U-Boot for Freescale i.MX6X

U-Boot has been ported to the and i.MX6X boards This document describes how to load, run and build U-Boot 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, SPI 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 U-Boot, 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 U-Boot up and running properly.

2.1 i.MX6X Switch Settings

2.1.1 i.MX6 Quad/Dual ARM2 board dip switch settings

Table 2.1.1.1: Dip Switch Setup for SD Slot1 Normal Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SW2	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW4	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Table 2.1.1.2: Dip Switch Setup for SD bit width

SW2	6
1-bit	OFF
4-bit	ON

Table 2.1.1.3: Dip Switch Setup for MMC Slot1 Normal Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF
SW2	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW4	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Table 2.1.1.4: Dip Switch Setup for MMC bus width

SW2	6	7	8
1-bit	OFF	OFF	OFF
4-bit	ON	OFF	OFF
8-bit	OFF	ON	OFF
4-bit DDR	ON	OFF	ON
8-bit DDR	ON	ON	OFF

Table 2.1.1.5: Dip Switch Setup for ESDHC Slot Selection

SW2	4	5
Slot 1	OFF	OFF
Slot 2	ON	OFF
Slot 3	OFF	ON
Slot 4	ON	ON

Table 2.1.1.6: Dip Switch Setup for SD/MMC normal boot and fast boot

SW1	5
Regular	OFF
Fast	ON

Table 2.1.1.7: Dip Switch Setup for SPI NOR Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	ON	ON	OFF	OFF
SW2	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW4	OFF	OFF	OFF	ON	ON	OFF	OFF	OFF

Table 2.1.1.8: Dip Switch Setup for One-Nand Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SW2	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW4	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Table 2.1.1.9: Dip Switch Setup for SATA Boot

Switch	1	2	3	4	5	6	7	8
SW1	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
SW2	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW4	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

2.1.2 i.MX6 Sabresd Switch Settings

Table 2.1.2.1: Dip Switch Setup for SD Slot1 Normal Boot

Switch	1	2	3	4	5	6	7	8	9	0
SW6	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF

Table 2.1.2.2: Dip Switch Setup for SD bit width

SW6	3
1-bit	OFF
4-bit	ON

Table 2.1.2.3: Dip Switch Setup for MMC Slot1 Normal Boot

Switch	1	2	3	4	5	6	7	8	9	10
S1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF

Table 2.1.2.4: Dip Switch Setup for MMC bus width

S1	3	4
1-bit	OFF	OFF
4-bit	ON	OFF
8-bit	OFF	ON

3 Image Install Instructions

3.1 Programming U-Boot into MMC

The following steps describe how to program U-Boot into an MMC/SD card using U-Boot:

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 bootstrap mode.
2. Use ATK to download the U-Boot binary with MMC boot to address 0.

3.2 Starting U-Boot from MMC

If U-Boot 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 U-Boot prompt should come up from the HyperTerminal after reset.

3.3 Starting U-Boot from MMC fastboot

eMMC 4.3 and 4.4 card support fastboot. To boot from fastboot, please follow below steps:

1. Make sure the slot can support fastboot.
   Please check spec for this information. On most of our boards, Slot3 support eMMC fastboot.
2. Program u-boot image to eMMC boot partition.
   Normal boot from SD first and program u-boot image to boot partition.

e.g

- Insert normal SD card to slot1 and eMMC 4.3/4.4 card to slot3.
- Normal boot from SD slot1.

- Write u-boot.bin to eMMC card boot partition.

e.g

```
> mmc dev 0
mmc0 is current device
> mmc read ${loadaddr} 0 0x200
MMC read: dev # 0, block # 0,count 512 ...
512 blocks read: OK
> mmc dev 1 1
mmc1(part 1) is current device
> mmc write ${loadaddr} 0 0x200
MMC read: dev # 1, block # 0,count 512 ...
512 blocks write: OK
```

3. Get eMMC bus width.

   Get boot bit bus mode. Use command “mmcinfo <dev>“ to get bus width.

```
> mmcinfo 1
Device: FSL_ESDHC
Manufacturer ID: 3
OEM: 5344
Name: SD04G
Tran Speed: 25000000
Rd Block Len: 512
SD version 2.0
High Capacity: Yes
Capacity: 3965190144
Bus Width: 4-bit
```

4. Set eMMC fast boot dip and boot bits.

- Set eMMC fast boot dip.
- Set bus width to the bus width we got from step 3.

