2 nquery 命令

nquery 命令用于查询指定设备指定 Nand Flash 芯片的 ID。ID 统一为五个字节，如果实际 Nand Flash 的 ID 少于五个字节，则用 0xff 补充。

格式：nquery dev 0

dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

例如，以下命令查询 PAVO 板上的 Nand Flash 芯片的 ID：

```
USBBoot :> nquery 0 0
ID of No.0 device No.0 flash:
Vendor ID    :0xec
Product ID   :0xd3
Chip ID      :0x14
Page ID      :0x25
Plane ID     :0x64
Operation status: Success!
```

4.3.3 nread 命令

nread 命令用于从 Nand Flash 中指定位置读取指定数量的的数据，读回来的数据经过 ECC 校验。该功能可以用来验证烧录的数据是否真正写入了 Nand Flash 中。

格式：nread sp len dev 0

sp：要读取数据的起始页号。
len：要读取的数据长度，以字节为单位。
dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

例如，下面命令将从 2048 页读取 100 个字节经过校验的数据：

```
USBBoot :> nread 2048 100 0 0
Reading from No.0 device No.0 flash....
0x00000000 :10 80 1f 3c 00 00 ff 27 00 90 80 40 00 98 80 40
0x00000010 :80 00 09 3c 00 68 89 40 40 00 08 3c 00 fc 08 35
0x00000020 :00 60 88 40 03 00 08 24 00 80 88 40 00 80 08 3c
0x00000030 :00 40 09 35 00 e0 80 40 00 e8 80 40 00 00 08 bd
0x00000040 :00 00 09 bd fd ff 09 15 20 00 08 25 00 00 00 00
0x00000050 :07 80 08 40 00 00 00 00 02 00 08 35 07 80 88 40
0x00000060 :00 00 00 00
```

4.3.4 nreadraw 命令

nreadraw 命令用于从 Nand Flash 中指定位置读取指定数量的数据，读回来的数据不经过 ECC 校验。该命令读到的数据是当前存储在 Nand Flash 中最原始的数据，可能包含错误。

格式：nreadraw sp len dev 0

sp：要读取数据的起始页号。
len：要读取的数据长度，以字节为单位。
dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

例如，下面命令将读取 2048 页的裸数据 100 个字节：

```
USBBoot :> nreadraw 2048 100 0 0
Reading RAW from No.0 device No.0 flash....
0x00000000 :10 80 1f 3c 00 00 ff 27 00 90 80 40 00 98 80 40
0x00000010 :80 00 09 3c 00 68 89 40 40 00 08 3c 00 fc 08 35
0x00000020 :00 60 88 40 03 00 08 24 00 80 88 40 00 80 08 3c
0x00000030 :00 40 09 35 00 e0 80 40 00 e8 80 40 00 00 08 bd
0x00000040 :00 00 09 bd fd ff 09 15 20 00 08 25 00 00 00 00
0x00000050 :07 80 08 40 00 00 00 00 02 00 08 35 07 80 88 40
0x00000060 :00 00 00 00
```

4.3.5 nreadoob 命令

nreadoob 命令用于从 Nand flash 中指定位置读取指定数量的 OOB 区数据，每次只能读取一页，超过 OOB 大小的数据用 0 补充。该命令可以用于确认 ECC 存放的位置。

格式：nreadoob sp len dev 0

sp：要读取数据的起始页号。
len：要读取的数据长度，以字节为单位。
dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

例如，下面的命令将读取 2048 页的 OOB 数据：

```
USBBoot :> nreadoob 2048 100 0 0
Reading OOB from No.0 device No.0 flash....
0x00000000 :ff ff 00 00 00 ff f8 33 d6 19 f0 b0 a5 58 a7 52
0x00000010 :53 e4 13 8d f4 c9 d6 c1 35 97 f9 e2 ee 0d aa 36
0x00000020 :0e d3 0b e2 ca 9c 74 ac 1f d0 ff ff ff ff ff ff
0x00000030 :ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x00000040 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000060 :00 00 00 00
```

从这次返回的数据不难看出，当前 ECC 的存放位置从 OOB 区偏移 6 的位置开始，ECC 总共 36 个字节。

4.3.6 nerase 命令

nerase 命令用于擦除 Nand Flash 中指定范围的数据，该操作以块为单位。

格式：nerase sb bn dev 0

sb：要擦除的起始块号。
bn：要擦除的块数。
dev：接入主机的设备的编号。

该命令以要擦除的块数为准则，也就是说擦除过程中如果出现坏块，该坏块不计数，以保证实际擦除的有效块数与 bn 相符。如果配置文件中指定了强制擦除坏块，则擦除命令不提供坏块个数的信息。

