i.MX25 PDK Linux

User’s Guide

Document Number: 924-76358
Rev. 2009.12
01/2010

Contents

About This Book

This document explains how to build and install the Freescale Linux BSP to the i.MX25 3-Stack board including board dip switch settings for image download and all kinds of boot mode, the steps to download image through ATK, redboot and u-boot, as well as the boot commands for each boot mode.

Audience

This document is intended for software, hardware, and system engineers who are planning to use the product and for anyone who wants to understand more about the product.

References

1. *i.MX Family Linux Software Development Kit Reference Manual*
2. *Advanced Toolkit Standard User's Guide*
3. Redboot documentation in Redboot release package

Chapter 1
Introduction

The i.MX25 3-Stack Linux BSP is a collection of binary, source code, and support files that can be used to create a Linux kernel image and a root file system for i.MX25 3-Stack board.

1.1 Boot Loader

The i.MX25 3-Stack Linux delivery package contains RedBoot bootloader binaries and u-boot bootloader binaries.

The `redboot_<version>.zip` file contains bootloader binaries for the i.MX 3-Stack boards. The following table lists the binary files for i.MX25 3-Stack boards:

Binary Name	Description
MX25_TO1_1_3stack_redboot.bin	Redboot binary for i.MX25 3-Stack TO1.1 board which support MMC/SD.
MX25_TO1_1_3stack_redboot-no-padding.bin	Redboot binary for i.MX25-Stack TO1.1 board which is used to program to the MMC card from a host PC or from Linux running on the target directly to offset 0x400 on the card.

For more information about RedBoot utilities, refer to `redboot_<version>/doc/redboot_MX25.pdf.`

The default u-boot-3ds.bin in the release package supports flash boot. To build u-boot-3ds.bin for MMC/SD boot, please refer to uboot user guide.

1.2 Linux Kernel image

This Freescale 3-Stack BSP contains the Freescale Linux 2.6.31 3-Stack kernel, driver source code, and a pre-built kernel image. You can obtain the i.MX25 3-Stack kernel image from the following location: `zImage and uImage.` zImage is used together with redboot, while uImage is used together with uboot.

1.3 Root File System

The root file system package provides busybox, common libraries, and other fundamental elements. The i.MX25 3-Stack BSP contains three kinds of root file systems:

```
rootfs.jffs2
```

```
rootfs.ext2.gz
```

rootfs.jffs2 and rootfs.ext2.gz file system includes Freescale specific libraries and gnome GUI.

Chapter 2
Building the Linux Platform

This chapter explains how to set up the build environment, install and build LTIB, set rootfs for NFS, and set up the host environment.

2.1 Setting Up the Build Environment

Setting up the build environment includes installing Linux OS and LTIB.

2.1.1 Install Linux OS using Linux Builder

Install a Linux distribution such as Ubuntu 9.04, Fedora 4/5 or RedHat on one computer. To build gnome mobile profile, ubuntu 9.04 is required.

2.1.2 Linux Host Configuration

The following instructions introduce how to setup TFTP and NFS server. The detailed setting needs to refer to help menu in your host machine. Here is one example how to set up Redhat:

1. Turn off the firewall, to enable the `tftp` to work:

```
 iptables -F
```

   OR, at the command line, type:

```
setup
```

2. Install the **tftp** server.
3. Install the **nfs** server.
4. Create the `tftboot` directory.

   The kernel images and anything that needs to be uploaded by `tftp` (such as the **zImage** kernel image) will be stored in this directory:

```
 mkdir /tftpboot
```

5. Edit `/etc/xinetd.d/tftp` to enable tftp as follows:

```
{
disable  =  no
socket_type = dgram.
protocol = udp.
wait = yes
user = root
server = /usr/sbin/in.tftpd.
server_args = /tftpboot
}
```

6. Run the following command on your Linux host machine:

```
vi /etc/exports
```

add this line in the file: $ROOTFS_DIR `*(rw,sync,no_root_squash)`

`Note: Please replace $ROOTFS_DIR with correct rootfs folder name. For example, /tools/rootfs.`

7. Restart the **nfs** and **tftp** servers on your host:

```
/etc/init.d/xinetd restart
/etc/init.d/nfs restart
```

NOTE

These instructions specify using an **nfs** server. Some Linux systems use **nfsserver**, rather than **nfs**. Use these instructions for either server type.

NOTE

A Windows tftp program "tftp.zip" is located under LTIB release package "Common/" folder. You can install it in Windows OS to setup Windows tftp server for downloading images.

2.2 Installing and Building LTIB

To install and build LTIB, use these steps.

NOTE

In some Linux systems, the following procedure must be done with "root" permissions. However, these instructions are for performing the procedure "not as root".

1. Install the LTIB package not as root:

```
tar zxf <ltib_release>.tar.gz
./<ltib_release>/install
```

This command installs LTIB to your directory.

2. Build LTIB:

```
cd <your LTIB directory>
./ltib -m config
```

Set platform choice to *"Freescale iMX reference boards"* and exit saving the changes. On the next menu, select platform type (`imx25_3stack`) and a package profile. Exit and save changes. On the *"Freescale iMX25 3-Stack Board"* screen, you may set *"Target* *image: (NFS only)"*, under *"Target Image Generation"*, *"Options"* (this would instruct LTIB to create a file system tree suitable for use with NFS).

Then run the following command:

```
./ltib
```

When the build completes, you can obtain the kernel image from `rootfs/boot/zImage`. Keep in mind that if your host system is missing packages that are needed by LTIB during its execution, you will be prompted to install additional packages to it.

NOTE

Default supported nand flash model is Samsung K9LBG08U0D-PCB0. If the nand flash model is Samsung K9LAG08U0M in your board, execute the following command on the jffs2 file after the image is generated. Only if the file system is jffs2 format file.

Below is the command line to make a jffs2 rootfs file from a rootfs directory tree generated with LTIB, for i.MX25 3-stack platform.

Mx25-3stack (Nand, Samsung K9LBG08U0D-PCB0, 4k page size, erase size:0x80000, MLC)

```
mkfs.jffs2 -r rootfs -e 0x80000  -s 0x1000 -n -o rootfs.jffs2
```

mx25-3Stack (Nand,Samsung K9LAG08U0M-PCB0, 2k page size, erase size:0x40000,MLC):

```
mkfs.jffs2 -r rootfs -e 0x40000  -s 0x800 -n -o rootfs.jffs2
```

2.3 Setting rootfs for NFS

There are two ways to set the rootfs for NFS on this package.

- Use the ext2 format rootfs package already provided in the distribution;
- Use the rootfs that is created after making the build of the kernel

Method 1: Using the rootfs package in the distribution

Use the following commands to set the `rootfs` directory for NFS (you must be the root user for this operation):

```
mkdir /mnt/rootfs
mkdir /tools
cp /<path_on_your_system>/rootfs.ext2.gz  /tools
cd /tools
gunzip rootfs.ext2.gz
mount -o loop -t ext2 rootfs.ext2 /mnt/rootfs
cp -r /mnt/rootfs .
export ROOTFS_DIR=/tools/rootfs
```

Method 2: Using the rootfs created after the kernel build

Instead of using the rootfs.ext2.gz, you could use the root file system in <`your LTIB directory`>.

```
%export ROOTFS_DIR=/<your LTIB directory>/rootfs
```

Other methods: For other ways to make a file system image file, see the *i.MX Family Linux Software Development Kit Reference Manual*.

2.4 Copying images to TFTP server

To use tftp server to download image, copy kernel image in the release package or LTIB to the tftp directory. For example,

```
cp iMX25_3stack/zImage /tftpboot
cp iMX25_3stack/rootfs.jffs2 /tftpboot
```

or

```
cp /<your LTIB directory>/rootfs/boot/zImage /tftpboot
cp /<your LTIB directory>/rootfs.jffs2 /tftpboot
```

Chapter 3 Board Dip switch setup

i.MX25 3stack board supports internal boot mode for NAND and MMC/SD. i.MX25 3-stack board consist of a CPU board, a Personality board and a Debug board. The boot modes are controlled by the dip switches on Debug board and personality board.