5. Reset board for booting.

3.4 Programming U-Boot to SATA device

The following steps describe how to program U-Boot to a SATA device.

1. Booting from MMC first.
2. Read u-boot.bin from mmc.

```
mmc dev 0
mmc read ${loadaddr} 0 0x200
```

3. Write to SATA device.

```
sata write ${loadaddr} 0 0x200
```

4. Set SATA boot dip setting.
5. Reset board and check output from terminal.

3.5 Programming the U-Boot into SPI-NOR Flash

1. Boot board with nand boot or mmc boot first.
2. Upload u-boot.bin to tftp root directory.
3. Load U-Boot Image to ram.

```
U-Boot> setenv serverip 10.193.100.158
U-Boot> setenv ethaddr  00:04:9F:00:EA:D7
U-Boot> setenv ipaddr   10.193.102.93
U-Boot> tftpboot ${loadaddr} u-boot.bin
```

4. Program u-boot.bin to spi-nor with address 0.

Initial SPI-NOR:

```
U-Boot> sf probe 1
```

Erase SPI-NOR before writing.

```
U-Boot> sf erase 0 0x40000
Erase is built in program.
```

Write data to SPI-NOR

```
U-Boot> sf write ${loadaddr} 0 0x40000
Writing SPI NOR flash 0x0 [0x40000 bytes] <- ram 0xXXXXXXXX
.........SUCCESS
```

3.6 Booting the system with an NFS filesystem

By default, U-Boot 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 boot arguments:

```
U-Boot > setenv bootargs 'console=ttyAM0,115200n8'
```

Setup boot command:

```
U-Boot > setenv bootcmd 'run bootcmd_net'
```

Setup NFS boot arguments:

```
U-Boot > setenv bootargs_nfs ‘setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp nfsroot=${serverip}:${nfsroot}
```

Setup net boot command:

```
U-Boot > setenv bootcmd_net 'run bootargs_nfs; dhcp; bootm'
```

Setup tftp server IP:

```
U-Boot> setenv serverip 10.193.100.158
```

Setup board mac address:

```
U-Boot> setenv ethaddr  00:04:9F:00:EA:D7
```

Setup board IP address:

```
U-Boot> setenv ipaddr   10.193.102.93
```

Setup kernel name:

```
U-Boot> setenv kernel   uImage
```

Setup rootfs path:

```
U-Boot> setenv nfsroot  /data/rootfs_home/rootfs
```

Save configurations:

```
U-Boot> 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 MMC

3.7.1 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
```

3.7.2 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>`

`cp.b <ram_address> <card_offset> <len>`

3. Restore the MBR.
```
root@freescale /$ dd if=./mbr.bin of=/dev/mmcblk0
```

3.7.3 Environment settings

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

Reset board.

4 Configuring U-Boot

By default, U-Boot 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).

```
U-Boot> setenv serverip 10.193.100.158
U-Boot> setenv ethaddr  00:04:9F:00:EA:D7
U-Boot> setenv ipaddr   10.193.102.93
U-Boot> setenv kernel   uImage
U-Boot> setenv nfsroot  /data/rootfs_home/rootfs
U-Boot> 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
U-Boot>
```

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 U-Boot 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:

```
U-Boot> printenv
bootdelay=3
baudrate=115200
loadaddr=0x10800000
netdev=eth0
ethprime=FEC0
U-Boot_addr=0xa0000000
U-Boot=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_U-Boot=tftpboot ${loadaddr} ${U-Boot}; protect off ${U-Boot_addr}
0xa003ffff; erase ${U-Boot_addr} 0xa003ffff; cp.b ${loadaddr} ${U-Boot_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 U-Boot

OS images can be downloaded using U-Boot 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 U-Boot 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 U-Boot prompt:

```
loady
```

U-Boot 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
  ```

5.3 Multi-environment store device

If need to modify the environment saving device, pls follow instructions below:

1. Environment in SD/MMC:

Normally, if CONFIG_DYNAMIC_MMC_DEVNO is defined, u-boot will use active SD slot for default environment storing device, that is, by default, u-boot will use the SD boot slot as the first choice for locating and saving environment data. Otherwise, SD environment will use CONFIG_SYS_MMC_ENV_DEV.

Need to modify boards' config files. The config file is under include/configs/<board_name>.h.
Make sure CONFIG_FSL_ENV_IN_MMC is not commented.
Make sure other CONFIG_FSL_ENV_IN_XXX is commented. Or, there will be a compiling error.

e.g.

```
/* #define CONFIG_FSL_ENV_IN_SF */
#define CONFIG_FSL_ENV_IN_MMC
```

2. Environment in SPI Flash:

Need to modify boards' config files. The config file is under include/configs/<board_name>.h.
Make sure CONFIG_FSL_ENV_IN_SF is not commented.
Make sure other CONFIG_FSL_ENV_IN_XXX is commented. Or, there will be a compiling error.

e.g.

```
#define CONFIG_FSL_ENV_IN_SF
/* #define CONFIG_FSL_ENV_IN_MMC */
```

3. Environment in SATA:

    Need to modify boards' config files. The config file is under include/configs/<board_name>.h.
    Make sure CONFIG_FSL_ENV_IN_SATA is not commented.
    Make sure other CONFIG_FSL_ENV_IN_XXX is commented. Or, there will be a compiling error.

e.g.

```
/* #define CONFIG_FSL_ENV_IN_SF */
#define CONFIG_FSL_ENV_IN_SATA
```

All environment settings listed in config files are supported and tested.
To enable them, you just need to enable accordingly configs.

5.4 Splash screen support

Splash screen is off by default. If need to show a splash screen on lcd panel, you should add it yourself.

1. In config file, enable splash screen.

Add:

```
#define CONFIG_SPLASH_SCREEN
```

then splash screen feature is now enabled.

2. Boot into u-boot console and set splash screen enviroment variables.

```
setenv splashimage '0x97c90000'
setenv splashpos '0,0'
setenv lvds_num 1
```

Note:

    splashimage is the image address that will be loaded.
    splashpos is the image position that will be displayed.
    If you want to use lvds0, then set lvds_num to 0.

3. How to change splash image.

Currently, we use converted image in board/freescale/common/fsl_bmp_600x400.c.
In order to change splash image, user need to:

- Prepare an bmp image for display, better to be 600x400. Other resolutions need to modify more source.
- Convert it to .c file. Pls ask us for bin2txt.py.
- Put converted bmp.c to board/freescale/common/fsl_bmp_600x400.c.
- Modify Makefile to build it.

e.g.

```
COBJS-${CONFIG_VIDEO_MX5}        += my_bmp.o
```

4. Now you can boot and check splash image displayed on panel.

5.5 EPDC splash screen support

EPDC splash screen is off by default. Thus if you need to show a splash screen on epdc lcd panel, you should add it yourself. The guide is listed below:

1. In config file, enable splash screen.

Find:

```
/*
#define CONFIG_SPLASH_SCREEN
*/
```

and remove comments indicator. Then splash screen feature is now enabled.

2. Program waveform file to sd card offset 0x100000.

- The waveform file epdc.fw.ihex is at \firmware\imx in linux-2.6-imx. But you may need another scrpt to convert it to raw waveform binary data.
  You should ask freescale for the scrpt ihex2bin.py.
- Use ihex2bin.py to get waveform binary data.
  `./ihex2bin.py –i epdc.fw.ihex –o wv_data.bin`
- Program wv_data.bin to sd card offset 0x100000. You can use ATK tool or dd command in Linux for this.
  `root@freescale /$ dd if=./wv_data.bin of=/dev/mmcblk0 bs=1024 seek=1024`
  Note:
  You can also program wv_data.bin to other offset in sd card. But you need to modify CONFIG_WAVEFORM_FILE_OFFSET in config file and rebuild u-boot.
  Now we only support waveform data file on sd/mmc card.
  If need to access other memory device for waveform data, e.g. spi nor, CONFIG_WAVEFORM_FILE_IN_MMC should be removed and CONFIG_WAVEFORM_FILE_IN_SF should be added.
  Codes to access spi nor should be added in function setup_waveform_file().

3. Boot your board and you will see splash screen on epdc lcd panel.

5.6 OTP fuse support

OTP Fuse is supported. Commands supported is listed below:
imxotp read <addr> - Read fuse at <addr>
imxotp blow [--force] <addr> <value> - Blow fuse at <addr> with hex value <value>
Note:
The parameter <addr> is the fuse bank offset in soc reference manual.

5.7 SATA support

SATA is supported. Commands supported is listed below:

```
sata init – init SATA sub system
sata info – show available SATA devices
sata device [dev] – show or set current device
sata part [dev] – print partition table
sata read addr blk# cnt
sata write addr blk# cnt
```

5.8 eMMC Fastboot

eMMC fastboot is supported.
For fastboot, please note that the bit width of card should match the dip settings.
For example, if mmcinfo shows eMMC 4.4 card is 8Bit DDR, then dip settings should be 8bit DDR. Then fastboot can work. Otherwise, fastboot will fail.

```
MX53-SMD U-Boot > mmcinfo 1
Device: FSL_ESDHC
Manufacturer ID: 45
OEM: 100
Name: SEM08
Tran Speed: 25000000
Rd Block Len: 512
MMC version 4.0
High Capacity: Yes
```

```
Capacity: 7944011776
Bus Width: 8-bit DDR
Boot Partition Size:  1024KB
Current Partition for boot: Boot partition 1
```

6 Building U-Boot

There are two ways to build U-Boot, from Ltib and from source.

6.1 Building U-Boot from ltib

1. Get Ltib.

```
git clone git://git.am.freescale.net/git/ltib.git
git checkout --track -b branch-imx-sdk origin/branch-imx-sdk
```

Note:

An easier to get ltib is using coltib in linux-infra.

```
~/linux-infra/tools/coltib
```

2. Get U-Boot source from ltib.

Change into ltib directory.

```
./ltib -m prep -p u-boot
```

Note:

If it is the first time that you run ltib, it will let you to set some configurations.
Please set [Freescale iMX reference boards] -> [imx25_3stack imx233 imx31_3stack imx35_3stack imx37_3stack iMX51_3stack iMX6Q iMX6S] in sequence.

3. Build U-Boot in ltib

```
./ltib -m scbuild -p u-boot
```

4. Install u-boot in ltib

```
./ltib -m scdeploy -p u-boot
```

6.2 Building U-Boot from source

Then you can get u-boot.bin in ltib/rootfs/boot directory.

Building U-Boot 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 6.2.0.1: Resources needed for Building U-Boot

Resource	Description	Source
u-boot-2009.08.tar.bz2	U-Boot base source for Linux	pkgs/
gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12-1.i386.rpm	Toolchain needed to build U-Boot under Linux	included
u-boot-v2009.08-imx_XX.XX.XX.tar.bz2	Freescale U-Boot source patch and binary files.	pkgs/

Unzip the Freescale U-Boot release package `L3.0.15_XX.XX.XX_source.tar.gz`. You will find the U-Boot source patch files and `u-boot-2009.08.tar.bz2` inside `L3.0.15_XX.XX.XX_source/pkgs`.

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

6.2.1 Setting up U-Boot Build Environment under Linux

To simplify the built process, it is recommended to use Freescale's pre-packaged tools to build U-Boot. 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_XX.XX.XX.tar.bz2` from this package.
2. Under Linux, with root user privilege, install `gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12-1.i386.rpm`.
   A more simple way is to run ltib in release firstly which will install the toolchain for host server.

6.2.2 Generating U-Boot Image

After the tools are setup properly, follow these instructions to build the U-Boot 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_XX.XX.XX.tar.bz2`:
   ```
   $ cd u-boot-2009.08
   ```
   Under Linux, do:
   ```
   $ tar jxvf u-boot-v2009.08-imx_XX.XX.XX.tar.bz2
   $ ./patches/patch-U-Boot.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 U-Boot. 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.

   Export toolchain first:

   export CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/arm-none-linux-gnueabi-

   For i.MX6 Quad arm2 board, do:
   ```
   make mx6q_arm2_config && make
   ```
   For i.MX6 Dual, do:
   ```
   make mx6dl_arm2_config && make
   ```
   For i.MX6 Quad sabresd board, do:
   ```
   make mx6q_sabresd_config && make
   ```
   For i.MX6 Dual sabresd board, do:
   ```
   make mx6dl_sabresd_config && make
   ```
   For i.MX6 Qual sabrelite board, do:
   ```
   make mx6q_sabrelite_config && make
   ```
   For i.MX6Solo sabreauto board, do:
   ```
   make mx6solo_sabreauto_config && make
   ```
   For i.MX6 Quad sabreauto board, do:
   ```
   make mx6q_sabreauto_config && make
   ```
   For i.MX6 Solo arm2 board, do:
   ```
   make mx6sl_arm2_config && make
   ```

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

7 U-Boot 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 U-Boot

```
setenv [env_var] [val]
saveenv
```

7.5 SPI-NOR commands

Initial SPI-NOR flash

```
sf probe
```

Read from SPI-NOR

```
sf read <dest_addr> <src_offset> <length>
```

Erase SPI-NOR

```
sf erase <src_offset> <length>
```

Write to SPI-NOR

```
sf write <dest_addr> <src_offset> <length>
```

7.6 MMC commands

1. Show or set current mmc device [partition]

   ```
   mmc dev [dev_no] [partition_no]
   ```

   Example:

   (1). Show current mmc device.