例如：

```
USBBoot :> nerase 16 5 0 0
Erasing No.0 device No.0 flash..... Finish!
Operation end position : 21
Force erase ,no bad block infomation !
```

这个命令将从第 16 块开始，连续擦 5 块，有坏块则跳过，最后实际擦除的块数一定是 5 块。

4.3.7 load 命令

load 用于将主机方的代码或者数据文件下载到设备的 SDRAM 的指定位置中，该命令配合 go 命令可用于将可执行的代码下载到 SDRAM 中运行，在产品研发阶段，可以用这个方法调试程序，而不必把程序烧录到 Nand Flash 中。

格式：load saddr filename dev

saddr：SDRAM 的起始地址，操作结束之后镜像文件的数据从该地址开始存放。
filename：主机方镜像文件的文件名，包括路径，下同。
dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

注意事项：

1） SDRAM 地址从 0x80000000 开始向后延伸，镜像文件大小不应该超过 SDRAM 容量。
2） SDRAM 中最后的 4MB 空间用于存放 USB Boot 第二阶段程序，不应将数据写入该区域。

例如，当前目录有 3.4MB 大小的 zImage，将其下载到 SDRAM 中的 6MB 偏移的地方，使用下面命令：

```
USBBoot :> load 0x80600000 zImage 0
Total size to send in byte is :3403477
Loading data to SDRAM :
#################################################################
#################################################################
#################################################################
#################################################################
####################################
```

4.3.8 go 命令

go 命令用于执行 SDRAM 中的代码，该命令将直接跳转到执行内存空间，配合 load 命令使用。

格式：go saddr dev

saddr：SDRAM 的地址，跳转的入口地址。
dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

注意事项：

1） 这个命令的操作很简单，直接跳转到 saddr，所以从 saddr 开始的代码必须直接可执行，go 命令不能传递参数。
2） SDRAM 地址从 0x80000000 开始，依次向后延伸。
3） 跳转之后设备会断开连接，脱离 USB Boot 的范畴。

例如，跳转执行 SDRAM 中 6MB 偏移的代码：

```
USBBoot :> go 0x80600000 0
Executing No.0 device at address 80600000 !
```

4.4 测试功能类命令

这类命令在 USB Boot 1.4 中包括两个，这类命令要在复位状态下使用。这类命令不依赖于 SDRAM，即使 SDRAM 不能正常工作，这两个功能也能执行。

4.4.1 memtest 命令

memtest 命令用于测试当前设备上的 SDRAM 是否能正常访问。测试的方法是对指定 SDRAM 空间按 word 方式进行读写测试，先写后读。

格式：memtest dev [saddr] [len]

dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。
saddr：该参数可选，用于指定要测试的 SDRAM 起始地址。
len：该参数可选，用于指定测试范围，相对于 saddr 的长度。

注意事项：

1） saddr 和 len 这两个参数可选，如果有必须同时出现。如果没有则以配置文件中指定的 SDRAM 大小为准。
2） 以下两种情况都说明内存测试失败，一是主机程序打印失败信息，二是主机程序出现假死，即停止响应。后者是因为测试代码没有能够执行到向主机返回测试结果就发生了未知异常。

例如，对 PAVO 板上的 64MBSDRAM 进行读写测试：

```
USBBoot :> memtest 0
```

```
Now test memory from 0 to 4000000:
Test memory pass!
```

4.4.2 gpios/gpioc 命令

gpios 和个 gpioc 命令分别用于让指定的 GPIO 输出高和输出低。

格式：gpios pin dev
gpioc pin dev

pin：GPIO 的顺序编号，约定 GPA0 的编号为 0，GPB0 的编号为 32，依此类推。
dev：接入主机的设备的编号，如果当前只有一个设备连接，dev 为 0。

