UBoot for Freescale i.MX35

UBoot has been ported to the i.MX35 3-stack board. This document describes how to load, run and build UBoot images on the board(s).

1 Overview

The U-Boot utility is a multi-platform, open-source, universal boot-loader with comprehensive support for loading and managing boot images, such as the Linux kernel. It supports the following features:

- Network download: TFTP, BOOTP, DHCP, NFS
- Serial download: s-record, binary (via Kermit)
- Flash management: copy, erase, protect, cramfs, jffs2
- Flash Types: CFI NOR-Flash, NAND-Flash
- Memory utilities: copy, dump. crc, check, mtest
- Boot from disk: raw block, ext2, fat, reiserfs
- Interactive shell: choice of simple or "busybox" shell with many scripting features

For more information on UBoot, refer to http://www.denx.de/wiki/U-Boot/WebHome.

2 Board Dip Switches Setup

Be sure to follow the switch settings required for the board in order to have UBoot up and running properly.

2.1 i.MX35 3-stack Switch Settings

The i.MX35 3-Stack consists of a CPU board, a Personality board and a Debug board. The Debug board has a RS232 interface and an Ethernet connector.

2.1.1 Personality Board Dip Switch Settings

The following table shows the dip switch settings for the Personality board.

Table 2.1.1.1: i.MX35 Personality Board Switch Setup For NOR Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	N/A	N/A	N/A	N/A
SW2	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Table 2.1.1.2:i.MX35 Personality Board Switch Setup For NAND Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	N/A	N/A	N/A	N/A
SW2 (K9LBG08U0D)	ON	OFF	OFF	ON	ON	ON	OFF	OFF
SW2 (K9LBG08U0M)	ON	OFF	OFF	ON	ON	OFF	OFF	OFF

Table 2.1.1.3:i.MX35 Personality Board Switch Setup For MMC Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	N/A	N/A	N/A	N/A
SW2	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF

2.1.2 Debug Board Dip Switch Settings

The following tables shows the dip switch settings for the Debug board. For MMC/SD boot use internal boot mode.

Table 2.1.2.1: i.MX35 Boot Mode Switch Settings

	SW5	SW6	SW7	SW8	SW9	SW10
Internal Boot	0	0	0	0	0	0
External Boot	0	0	0	0	1	0

Table 2.1.2.2: Debug Board SW4 settings

Switch	1	2	3	4	5	6	7	8
SW4	ON	OFF	OFF	OFF	OFF	OFF	OFF	ON

3 Image Install Instructions

Get the U-Boot bin file u-boot-3ds.bin from release.
There are two ways to store a UBoot image into the onboard flash. One way is using the Advanced Toolkit.

3.1 Programming Through Advanced Toolkit (ATK)

Advanced Toolkit could be used to download UBoot on the boards. Before downloading ensure the following dip switches are set on the Personality and Debug boards. Follow the instructions in the ATK user guide to program UBoot.

Table 3.1.0.1: i.MX35 Personality Board Switch Setup For ATK Downloading

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	N/A	N/A	N/A	N/A
SW2	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF

Table 3.1.0.2: i.MX35 Debug Board Switch Setting For ATK Downloading

SW5	SW6	SW7	SW8	SW9	SW10
0	0	0	0	0	0

3.2 Programming the UBoot into NAND/NOR Flash

Use ATK tool to program UBoot to Nand/Nor flash with address 0.

3.3 Starting UBoot from NAND/NOR Flash

If UBoot is already programmed into the Nand/Nor flash, follow the instructions in **Section 2 “Board Dip Switches Setup“** to setup the board in Nand/Nor boot mode. The UBoot prompt should come up from the HyperTerminal after reset.

3.4 Programming UBoot into MMC

The following steps describe how to program UBoot into an MMC/SD card using UBoot:

1. Insert a MMC/SD card into the slot and follow the instructions in **Section 2 “Board Dip Switches Setup”** to setup the board in internal MMC boot mode.
2. Use ATK to download the UBoot binary with MMC boot to the address 0.

3.5 Starting UBoot from MMC

If UBoot is already programmed into the MMC/SD card, follow the instructions in **Section 2 "Board Dip Switches Setup"** to setup the board in internal MMC boot mode. The UBoot prompt should come up from the HyperTerminal after reset.

3.6 Booting the system with an NFS filesystem

By default, UBoot is configured to boot from NFS. To boot from NFS, some configurations need to be set first. Press any key to break the boot progress and set configurations.