3.1 Dip switch setup for ATK downloading

Table 3.1-1 Debug board switch setup for ATK downloading

SW5	SW6	SW7	SW8	SW9	SW10
OFF	OFF	OFF	OFF	ON	ON

3.2 Dip switch setup for boot modes

Table 3.2-1 Debug board SW4 switch setup for boot

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

Table 3.2-2 Debug board SW5-SW10 switch setup for boot

Switch	SW5	SW6	SW7	SW8	SW9	SW10
Internal Boot (MMC/SD and NAND)	OFF	OFF	OFF	OFF	OFF	OFF
External Boot	OFF	OFF	OFF	OFF	ON	OFF

Table 3.2-3 Personality board switch setup for NAND internal boot

Samsung K9LAG08U0M-PCB0 (MCIMX25WPDK)								
Switch	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
SW21	ON	OFF	OFF	ON	ON	OFF	OFF	OFF
SW22	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
Samsung K9LBG08U0D–PCB (MCIMX25WPDKJ)								
Switch	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
SW21	ON	OFF	OFF	ON	OFF	ON	OFF	OFF
SW22	OFF	OFF	ON	ON	OFF	OFF	OFF	OFF

Table 3.2-4 Personality board switch setup for MMC/SD internal boot

Switch	1	2	3	4	5	6	7	8
SW21	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
SW22	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

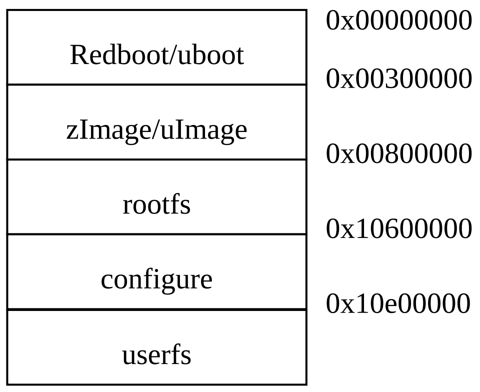
Chapter 4 Flash memory map

The next two chapters will make use of flash memory address frequently, so this chapter gives out a sketch of flash memory map. Flash memory scheme mainly depends on the bootloader, kernel image, rootfs and flash size. Flash physical start address are fixed by hardware design. On i.MX25 3-stack platform, NAND flash and MMC/SD flash start address are both 0x00000000 as they use different boot modes. The distribution of bootloder, kernel image and rootfs on flash can be configured according to flash size and its characteristics.

4.1 NAND flash memory map

NAND flash scheme is configured statically by software. It can be adjusted according to different requirements. Figure 4.1-1 illustrates default NAND flash map on i.MX25 3-3satck platform.

Figure 4.1-1 NAND flash memory scheme



4.2 MMC/SD flash memory map

MMC/SD flash scheme is some different from NAND flash. MMC/SD flash must keep first sector (512 bytes) as MBR (Master Boot Record). At boot up, MBR is executed to look up the partition table to determine which partition to use for booting. Bootloader should be put at the end of MBR. Kernel Image and rootfs can be put any address after bootloader. Please refer to the Appendix for the partition method of MMC/SD.

Chapter 5 Downloading Bootloader, zImage and rootfs to Target using ATK

This chapter explains how to install the ATK tool, erase FLASH, download Redboot, u-boot, zImage and rootfs to NAND and MMC/SD card.

5.1 Installing the ATK Tools

Download the ATK tool install package from the Freescale release web site or from Freescale support.

5.2 NAND Flash

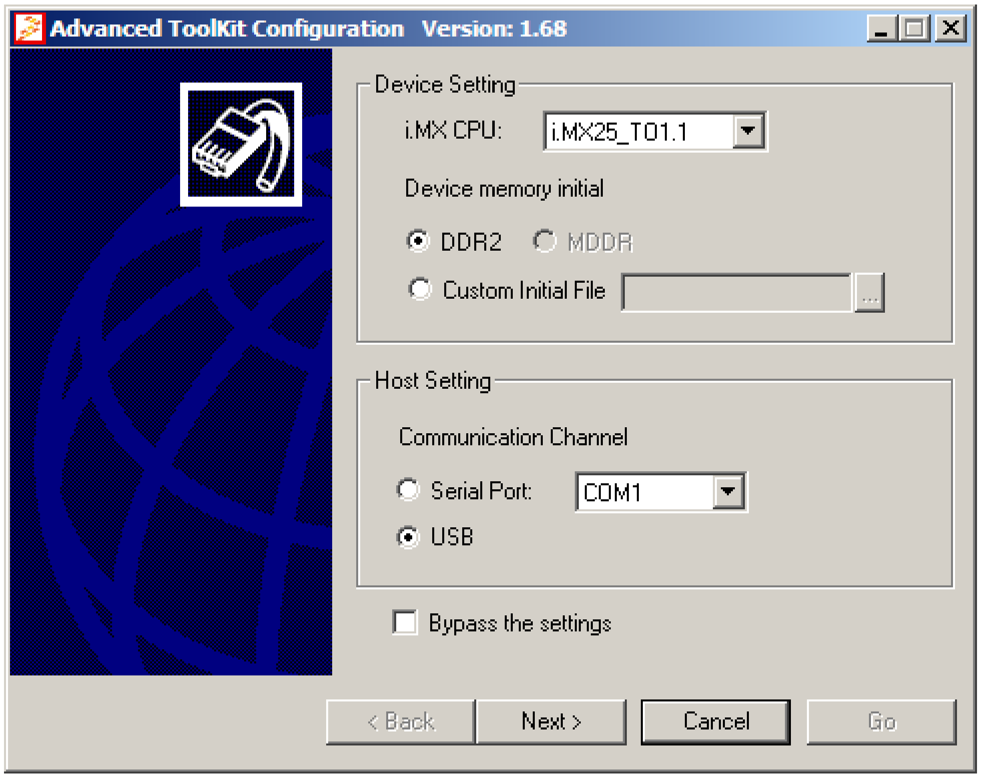
5.2.1 Erasing the NAND Flash

If the board does not boot up or identifies an excessive number of bad blocks when booting up, erase NAND flash and power up the boards again. This problem may occur because a different NAND bad block handling mechanism had been used previously.

To erase the NAND Flash, use these steps:

1. Set dip switch by referring to table 3.1-1 and 3.1-2 on Section Dip switch setup for ATK downloading.
2. Power on i.MX25 3stack board and Run the ATK, and then select the options for your platform, for **Device memory initial** (**DDR2** or **MDDR**). Select “DDR2” for i.MX25 TO1.1 3stack. For **Communication Channel**, both serial and USB can be used. However, we suggest to use USB since it is faster than serial. Figure 5.2.1 illustrates ATK configuration for i.MX25 TO1.1 3stack.
3. Click **next** and select **Flash Tool**, then click **Go.**

Figure 5.2-1 i.MX25 TO1.1 ATK configuration


Advanced ToolKit Configuration Version: 1.68

Device Setting

i.MX CPU: i.MX25_TO1.1

Device memory initial

[✓ DDR2] MDDR

Custom Initial File

Host Setting

Communication Channel

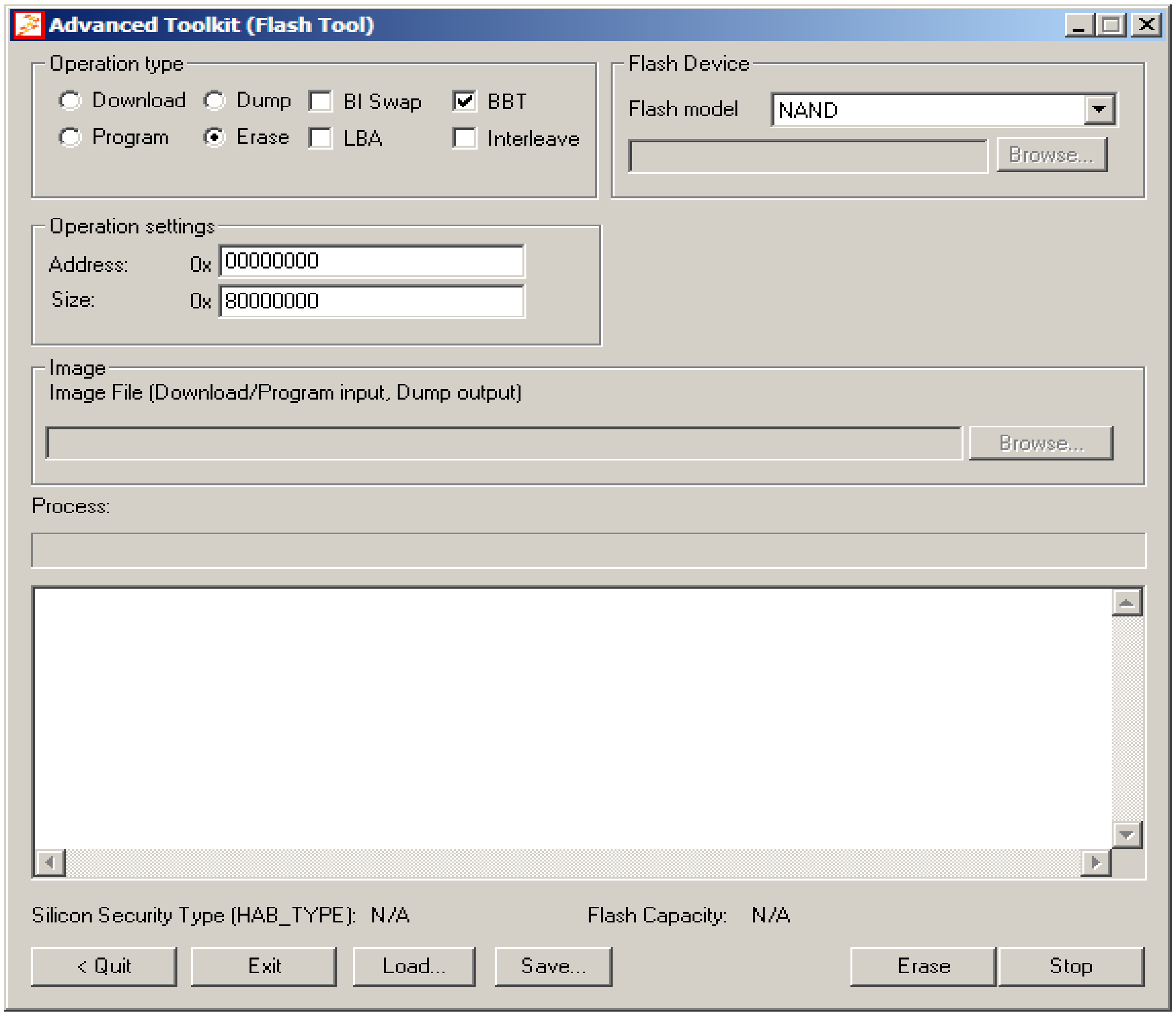
Serial Port: COM1

[✓ USB]