注意，pin 的值不应该大于 GPIO 个数的总和。Jz4740 和 Jz4750 的编号方法相同。

例如，下面的命令让 GPC20 输出高电平：

```
USBBoot :> gpios 84 0
GPIO 84 set!
```

5 烧录实例

本节介绍实际的烧录方法，本节将以君正网站发布的 Linux BSP 的整个烧录过程为例，具体介绍烧录工具和命令的用法。下面涉及到的配置文件的改动都是针对君正发布的 Linux BSP，如果是别的系统，或者是修改过的 Linux BSP，这些配置需要根据实际情况修改。硬件方面，本节以君正的参考设计 PAVO 开发板和 APUS 开发板为例。

请用户务必明确，烧录的根本原则是：烧录的设置（ECC 偏移，ECC 算法，plane 方式，坏块标记位置等）要与读取烧录内容是程序匹配。例如，烧录 bootloader 的设置要与芯片内 bootrom 代码定义匹配；烧录 kernel 要与 bootloader 的定义匹配；烧录文件系统要与 kernel 的文件系统驱动匹配。

5.1 Boot 操作

从君正网站下载 USB Boot 1.4 工具包并解压。

5.1.1 PAVO 板

PAVO 开发板搭载 Jz4740 处理器，PAVO 板从 usb device 启动的方法比较简单：

1） 按住 sw5 按键不放的同时，按下复位键。这时 PAVO 板将从 usb device 启动，然后用 usb 电缆将 PAVO 板与 PC 连接，这时候在 PC 上应该能看到一个新的 usb 设备。如果第一次使用，请根据 windows 的提示安装驱动程序
2） 运行 usb_boot.exe 程序，如果连接已经成功，执行 list 命令，应该能看到设备个数非 0。
3） PAVO 板上的 SDRAM 为 64MB，为了适应其他小内存的开发板这里按 16MB 来使用，Nand Flash 型号为 K9G8G08XX，将工具包中的 USBBoot_PAVO.cfg 文件重命名为 USBBoot.cfg。USBBoot_PAVO.cfg 包含了针对 PAVO 板的默认配置。
4） 执行 Boot 命令，应该看到以下信息：

```
USBBoot :> boot 0
Checking state of No.0 device: Unboot
Now booting No.0 device:
Download stage one program and execute at 0x80002000: Pass
Download stage two program and execute at 0x80c00000: Pass
Boot success!
Now configure No.0 device:
Now checking whether all configure args valid:
Current device information: CPU is Jz4740
Crystal work at 12MHz, the CCLK up to 336MHz and PMH_CLK up to 112MHz
Total SDRAM size is 16 MB, work in 4 bank and 16 bit mode
```

```
Nand page size 2048, ECC offset 28, bad block ID 0, use 1 plane mode
Configure success!
```

5.1.2 APUS 板

APUS 开发板搭载 Jz4750 处理器，APUS 板从 usb device 启动的方法稍微有些不同：

1） 首先根据当前开发板上晶体的频率，进行跳线的设置，跳线的设置方法与 Jz4750 芯片的版本有关，具体请参考 bootrom 文档。如果开发板上有 RTC 晶体，则可以跳过这一步。
2） 按住 SW6（APUS v1.1.2）的同时，按下复位键。具体按哪些按键与 APUS 板的版本有关，请参考对应的原理图。如果成功，在 PC 上应该能看到一个新的 usb 设备。
3） 运行 usb_boot.exe 程序，如果连接已经成功，执行 list 命令，应该能看到设备个数非 0。
4） APUS 板上的 SDRAM 为 64MB，为了适应其他小内存的开发板这里按 16MB 来使用，Nand Flash 型号为 K9G8G08XX，将工具包中的 USBBoot_APUS.cfg 文件重命名为 USBBoot.cfg。USBBoot_APUS.cfg 包含了针对 PAVO 板的默认配置。
5） 执行 Boot 命令，应该看到以下信息：

```
USBBoot :> boot 0

Checking state of No.0 device: Unboot
Now booting No.0 device:
Download stage one program and execute at 0x80002000: Pass
Download stage two program and execute at 0x80c00000: Pass
Boot success!
Now configure No.0 device:
Now checking whether all configure args valid:
Current device information: CPU is Jz4750
Crystal work at 24MHz, the CCLK up to 336MHz and PMH_CLK up to 112MHz
Total SDRAM size is 16 MB, work in 4 bank and 16 bit mode
Nand page size 4096, ECC offset 24, bad block ID 127, use 1 plane mode
Configure success!
```

