# USB Boot Tool User Manual

Revision: 1.4
Date: Feb 2009

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

# USB Boot Tool User manual

**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.

**Ingenic Semiconductor Co. Ltd**

**Room 108, Information Center Block A**<br>
**Zhongguancun Software Park**<br>
**8 Dongbeiwang west Road, Haidian District**<br>
**Beijing China, 100193**

**Tel: 86-10-82826661**<br>
**Fax: 86-10-82825845**<br>
**Http: //www.ingenic.cn**

# CONTENTS

# 1 Overview

The USB boot tool is developed by Ingenic. It’s useful for the product development and production. With the USB boot tool, you can test the hardware problem and program the NAND flash. It can also be used to upgrade the firmware of your product. All you need is an XBurst-based microprocessor with USB boot functionality supported.

The USB boot tool includes the following features：

- Support Jz4740 and Jz475x.
- Transfer and receive data through USB with high speed.
- One host can control multiple devices simultaneously.
- And one device can connect one or more NAND flashes.
- Support Nand Flash page size 512, 2048 and 4096, and two-planes programming.
- Provide simple hardware test functions.

# 2 Name Agreement

Host: A computer running windows XP/2K, and require at least one USB host port.

Device: A development board or product using Ingenic CPU.

Data area: An area in NAND Flash physical structure, used to store the actual data. Its size can be 512B, 2048B and 4096B.

OOB area: An area in NAND Flash physical structure, following the data area, used to store ECC and filesystem information. Its size is 1/32 of the data area size.

Stage 1: The first stage of the USB boot tool work cycle. The tool initializes the PLL, SDRAM and URAT of the device at this stage.

Stage 2: The second stage of the USB boot tool work cycle. The tool receives commands from host and handles these commands at this stage. Stage 2 will never exit until reset.

Boot operation: An operation that can make device into stage 2.

Reset stage: The stage when the device reset and boot from the usb device.

Boot stage: The stage after boot operation.Also the stage 2 in other words.

File type: The file type in the view of the USB boot tool. There are three types of files: NO OOB, WITH OOB BUT NOT ECC, BOTH OOB AND ECC.

# 3 Tutorial

## 3.1 Preparation

The current USB boot tool can just be run on the Windows XP/2000/Vista host. It supports the USB high speed transfer. The USB boot tool package includes following files:

- usb_boot.exe: it is a win32 application running on the PC host.
- fw.bin: it is the binary code of stage 1. The host application will download it to the device when executing the boot operation.
- usb_boot.bin: It is the binary code of stage 2. The host application will download it to the device after the boot operation and then never return.
- Usb_Boot_Driver.sys: it is the Windows driver of the USB boot tool. You have to install this driver when you use the tool for the first time to make the tool work normally. The host application will send commands to the device via the driver.
- Usb_Boot_Driver.inf: is the driver configuration file.
- USBBoot.cfg: is the configuration file for the USB boot tool. The configuration file specifies parameters of the target board, including PLL, SDRAM, UART, NAND etc. You must ensure that all the parameters are configured correctly, or the USB boot tool can't work normally.

## 3.2 Configuration File

The host application usb_boot.exe will get most critical information from the configuration file. You must keep the settings in the configuration file consistent with the device before any operations, otherwise it will cause unkown errors. The configuration file can be modified at any time. Once it was changed, another boot operation is needed.

This is a template of the configuration file:

```
;-------------------------------------------------------------------------------
; USB Boot configuration file for the Jz4740 or Jz4750 board
;-------------------------------------------------------------------------------
[PLL]
EXTCLK            12          ;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             0               ;Use which uart, 0/1 for jz4740,0/1/2/3 for jz4750

[SDRAM]
SDRAMWIDTH          32              ;Define the SDRAM bus width 32/16
BANKNUM             4               ;Define the SDRAM bank number 2/4
ROWADDR             13              ;Define the SDRAM row address 11 to 13
COLADDR             9               ;Define the SDRAM row address 8 to 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            2048            ;The page size of the NAND chip in bytes
PAGEPERBLOCK        128             ;The page number per block
FORCEERASE          1               ;The force to erase flag (0|1)
OOBSIZE             64              ;oob size in byte
ECCPOS              6               ;ecc position
BADBLACKPOS         0               ;bad black ID position
BADBLACKPAGE        127             ;bad black ID page
PLANENUM            1               ;The planes number of target nand flash
BCHBIT              4               ;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) *
BANKNUM * (SDRAMWIDTH / 8)
;The CPUSPEED has restriction as: ( CPUSPEED % EXTCLK == 0 ) && ( CPUSPEED % 12 == 0 )
;For jz4750, the program just init BANK0(DSC0).
;Beware all settings must be set correctly!
```

There are three sections in the configuration file: PLL, SDRAM and NAND.The information in the PLL section and the SDRAM section will affect the boot process, while information in the NAND section will affect the NAND operation.

