VT8500 U-BOOT USER GUIDE

Jan.09, 2008

Document History:

Version	Date	Author	Description
0.1	2008-11-05	Luis Zhou	Initial version
0.2	2008-12-02	Luis Zhou	Add stuff about showing logo in u-boot
0.3	2009-01-09	Luis Zhou	Add stuff about “scriptcmd” “gocmd”, and do some subtle modification

VT8500 U-BOOT USER GUIDE

1. Memory map of VT8500 SPI/LPC flash

Bootloader (v-load & u-boot) will be stored in LPC flash or SPI flash, that depends on what kind of flash a board uses. The u-boot will display the information of SPI flash, such as figure 1.1 displays information 4Mb (512KB) SPI flash and figure 1.2 is log of only LPC flash

```
SPI Flash Bank[0]: Atmel(ID=1f4401), Unprotect it...
SPI Flash Bank[1]: No SPI Flash

SPI Flash: 512 kB
```

Figure 1.1 SPI information

```
SPI Flash Bank[0]: No SPI Flash
SPI Flash Bank[1]: No SPI Flash

ONLY LPC flash or no supported SPI flash
```

Figure 1.2 Log of LPC flash

The start address of SPI/LPC flash (start_flash_addr) is 0xff800000, which is a constant for all vt8500 boards. And the start address of v-load is calculated by the following formula:

$$Addr\_of\_vload = Start\_flash\_addr + size\_of\_flash - size\_of\_one\_sector \quad (1.0)$$

That means, the v-load resides in the last sector of SPI/LPC flash. And the parameter of u-boot resides second last sector which follow v-load. For example, the board is with 8M SPI flash, so the address of v-load is 0xff800000 + 0x800000 – 0x10000 = 0xffff0000. And the address of u-boot parameter is 0xfffe0000 (0xffff0000 – 0x20000).
And the address of u-boot reside in lower 5 sectors from space of parameter of u-boot, So if the start address of u-boot parameter is 0xfffe0000, the start address of u-boot is 0xfff90000. The SPI flash In bank 0 start from 0xff800000, map 8M address space; the SPI flash In bank 1 start from 0xff000000, map also 8M address space. And in this 8M address space, access the address higher than its actual size will loop back. The formula is:

$$Actual\_access\_Addr = start\_addr\_of\_bank + (addr \,\&\, (flash\_size - 1)) \quad (1.1)$$

For example, 512K (0x80000) SPI flash in bank0, accessing 0xff810000 is same as access 0xff800000 + (0xff810000 & (0x80000-1)). Note that, for vt8500, 0xfffe0000 is always true start address of v-load no matter what size is the SPI/LPC flash, for the same reason, 0xfff90000 is always true start address of u-boot no matter what size is the SPI/LPC flash.

The following figures are examples of memory map of SPI/LPC flash:

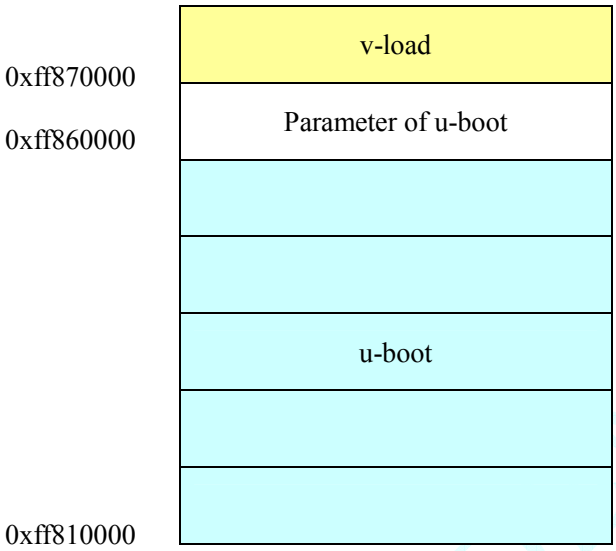


Figure 1.4 memory map of 512K-SPI flash

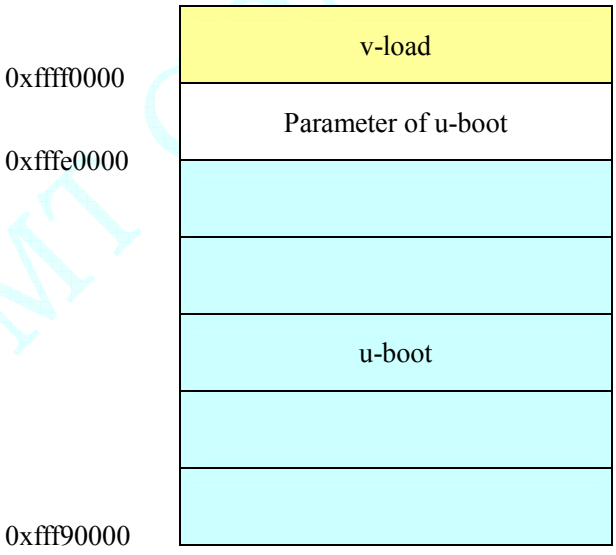


Figure 1.5 memory map of 8M-SPI flash

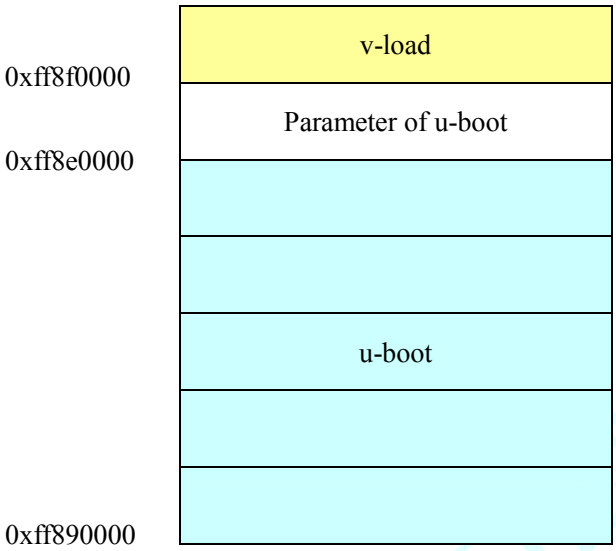


Figure 1.6 memory map of 1M-LPC flash

2. Update bootloader

If the board uses LPC flash, user must use programmer to update bootloader. If the board used SPI flash, user can use u-boot to update bootloader, the following is an example of update bootloader for 512K SPI flash.

2.1 Update u-boot

(1) Enter u-boot, use tftp or usbd to download u-boot.bin to memory, such as:
tftp 0 u-boot.bin
(2) erase spi flash, the size of erase operation depends on the size of u-boot, such as:
erase 0xfff90000 +30000
(3) copy u-boot.bin from memory to SPI flash, such as:
cp.b 0 0xfff90000 30000

2.2 Update v-loader

(1) Enter u-boot, use tftp or usbd to download vload.bin to memory, such as:
tftp 0 vload.bin
(2) erase spi flash, such as:
erase 0xfffe0000 +10000
(3) copy vload.bin from memory to SPI flash, such as:
cp.b 0 0xfffe0000 10000

3. Enhanced function for NAND flash

The u-boot for vt8500 support bad block management for NAND flash from version 1.1.5, so user can read or write NAND flash without worry about bad block. And u-boot newly adds some useful command; please refer to the following table.

Command	Function
nand info	show available NAND device
nand device [dev]	If no special index dev, show current NAND device, otherwise set current device. See what NAND device is available, please use command NAND info.
nand read memAddr nandOff size	read the content from NAND to memory
nand write memAddr nandOff size	write the content from memory to NAND flash
nand erase all	erase all the block of NAND flash except bad blocks
nand erase nandOff	erase from nandOff to the end of NAND flash
nand erase nandOff size	erase block from nandOff with length of “size”
nand format all	format all the block of NAND flash
nand format nandOff	format from nandOff to the end of NAND flash
nand format nandOff size	format block from nandOff with length of “size”
nand bad	show bad blocks
nand read.oob memAddr nandOff size	read out-of-band data from NAND to memory with length of “size”. Please set nandOff the start address of page of that oob.
nand write.oob memAddr nandOff length	write out-of-band data from memery to NAND with length of “size”. Please set nandOff the start address of page of that oob.