Setup tftp server IP:
```
UBoot> setenv serverip <server_ip_address>
```
Setup board mac address:
```
UBoot> setenv ethaddr  <ethernet MAC address>
UBoot> setenv eth1addr <FEC MAC address>
UBoot> setenv fec_addr <FEC MAC address>
```
Setup board IP address:
```
UBoot> setenv ipaddr   <IP address>
```
Setup kernel name:
```
UBoot> setenv kernel   uImage
```
Setup rootfs path:
```
UBoot> setenv nfsroot  <rootfs_directory>
```
Save configurations:
```
UBoot> saveenv
```

Note:

To avoid a conflict of IP address, the `dhcp` *command can be used for a valid IP address.*

Reset board and kernel will be launched.

3.7 Booting kernel from Nor

3.7.1 Programming kernel and rootfs to nor

You need to use ATK tool to program kernel image to 0xa0080000 and rootfs image to 0xa2280000 first.
Other methods to program kernel image and rootfs image to nand:
(1) Use serial download to load image to ram, then use `cp` command in UBoot to program image to nand.
(2) Use `tftpboot` command to load image to ram, then use `cp` command in UBoot to program image to nand.

3.7.2 Environment settings

Setup configurations:
```
UBoot> setenv bootargs_nand 'setenv bootargs ${bootargs} root=/dev/mtdblock3
ip=dhcp rootfstype=jffs2'
UBoot> setenv bootcmd_nand 'run bootargs_base bootargs_nand;cp.b 0xa0080000
${loadaddr} 0x200000;bootm'
UBoot> saveenv

Reset board.
```

3.8 8Booting kernel from MMC

3.8.1 Formatting mmc card

1. Insert mmc card into a Linux machine.

The card will be /dev/mmcblkx, e.g /dev/mmcblk0. (You can use command `dmesg` to get the detailed information.).

2. Use command `fdisk` to format the mmc card to 2 partitions.

```
root@freescale ~$ fdisk /dev/mmcblk0
```

3. Type `p` in fdisk prompt to know the size of a cylinder.

```
Command (m for help): p

Disk /dev/mmcblk0: 3965 MB, 3965190144 bytes
4 heads, 16 sectors/track, 121008 cylinders
Units = cylinders of 64 * 512 = 32768 bytes
```

So the cylinder size is 32768Bytes = 0x8000.

4. Caculate the start cylinder of first partition.

The boot area consists of three parts, MBR, u-boot.bin and uImage. Normally the MBR is 512 Bytes. u-boot.bin is about 180KB, uImage is 2MB. So we can use the first 4MBytes (3MB > 2MB + 180KB + 512B) in mmc card as boot area.

So the begin cylinder for first partition is from 0x400000 / 0x8000 = 128.

5. Caculate the end cylinder of first partition.

Get rootfs.ext2.gz from release and uncompress it. The ext2 rootfs is about 380MB. So we choose 500MB (500MB > 380MB) as the first partition size.
The end cylinder is (500 * 0x100000) / 0x80000 = 16000.

6. Create the first partition.

Type `n` to create a partition.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
```

Type `p` to create a primary partition.
Type 1 as the first partition number.

```
Partition number (1-4): 1
```

Type 96 as the first cylinder.

```
First cylinder (1-121008, default 1): 128
```

Type 16000 as the last cylinder.

```
Last cylinder or +size or +sizeM or +sizeK (96-121008, default 121008): 16000
```

7. Create the second partition.

Type `n` to create a partition.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
```

Type `p` to create a primary partition.
Type 2 as the second partition number.

```
Partition number (1-4): 2
```

Type 16001 as the first cylinder.

```
First cylinder (1-121008, default 1): 16001
```

Type enter directly to use the default cylinder as the last cylinder.

```
ast cylinder or +size or +sizeM or +sizeK (16001-121008, default 121008): Using
default value 121008
```

8. Verify and write the partition information.

Type `p` to show the partition information.

```
Command (m for help): p

Disk /dev/mmcblk0: 3965 MB, 3965190144 bytes
4 heads, 16 sectors/track, 121008 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

        Device Boot      Start         End      Blocks  Id System
/dev/mmcblk0p1              96       16000      508960  83 Linux
/dev/mmcblk0p2           16001      121008     3360256  83 Linux
```

If the information is correct, type `w` to write the partition information to MBR.

```
Command (m for help): w
The partition table has been altered!

Calling mmcblk0: ioctl() to re-read partition ta p1ble
 p2
```

3.8.2 Program u-boot.bin, kernel and rootfs to mmc card

1. Save the MBR.

```
root@freescale /$ dd if=/dev/mmcblk0 of=./mbr.bin bs=512 count=1
```

Note:

Another way to avoid this step is that you can customize the u-boot.bin image by disabling the flash header in image and in the next step, you should program u-boot.bin to address 0x400.

