GM8136

U-BOOT

User Guide
Rev.: 1.1
Issue Date: September 2014

REVISION HISTORY

GM8136 U-BOOT User Guide

Date	Rev.	From	To
Sept. 2014	1.0	-	Original

Printed in Taiwan 2014

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Overview

U-BOOT is closely related to Linux. Some parts of the source code are originated in the Linux source tree, so that some header files are in common and the special provision supports booting from the Linux images. When the peripheral chip is different from the on-board peripheral chip, the drivers should be modified according to the new hardware design and the new definition should be configured as required.

Directory hierarchy:

- board: Board-dependent files
- common: Misc. architecture-independent functions
- cpu: CPU-specific files
- disk: Code for handling the disk drive partition
- doc: Basic documentation
- drivers: Commonly-used device drivers
- examples: Example codes for standalone applications
- include: Header files
- lib_arm: Generic files for the ARM architecture
- lib_avr32: Generic files for the AVR32 architecture

- lib_generic: Generic files for all architectures
- lib_i386: Generic files for the i386 architecture
- lib_m68k: Generic files for the m68k architecture
- lib_mips: Generic files for the MIPS architecture
- lib_nios: Generic files for the NIOS architecture
- lib_ppc: Generic files for the PowerPC architecture
- lib_sparc: Generic files for the SPARC architecture
- net: Networking code
- post: Power-On Self-Test
- tools: Tools used to build the S-Record or U-BOOT images

Chapter 2

Compiler

This chapter contains the following sections:

The released U-BOOT for the GM8136 Flash system can be found in the "u-boot" directory. In this directory, users can find the pre-compiled image and the source archive.

2.1 Unpack

Users can unpack the source archive by issuing the following command:

```
$ tar xvfz u-boot-2013.01.tar.gz
```

Change to the created u-boot directory:

```
$ cd u-boot-2013.01
```

2.1.1 Building U-BOOT

U-BOOT is used by FA6-Linux as the OS loader.

2.1.1.1 Configuring U-BOOT

The FA6 U-BOOT maintains the configuration file, *GM.h*, to configure different hardware environments. The *GM.h* file is located at **/usr/src/arm-linux-3.3/u-boot-2013.01/include/configs/GM8136.h**. Users can modify this file depending on the circumstances.

In addition, users can modify the MAC address to download the Linux code to the MediaCreative platform. At this stage, users should modify the definition, CONFIG_ETHADDR, of the MAC address at **u-boot-2013.01/include/configs/GM8136.h** (As shown in Figure 2-1).

```
#define CONFIG_ETHADDR                  00:42:70:00:30:22
#define CONFIG_NETMASK                  255.0.0.0
#define CONFIG_IPADDR                   10.0.1.52
#define CONFIG_SERVERIP                 10.0.1.51
#define CONFIG_GATEWAYIP                10.0.1.51
```

Figure 2-1. Modifying MAC and IP for U-BOOT in config_GM8136.h

2.1.1.2 Making U-BOOT

Once GMAC and IP are modified, users can build U-BOOT with the following commands:

 # cd /usr/src/arm-linux-3.3/u-boot-2013.01

 # ./make_8136

These two commands will create the file, *u-boot.bin*, in the folder as shown in the first line. Users should follow the instruction to burn U-BOOT into the GM8136 Flash and write the specific image, **u-boot.bin**, to the Flash address, 0x140000 (For the NAND system). To upgrade the SPI NOR Flash or NAND Flash, the PCTOOL command or SPI/NAND command should be used. Please refer to the Flash user guide for programming the SPI NOR Flash or NAND Flash.

2.2 Linux Loader – U-BOOT

U-BOOT is a well-known OS loader in the Linux world that is capable of loading images from a terminal protocol (Such as **Kermit**) and booting the Linux kernel. It provides the Flash utilities and Ethernet TFTP transfer functions. There are three CPU cores in GM8136, and U-BOOT can only service for the FA6 Linux image.