Bypass the settings

4. Select the erase options as Figure 5.2-2
5. Click **Erase** to erase the flash.

Figure 5.2-2 i.MX25 TO1.1 NAND flash Erase configuration


Advanced Toolkit (Flash Tool)

Operation type: Download, Dump, BI Swap, [✓ BBT], Program, [✓ Erase], LBA, Interleave

Flash Device — Flash model: NAND

Operation settings — Address: 0x 00000000; Size: 0x 80000000

Image — Image File (Download/Program input, Dump output)

Process:

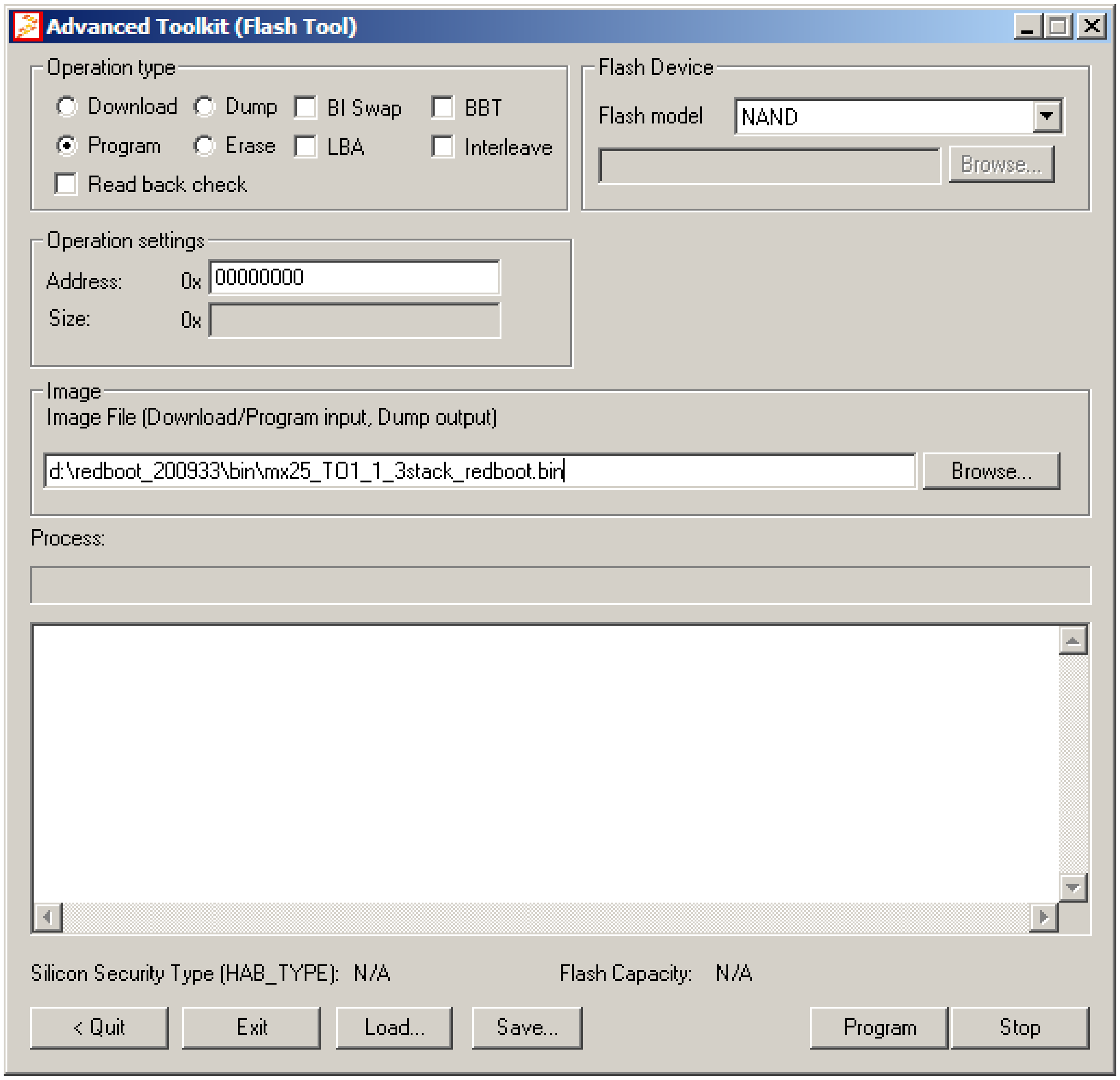
Silicon Security Type (HAB_TYPE): N/A   Flash Capacity: N/A

5.2.2 Download RedBoot/u-boot zImage and rootfs to NAND Flash

To download the bootloader, use these steps:

1. Set ATK as same steps 1-3 in 5.2.1
2. Program RedBoot/u-boot by selecting the options as Figure 5.2-3
3. Click **Browse** to select Redboot/u-boot and click **Program** to download.

Figure 5.2-3 i.MX25 programming bootloader to NAND flash



Operation type: [✓ Program]

Flash model: NAND

Address: 0x 00000000

Size: 0x

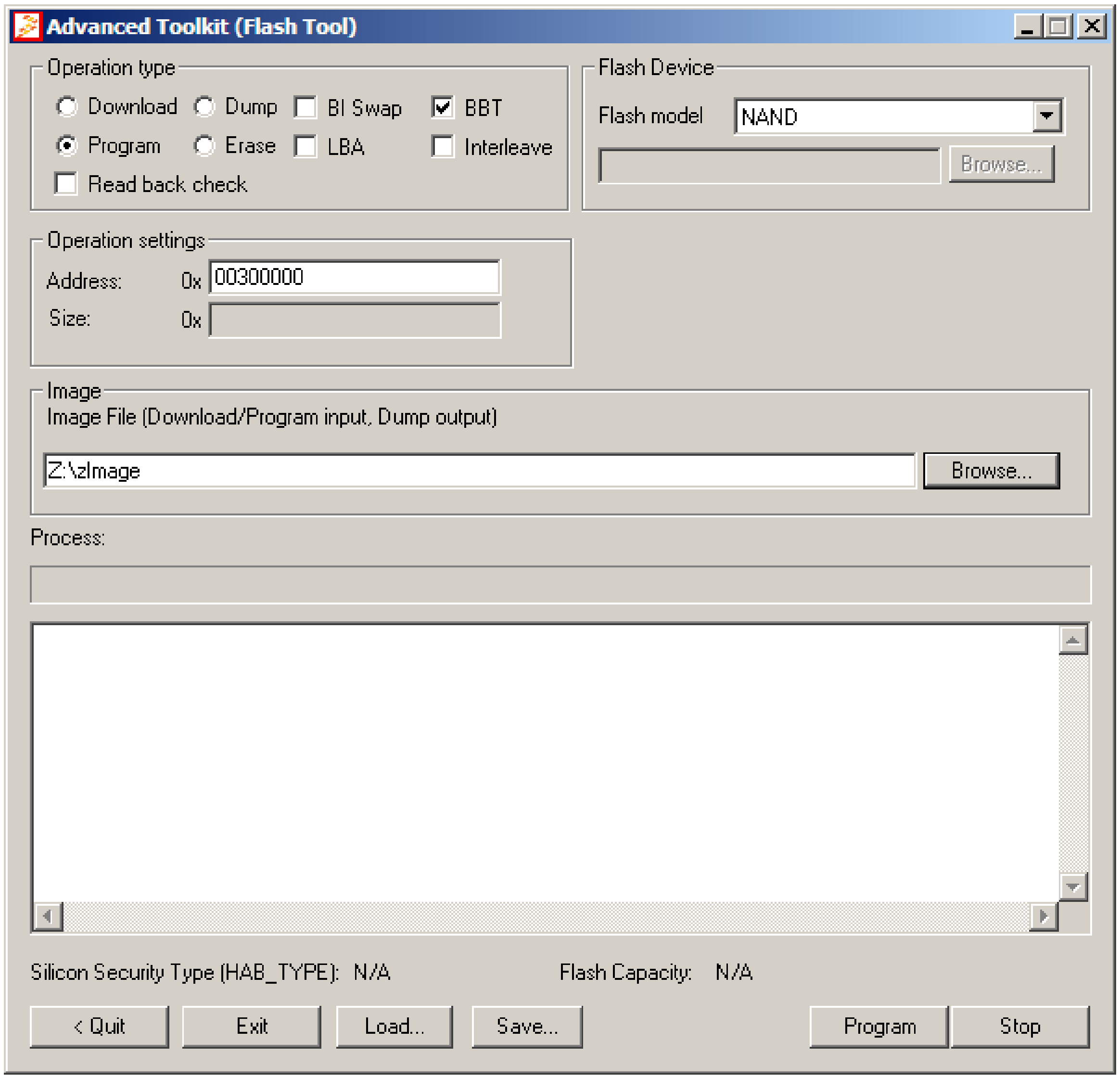
d:\redboot_200933\bin\mx25_TO1_1_3stack_redboot.bin