2. Program u-boot.bin, kernel and rootfs to mmc card

Use ATK tool to program u-boot.bin to address 0, kernel uImage to 0x100000, rootfs to 0x400000.

Note:

Another way is download images to ram via tftp and use cp command for programming.

```
tftpboot <ram_address> <image_name>
mmc write 0 ${loadaddr} <block_offset> <block_count>
```

3. Restore the MBR.

```
root@freescale /$ dd if=./mbr.bin of=/dev/mmcblk0
```

3.8.3 Environment settings

```
UBoot> setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/mmcblk0p1
ip=dhcp rootfstype=ext2'
UBoot> setenv bootcmd_mmc 'run bootargs_base bootargs_mmc;mmc read 0 ${loadaddr}
0x800 0x1000;bootm'
UBoot> setenv bootcmd ‘run bootcmd_mmc’
UBoot> saveenv
```

Reset board.

4 Configuring UBoot

By default, UBoot is configured to display the command prompt and receive serial keyboard input on certain UART ports with 115,200-8-N-1 settings.

The system configuration also needs to be set using the `setenv` command. The following is one example of how to set it up by assigning a static IP address to the board. If the network supports BOOTP/DHCP and one wants to use it, set "`true`" when prompted with "`Use BOOTP for network configuration`" in the following example and skip the configurations for `Gateway IP address`, `Local IP address` and `Local IP address mask`. Note that the new configuration doesn't take effect until the board is reset (reset button or `reset` command).

```
UBoot> setenv serverip <server ip address>
UBoot> setenv ethaddr  <Board MAC address>
UBoot> setenv eth1addr <FEC MAC address>
UBoot> setenv fec_addr <FEC MAC address>

UBoot> setenv ipaddr   <local ip address>
UBoot> setenv kernel   uImage
UBoot> setenv nfsroot  <rootfs directory>
UBoot> saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Erase is built in program.
Writing to SPI flash...Writing SPI NOR flash 0x100000 [0x20000 bytes] <- ram
0x975e06e8
......SUCCESS

done
UBoot>
```

Each of the parameters should be modified for the specific network and usage. It is very important that the board specifics parameters be set correctly. These parameters are not used by the UBoot but are used by some operating systems.

Note that some DHCP servers are not configured to support BOOTP requests and as a result a static configuration will be required.

The board specifics parameter is a bit mask of board options. At this time, there are no options and board specifics should be 0.

The configuration can be listed:

```
UBoot> printenv
bootdelay=3
baudrate=115200
loadaddr=0x80800000
netdev=eth0
ethprime=FEC0
uboot_addr=0xa0000000
uboot=u-boot.bin
bootargs_base=setenv bootargs console=ttymxc0,115200
bootargs_nfs=setenv bootargs ${bootargs} root=/dev/nfs ip=dhcp
nfsroot=${serverip}:${nfsroot},v3,tcp
bootcmd=run bootcmd_net
bootcmd_net=run bootargs_base bootargs_nfs; tftpboot ${loadaddr} ${kernel};
bootm
prg_uboot=tftpboot ${loadaddr} ${uboot}; protect off ${uboot_addr} 0xa003ffff;
erase ${uboot_addr} 0xa003ffff; cp.b ${loadaddr} ${uboot_addr} ${filesize};
setenv filesize; saveenv
```

```
ethact=FEC0
ethaddr=00:04:9F:00:EA:D7
bootargs=console=ttymxc0,115200 root=/dev/nfs ip=dhcp nfsroot=:/opt/eldk/
arm,v3,tcp
ipaddr=10.193.102.93
serverip=10.193.100.158
nfsroot=/data/rootfs_home/rootfs
kernel=uImage.r450
stdin=serial
stdout=serial
stderr=serial

Environment size: 830/131068 bytes
Boot>
```

5 Using UBoot

OS images can be downloaded using UBoot via Ethernet or the serial port. The downloaded images can be immediately decompressed into SDRAM and executed or they can be stored into flash and decompressed into SDRAM at a later time. All of these instructions assume that UBoot has been installed onto the board and configured.

5.1 Serial Download

To download an image through serial, the terminal should support y moderm protocol, e.g. HyperTerminal in windows. Issue the following command under UBoot prompt:

```
loady
```

UBoot now is ready to receive data. To send a file, click on "*Transfer -> Send File -> Ymodem (under Protocol) -> Send...*", choose the file to download.

5.2 Ethernet Download

Note that TFTP is not a reliable protocol and downloads may sometimes fail (with no error message) on busy networks. Always check the byte count of the download to make sure the download is complete. Besides, some EVB/ADS boards don't have the correct MAC address programmed in the EEPROM which is used by the Ethernet controller.

To download a file via the Ethernet:

1. Set up and configure a TFTP server.
   ```
   setenv ethaddr <board_mac_addr>
   setenv serverip <Server_IP_Addr>
   setenv ipaddr <Board_IP_Addr>
   ```
   (This address can be retrived by command dhcp)
2. Configure the server to load files from the appropriate directories or copy the desired image to the server directory.
3. Test the network configuration:
   ```
   ping <TFTP server IP address>
   ```

The following discuss how to download and execute different kinds of images.

- To download and execute an uncompressed image (check that the number of bytes downloaded is correct):
  ```
  tftpboot ${loadaddr} <file name>
  bootm
  ```

6 Building UBoot

6.1 Building UBoot from source

Building UBoot can be achieved under either Windows using Cygwin or Linux. Here we only supply files for building under Linux. All the following files are included in release package.

Table 6.1.0.1: Resources needed for Building UBoot

Resource	Description	Source
`u-boot-2009.08.tar.bz2`	UBoot base source for Linux	pkgs/
`tc-fsl-x86lnx-armeabi-nptl-4.1.2-3.i386.rpm`	Toolchain needed to build UBoot under Linux	included
`u-boot-v2009.08-imx_09.12.01.tar.bz2`	Freescale UBoot source patch and binary files.	pkgs/

Unzip the Freescale UBoot release package `L2.6.31_09.12.01_SDK_source.tar.gz`. You will find the UBoot source patch files and `u-boot-v2009.08-imx_09.12.01.tar.bz2` inside `L2.6.31_09.12.01_SDK_source/pkgs`.

The following describes how to setup the build environment first and then gives instructions on how to build the UBoot images.

6.1.1 Setting up UBoot Build Environment under Linux

To simplify the built process, it is recommended to use Freescale's pre-packaged tools to build UBoot. The following has been verified working under Redhat 9.0 and Ubuntu 9.10 Linux distribution releases.

1. Locate `u-boot-2009.08.tar.bz2` and `u-boot-v2009.08-imx_09.12.01.tar.bz2` from this package.
2. Under Linux, with root user privilege, install `tc-fsl-x86lnx-armeabi-nptl-4.1.2-3.i386.rpm.` A more simple way is to run ltib in release firstly which will install the toolchain for host server.

6.1.2 Generating UBoot Image

After the tools are setup properly, follow these instructions to build the UBoot image:

1. Decompress the `u-boot-2009.08.tar.bz2` base line source code into some directory, etc, your home directory. There should be a `packages` directory under your home if this is done correctly.
   ```
   $ tar jxvf u-boot-2009.08.tar.bz2
   ```
2. Change into the `u-boot-2009.08` subdirectory and apply the patches from the `u-boot-v2009.08-imx_09.12.01.tar.bz2`:
   ```
   $ cd u-boot-2009.08
   ```
   Under Linux, do:
   ```
   $ tar jxvf u-boot-v2009.08-imx_09.12.01.tar.bz2
   $ ./patches/patch-uboot.sh
   ```
3. Check for a non-zero return value of the patch command (`echo $?`), or check to see if there are any rejected patch fragments (`find . -name '*.rej' -print`). Either of these findings would indicate that the patch did not apply properly.
4. Build UBoot. The following commands can be done from any directory and it is recommended to create a separate directory from the source in order to have a clean build.

   For i.MX35 3-Stack, do:

```
            make CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-
nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi- mx35_3stack_config
            make CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-
nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-
```

This creates the UBoot image (`u-boot.bin`) under uboot source directory. This image can run from either SDRAM or flash.