5.2 烧录 Bootloader

Linux 使用 uboot 为 bootloader，其他系统同理。从 uboot 源码中编译生成 u-boot-nand-pavo.bin 和 u-boot-nand-apus.bin 两个文件。通过下面方法分别烧录到 PAVO 和 APUS 板中。

PAVO 板使用 USBBoot_PAVO.cfg 的配置即可，执行命令：

USBBoot :> nprog 0 u-boot-nand-pavo.bin 0 0 –n

APUS 板使用 USBBoot_APUS.cfg 的配置即可，执行命令：

USBBoot :> nprog 0 u-boot-nand-apus.bin 0 0 –n

5.3 烧录 Kernel

从 Linux 的源码树编译生成 kernel 文件 uImage_pavo 和 uImage_apus，分别对应 PAVO 板和 APUS 板。然后通过下面方法烧录。

PAVO 板使用 USBBoot_PAVO.cfg 的配置即可，执行命令：

USBBoot :> nprog 2048 uImage_pavo 0 0 –n

APUS 板使用 USBBoot_APUS.cfg 的配置即可，执行命令：

USBBoot :> nprog 1024 uImage_apus 0 0 –n

5.4 烧录 YAFFS2 文件系统

烧录 yaffs2 文件系统的步骤分为两个：制作镜像和烧录。

首先准备根文件系统，假设放在 root26 目录中。在 Linux 操作系统上通过下面的命令制作 yaffs2 文件系统镜像：

PAVO： # mkyaffs2image 1 root26 root26.yaffs2 (2k 页)
APUS： # mkyaffs2image 2 root26 root26.yaffs2 (4k 页)

之后在当前目录获得 root26.yaffs2 这个文件，对于 PAVO 该文件的结构是 2KB 数据和 64 字节 OOB 交替，对于 APUS 该文件的结构是 4KB 数据和 128 字节 OOB 交替。均属于有 OOB 无 ECC 的类型。

mkyaffs2image 第一个参数会依据 nand 参数和分区信息修改为 1、2 或 3，执行./mkyaffs2image 可查看帮助信息。

然后修改配置文件，Nand Flash 相关参数按照附录的列表修改。修改之后请务必再执行 Boot 命令，来更新配置信息。然后通过下面的命令烧录到 MTD 第二个分区：

PAVO 板使用 USBBoot_PAVO.cfg 的配置，执行命令：
USBBoot :> nprog 4096 root26.yaffs2 0 0 –o

APUS 板使用 USBBoot_APUS.cfg 的配置，执行命令：
USBBoot :> nprog 2048 root26.yaffs2 0 0 –o

5.5 烧录其它文件系统

其他文件系统的烧录也要两个过程：制作镜像和烧录。

由于文件系统的特点不同，制作镜像的过程可能不一样。有的可以通过专门的工具，有的需要通过母片的方式。无论哪种方法，最后都生成的镜像文件都必须归类到三种文件类型当中。USB Boot 工具只能烧录符合文件类型约定的镜像。

例如 FAT 文件系统，通过母片的方式把数据完整读出来，生成文件 root26.fat。以 PAVO 为例，这个文件的结构是 2KB 数据和 64 字节 OOB 交替，并且 OOB 中包含了 ECC 信息，属于有 OOB 又有 ECC 的类型。

然后修改配置文件，Nand Flash 相关参数按照附录的列表修改。修改之后请务必再执行 Boot 命令，来更新配置信息。然后通过下面的命令烧录到 MTD 最后一个分区：

```
USBBoot :> nprog 262144 root26.fat 0 0 -e
```

5.6 Two-Plane 烧录

Two-plane 方法能够提高 Nand Flash 的读写性能，目前 Two-plane 方式只用在文件系统部分。之前介绍的烧录方法都是 single-plane 的，two-plane 使用的文件系统镜像与 single-plane 不同，两者不能混用。

Two-plane 的文件系统镜像是 single-plane 的两倍。例如，如果 Nand Flash 的物理规格是 2KB+64BOOB，则 single-plane 的镜像结构是 2KB 数据和 64BOOB 交替，而 two-plane 的镜像结构是 4KB 数据和 128BOOB 交替。如果 Nand Flash 的物理规格是 4KB+128BOOB，则 single-plane 是 4KB 数据和 128BOOB 交替，而 two-plane 是 8KB 数据和 265BOOB 交替。USB Boot 工具以 two-plane 方式烧录的时候会把镜像文件的数据进行拆分，例如将 4KB 数据和 128BOOB 交替的文件拆分为两个 2KB 数据和 64BOOB，烧录到 Nand Flash 中。