### 3.2.1 PLL Section

The PLL section defines the information about frequency:

- EXTCLK: This variable defines the external crystal frequency. It should be set to 12MHz for Jz4740. For Jz4750, it can be 12, 13, 19.2, 24, 26 or 27MHz. If the external crystal frequency is 19.2MHz, please set EXTCLK to 19.
- CPUSPEED: This variable defines the CCLK frequency after boot operation. It means CPU will work at CPUSPEED MHz after boot operation. Notice, its value must be divisible by 12 and EXTCLK.
- PHMDIV: This variable defines the ratio between PCLK，HCLK，MCLK and CCLK after boot operation. For example, 3 means 3：3：3：1.If EXTCLK=12，CPUSPEED=336，PHMDIV=3，then PCLK=HCLK=MCLK=112MHz=1：3=CCLK= CPUSPEED=336MHz after boot operation.
- BOUDRATE: This variable defines the baud rate after boot operation.It should be set to 57600 under normal circumstances.
- USEUART: This variable defines which UART is used for output. It can be 0 or 1 for Jz4740 while 0, 1, 2 or 3 for Jz4750.

### 3.2.2 SDRAM Section

The SDRAM section defines the SDRAM configuration of the device. This section must be set correctly, otherwise the device can not be booted successfully.The settings in SDRAM section will be passed to the device while doing the boot operation.

- SDRAMWIDTH：This variable defines the bus width of SDRAM. It can be 32bit or 16bit.
- BANKNUM：This variable defines the bank number of SDRAM. It can be 2 or 4 banks.
- ROWADDR：This variable defines the row addess of SDRAM.
- COLADDR：This variable defines the column addess of SDRAM.
- ISBUSSHARE：This variable only used for Jz4750. Defines whether SDRAM bus share with NAND.

The USB boot tool will calculate the total size of SDRAM by values of SDRAMWIDTH, BANKNUM, BANKNUM and COLADDR. If the total size of SDRAM given by the configuration file exceeds the actual size of device, then the program of stage 2 can not work.

The formula is as follow:

SDRAM size = 2^(ROWADDR + COLADDR) * BANKNUM * (SDRAMWIDTH / 8)

For example:

SDRAMWIDTH=32，BANKNUM=4，ROWADDR=13，COLADDR=9；then SDRAM size=64MB；
SDRAMWIDTH=16，BANKNUM=2，ROWADDR=13，COLADDR=9；then SDRAM size=16MB。

### 3.2.3 NAND Section

The NAND section defines the information of Nand Flash.

- BUSWIDTH: This variable defines the bus width of Nand Flash interface. It can be 8bit or16bit.
- ROWCYCLES: This variable defines the row cycles of commands. It can be 2 or 3.
- PAGESIZE: This variable defines the page size of Nand Flash. It can be 512, 2048 or 4096.
- PAGEPERBLOCK: This variable defines the page number of a block. It can be 32, 64 or 128.
- FORCEERASE: This variable defines whether force erase the bad block or not.1 means yes while 0 means no. if this variable was set to 1, the bad block information will lost after an erase operation.
- OOBSIZE: This variable defines the OOB area size of Nand Flash. It can be 16, 64 or 128.
- ECCPOS: This variable defines the start position of the stored ECC information in OOB area. Its value is from 0 to (OOBSIZE – 1).Its value is determined by the program who will read the data in this Nand Flash in the future.
- BADBLACKPOS: This variable defines the bad block information position in OOB area. Its value is from 0 to (OOBSIZE – 1).
- BADBLACKPAGE: This variable defines which page within a block contains the bad block information. Its value is from 0 to (PAGEPERBLOCK – 1).
- PLANENUM: This variable defines use two-plane mode or not.1 means yes and 0 means no.
- BCHBIT: This variable defines using which BCH algorithm.It is only used for Jz4750, and it can be 4 or 8.
- WPPIN: This variable defines the write-protect pin number. It is hardware related. If there is not write-protect function in device, then just set it to 0.
- BLOCKPERCHIP: This variable is not used now.

## 3.3 Boot from USB device

The device needs to boot from USB device before using USB boot tool. The program in the CPU bootrom will select the boot mode through Boot_sel[0:1] at reset. When Boot_sel[0:1] = 01, the process will boot from USB device. For example: For the PAVO v1.3 dev broad,hold on sw5 and reset, the CPU will boot from USB device.

Notice, Jz4730 can not boot from USB device. So the USB boot tool can’t be used for Jz4730.