Silicon Security Type (HAB_TYPE): N/A

Flash Capacity: N/A

4. Program zImage/uImage by selecting the options as Figure 5.2.4. Address is 0x300000 for both zImage and uImage.
5. Click **Browse** to select zImage/uImage and click **program** to download

Figure 5.2-4 i.MX25 program Kernel image to NAND flash



Operation type: Download [✓ Program] Dump Erase BI Swap LBA [✓ BBT] Interleave Read back check

Flash model: NAND

Address: 0x 00300000

Size: 0x

Image File (Download/Program input, Dump output)

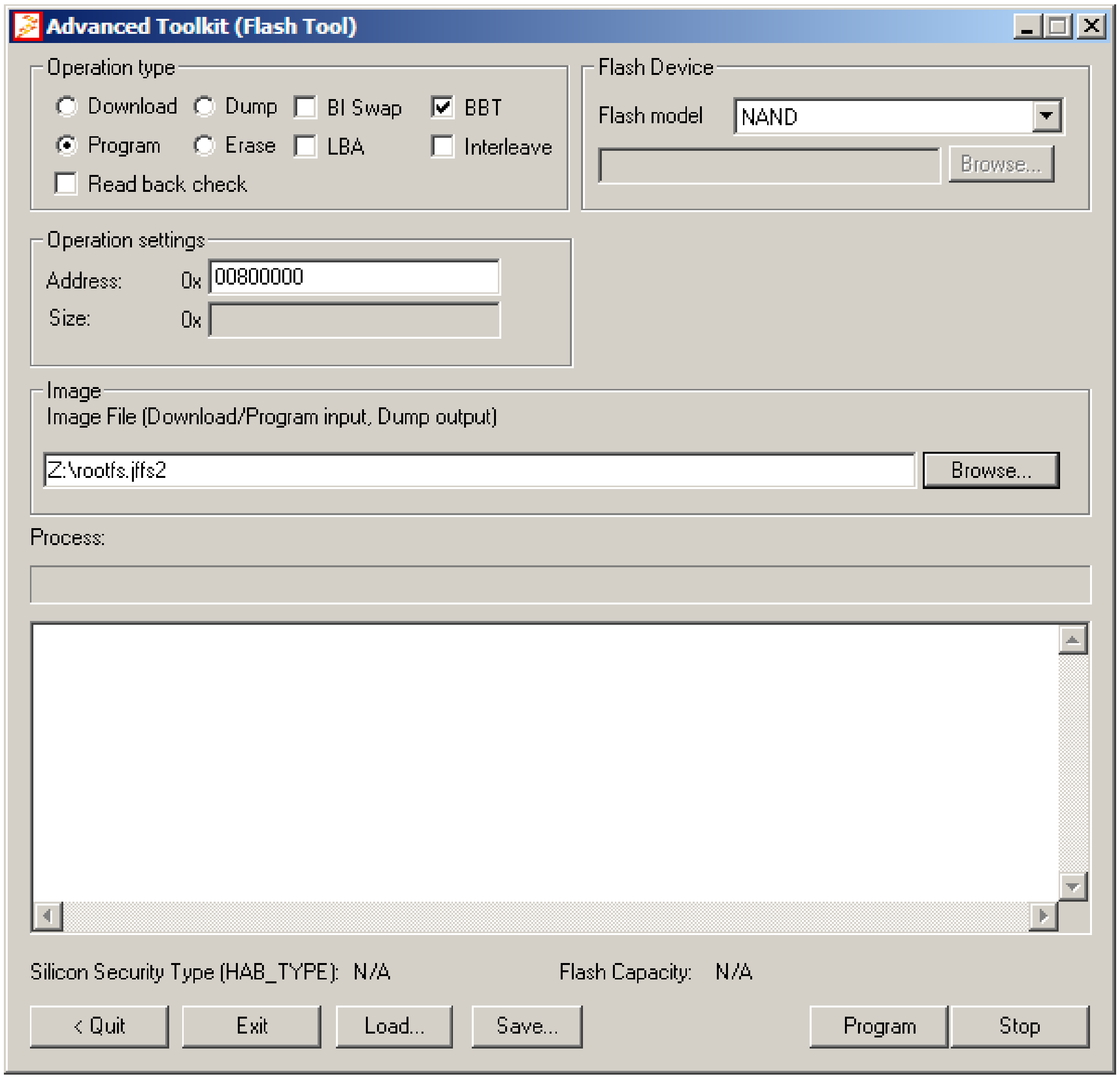
Z:\zImage

Silicon Security Type (HAB_TYPE): N/A

Flash Capacity: N/A

6. Program rootfs by selecting the options as Figure 5.2.5
7. Click **Browse** to select rootfs.jffs2 and click **Program** to download.

Figure 5.2-5 i.MX25 program rootfs to NAND flash


Advanced Toolkit (Flash Tool)

Operation type: ( ) Download ( ) Dump [ ] BI Swap [✓ BBT] [✓ Program] ( ) Erase [ ] LBA [ ] Interleave [ ] Read back check

Flash Device: Flash model: NAND

Operation settings:
Address: 0x 00800000
Size: 0x

Image
Image File (Download/Program input, Dump output)
Z:\rootfs.jffs2

Process:

Silicon Security Type (HAB_TYPE): N/A
Flash Capacity: N/A

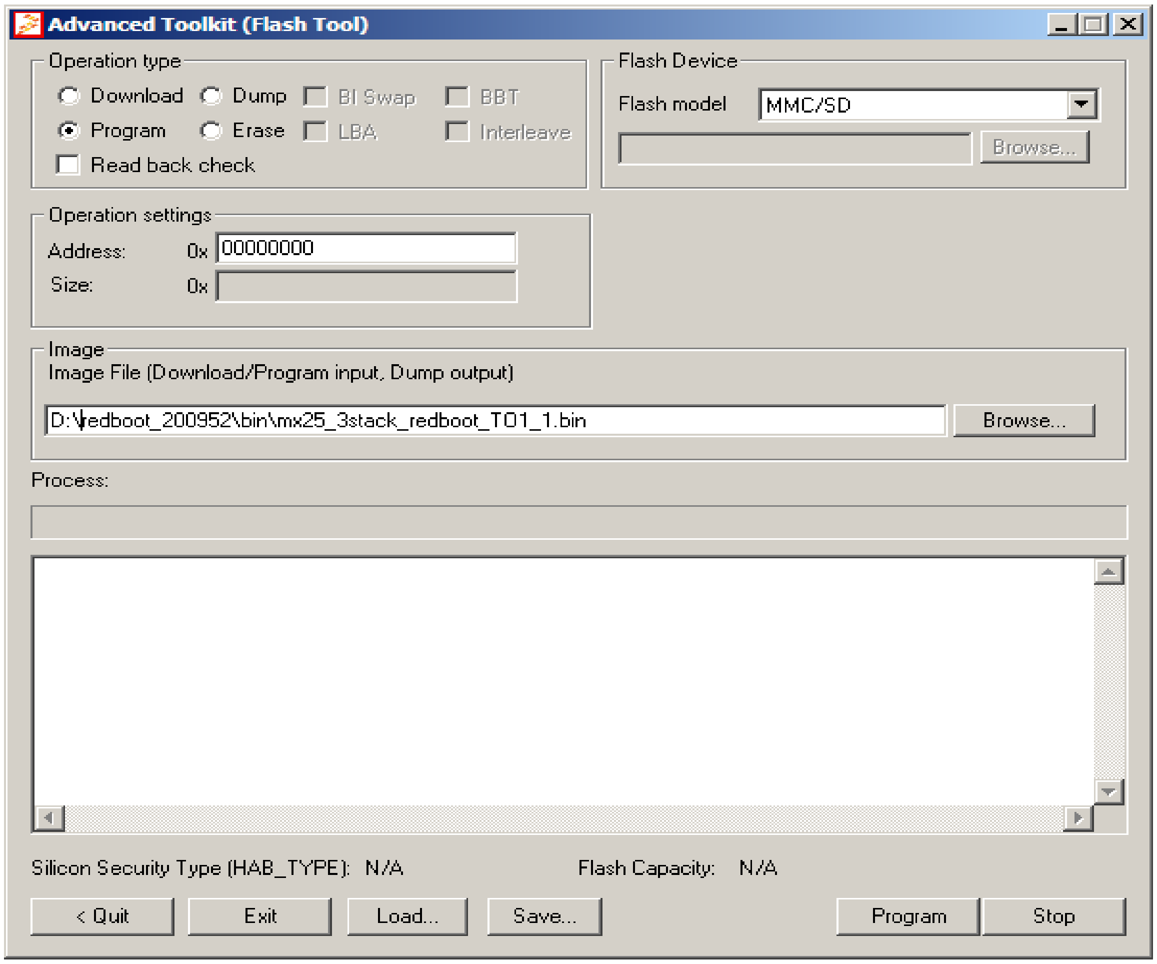
5.3 MMC/SD

5.3.1 Download RedBoot/u-boot and zImage/uImage

To download the RedBoot/u-boot bootloader to MMC/SD card, use these steps:

1. Set ATK as same steps 1-3 in 5.2.1
2. Program RedBoot/u-boot by selecting the options as Figure 5.3-1.
3. Click **Browse** to select the bootloader and click **Program** to download.