2.2.1 Running U-BOOT

The FA6-Linux distribution package provides the U-BOOT code to perform the following tasks:

- Programming Flash
- Transferring data from PC to the target by UART (Kermit) or Ethernet (TFTP, please use the GMAC0 port.)
- Loading or branching Linux kernel

Note: Users can run the U-BOOT code from Flash or via ICE.

2.2.1.1 Running U-BOOT from Flash

If the boot code (nsboot.bin) and the U-BOOT code are ready in a Flash, users can run U-BOOT from the Flash by the keystroke, "ESC", at the reset time.

2.2.1.2 Running U-BOOT via ICE

Users may run U-BOOT via ICE by following the procedures:

1. Connect the FA6 target to the user PC with JTAG ICE
2. Open the OPENice debugger and load **u-boot.bin** to the memory address, 0x0
3. Set PC to 0x0 and run

2.2.2 U-BOOT Environment Variables

U-BOOT maintains a number of environment variables for various functions. These environment variables can be displayed by using the following command:

=> printenv

Users can set the environment values by using the "setenv name value" command, where "name" denotes the name of the environment variable and "value" denotes the value to be set. The following command is an example of setting the IP address environment:

=> setenv ipaddr 192.168.68.48

Table 2-1 lists and describes the environment variables used in this document.

Table 2-1. Environment Variables

Environment Variable	Description
Ipaddr	The target IP address
serverip	The IP address of the TFTP server
ethaddr	The MAC address value
netmask	The netmask value of the IP address
printenv	Print the environment variables of U-BOOT
setenv [env] [value]	Set the environment variables
saveenv	Save the environment variables

2.2.3 U-BOOT Command Reference

Users can type "help" on the U-BOOT terminal to display the command list.

2.3 Modify DRAM Size

Before compiling U-BOOT, users can open the GM8136 U-BOOT configuration file, include/configs/gm8136.h. The default value is 64MB, and users can enable "DDR_SIZE_128MB" and disable "DDR_SIZE_64MB" if the DRAM size is 128MB.

```
//#define DDR_SIZE_128MB
#define DDR_SIZE_64MB
```

2.4 Build SPI System

Before compiling U-BOOT, the SPI configuration function has to be enabled. Users can open the GM8136 U-BOOT configuration file, include/configs/gm8136.h, and ensure that the "CONFIG_SPI***" option is defined and other options are disabled.

```
#define CONFIG_CMD_SPI
//#define CONFIG_SPI_NAND_GM
```

2.4.1 Binaries

Once U-BOOT has successfully compiled the following files, the compiled files will be stored in the "u-boot" directory.

File	Description
u-boot	Compiled ELF image
u-boot.bin	u-boot is converted to a raw binary.
u-boot.srec	u-boot.bin is converted to the Motorola S-Record format.

2.5 Build NAND System

Before compiling U-BOOT, the NAND configuration has to be enabled. Users can open the GM8136 u-boot configuration file, include/configs/gm8136.h, to ensure that the "CONFIG_SPI_NAND_GM" option is defined and "CONFIG_CMD_SPI" is disabled.

```
//#define CONFIG_CMD_SPI
#define CONFIG_SPI_NAND_GM
```

Then, issue the following command to build an image:

```
$ ./make_8136
```

2.5.1 Binaries

Once U-BOOT has successfully compiled the following files, the compiled files will be stored in the "u-boot" directory.

File	Description
u-boot	Compiled ELF image
u-boot.bin	u-boot is converted to a raw binary.
u-boot.srec	u-boot.bin is converted to the Motorola S-Record format.

Chapter 3

Add New Flash Chip

This chapter contains the following sections:

3.1 SPI Flash

If users want to add a new SPI Flash, the files listed below should be modified:

```
u-boot-2013.01/drivers/mtd/spi/
winbond.c or macronix.c or eon.c …
and modify flash_table[] to add new chip
```

Users can update the SPI Flash settings as required (These definitions are the supported the Flash types):

```
u-boot-2013.01/include/configs/gm8136.h
#define CONFIG_SPI_FLASH            1
#define CONFIG_SPI_FLASH_EON        1
#define CONFIG_SPI_FLASH_MACRONIX   1
#define CONFIG_SPI_FLASH_SPANSION   1
//#define CONFIG_SPI_FLASH_SST        1
//#define CONFIG_SPI_FLASH_STMICRO    1
#define CONFIG_SPI_FLASH_WINBOND    1
```

3.2 SPI-NAND Flash

If users want to add a new SPI-NAND Flash, the following file should be modified. (Search the "spi_nand second ID" string and add it to the file)

```
u-boot-2013.01/drivers/mtd/nand/nand_ids.c
```

Chapter 4

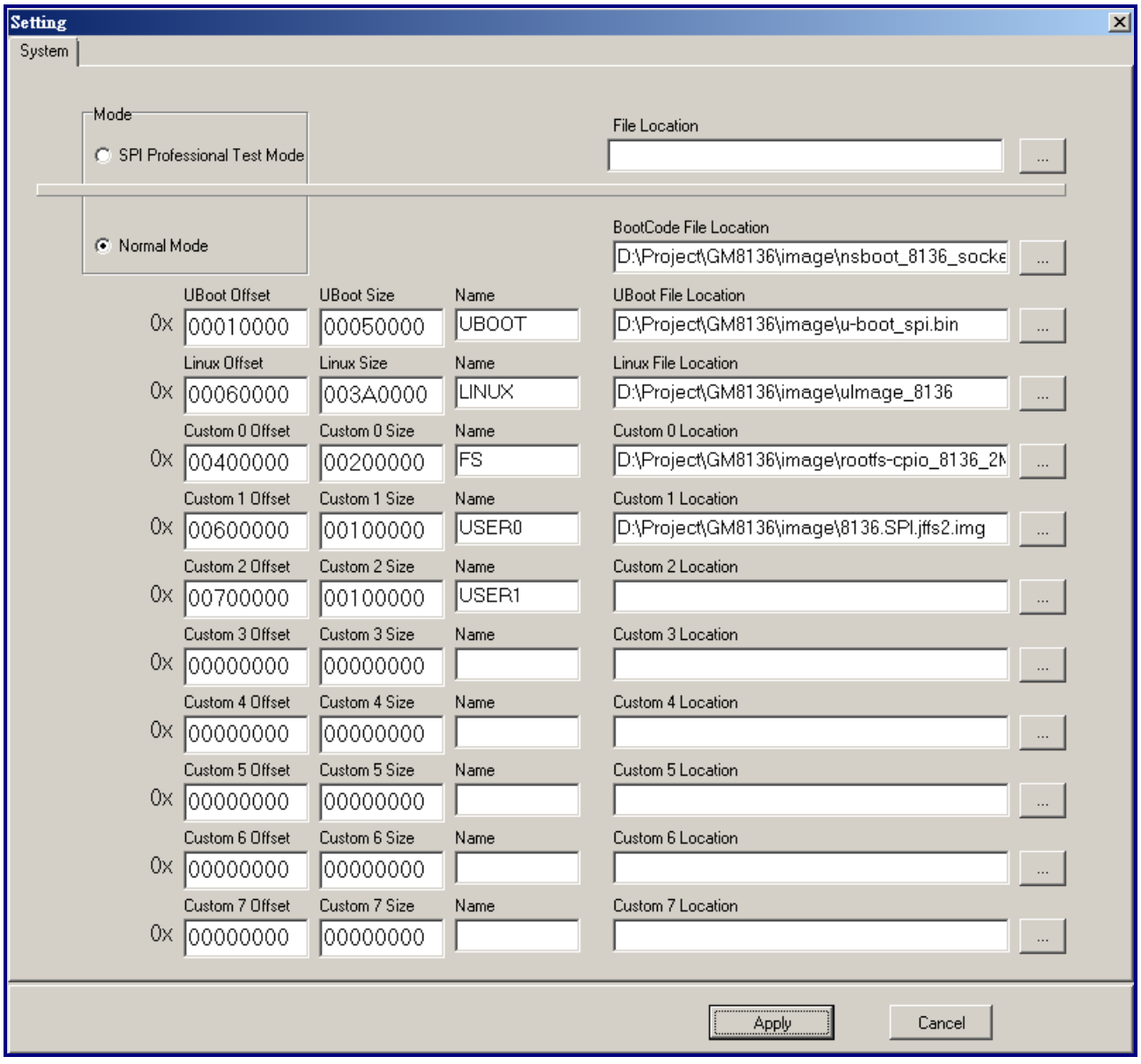
Modify Flash Address Setting

This chapter contains the following sections:

If users want to add a new SPI Flash, the following file should be modified:

`u-boot-2013.01/include/configs/gm8136.h`

U-BOOT will auto-load the FA6 images into DRAM and run the FA6 image. Users only need to use the PC-tools and modify the image offset/size to match the user environment, and do not change the first two images. U-BOOT will automatically load the second image as the FA6 image.



4.1 SPI Flash

Users can use the following defines to set the address of SPI Flash.

```
#define CONFIG_ENV_OFFSET       0x50000   /* environment value address */
#define CONFIG_ENV_SIZE            0x10000      /* erase block size */
#define CONFIG_ENV_SECT_SIZE   0x10000
```

If users have set the “saveenv” command and users want to clear and use the default setting, users can set the command as follows:

```
sf probe 0:0; sf erase 0x50000 0x10000
```

4.2 NAND Flash

Users can use the following defines to set the address of NAND Flash. (SPI-NAND is the same.)

```
#define CONFIG_ENV_OFFSET      0x220000     /* environment value address */
#define CONFIG_ENV_SIZE        0x20000  /* erase block size */
```

If users have set the “saveenv” command and users want to clear and use the default setting, users can set the command as follows:

```
nand erase 0x220000 0x20000
```

Chapter 5

Flash Command

This chapter contains the following sections:

If users want to run the read, write, or erase command, the command should be set as follows:

5.1 SPI NOR Flash

Command

```
=> help sf
sf probe [bus:]cs [hz] [mode]   - init flash device on given SPI bus
                                  and chip select
sf read addr offset len         - read `len' bytes starting at
                                  `offset' to memory at `addr'
sf write addr offset len        - write `len' bytes from memory
                                  at `addr' to flash at `offset'
sf erase offset len             - erase `len' bytes from `offset'
```

For detecting a chip, if users want to run SPI Flash with the read, write, or erase command, this command should be set first.

Example

```
=> sf probe 0:0
#SF: Got idcode ef 40 17
8192 KiB W25Q64CV at 0:0 is now current device
=>sf read 0x10000 0x6000 0x1000
```

5.2 NAND Flash

Command

```
nand read addr offset len          - read `len' bytes starting at
                                   `offset' to memory at `addr'
nand write addr offset len         - write `len' bytes from memory
                                   at `addr' to flash at `offset'
nand erase offset len              - erase `len' bytes from `offset'
nand markbad off [...]             - mark bad block(s) at offset (UNSAFE)
```

Example

```
=> nand info

Device 0: NAND 128MiB 3,3V 8-bit, sector size 256 KiB
=> nand read 0x10000 0x480000 0x1000
=> nand markbad 0x4000000
```

Note: Because block 0 contains the bad block information, users should not erase block 0.

Chapter 6

MAC Setting

If users want to transfer the MAC setting values into Linux kernel, users can modify the “bootargs” argument as follows:

ethaddr = 00:42:70:00:30:22

ipaddr 192.168.68.1

then, use the “saveenv” command to save the MAC setting values to Flash.

```
GM # set bootargs mem= 128M gmmem=90M console=ttyS0,115200 e

thaddr=00:42:70:00:30:22

GM # saveenv

Saving Environment to NAND...

Erasing Nand...

Erasing at 0x280000 -- 100% complete.

Writing to Nand... done

GM # printenv
```

```
baudrate=115200

bootargs=mem= 128M gmmem=90M console=ttyS0,115200 ethaddr=00

:42:70:00:30:22

bootcmd=nand read 0x02000000 z

bootdelay=3

ethact=eth0

ethaddr=00:40:25:00:00:02

gatewayip=10.0.1.51

ipaddr=10.0.1.52

netmask=255.0.0.0

serverip=10.0.1.51

stderr=serial

stdin=serial

stdout=serial


Environment size: 399/131068 bytes

GM #
```

Boot Linux kernel

When the GMAC driver is inserted, these arguments will be obtained by driver and set by itself.

```
ftgmac100: Loading version 0.1 ...

ftgmac100-0-mdio: probed

ftgmac100-0 ftgmac100-0.0: eth0: 1 tx queue used (max: 2)

ftgmac100-0 ftgmac100-0.0: eth0: 1 rx queue used (max: 1)

ftgmac100-0 ftgmac100-0.0: eth0: irq 3, mapped at 90868000

ftgmac100-0 ftgmac100-0.0: eth0: mac=00:42:70:00:30:22

…
```

```
/ # ifconfig eth0 up
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:42:70:00:30:22
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2233 (2.1 KiB)  TX bytes:0 (0.0 B)
          Interrupt:3 Base address:0x6000

/ #
```

Chapter 7

ICache On/Off

Command

Users should use the following instruction to enable or disable ICache:

```
icache [on, off]- enable or disable instruction cache
```

Chapter 8

DRAM Read/Write

If users want to test DRAM, the following commands should be used:

```
cmp     - memory compare
cp      - memory copy
md      - memory display
mm      - memory modify (auto-incrementing)
mtest   - simple DRAM test
mw      - memory write (fill)
```

Command

```
Simple ram test:mtest [start [end [pattern]]]
```

Example

```
=> mtest 100000 200000
Testing 00100000 ... 00200000:
Pattern 0000000F Writing... Reading...
=>
```

Test mem

Testing 00100000 ... 00200000: Pattern 0000000F Writing... Reading...=> this test writes to the memory, thus modifies the contents of a memory. It will fail when this test is applied to the ROM or Flash memory. This command may crash the system when the tested memory range includes areas that are needed for operating the U-BOOT firmware (Such as the exception vector code, U-BOOT internal program code, stack, or heap memory areas).

Chapter 9

Transfer Argument to Kernel

If U-BOOT is used to pass the parameter to Linux, users must transfer the Linux image first.

(1) Transfer command is as follows:

```
./mkimage -A arm -O linux -T kernel -C none -a 2000000 -e 2000040 -n gm8136 -d arch/arm/boot/zImage
uImage
```

```
./mkimage -A arch -O os -T type -C comp -a addr -e ep -n name -d data_file[:data_file…] image
-A ==> Set architecture 'o 'a'ch'
-O ==> Set operating system 'o 'os'
-T ==> Set image type 'o 't'pe'
-C ==> Set compression ty'e 'c'mp'
-a ==> Set load address 'o 'a'dr' (Hex)
-e ==> Set entry point 'o 'ep' (Hex)
-n ==> Set image name 'o 'n'me'
-d ==> Use image data fr'm 'dataf'le'
-x ==> Set XIP (Execute in place)
```

(2) By saving the environment arguments, it will be automatically booted next time. The following example is the U-BOOT operation steps for the SPI Flash.

Example

```
sf probe 0:0
setenv bootargs mem=256M gmmem=190M console=ttyS0,115200 user_debug=31 init=/squashfs_init
root=/dev/mtdblock2 rootfstype=squashfs
saveenv
```

(3) By default, U-BOOT will automatically set the command line to Linux. If users want to modify the command line, users must issue the "setenv" command and run the "saveenv" command to Flash, and it will automatically load the command line when using it next time.

(4) For the Linux image with squash rootfs, users must burn its rootfs to Flash, and then load the Linux image by the tftp command. When Linux is booting, it will search its rootfs in Flash and mount it as the root filesystem if squashfs can be found in Flash. Otherwise, it will be boot failure. For the rootfs operation, please refer to the Linux user guide.

(5) If users want to modify the default setting of the U-BOOT command line, users must modify the u-boot-2013.01/include/configs/gm8136.h file. Please search the "CONFIG_EXTRA_ENV_SETTINGS" string and modify all settings to match the user requirements.

Grain Media only supports the following functions to pass the argument to the Linux kernel

- Change DDR size
- UART baudrate
- MAC address
- MTD partition

Note: For partitions of the SPI and NAND Flashes, PCTOOL will be used for this purpose and the Flash size can be automatically detected and it will not be used to pass the argument.

Users can update the boot command or save the Grain Media command by the "saveenv" command.

Linux boot message

```
## Booting kernel from Legacy Image at 02000000 ...
   Image Name:   gm8136
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    4260024 Bytes = 4.1 MiB
   Load Address: 02000000
   Entry Point:  02000040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...



Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0
Linux version 3.3.0 (root@ftclab1) (gcc version 4.4.0 20100318 (experimental) (Buildroot 2012.02) )
#96 PREEMPT Thu Sep 11 20:01:29 CST 2014
CPU: FA6 [66056263] revision 3 (ARMv5TE), cr=0000397f
CPU VIPT aliasing data cache, unknown instruction cache
Machine: Grain-Media GM8136 series
```

If users use PCTOOL to burn the images, the partition setting will be recorded into Flash and nsboot /u-boot /Linux will automatically parse the partition setting. Users do not need to set the partition again. If users want to set the partition from U-BOOT, users should use the corresponding command for each Flash as follows:

SPI Flash

setenv bootargs mem=128M gmmem=90M console=ttyS0,115200 user_debug=31 init=/squashfs_init root=/dev/mtdblock2 rootfstype=squashfs mtdparts mtdparts=nor-flash:0x50000@0x10000(uboot),0x3A0000@0x60000(linux),2M@4M(fs),1M@6M(user0),-(reserved)

NAND Flash

setenv bootargs mem=128M gmmem=90M console=ttyS0,115200 user_debug=31 init=/squashfs_init root=/dev/mtdblock2 rootfstype=squashfs mtdparts

mtdparts=nand-flash:0x100000@0x140000(uboot),5M@16M(linux),5M@32M(fs),5M@40M(user0),-(reserved)

Chapter 10

LCD Logo

If users want to show logo to LCD, please enable this definition and load the logo content from Flash to this address defined in CONFIG_VIDEO_FB_BASE. This definition is placed in the following file:

`u-boot-2013.01/include/configs/gm8136.h`

```
#define CONFIG_VIDEO
#define CONFIG_VIDEO_FB_BASE       0xE000000
```

CONFIG_VIDEO_FB_BASE means the frame buffer address in the DDR memory for LCD to read. Users can change this address according to real environment.

If there are multiple LCD controllers, only LCDC0 will be used for showing the logo. Others are disabled.

The command is listed below.

```
bootlogo {hdmi}
```

where the parameter, hdmi, is optional. If no argument is specified, the logo is displayed through VGA DAC only. If the parameter, hdmi, is specified, the LCD control shows the logo image through both VGA DAC and HDMI simultaneously.

Please be aware that if HDMI is used, the HDMI cable should be connected to the board; otherwise, HDMI driver will keep waiting the HDMI connection until the connection is established.

Example: Show logo through VGA DAC and HDMI

```
GM#bootlogo hdmi
```

Exmpale: Show logo through VGA DAC only

```
GM#bootlogo
```

Related source code:

u-boot-2013.1/common/cmd_bootlogo.c

u-boot-2013.1/common/HDMI_function.c

u-boot-2013.1/common/HDMI_header.h

u-boot-2013.1/common/HDMI_TxRegs.h

Note: In cmd_bootlogo.c, flcd_main() is the entry point to initialize the LCDC driver. By default, the input resolution is VIN_1024x768 and the output resolution is 1920x1080, which is fullHD.

The following table shows the supported input resolutions:

VIN_1280x720	1280x720
VIN_1920x1080	1920x1080
VIN_1024x768	1024x768

The following table shows the supported output resolutions:

OUTPUT_VGA_1280x720	1280x720
OUTPUT_VGA_1920x1080	1920x1080

If users want to change the input resolution or the output resolution, users should change the parameters to the flcd_main() function, sli10121_polling_thread(), and HDMI_setting.vid of Monitor_CompSwitch() if HDMI is enabled.

```
HDMI_setting.vid = VID_16_1920x1080p;      /*HDMI outputs 1920x1080 resolution*/
HDMI_setting.vid = VID_04_1280x720p;       /*HDMI outputs 1280x720 resolution */
```

Chapter 11

Manual Control to Boot System

If users want to boot the system with the manual control instead of auto boot, users need to use this method as follows:

(A) Users can use the run ./build_uImage_8136 command to generate the kernel image - uImage_8136.
(B) Command line for the Linux kernel, U-BOOT will automatically load them. If users do not want to auto-load, users can enter the “ESC” button and set the commands by the following steps:

(Step 1)
Set the command, "sf probe 0:0" (NAND Flash does not need to run it.)

(Step 2)
Set the command, "run lm"

(Step 3)

If users want to modify the command line, users should set the commands as follows: (If users do not want to modify, please ignore it.)

setenv bootargs mem=256M gmmem=190M console=ttyS0,115200 user_debug=31 init=/squashfs_init root=/dev/mtdblock2 rootfstype=squashfs

(Step 4)

If users want to modify the image, users should set the commands as follows: (If users do not want to modify, please ignore it.)

uImage_8136 please load to DRAM address 0x02000000

(Step 5)

Set the command, "bootm 0x2000000"

Chapter 12

Firmware Upgrade

This chapter contains the following sections:

Users can upgrade the latest version of the firmware by using the USB memory stick.

12.1 Configuration Setting

Please edit the file, "gm8136.h", under the path, "u-boot-2013.01/include/configs", and enable the following settings.

#define CONFIG_FOTG210

#define CONFIG_AUTO_UPDATE

12.2 Command Usage

The command name is "**fwupd**" and the description of the command usage is shown below.

```
fwupd - firmware upgrade from usb device for specified filename

Usage:
fwupd <part> [<filename>]
    - Load binary file 'filename' from usb device
      to update content of partition 'part' on flash.
      If 'filename' is omitted, environment variable "auimg?"
      is looking for, otherwise, command is aborted.
      All numeric parameters are assumed to be decimal.
```

The parameter, "<part>", specifies partition should be updated and cannot be ignored. For the detailed information of the partitions, please refer to the setting of PCTOOL.

The parameter "<filename>" is used to specify the filename to be read on the USB memory stick and can be ignored, if the filename is omitted, the environment variable "auimg?" will be looking, otherwise, the command will be aborted.

12.3 Environment Setting

Some environment variables are used for running the firmware upgrade. The variable, "auimg?", is used to specify the filename for updating the content of a partition. For example, the variable, "*auimg0*", specifies the required filename to update partition 0; the variable, "*auimg1*", specifies the required filename to update partition 1.