The word ‘reset’ in this document means reset and boot from USB device.

## 3.4 Install host windows driver

It needs to install the driver when the device first connects to PC after reset.The driver is given by the two files: Usb_Boot_Driver.inf and Usb_Boot_Driver.sys.

# 4 USB Boot Tool Commands

This section lists all commands supported by the USB boot tool. There are three types of commands: help commands, task commands and test commands.

## 4.1 Reset and Boot Stages

The ‘boot’ command is the only way to change the device from reset stage to boot stage.

The device is placed at the reset stage when it is first connected with the host. After executing the ‘boot’ command from the host, the device is transformed to the boot stage. At the boot stage, you can perform the task commands to do the actual jobs. If you want to restore the device to reset stage, please hardware reset it.

The usage of the ‘boot’ command is as below:

Format: boot dev

Where dev is the device number that are connected to the host. If there is only one device, the dev value is 0.

If you want to run the task commands, you should run the ‘boot 0’ command at first. If you want to run the test commands, please don’t run it.

## 4.2 Help Commands

The help commands can be run at anytime.

### 4.2.1 ‘help’ Command

The ‘help’ command will print out all of the commands supported by current USB boot tool. And ‘help xxx’ will show the usage of a specific command.

### 4.2.2 ‘version’ Command

Run ‘version’ command to query the version of the tool. For example:

```
USBBoot :> version
```

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

### 4.2.3 ‘exit’ Command

Run ‘exit’ command to quit from the USB boot tool.

### 4.2.4 ‘list’ Command

Run ‘list’ command to detect how many devices connected to the host. You can use this command to check whether the device is booted successfully from USB boot mode. For example:

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

## 4.3 Task Commands

All of the task commands must be run when the device is under the boot stage. See the ‘boot’ command to learn how to put the device at the boot stage.

### 4.3.1 ‘nprog’ Command

The ‘nprog’ command is used to program the NAND flash. It is the most frequently used commad. It can be used to program three types of binary files to the NAND flash with the Error Code Correction (ECC) and the write verification.

Format: nprog sp filename dev flsh filetype

sp: is the starting page number, the first page is 0.
filename: is the binary filename
dev: is the device number connected to the host, if there is only one device, dev is 0.
flsh: is the flash number on the board, if there is only one flash on the board, flsh is 0.
filetype: is the binary type. There are three types can be supported currently. ‘-n’ means the binary file just includes page data but not OOB data. ‘-o’ means the binary includes both the page data and the OOB data, but the OOB data does not include the ECC. ‘-e’ means the binary includes both the page data and the OOB data, and the OOB data includes the ECC.

For example, the boot loader binary (e.g. u-boot-nand.bin) belongs to the first type (‘-n’). And the yaffs2 image generated by the mkyaffs2image belongs to the second type (‘-o’).

Important notes:

1) The ‘nprog’ command will perform the write verification during the programming process. It reads back a whole block data after finishing a block programming operation and check whether it’s correct or not. The programming process will be continued even if the check was failed. You can see the status message on the serial console.
2) The first 16KB data on the first block is special and the programming method is fixed. You can’t change it by modifying the configuration file. This is decided by the CPU boot mode. For JZ4740, the first 16KB data will be programmed with the hardware RS ECC, and the ECC will be placed at the ECCPOS 6. For JZ4750, the first 16KB data will be programmed with the hardware BCH 8-bit ECC, and the ECC will be placed at the ECCPOS 3.
3) The ‘nprog’ command will perform the ‘nerase’ operation at first. It just erases the block that it uses. The remaining blocks of the partition will keep unchanged.

### 4.3.2 ‘nquery’ Command

The ‘nquery’ command is used to query the NAND flash ID.

Format: nquery dev flsh

dev: is the device number connected to the host, if there is only one device, dev is 0.
flsh: is the flash number on the board, if there is only one flash on the board, flsh is 0.

For example:

```
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’ Command

The ‘nread’ command is used to read back the page data from the NAND flash. It will perform the ECC during the read process.

Format: nread sp len dev flsh

sp: is the starting page number, the first page is 0.
len: is the data length in byte.
dev: is the device number connected to the host, if there is only one device, dev is 0.
flsh: is the flash number on the board, if there is only one flash on the board, flsh is 0.

For example:

```
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’ Command

The ‘nreadraw’ command is used to read back the page data from the NAND flash. It will not perform the ECC during the read process.

Format: nreadraw sp len dev flsh

sp: is the starting page number, the first page is 0.
len: is the data length in byte.
dev: is the device number connected to the host, if there is only one device, dev is 0.
flsh: is the flash number on the board, if there is only one flash on the board, flsh is 0.