Figure 5.3-1 i.MX25 programming bootloader to MMC/SD card



Operation type: ( ) Download ( ) Dump [✓ Program] ( ) Erase ☐ Read back check

Flash model: MMC/SD

Operation settings

Address: 0x 00000000

Size: 0x

Image

Image File (Download/Program input, Dump output)

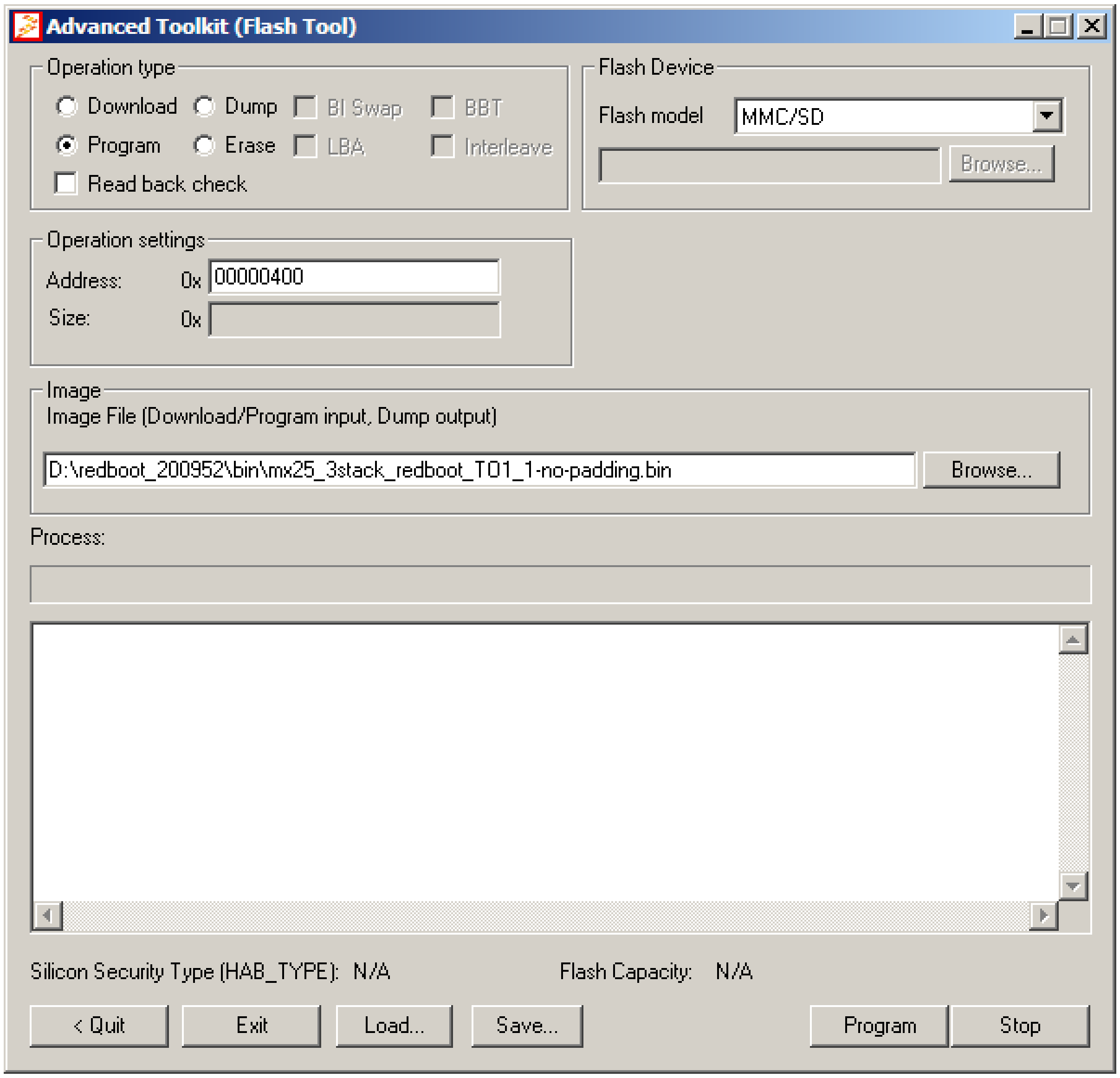
D:\redboot_200952\bin\mx25_3stack_redboot_TO1_1.bin

Silicon Security Type (HAB_TYPE): N/A

Flash Capacity: N/A

Please note that this operation will delete the partition table present on the medium. You need to program mx25_3stack_redboot_TO1_1-no-padding.bin for redboot with address 0x400 if you don’t want to delete the partition table as Figure 5.3-2.

Figure 5.3-2 i.MX25 programming no-padding redboot to MMC/SD card


Advanced Toolkit (Flash Tool)

Operation type: Download [✓ Program] Dump Erase BI Swap LBA BBT Interleave Read back check

Flash Device — Flash model: MMC/SD

Operation settings — Address: 0x 00000400 — Size: 0x

Image — Image File (Download/Program input, Dump output): D:\redboot_200952\bin\mx25_3stack_redboot_TO1_1-no-padding.bin

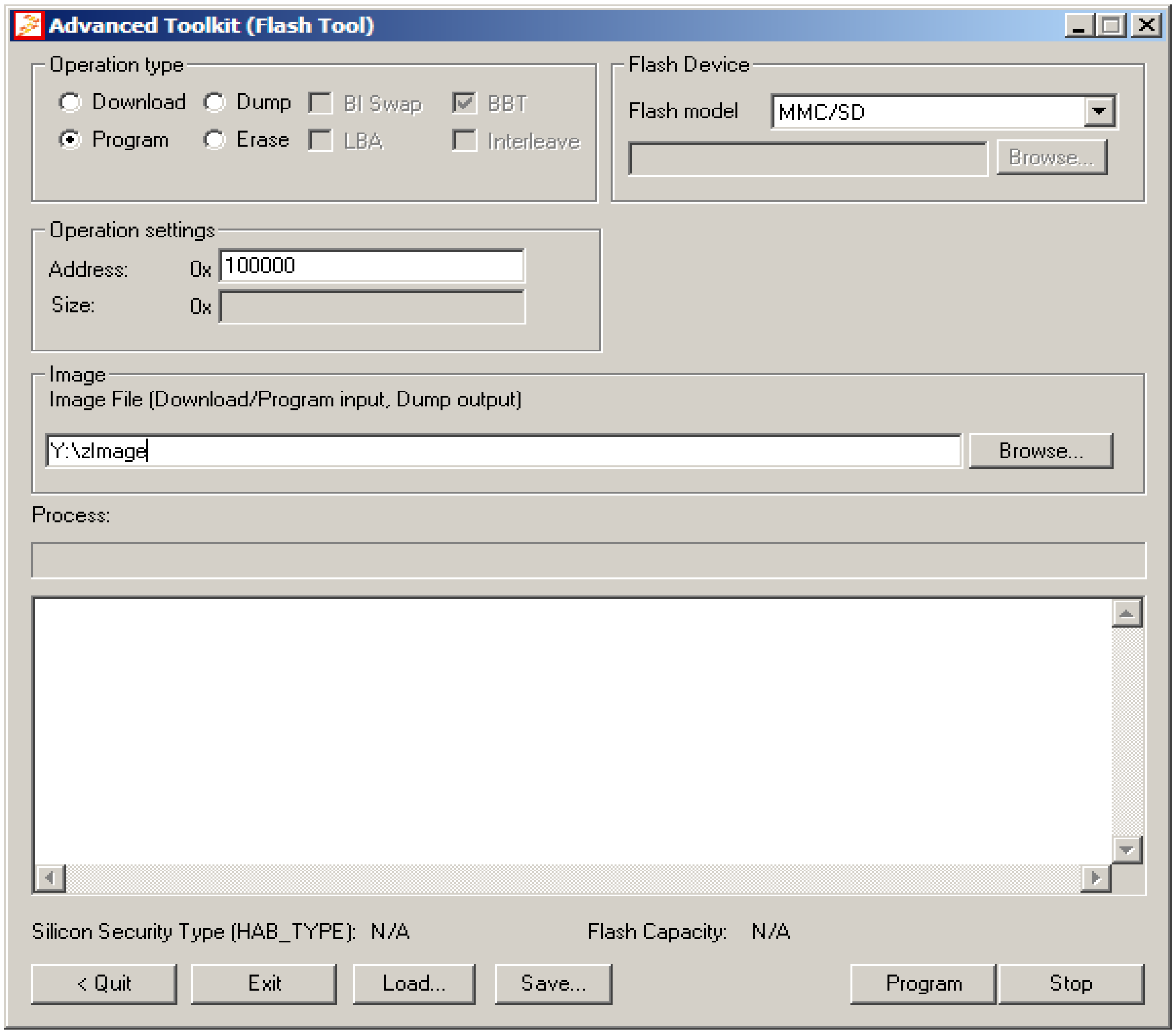
Process:

Silicon Security Type (HAB_TYPE): N/A   Flash Capacity: N/A

4. Program zImage/uImage by selecting the options as Figure 5.3-3. Address is 0x100000 for programming. Make sure you have enabled ext3 file system support of the zImage/uImage as chapter 6.2.2 if you use exe3 file system in MMC/SD card.

5. Click **Browse** to select the zImage/uImage and click **Program** to download.

Figure 5.3-3 i.MX25 Programming Kernel Image to MMC/SD Flash



Advanced Toolkit (Flash Tool)

Operation type: Download ( ) Dump ( ) BI Swap ☐ BBT ☑ Program (●) Erase ( ) LBA ☐ Interleave ☐

Flash Device: Flash model: MMC/SD

Operation settings: Address: 0x 100000 Size: 0x

Image File (Download/Program input, Dump output): Y:\zImage

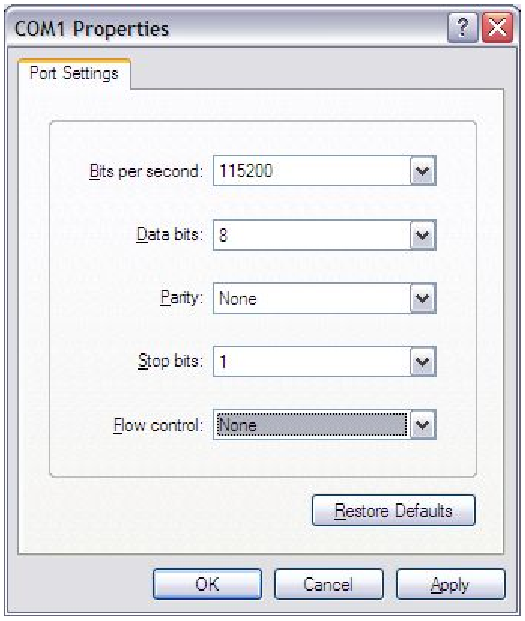
Silicon Security Type (HAB_TYPE): N/A Flash Capacity: N/A

Chapter 6 Download Kernel and rootfs by bootloader

6.1 Setup Terminal

The i.MX25 3-Stack board can communicate with Host server (Windows or Linux) via the serial cable. The common communication programs such as HyperTerminal, Tera Term, PuTTY can be found from the web pages. Here is the example to setup Terminal via HyperTerminal in Windows Host:

6. Connect the target and the Windows PC via a serial cable.
7. Open HyperTerminal on the Windows PC, and select the settings shown in Figure 6.1-1.



Bits per second: 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Figure 6.1-1 HyperTerminal Settings

8. Power up the target by setting up the right dips, and wait until the system enters bootloader prompt.

6.2 Download by Redboot

6.2.1 NAND

NOTE

Cannot download by this method if rootfs.jffs2 size is more than 64M since total SDRAM is 64M. Please use ATK to download images.

1. Initialize and erase flash

```
RedBoot> fis init -f
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
... Erase from 0x00040000-0x00080000: .
... Erase from 0x000c0000-0x80000000: ......
```

2. Download the kernel image (zImage under /tftpboot) to RAM:

```
RedBoot> load -r -b 0x100000 zImage -h <host IP address>
```

3. Program the zImage into NAND flash:

```
RedBoot> fis create -f 0x300000 kernel
... Read from 0x07ec0000-0x07eff000 at 0x00080000: ..
... Read from 0x07ec0000-0x07eff000 at 0x00080000: ..
... Read from 0x07ec0000-0x07eff000 at 0x00080000: ..
... Erase from 0x00100000-0x00300000: ........
... Program from 0x00100000-0x002d1254 at 0x00100000: ........
... Erase from 0x00080000-0x000c0000: .
... Program from 0x07ec0000-0x07f00000 at 0x00080000: ..
```

4. Download the jffs2 format rootfs (rootfs.jffs2 under /tftpboot) to RAM:

```
RedBoot> load -r -b 0x100000 rootfs.jffs2 -h <host IP address>
```

Program the rootfs into NAND flash:

```
RedBoot> fis create -f 0x800000 root
... Read from 0x07ec0000-0x07eff000 at 0x00080000: ..
... Read from 0x07ec0000-0x07eff000 at 0x00080000: ..
... Read from 0x07ec0000-0x07eff000 at 0x00080000: ..
... Erase from 0x00600000-0x07a40000: ..
```

5. Boot up system

```
RedBoot> fis load kernel
RedBoot> exec -c "noinitrd console=ttymxc0 115200 root=/dev/mtdblock2 rw
ip=dhcp rootfstype=jffs2"
```

6.2.2 MMC/SD

The following steps are based on using ext3 file system, as EXT3 type file system is more suitable for MMC/SD.

6. Rebuild the kernel image with support for EXT3 MMC/SD boot. Look for, and mark the following options with an asterisk [*] on the LTIB configuration screens. You can access them with the `./ltib -c` command., then mark *"Configure the kernel"* with an [*], Exit and save the changes. The *"Linux Kernel Configuration"* screen should come up. After you set the four options below on the kernel, make sure `e2fsprogs` is selected on the "*Package list*". Exit all screens, and remember to copy the new kernel to the tftp directory of your host after the build completes.

```
Device Drivers  --->
          <*> MMC/SD card support  --->
                      <*>   Freescale i.MX Secure Digital Host Controller
Interface support

File systems  --->
                      <*> Ext3 journalling file system support
```

2. Initialize the MMC/SD flash

```
RedBoot> fis init
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
... Erase from 0x00040000-0x00060000:
... Program from 0x03ee0000-0x03f00000 at 0x00040000:.
```

3. Create redboot setting on first time boot

```
... Read from 0x03ee0000-0x03eff000 at 0x00040000: .
... Read from 0x03ee0000-0x03eff000 at 0x00040000: .
... Erase from 0x00040000-0x00060000:
... Program from 0x03ee0000-0x03f00000 at 0x00040000: .
```

4. Download the kernel image (zImage under /tftpboot) to RAM

```
RedBoot> load -r -b 0x100000 zImage -h <host IP address>
Using default protocol (TFTP)
Raw file loaded 0x00100000-0x002fd777, assumed entry at 0x00100000
```

5. Program the kernel image into MMC/SD

```
RedBoot> fis create -f 0x100000 kernel
... Read from 0x03ee0000-0x03eff000 at 0x00040000: .
... Read from 0x03ee0000-0x03eff000 at 0x00040000: .
An image named 'kernel' exists - continue (y/n)? y
... Erase from 0x00100000-0x00600000:
... Program from 0x00100000-0x002fd778 at 0x00100000: .
... Erase from 0x00040000-0x00060000:
... Program from 0x03ee0000-0x03f00000 at 0x00040000: .
```

6. Boot up the system through NFS

```
RedBoot> fis load kernel
... Read from 0x07ee0000-0x07eff000 at 0x00060000: .
... Read from 0x00100000-0x002d1254 at 0x00100000: .
RedBoot> exec -c "noinitrd console=ttymxc0,115200 root=/dev/nfs
nfsroot=<the IP address of the host machine>:<rootfs directory> rw
ip=dhcp"
```

7. Create the partitions on MMC/SD card

```
root@freescale /$ fdisk /dev/mmcblk0
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklab
```

```
el
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that the previous content
won't be recoverable.


The number of cylinders for this disk is set to 62752.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):p
Disk /dev/mmcblk0: 2056 MB, 2056257536 bytes
4 heads, 16 sectors/track, 62752 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

        Device Boot      Start         End      Blocks  Id System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-62497, default 1): 192
Last cylinder or +size or +sizeM or +sizeK (192-62497, default 62497):
Using def
ault value 62497

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

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

7. Generate MBR.bin and save it as backup. (This step is optional, the saved MBR.bin can be used for MBR recovery)

```
dd if=/dev/mmcblk0 of=/mbr.bin bs=512 count=1
```

8. Format the MMC/SD partitions as ext3 type

```
root@freescale ~$ mkfs.ext3 /dev/mmcblk0p1
mke2fs 1.41.4 (27-Jan-2009)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
125184 inodes, 499976 blocks
24998 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=515899392
16 block groups
32768 blocks per group, 32768 fragments per group
7824 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912
```

```
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
 ount -t ext2e /$ mkfs.009)typebin
generation●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
●●●●●●●●●●●●●●●●●●180 days, whichever comes first.  Use tune2fs -c or -i
to override
```

9. Copy the rootfs contents into MMC/SD card (copy the rootfs.ext2 to NFS rootfs)