6.1.3 Customize UBoot Image for MMC boot

Change into the `u-boot-2009.08` subdirectory.

```
make CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-
none-linux-gnueabi/bin/arm-none-linux-gnueabi- mx35_3stack_mmc_config
make CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-
none-linux-gnueabi/bin/arm-none-linux-gnueabi-
```

6.1.4 Customize UBoot Image for MMC boot

Change into the `u-boot-2009.08` subdirectory and edit the file include/configs/mx25_3stack_config.h.
1. Undefine Nand configs.
Comments the Nand command definition.

```
/* #define CONFIG_CMD_NAND */
/* #define CONFIG_MXC_NAND */
```

2. Enable the MMC configs by uncommenting it.

```
/*
 * MMC Configs
 * */
#define CONFIG_FSL_MMC          1

#define CONFIG_MMC              1
#define CONFIG_CMD_MMC
#define CONFIG_DOS_PARTITION    1
#define CONFIG_CMD_FAT          1
#define CONFIG_MMC_BASE         0x0
*/
```

7 UBoot Commands

The following commands are supported for various Freescale boards. The intent is to provide a user-friendly and easy-to-use interface for setting up or using certain hardware features such as NAND flash, MAC address in the EEPROM for CS8900A Ethernet controller and reading/writing e-fuses, etc. This section shows these commands and the corresponding help messages.

7.1 Run

Run an commands in the environment variable(s) 'var'.

```
      run [env_variable]
```

7.2 Net commands

Boot image via network using BOOTP/TFTP protocol

```
      bootp [loadAddress] [[hostIPaddr:]bootfilename]
```

Boot image via network using TFTP protocol

```
      tftpboot [loadAddress] [[hostIPaddr:]bootfilename]
```

Boot image via network using DHCP/TFTP protocol
```
dhcp [loadAddress] [[hostIPaddr:]bootfilename]
```
Boot image via network using RARP/TFTP protocol
```
rarpboot [loadAddress] [[hostIPaddr:]bootfilename]
```

7.3 Boot commands

Boot application image from memory
```
bootm [addr [arg ...]]
```
Boot image via network using TFTP protocol
```
tftpboot [loadAddress] [[hostIPaddr:]bootfilename]
```
Boot image via network using DHCP/TFTP protocol
```
dhcp [loadAddress] [[hostIPaddr:]bootfilename]
```
Boot image via network using RARP/TFTP protocol
```
rarpboot [loadAddress] [[hostIPaddr:]bootfilename]
```

7.4 Environment commands

Set and save environment variable.Enable either NOR or NAND flash media for UBoot
```
setenv [env_var] [val]
saveenv
```

7.5 Nand commands

Show available NAND devices
```
nand info
```
Read from Nand
```
nand read <dest_addr> <src_offset> <length>
```
Erase Nand
```
nand erase <src_offset> <length>
```
Write to Nand
```
nand write <dest_addr> <src_offset> <length>
```
Show bad blocks
```
nand bad
```

7.6 MMC commands

Display MMC card info
```
      mmcinfo
Read MMC data from card
      mmc read <dev_num> addr blk# blk_count
Write data to MMC card
      mmc write <dev_num> addr blk# blk_count
Rescan MMC device
      mmc rescan <dev_num>
List avaiable devices
      mmc list
```

7.7 FAT commands

Load binary file from a dos filesystem
```
fatload <interface> <dev[:part]>  <addr> <filename> [bytes]
```
List files in a directory (default /)

```
      fatls <interface> <dev[:part]> [directory]
```
Print information about filesystem
```
      fatinfo <interface> <dev[:part]>
```
Write to Nand
```
      nand write <dest_addr> <src_offset> <length>
```
Show bad blocks
```
      nand bad
```

7.8 I2C commands

i2c memory display
```
      imd address[.0, .1, .2]  [# of objects]
```
i2c memory modify (auto-incrementing)
```
      imm address[.0, .1, .2]
```
i2c memory modify (constant address)
```
      imn address[.0, .1, .2]
```
i2c memory write (fill)
```
      imm address[.0, .1, .2] value [count]
```
checksum calculation
```
      icrc32 address[.0, .1, .2] count
```
probe to discover valid I2C chip addresses
```
      iprobe
```

8 Frequently Asked Questions

8.1 Why there is no UBoot prompt on an internal UART port?

Double check the dip switch settings as mentioned earlier in this document.

8.2 How to make no-padding u-boot images?

Use dd command.
```
      dd if=./u-boot.bin of=./u-boot-no-padding.bin bs=1024 skip=1
```