以 yaffs2 文件系统为例，对于物理规格是 2KB+64BOOB 的 Nand Flash，用以下命令：

```
# mkyaffs2image 2 root26 root26_4k.yaffs2
```

命令中的第一个参数为 2，这样生成的 root26_4k.yaffs2 是 4KB 数据和 128BOOB 交替的。

然后修改配置文件，Nand Flash 相关参数按照附录中 4KB 页的内容来修改。修改之后请务必再执行 Boot 命令，来更新配置信息。然后通过下面的命令烧录到 MTD 第二个分区：

```
USBBoot :> nprog 4096 root26_4k.yaffs2 0 0 –o
```

6 烧录原理简介

6.1 Boot 流程介绍

Boot 过程需要经历两个阶段。其中第一个阶段负责初始化第二阶段程序运行的环境，第二阶段才能提供烧录等功能。

USB Boot 1.4 版本的 Boot 过程详细如下：

1） 用户执行 boot 命令。
2） 主机方程序扫描配置文件，检查配置文件的正确性，然后设置第一阶段的参数。
3） 主机方程序将从配置文件中获得的参数和 fw.bin 一起发送给目标开发板。
4） 目标开发板在 cache 中执行 fw.bin，fw.bin 读取参数，初始化目标开发板。
5） 目标开发板从 fw.bin 返回，主机检查第一阶段是否运行成功。
6） 主机程序根据当前目标开发板的 SDRAM 大小计算第二阶段的执行位置，执行位置位于 SDRAM 中的最后 4M，然后将 usb_boot.bin 发送到目标板 SDRAM 中的执行位置。
7） usb_boot.bin 在目标板 SDRAM 中执行，其最初的代码根据执行位置修改程序本身的 GOT 表，使之能够在当前地址执行。
8） GOT 修改完毕，usb_boot.bin 跳转到主函数入口，usb_boot.bin 正式运行。
9） 主机程序将完整的配置文件参数通过第二阶段协议发送给目标板，目标板记录这些设置。
10） 主机程序验证 usb_boot.bin 是否正常运行，Boot 过程结束

1.4 版本的 Boot 过程更加清晰，哪一步运行不正常都能通过打印正确反映出来。

6.2 硬件测试流程介绍

1.4 版本提供了硬件测试功能，分别是内存测试和 GPIO 测试，这两个功能可以在仅有 CPU 的情况下使用。即使 SDRAM 不能正常工作，这两个功能也能执行。

6.2.1 内存测试流程

该更能用来测试 SDRAM 是否能够正常工作，测试的方法是检验 SDRAM 访问的正确性。内存测试的流程如下：

1） 主机程序将测试代码发送到目标开发板的 cache 中执行。
2） 测试代码首先根据配置文件对 SDRAM 进行初始化，但是不进行升频。
3） 测试代码对指定范围的 SDRAM 按 word 对齐方式进行读写测试，并将测试结果返回给主机。
4） 主机处理测试结果，测试流程结束。

6.2.2 GPIO 测试

GPIO 测试功能相对简单，通过命令可以把选定的 GPIO 管脚拉高或者拉低。GPIO 测试的流程如下：

1） 主机程序将测试代码发送到目标开发板的 cache 中执行。
2） 测试代码将选定的 GPIO 设置为输出功能。
3） 测试代码将选定的 GPIO 拉高或者拉低。
4） 测试流程结束。

注意该功能是没有交互过程的。

7 常见问题

以下是常见问题以及解决的思路，出现问题请首先参考本文档检查，如仍未解决，可以尝试修改USB Boot 1.4 的源码。如发现程序漏洞请及时联系：yliu@ingenic.cn。

问题 1：Boot 不成功

Boot 不成功的表现有多种，请依次检查以下内容：

- 首先通过 list 命令检查设备是否正确识别。
- 检查配置文件中的 PLL 段，设置是否与实际硬件相符合，是否是合法的值。
- 检查配置文件中的 SDRAM 段，设置是否与实际硬件相符合，所得到的 SDRAM 大小有没有超过实际大小。
- 检查板级硬件，确认 CPU，晶体，供电，SDRAM，USB 通讯等都工作正常。
- 检查主机方驱动是否安装了最新的，例如主机之前已经安装了其他公司定制的驱动，再使用本工具，建议把旧的驱动彻底卸载，然后安装新驱动。