```
U-Boot > mmc dev
mmc0 is current device
```

(2). Change to mmc device 1.

```
U-Boot > mmc dev 1
Mmc1(part 0) is current device
```

(3). Change to mmc device 1, partition 1

```
U-Boot > mmc dev 1 1
Mmc1(part 1) is current device
```

2. Read data from mmc

```
Mmc read <addr> <blk> <cnt>
```

Example:

(1). Read data from device 0. Current device is 0.

```
U-Boot > mmc read 0x70100000 0 0x200

MMC read: dev # 0, block # 0, count 512 ... 512 blocks read: OK
```

(2). Read data from device 1. Current device is 0.

```
U-Boot > mmc dev 1
Mmc1(part 0) is current device
U-Boot > mmc read 0x70100000 0 0x200

MMC read: dev # 1, block # 0, count 512 ... 512 blocks read: OK
```

3. Write data to mmc

Mmc write <addr> <blk> <cnt>

Example:

(1). Write data to device 0. Current device is 0.

```
U-Boot > mmc write 0x70100000 0 0x200

MMC write: dev # 0, block # 0, count 512 ... 512 blocks write: OK
```

(2). Write data from device 1. Current device is 0.

```
U-Boot > mmc dev 1
Mmc1(part 0) is current device
U-Boot > mmc write 0x70100000 0 0x200

MMC read: dev # 1, block # 0, count 512 ... 512 blocks read: OK
```

4. Show or set boot partition (for eMMC card)

Mmc bootpart [dev] [part]

Example:

(1). Show boot partition

```
U-Boot > mmc bootpart
Card doesn't support boot partition feature
U-Boot > mmc bootpart 1
Device 1: boot partition 1 is for boot
```

(2). Set boot partition

```
U-Boot > mmc bootpart 1 1
Device 1: boot partition 1 is for boot
U-Boot > mmc bootpart 1 0
Switch boot partition to partition #0, OK
Device 1: boot partition 0 is for boot
```

5. Lists available partition on current mmc device

Mmc list

Example:

```
U-Boot > mmc list
```

```
FSL_ESDHC: 0
FSL_ESDHC: 1
```

6. Erase data
   Mmc erase <blk> <cnt>
   Example:
   (1). Erase 0x200 blocks on blk offset 0 on current device

```
U-Boot > mmc erase 0x0 0x200

MMC erase: dev # 1, block # 0, count 512 ...

Caution! Your devices Erase group is 0x400
The erase range would be change to 0x0~0x3ff

512 blocks erase: OK
```

7. Rescan device
   Mmc rescan
   Example:

```
U-Boot > mmc rescan
U-Boot > mmc list
FSL_ESDHC: 0
FSL_ESDHC: 1
```

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 EXT2 commands

List files in a directory (default /)

```
ext2ls <interface> <dev[:part]> [directory]
```

Load binary file from a Ext2 filesystem

```
ext2load <interface> <dev[:part]> [addr] [filename] [bytes]
```

7.9 PATA commands

Show available PATA devices

```
pata info
```

Show or set current device

```
pata device [dev]
```

Print partition table

```
pata part [dev]
```

Read data from pata device

```
pata read addr blk# cnt
```
Write data to pata device
```
pata write addr blk# cnt
```

7.10 SATA commands

init SATA sub system
```
sata init
```
show available SATA devices
```
sata info
```
show or set current device
```
sata device [dev]
```
print partition table
```
sata part [dev]
```
read data from sata
```
sata read addr blk# cnt
```
write data to sata
```
sata write addr blk# cnt
```

7.11 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 U-Boot 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
```

8.3 How to set mac address to FEC in boot?

Pls set the fec_addr environment data.
```
setenv fec_addr 00:ab:cd:df:gf:hj
```

FEC network init code will check fec_addr. If this environment data is set, the value will be write to FEC registers.