The variable, "autoupdate", indicates running the firmware upgrade process as the system booting. The meaning for "*yes*" is to allow this feature, while the meaning of "*no*" is not to allow this feature.

The default value of the environment variables are shown as below:

```
auimg0=u-boot_spi_rmii.bin
auimg1=uImage_8136
auimg2=rootfs-cpio_8136.squashfs.img
auimg3=mtd.img
autoupdate=no
```

Users can change the default settings. Please edit the file, "gm8136.h", under the path, "u-boot-2013.01/include/configs", and modify the values of the following definitions.

```
#define AU_ENABLE "no"
#define AU_IMG_FILE_0   "u-boot_spi_rmii.bin"
#define AU_IMG_FILE_1   "uImage_8136"
#define AU_IMG_FILE_2   "rootfs-cpio_8136.squashfs.img"
#define AU_IMG_FILE_3   "mtd.img"
```

Please be aware that any behavior of updating U-BOOT will erase all environment settings saved on Flash.

12.4 Example

If users want to update U-BOOT on Flash and to specify the filename:

```
GM # fwupd 0 u-boot.bin
```

If users want to update U-BOOT on Flash but not to specify the filename, the filename will be obtained from the environment variable, "auimg0":

```
GM # fwupd 0
```

If users want to enable the auto-update feature, users should set the value of the variable, "autoupdate", to "yes" and save it on Flash:

```
GM # setenv autoupdate yes
GM # saveenv
```

Chapter 13

LCD BootLogo

If users want to show logo to LCD, please enable this definition and load the logo content from Flash to this address defined in the CONFIG_VIDEO_FB_BASE. This definition is placed in the following file.

`u-boot-2013.01/include/configs/gm8136.h or gm8139.h`

```
#define CONFIG_VIDEO
#define CONFIG_VIDEO_FB_BASE       0x4000000
#define CONFIG_CAT6611    /* HDMI */
```

CONFIG_VIDEO_FB_BASE means the frame buffer address in DDR memory for LCD to read. Users **MUST** change this address according to real environment.

The command is listed below.

bootlogo {hdmi}

where paramter hdmi is optional. If no argument is specified, the logo is displayed through VGA or CVBS only. If parameter hdmi is specified, LCD control shows the logo image through both VGA and HDMI simultaneously.

Please be aware that if HDMI is used, the HDMI cable should be connected to the board, otherwise HDMI

driver will keep waiting the HDMI connection until the connection is established.

Note: When bootlogo is turned on, LCD driver will output color bar automatically for testing the hardware path. Users can turn it off by disabling flcd210_set_patterngen() in cmd_bootlogo.c.

Example: show logo through VGA DAC and HDMI

```
GM#bootlogo hdmi
```

Exmpale: show logo through VGA DAC only

```
GM#bootlogo
```

Related source code:
u-boot-2013.1/common/cmd_bootlogo.c
u-boot-2013.1/common/cat6611 directory

Note:
In cmd_bootlogo.c, flcd_main() is the entry point to initialized LCDC driver. By default, the input resolution is VIN_1920x1080 and the output resolution is 1920x1080 which is fullHD. Two variables, g_vin and g_output, will define the input resolution and output resolution. If users want to change either the input or output resolution, please change the corresponding values.

Besides, the PLL3 should be correct in order to provide the correct LCD pixel clock. For example for fullHD, the PLL3 should be 594Mhz instead of 540Mhz which is default PLL3 clock.

The following table shows the supported input resolutions:

VIN_1280x720	1280x720
VIN_1920x1080	1920x1080
NTSC / PAL	720x480 or 720x576

The following table shows the supported output resolutions:

OUTPUT_VGA_1280x720	1280x720
OUTPUT_VGA_1920x1080	1920x1080

For HDMI, OUTPUT_VGA_1280x720 RGB is not supported yet.