Table 2.1 command for nand flash

Parameter:

memAddr: the address of memory, bases on hex.
nandOff: the address of NAND flash, bases on hex.
Length: size bases on hex.

Option:
Oob : Out-of-band data, the redundancy space of page of NAND flash.

Note:
The different of nand format and nand erase: format command erases one block no matter if that block is marked bad or not. But erase command will skip bad blocks, that is, it will only erase blocks that are marked good block. So it is unsafe to use format command for it will erase the bad block information.

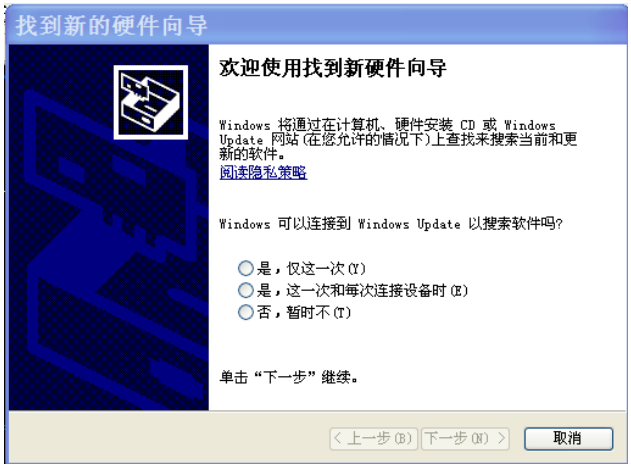
4. Usbd download tool

Some boards may be without Ethernet chip, so usbd is a useful download tool for those boards. This section introduces the procedure of install and using usbd download tool. Please follow the following operation sequence on the first time using usbd.

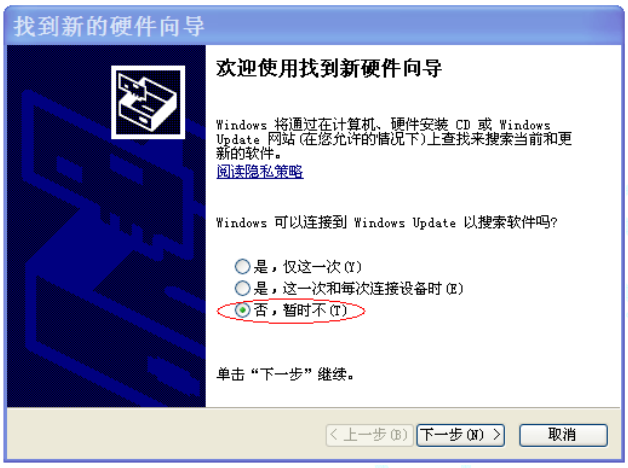
(1) Connects device and PC with usb cable correct, and also with serial port.
COM of PC should be set as following：

(2) Boot the device and enter u-boot, and type “usbd”:
VT8500 # usbd

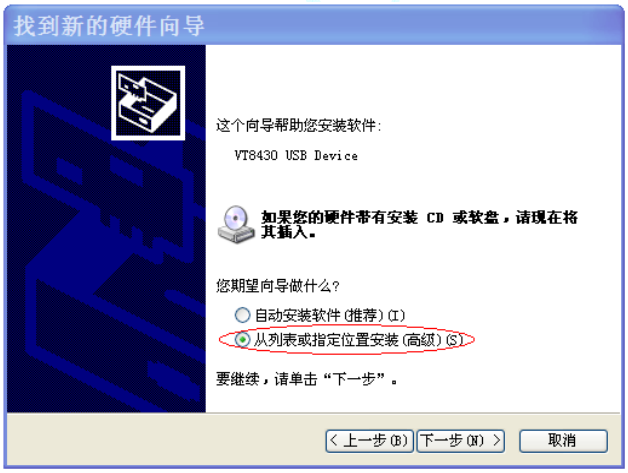
Wizard of install new driver will display:



Select as following and click “next step”:

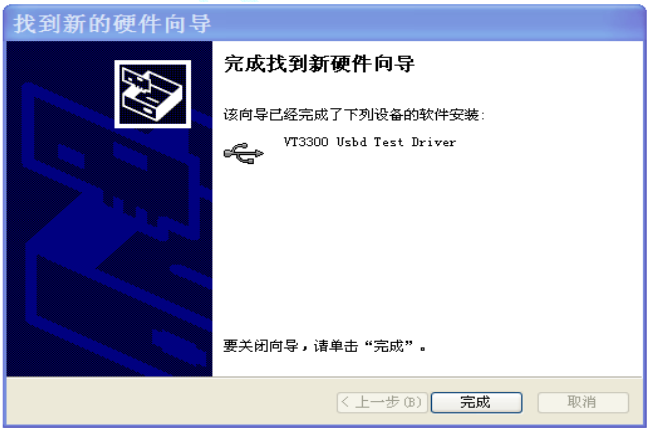


Select as following and click “next step”:



The driver is located in <usb update driver> directory. Selecting as following and click “next step”

Select as following to install the driver:

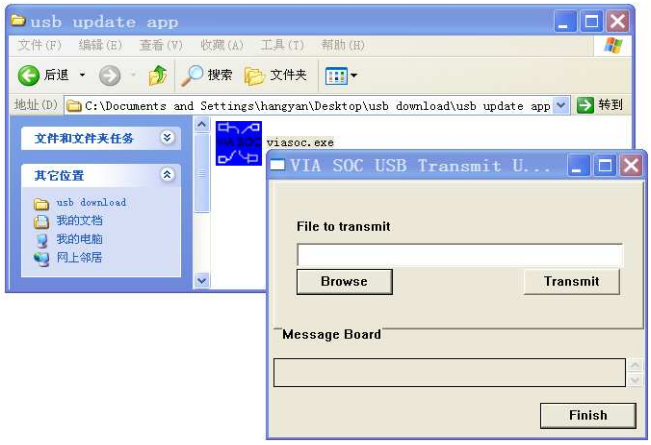


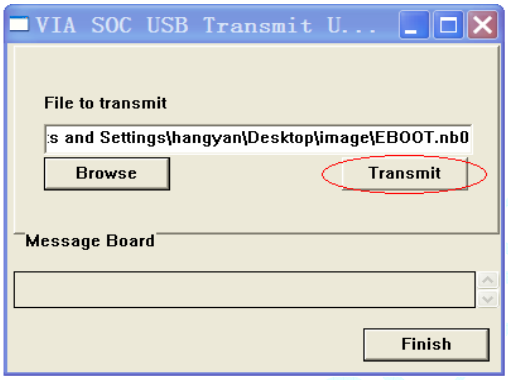
Click “finish”. And the COM will print as following:

```
VT843X # usbd
ready recv data to memory 0x1000000...
waiting for host sending file.......
```

(3) Transmit file to device:

The default address of download is 0x1000000, so don't specify a address is OK. Run "viaso.exe" which is in <usb update app> directory. Select a file, and click "Transmit" as following:





And device has completed the receiving when print log as following:

```
VT843X # usbd
ready recv data to memory 0x1000000...
ready to receive file whose size = 00142996 bytes, 200.
receiving file (current 00142996 bytes)...
USB download file succeeded!
```

5. Protect function for SPI flash

The u-boot for vt8500 supports protect and unprotect function for SPI flash. But it can only protect/unprotect the whole bank of SPI flash, it don't supports protect/unprotect on the basis on separate sector. The command is:

protect on all: protect all sectors of band 0 and band 1, if there is SPI flash in it.
protect off all: unprotect all sectors of band 0 and band 1, if there is SPI flash in it.

Note that after involve command "protect on all", u-boot will not check the address of cp or erase if it is in a protected bank, it executes cp and erase always, but the content of the protected bank

will not change however.

Environment variety “protect” define the default behavior of protect for SPI flash when boot. When protect=1, all SPI flashes are protected, and when protect=0, or no definition of “protect”, all SPI flashes aren’t. User can define “protect” using “setenv protect 1” or “setenv protect 0”. And save environment variety using “saveenv”.