```
root@freescale ~$ mount -t ext2 -o loop /rootfs.ext2 /mnt/cdrom
root@freescale ~$ cd /mnt
root@freescale /mnt$ mkdir mmcblk0p1
root@freescale /mnt$ mount -t ext3 /dev/mmcblk0p1 /mnt/mmcblk0p1
kjournald starting.  Commit interval 5 seconds
EXT3 FS on mmcblk0p1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
root@freescale /mnt$ cp -rf /mnt/cdrom/* /mnt/mmcblk0p1/
root@freescale /mnt$umount /mnt/mmcblk0p1
root@freescale /mnt$umount /mnt/cdrom
```

10. Boot up system from zImage on MMC/SD

```
fis load kernel
exec -c "noinitrd console=ttymxc0 root=/dev/mmcblk0p1 rootfstype=ext3 rw
ip=dhcp"
```

6.3 Download by u-boot

6.3.1 NAND

NOTE

Cannot download by this method if rootfs.jffs2 size is more than 64M since total SDRAM is 64M. Please use ATK to download images or boot from NFS and copy rootfs.jffs2 to nand.

1. Setup the tftp server, ipaddr, loadaddr environment by u-boot setenv command

```
MX25 U-Boot >setenv serverip xx.xx.xx.xx
MX25 U-Boot >setenv ipaddr xx.xx.xx.xx
MX25 U-Boot >setenv loadaddr 0x80100000
MX25 U-Boot >setenv eth1addr 00:04:9F:00:EA:CF
MX25 U-Boot >saveenv
```

2. Load the kernel into RAM from tftp server. Default prime Ethernet is smc911x-0, so make sure Ethernet cable is inseted to LAN9217 in debug board. Please refer to uboot user guider doc if you need to load kernel from FEC.

```
MX25 U-Boot > bootp 0x80100000 uImage.MX25
smc911x: detected LAN9217 controller
smc911x: phy initialized
smc911x: MAC 00:04:9f:00:95:52
```

```
BOOTP broadcast 1
*** Unhandled DHCP Option in OFFER/ACK: 44
*** Unhandled DHCP Option in OFFER/ACK: 46
*** Unhandled DHCP Option in OFFER/ACK: 44
*** Unhandled DHCP Option in OFFER/ACK: 46
DHCP client bound to address 10.193.102.151
Using smc911x-0 device
TFTP from server 10.193.100.212; our IP address is 10.193.102.151; sending
through gateway 10.193.102.254
Filename 'uImage.MX25'.
Load address: 0x80100000
Loading: #################################################################
         #################################################################
         ########
done
Bytes transferred = 2021408 (1ed820 hex)
```

3. Erase NAND block for uImage (Note: the erase size is block alignment and larger than uImage size)

```
MX25 U-Boot > nand erase 0x300000 0x1ee000

NAND erase: device 0 offset 0x300000, size 0x1ee000
Erasing at 0x0 -- 4718592% complete.
OK
```

4. Write the uImage into NAND flash

```
MX25 U-Boot > nand write 0x80100000 0x300000 0x1ee000
NAND write: device 0 offset 0x300000, size 0x1ee000
 2023424 bytes written: OK
```

5. Load rootfs.jffs2 into RAM

```
MX25 U-Boot > bootp 0x80100000 rootfs.jffs2
smc911x: detected LAN9217 controller
smc911x: phy initialized
smc911x: MAC 00:04:9f:00:95:52
BOOTP broadcast 1
*** Unhandled DHCP Option in OFFER/ACK: 44
*** Unhandled DHCP Option in OFFER/ACK: 46
*** Unhandled DHCP Option in OFFER/ACK: 44
*** Unhandled DHCP Option in OFFER/ACK: 46
DHCP client bound to address 10.193.102.151
Using smc911x-0 device
TFTP from server 10.193.100.212; our IP address is 10.193.102.151; sending
through gateway 10.193.102.254
Filename 'rootfs.jffs2'.
Load address: 0x80100000
Loading: #################################################################
         #################################################################
         #################################################################
         ########################################################
done
Bytes transferred = 3670016 (380000 hex)
```

6. Fill 0xFFFFFFFF data in the spare space if the rootfs is not page alignment

```
MX25 U-Boot > nw spare_start_addr 0xffffffff spare_size
Here spare_start_addr = 0x80100000 + 0x380000
Rootfs.jffs2 is block alignment, so spare_size = 0
```

7. Erase the NAND block for rootfs

```
MX25 U-Boot > nand erase 0x800000 0x380000

NAND erase: device 0 offset 0x800000, size 0x380000
Erasing at 0x0 -- 11534336% complete.
OK
```

8. Write the rootfs.jffs2 into NAND flash

```
MX25 U-Boot > nand write 0x80100000 0x800000 0x380000

NAND write: device 0 offset 0x800000, size 0x380000
 3670016 bytes written: OK
```

9. Set bootup command environment

```
MX25 U-Boot > setenv bootargs_nand 'setenv bootargs ${bootargs}
root=/dev/mtdblock2 ip=dhcp rootfstype=jffs2'
MX25 U-Boot > setenv bootcmd_nand 'run bootargs_base bootargs_nand; nand
read ${loadaddr} 0x300000 0x200000;bootm'
MX25 U-Boot > saveenv
```

10. bootup system

```
MX25 U-Boot > run bootcmd_nand
```

6.3.2 MMC/SD

1. Rebuild the image which supports EXT3 MMC/SD boot

```
File systems  --->
                      <*> Ext3 journalling file system support
```

2. Setup the tftp server, ipaddr, loadaddr environment by u-boot setenv command

```
MX25 U-Boot >setenv serverip xx.xx.xx.xx
MX25 U-Boot >setenv ipaddr xx.xx.xx.xx
MX25 U-Boot >setenv eth1addr  xx:xx:xx:xx:xx:xx
MX25 U-Boot >setenv loadaddr 0x80100000
MX25 U-Boot >saveenv
```

3. Load the kernel uImage into RAM. Default prime Ethernet is smc911x-0, so make sure Ethernet cable is inseted to LAN9217 in debug board. Please refer to uboot user guider doc if you need to load kernel from FEC.

```
MX25 U-Boot > bootp 0x80100000 uImage.MX25
smc911x: detected LAN9217 controller
smc911x: phy initialized
smc911x: MAC 00:04:9f:00:95:52
BOOTP broadcast 1
*** Unhandled DHCP Option in OFFER/ACK: 44
*** Unhandled DHCP Option in OFFER/ACK: 46
*** Unhandled DHCP Option in OFFER/ACK: 44
*** Unhandled DHCP Option in OFFER/ACK: 46
DHCP client bound to address 10.193.102.151
Using smc911x-0 device
TFTP from server 10.193.100.212; our IP address is 10.193.102.151; sending
through gateway 10.193.102.254
Filename 'uImage.MX25'.
Load address: 0x80100000
```

```
Loading: #################################################################
         #################################################################
         ########
done
Bytes transferred = 2021408 (1ed820 hex)
```

4. Write uImage into MMC/SD. The below command writes the image with the size 0x200000 from ${loadaddr} to the offset 0x100000 of the MMC/SD card. Here 0x800 =0x100000/512, 0x1000=0x200000/512. The block size of this card is 512.

```
MX25 U-Boot > mmc write 0 0x80100000 0x800 0x1000
MMC write: dev # 0, block # 2048, count 4096 ... 4096 blocks written: OK
```

5. Boot from NFS

```
MX25 U-Boot > setenv kernel uImage.MX25
MX25 U-Boot > setenv nfsrootfs /data/rootfs_home/rootfs_mx25
MX25 U-Boot > setenv bootcmd_net 'run bootargs_base bootargs_nfs; tftpboot
${loadaddr} ${kernel};bootm'
MX25 U-Boot > saveenv
MX25 U-Boot > run bootcmd_net
```

6. Create the partitions on MMC/SD card

```
root@freescale /$ fdisk /dev/mmcblk0
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklab
el
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that the previous content
won't be recoverable.