问题 2：烧录过程频繁报告校验错误

烧录过程中频繁报告 Check Error 这样的错误，例如每个块都出错，请依次检查下列内容：

- 首先检查配置文件中的 NAND 段，设置是否与实际硬件相符合。
- 检查配置文件中的 ECCPOS 的值，是否跟镜像文件的结构相符合。例如有 OOB 无 ECC 的文件类型。
- 检查硬件是否有写保护，相应设置 WPPIN 的值。
- 检查板级硬件，确认 NAND 能够正常工作。
- 最后可能是时序问题，即程序的时序太快了，可以通过修改配置文件 CPUSPEED 的值降低主频。

问题 3：烧录过程偶然报告校验错误

烧录过程中大多数的块都烧录校验通过，只有极其少数的块出现 Check Error，这个问题有以下可能原因：

- 如果出错的地方总是出现在最后几个块，有可能是非常特殊的块没有擦除，请手动将这几个块擦除再重新烧录。
- 有可能是某些块中出现不可纠正的错误，这种情况程序能够处理。

问题 4：烧录成功，但是无法启动

烧录过程没有报错，但是启动之后报错，无法启动，或者出现各种各样的错误。这是由烧录程序写的方式与 boorom，bootloader 和内核驱动程序等读的方式不匹配引起的。从以下方面检查：

- 首先检查配置文件中的 NAND 段，设置是否与实际硬件相符合。
- 然后检查 ECCPOS，BADBLACKPOS，BADBLACKPAGE，PLANENUM 和 BCHBIT 是否与读的程序相符。
- 检查烧录文件本身、烧录的命令、烧录的位置。

问题 5：烧录文件系统成功，但是无法正常启动文件系统

这个问题是前一个问题的特例，重点检查烧录文件本身的正确性、ECCPOS。

问题 6：烧录 yaffs2 文件系统成功，文件系统成功启动，但是启动过程报告错误

烧录 yaffs2 文件系统成功，并且启动也成功，但是启动过程频繁报告 yaffs tragedy: attempting to use non-directory as a directory in scan. Put in lost+found 这样的错误。解决方法是将整个 yaffs2 分区完整地擦除。

8 附录

在 PAVO 和 APUS 板烧录各种目标文件时烧录工具的默认配置表：

在君正 Jz4740 的 pavo 开发板和 Jz4750 的 apus 开发板上使用 K9G8G08U0B(2KB 页大小)或 K9GAG08U0M(4KB 页大小)的 NAND Flash 烧录各种目标文件时烧录工具的默认配置表如下：

其中,参数 PLANENUM 和 OOBSIZE 只在 usb boot 工具配置文件中存在;JDI 只支持单 plane 操作,oobsize 根据页大小自动算出。

参数 OOB 和 OOBECC 只在 JDI 工具中存在；在 usb boot 工具中，被烧录文件是否包含 OOB,以及 OOB 中是否包含 ECC 是由命令行参数指定。

	PAVO 板								APUS 板									
	2KB 页 NAND				4KB 页 NAND				2KB 页 NAND					4KB 页 NAND				
	uboot	uImage	yaffs2	vfat	uboot	uImage	yaffs2	vfat	uboot	uImage	uImage.oob	yaffs2	vfat	uboot	uImage	uImage.oob	yaffs2	vfat
BUSWIDTH	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
ROWCYCLES	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
PAGESIZE	2048	2048	2048	2048	4096	4096	4096	4096	2048	2048	2048	2048	2048	4096	4096	4096	4096	4096
PAGEPERBLOCK	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
FORCEERASE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECCPOS	28	28	28	28	28	28	28	28	24	24	24	24	24	24	24	24	24	24
BADBLACKPOS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BADBLACKPAGE	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127
BCHBIT	-	-	-	-	-	-	-	-	4	4	4	4	4	8	8	8	8	8
PLANENUM	1	1	1	2	1	1	1	2	1	1	1	1	2	1	1	1	1	2
OOBSIZE	64	64	64	64	128	128	128	128	64	64	64	64	64	128	128	128	128	128
OOB	0	0	1	1	0	0	1	1	0	0	1	1	1	0	0	1	1	1
OOBECC	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1