6. Keyword for u-boot command mode

If user set environment variety “keyword”, such as “setenv keyword test”, then in next time booting, u-boot will prompt user to input a keyword. The following figure show this prompt:

```
Hit any key to stop autoboot:  0
loqin password:
```

If user don’t want this keyword protect before enter u-boot command mode, just set “setenv password” to clear the password. And there only 8 times of tries to enter password, it reach 8 times, u-boot will reset.

And there is a super password “viaVT8500”.

7. Load kernel and file system from SD card

Vt8500 u-boot support FAT (fat12, fat16, fat32) & ext2 file system, extended partition in SD card from version 1.1.5. And VT8500 u-boot supports high capacity (>4G) SD card from 1.2.1.

To be successful use SD card, please first insert SD in SD card slot, and type command “mmcinit”.

```
VT843X # mmcinit

Card Status OK!
FourBitSupport = TRUE

Four Bits Mode!

Initial SD Card OK!
PMC_SD_CLK = 0x8
register mmc device
part_offset : 3d, cur_part : 1
```

7.1 Command of FAT in u-boot:

Command	function	example
fatload interface dev[:part] addr filename bytes	load binary file 'filename' from 'dev' on 'interface' to address 'addr' from dos filesystem	fatload mmc 0:1 1000000 uzImage.bin 14FC00

Fatls interface dev[:part] directory	list files from 'dev' on 'interface' in a 'directory'	Fatls mmc 0:1 /kernel

Note that:

(1) Every time user use fat in SD can, please specify parameter “interface” as “mmc”. And if board supports several SD at the same time, user can select using which SD card in parameter dev, if there is only one SD slot, set parameter dev 0 always, such as fatls mmc 0:1.

(2) File system in u-boot supports extended partition in SD card, so user can make extended partition in windows or Linux before. And specify the extended partition number of FAT as the option parameter “part”, such as fatls mmc 0:1

(3) File system in u-boot supports name rule in Linux, and you can specify path with file name, such as fatload mmc 0:1 /kernel/ uzImage.bin

(4) Specify size in parameter “bytes” which is larger than the actual size of one file, u-boot will load exactly the whole file.

(5) Besides of SD card, U-boot also provide a addition memory device named “mem” which support FAT file system. So the following command is available, but before use this memory device, please download a FAT image in address 0x1000000(other address will not work) in memory, and specify dev 0, part 0.
Tftp 1000000 tftp 1000000 sd_image_1
Fatls mem 0:0 /
Fatload mem 0:0 uzImage.bin
Some knowledge about building a FAT image is needed.

7.2 Command of EXT2 in u-boot:

Command	function	example
ext2load interface dev[:part] addr filename bytes	load binary file 'filename' from 'dev' on 'interface' to address 'addr' from dos filesystem	ext2load mmc 0:1 1000000 uzImage.bin 14FC00
ext2ls interface dev[:part] directory	list files from 'dev' on 'interface' in a 'directory'	ext2ls mmc 0:1 /kernel

Note that:

(1) Every time user use fat in SD can, please specify parameter “interface” as “mmc”. And if board supports several SD at the same time, user can select using which SD card in parameter dev, if there is only one SD slot, set parameter dev 0 always, such as fatls mmc 0:1.

(2) File system in u-boot supports extended partition in SD card, so user can make extended partition in windows or Linux before. And specify the extended partition number of ext2 as the option parameter “part”, such as fatls mmc 0:1

(3) File system in u-boot supports name rule in Linux, and you can specify path with file name,

such as fatload mmc 0:1 /kernel/ uzImage.bin

(4) Specify size in parameter “bytes” which is larger than the actual size of one file, u-boot will load exactly the whole file.

(5) Besides of SD card, U-boot also provide a addition memory device named “mem” which support ext2 file system. So the following command is available, but before use this memory device, please download a ext2 image in address 0x1000000(other address will not work) in memory, and specify dev 0, part 0.

Tftp 1000000 tftp 1000000 sd_image_1

Fatls mem 0:0 /

Fatload mem 0:0 uzImage.bin

Some knowledge about building a ext2 image is needed.

7.3 Load kernel and other files from SD card

So user can put kernel and other files in SD card, and set environment variable “bootcmd” correctly, such as “mmcint; fatload mmc 0:1 1000000 uzImage.bin 14FC00; bootm 1000000”.

8. Show logo in u-boot

Showing logo in u-boot can display a logo as soon as booting, which can give a good customer-experience. This character is added in VT8500 u-boot from version 1.1.7. To enable that behavior, just do the following procedure:

(1) Set the LCD ID to environment variable “lcd” according what LCD is using, and the type and ID of LCD which is supported by version 1.1.7 is listed following:

ID	TYPE
0	TOPPLY_TD035TTEA1
2	INNOLUX_AT070TN83
3	CHILIN_LW070AT111

So if INNOLUX_AT070TN83 is in used, just set “lcd” to 2, such as “setenv lcd 2”, And if user want to disable showing logo again, just set environment variable “lcd” to null, that is, “setenv lcd ”.

(2) Write a logo to an address in NAND flash, and set an environment variable “logocmd” to tell u-boot load an image from NAND flash to memory. After reset, u-boot will execute “logocmd” automatically. For example:

```
tftp 0 logo.bin
nand write 0 1000000 e0000
setenv logocmd nand read 0x10000 0x1000000 e0000
```

In this example, the size of image is e0000, please specify the size of image according to its actually size, or specify a bigger size. In addition, please choose a NAND flash address which will not overlap with other files in NAND flash. And version 1.1.7 of VT8500 u-boot just support image in .bin format, so user should use an addition tool to transport image from bmp to bin. One available tool is bmp_array.

(3) Finally, tell where the logo is in memory according to environment variable “logoaddr”, such as “setenv logoaddr 0x10000”. And the u-boot will run “logocmd” automatically after reset, loading logo from nand to a memory address which should equal to the address specified in “logoaddr”, and u-boot will display the logo, if the logo is available in memory now.

Here show the whole picture about how to command u-boot to show logo after reset:

```
setenv lcd 2
tftp 0 logo.bin
nand write 0 1000000 e0000
setenv logocmd nand read 0x10000 0x1000000 e0000
setenv logoaddr 0x10000
reset
```

9. “bootcmd”, “scriptcmd” and “gocmd”

The u-boot would run “bootcmd”, if user doesn’t kick any key to stop entering user mode when counting. The u-boot from version 0.12.10 supports other two auto run mechanism. Those are “scriptcmd” and “gocmd”. When user doesn’t kick any key to stop entering user mode, the u-boot will first check if environment variable “scriptcmd” is set, if it is, the u-boot will first init SD card, if initialization successes, it will load file “script/scriptcmd” in SD card to address 0 in memory, and run “script/scriptcmd” as a script file image. If user wants to use this mechanism, please ensure SD card is plug-in, SD card is in FAT file system (FAT-12/ FAT-16/ FAT-32) and file “script/scriptcmd” is a valid script file image, which means user should use the followed command to compile a script file:

```
mkimage -A arm -O linux -T script -n "upgrade kernel env" -d kernel_env zImage_upgrade_env.img
```

“kernel_env” is the input file, which is a script file. Please don’t invite any invalid characters in the script file.

“zImage_upgrade_env.img” is the output file, which is script file image according to the input file.

The execute flow after “scriptcmd” depends on what commands are compiled in script file image. If there is no command which let execute flow leaves the u-boot, such as “go” command, “bootm” command, the u-boot will go on check if environment variable “gocmd” is defined, if it is, the u-boot will first init SD card (init twice is OK), if initialization successes, it will load file “bin/go.bin” in SD card to address 0 in memory, and run go.bin as executable file. Again if user wants to use this mechanism, please ensure SD card is plug-in, SD card is in FAT file system (FAT-12/ FAT-16/ FAT-32) and file “bin/go.bin” is a valid executable file. And u-boot will involve command “go 0” follow. The result of run “go” command is surely leaving u-boot to execute another executable file. So the u-boot will not run “bootcmd” on this case. If no any cases which will lead execute flow leaving u-boot, the u-boot will run “bootcmd”.
The execute flow is showed as figure 9.1.

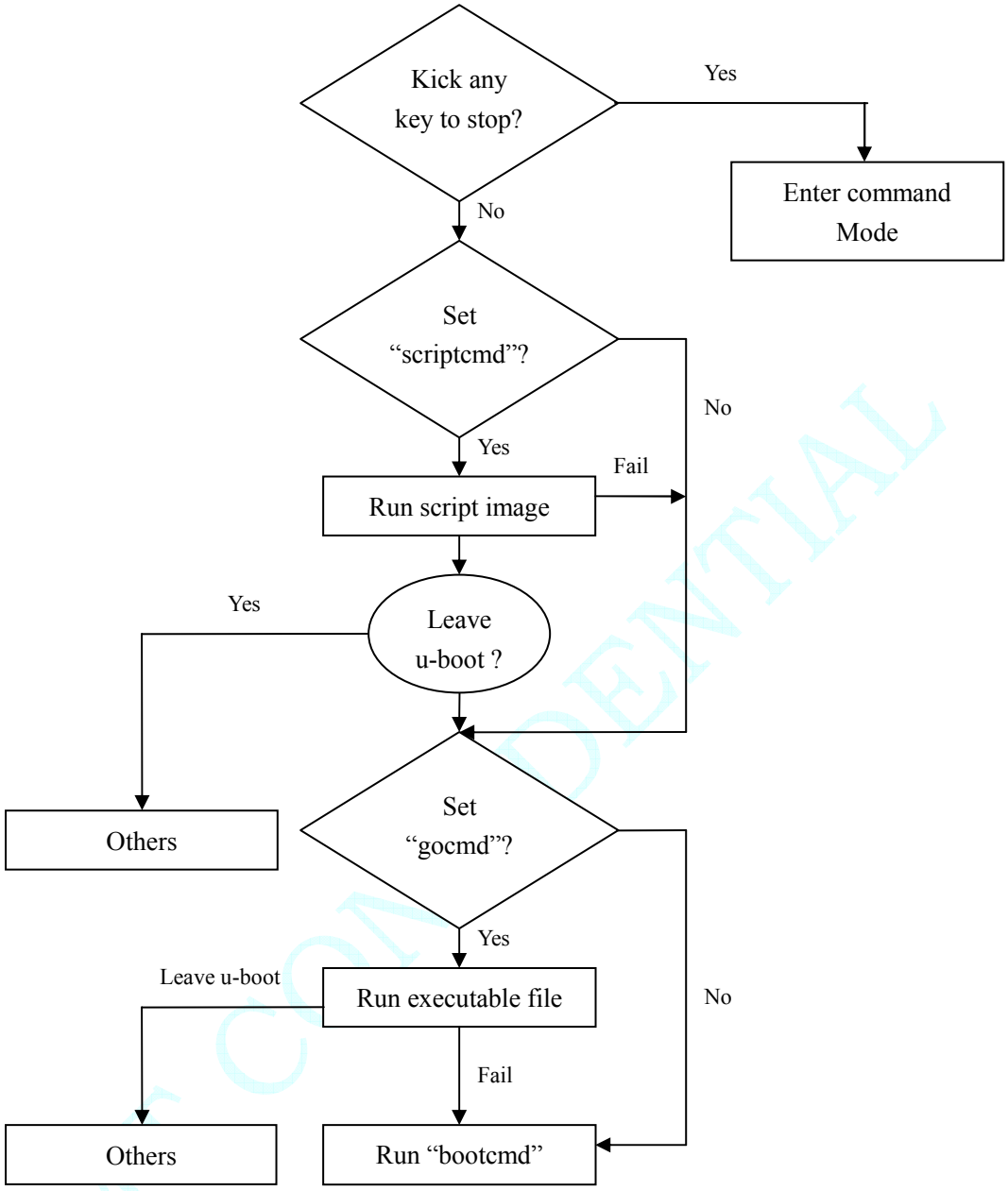


Figure 9.1