The number of cylinders for this disk is set to 62752.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/mmcblk0: 2056 MB, 2056257536 bytes
4 heads, 16 sectors/track, 62752 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

        Device Boot      Start         End      Blocks  Id System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-62752, default 1): 192
Last cylinder or +size or +sizeM or +sizeK (192-62752, default 62752):
Using def
ault value 62752

Command (m for help): p
```

```
Disk /dev/mmcblk0: 2056 MB, 2056257536 bytes
4 heads, 16 sectors/track, 62752 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

        Device Boot      Start         End      Blocks  Id System
/dev/mmcblk0p1             192       62752     2001952  83 Linux

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

Calling ioctl() to re-r mmcblk0:ead partition table p1
```

7. Format the MMC/SD partitions as ext3 type

```
root@freescale /$ mkfs.ext3 /dev/mmcblk0p1
mke2fs 1.41.4 (27-Jan-2009)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
125184 inodes, 500488 blocks
25024 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=515899392
16 block groups
32768 blocks per group, 32768 fragments per group
7824 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

8. Copy the rootfs contents into MMC/SD card (copy the rootfs.ext2 to NFS rootfs)

```
root@freescale ~$ mount -t ext2 -o loop /rootfs.ext2 /mnt/cdrom
root@freescale ~$ cd /mnt
root@freescale /mnt$ mkdir mmcblk0p1
root@freescale /mnt$ mount -t ext3 /dev/mmcblk0p1 /mnt/mmcblk0p1
kjournald starting.  Commit interval 5 seconds
EXT3 FS on mmcblk0p1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
root@freescale /mnt$ cp -rf /mnt/cdrom/* /mnt/mmcblk0p1/
root@freescale /mnt$umount /mnt/mmcblk0p1
root@freescale /mnt$umount /mnt/cdrom
```

9. Set bootup command environment

```
MX25 U-Boot > setenv bootargs_mmcsd 'setenv bootargs ${bootargs}
root=/dev/mmcblk0p1  ip=dhcp rootfstype=ext3'
MX25 U-Boot > setenv bootcmd_mmcsd 'run bootargs_base bootargs_mmcsd; mmc
read 0 ${loadaddr} 0x800 0x1000;bootm'
MX25 U-Boot > saveenv
```

10. Boot up system

```
MX25 U-Boot > run bootcmd_mmcsd
```

Chapter 7
Running the Image on the Target

This chapter explains how to run an image on the target from downloaded flash and NFS. These instructions assume that you have downloaded the kernel image and rootfs using the instructions in Chapter 5 or Chapter 6.

7.1 Run the Image from NAND Flash

To boot the kernel from NAND flash as following steps:

1. Set dip switch as chapter 3.2 and power on i.MX25 3-stack.
2. Run the following command in the RedBoot/u-boot prompt.

Redboot:

```
RedBoot> fis load kernel
RedBoot> exec -c "noinitrd console=ttymxc0,115200 root=/dev/mtdblock2 rw
rootfstype=jffs2 ip=dhcp"
```

Note that needs to create redboot setting on first time boot if ATK is used to download zImage:

```
fis create -f 0x300000 -l 0x500000 -n kernel -e 0x300000 -b 0x300000
```

U-Boot:

```
MX25 U-Boot > setenv bootargs nand 'setenv bootargs ${bootargs}
root=/dev/mtdblock2 ip=dhcp rootfstype=jffs2'
MX25 U-Boot > setenv bootcmd_nand 'run bootargs_base bootargs_nand; nand
read ${loadaddr} 0x300000 0x200000;bootm'
MX25 U-Boot > saveenv
MX25 U-Boot > run bootcmd_nand
```

7.2 Run the image from MMC/SD flash

To boot the kernel from MMC/SD flash as following steps:

1. Set dip switch as chapter 3.2 and power on i.MX25 3-stack.
2. Run the following command in the RedBoot/u-boot prompt.

Redboot:

```
RedBoot> fis load kernel
RedBoot> exec -c "noinitrd console=ttymxc0 root=/dev/mmcblk0p1
rootfstype=ext3 rw"
```

U-Boot:

```
MX25 U-Boot > setenv bootargs mmcsd 'setenv bootargs ${bootargs}
root=/dev/mmcblk0p1  ip=dhcp rootfstype=ext3'
MX25 U-Boot > setenv bootcmd_mmcsd 'run bootargs_base bootargs_mmcsd; mmc
read 0 ${loadaddr} 0x800 0x1000;bootm'
MX25 U-Boot > saveenv
MX25 U-Boot > run bootcmd_mmcsd
```

7.3 Run the image from NFS flash

Redboot:

```
load -r -b 0x100000 zImage
exec -c "noinitrd console=ttymxc0 root=/dev/nfsroot rootfstype=nfsroot
nfsroot=<the IP address of the host machine>:/tools/rootfs rw ip=dhcp"
```

NOTE

You can have the target run automatically by configuring RedBoot via "fconfig" commands. Set "true" when prompted with "Run script at boot". Enter the script that is to be auto executed after power on as indicated in the example below. The IP address of the tftp server may be entered at the "Default server IP address" prompt.

```
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> load r –b 0x100000 /tftpboot/zImage
>> exec –b 0x100000 –l 0x200000 -c "noinitrd console=ttymxc0
root=/dev/nfsroot rootfstype=nfsroot nfsroot=<the IP address of the host
machine>:/tools/rootfs rw ip=dhcp"
>>
Boot script timeout (1000ms resolution): 1
Use BOOTP for network configuration: true
Default server IP address: 10.192.221.167
Board specifics: 0
Console baud rate: 115200
Set eth0 network hardware address [MAC]: false
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Default network device: lan92xx_eth0
Update RedBoot non-volatile configuration - continue (y/n)? y
... Read from 0x07ee0000-0x07eff000 at 0xeff80000: .
... Erase from 0xeff80000-0xeffa0000: .
... Program from 0x07ee0000-0x07f00000 at 0xeff80000: .
RedBoot>
```

U-Boot:

```
=> setenv serverip 10.193.100.213
=> setenv kernel uImage.MX25
=> setenv nfsrootfs /data/rootfs_home/rootfs_MX25
=> setenv bootcmd_net 'run bootargs_base bootargs_nfs; bootp ${loadaddr}
${kernel}; bootm'
=> setenv bootcmd run bootcmd net
=> saveenv
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x100000 -- 100% complete.
Writing to Nand... done
=> run bootcmd_net
```

Chapter 8 Appendix

8.1 Using a Linux Host to set up an SD/MMC card

This chapter describes the steps to prepare an SD/MMC card to boot off an i.MX SoC.

Note that these instructions do not apply to i.MX31 and i.Mx23, as their ROM was designed with different expectations.

8.1.1 Requirements

An SD/MMC card reader, like a USB card reader, is required. It will be used to transfer the boot loader and kernel (zImage) images, to initialize the partition table and copy the root file system. To make the instructions easier, we make the assumption that an 4GB SD/MMC card is used.

Any Linux distribution can be used for the following steps. You might want to use a Linux distribution that LTIB has been tested against (like Fedora, Ubuntu, etc).

The Linux kernel running on the Linux host will assign a device node to the SD/MMC card reader. The kernel might decide the device node name or udev rules might be used. In the following instructions, it is assumed that udev is not used.

To identify the device node assigned to the SD/MMC card, please run:

```
$ cat /proc/partitions

major minor  #blocks  name

   8     0   78125000 sda

   8     1   75095811 sda1

   8     2          1 sda2

   8     5    3028221 sda5

   8    32  488386584 sdc

   8    33  488386552 sdc1

   8    16    3921920 sdb

   8    18    3905535 sdb1
```

In this case, the device node assigned is /dev/sdb (a block is 1kB large)

8.1.2 Copying the boot loader image

The RedBoot/U-Boot images can be found in the release package.

The following command will copy the boot loader image to the SD/MMC card:

```
$ sudo dd if= mx25_3stack_redboot_TO1_1.bin of=/dev/sdb bs=512  && sync &&
sync
```

Please note that this operation will delete the partition table present on the medium. Should you want to update redboot to another version, please run the following command instead:

```
$ sudo dd if= mx25_3stack_redboot_TO1_1-no-padding.bin of=/dev/sdb bs=512
seek=2 && sync && sync
```

The first 1kB, that includes the partition table, will be preserved.

8.1.3 Copying the kernel image (zImage)

The following command will copy the kernel image to the SD/MMC card

```
$ sudo dd if=zImage of=/dev/sdb bs=512 seek=2048 && sync && sync
```

This will copy the zImage to the medium at offset 1MB.

8.1.4 Copying the file system (rootfs)

A partition table must be first created. If one already exists and if the partition you want to use is big enough for the file system you want to deploy, then you can skip this step

The following command will create a partition, at offset 8192 (in sectors of 512 bytes)

```
$ sudo fdisk /dev/sdb
```

Type the following fdisk commands (each followed by <ENTER>):

```
u                      [switch the unit to sectors instead of cylinders]
d                      [repeat this until no partition is reported by the
'p' command ]
n                      [create a new partition]
p                      [create a primary partition]
1                      [the first partition]
8192                   [starting at offset sector #8192, i.e. 4MB, which
leaves enough space for the kernel, the boot loader and its configuration
data]
<enter>                [using the default value will create a partition
that spans to the last sector of the medium]
w                      [ this writes the partition table to the medium and
fdisk exits]
```

The next step is to format the partition. The file system format ext3 is a good candidate for removable medium, thanks to the built-in journaling. Run the following command to format the partition:

```
$ sudo mkfs.ext3 /dev/sdb1
```

The next step is to copy the target file system to the partition.

```
$ mkdir /home/user/mountpoint
$ sudo mount /dev/sdb1 /home/user/mountpoint
```

Let’s assume the root file system files are located in /home/user/rootfs

```
$ cd /home/user/rootfs
$ sudo cp –rpa [A-z]* /home/user/mountpoint
$ sudo umount /home/user/mountpoint
```

The file system contents is now on the medium,

8.1.5 Final configuration

Redboot needs to be configured and its partition table initialized. A kernel entry in the partition table must be created:

```
RedBoot> fis init
RedBoot> fis create -f 0x300000 -l 0x500000 -n kernel -e 0x300000 -b
0x300000
```