For example:

```
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’ Command

The ‘nreadoob’ command is used to read back the OOB data from the NAND flash. You can use this command to check the ECC position and ECC data.

Format: nreadoob sp len dev flsh

sp: is the starting page number, the first page is 0.
len: is the data length in byte.
dev: is the device number connected to the host, if there is only one device, dev is 0.
flsh: is the flash number on the board, if there is only one flash on the board, flsh is 0.

For example:

```
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
```

From the data, we can see that the ECC position is from 6 and the ECC data includes 36 bytes totally.

### 4.3.6 ‘nerase’ Command

The ‘nerase’ command is used to erase the NAND flash in block unit.

Format: nerase sb bn dev flsh

sb: is the starting block number, the first block is 0.
bn: is the block count.
dev: is the device number connected to the host, if there is only one device, dev is 0.
flsh: is the flash number on the board, if there is only one flash on the board, flsh is 0.

The command will check the bad block during the operation. It will skip to next block if it meets a bad block. You can specify the FORCEERASE to 1 to tell the command to erase the bad block forcely.

For example:

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

The above command will erase 5 blocks of the NAND flash starting from block 16.

### 4.3.7 ‘load’ Command

The ‘load’ command is used to load code or data to the SDRAM of the target board. The loaded code can then be executed by running the ‘go’ command subsequently.

Format: load saddr filename dev

saddr: is the starting address of the SDRAM.
filename: is the file name to be loaded.
dev: is the device number connected to the host, if there is only one device, dev is 0.

Important notes:

1) The SDRAM base address is always starting from 0x80000000. And the file size should be less than the SDRAM size.
2) The last 4MB space of the SDRAM is reserved and used by the USB boot code. User program should not use it.

For example:

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

### 4.3.8 ‘go’ Command

The ‘go’ command is used to execute the code in the SDRAM. It will jump to the target address directly. It should be run after loading data with the ‘run’ command.

Format: go saddr dev

saddr: is the starting address of the SDRAM.
dev: is the device number connected to the host, if there is only one device, dev is 0.

For example, after loading zImage to 0x80600000, run ‘go’ command to execute it:

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

## 4.4 Test Commands

There are two test commands in current version. They can only be used under reset stage. These types of commands don’t depend on the SDRAM. So even if the SDRAM can’t work normally, you can also use them.

### 4.4.1 ‘memtest’ Command

The ‘memtest’ command is used to test the SDRAM. The command will first write data to SDRAM and then read back to check. This command will help you to test whether the SDRAM works normally or not.

Format: memtest dev [saddr] [len]

dev: is the device number connected to the host, if there is only one device, dev is 0.
saddr: is the starting address of the SDRAM. It is default to 0x80000000.
len: is the length in byte.

For example:

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

### 4.4.2 ’gpios’/’gpioc’ Command

The ’gpios’ and ’gpioc’ commands are used to output a high or low level to the GPIO pin. ’gpios’ will output high level while ’gpioc’ will output low level. These two commands can be used to test and check your hardware problem.

Format:

```
gpios pin dev
gpioc pin dev
```

pin: is the GPIO pin number. Group A is from 0 to 31, Group B is from 32 to 63, and so on.
dev: is the device number connected to the host, if there is only one device, dev is 0.

For example:

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

This command will output a high level to pin 84.

# 5 Table: Configuration Examples

| Board | JZ4740_PAVO Board | | | | | | | | JZ4750_APUS Board | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NAND Type | 2KB PageSize NAND | | | | 4KB PageSize NAND | | | | 2KB PageSize NAND | | | | 4KB PageSize NAND | | | |
| Binary Type | uboot | UImage | yaffs2 | vfat | uboot | uImage | yaffs2 | vfat | uboot | uImage | yaffs2 | vfat | uboot | uImage | yaffs2 | vfat |
| BUSWIDTH | 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 |
| PAGESIZE | 2048 | 2048 | 2048 | 2048 | 4096 | 4096 | 4096 | 4096 | 2048 | 2048 | 2048 | 2048 | 4096 | 4096 | 4096 | 4096 |
| PAGEPERBLOCK | 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 |
| ECCPOS | 6 | 6 | 28 | 28 | 6 | 6 | 56 | 56 | 3 | 3 | 28 | 28 | 3 | 3 | 72 | 72 |
| BADBLACKPOS | 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 |
| BCHBIT | - | - | - | - | - | - | - | - | 8 | 8 | 4 | 4 | 8 | 8 | 4 | 4 |
| PLANENUM | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 |
| OOBSIZE | 64 | 64 | 64 | 64 | 128 | 128 | 128 | 128 | 64 | 64 | 64 | 64 | 128 | 128 | 128 | 128 |
| OOB | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| OOBECC | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |