i.MX51 3-Stack Linux BSP

User's Guide

Document Number: 926-77208
Rev. 5.0.0
09/2009

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

About This Book

This document explains how to build and install the Freescale Linux BSP to the i.MX51 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.MX51 3DS 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.MX51 3DS board.

1.1 Boot Loader

The i.MX51 3DS Linux delivery package contains RedBoot bootloader binaries and u-boot bootloader binaries.

The `redboot_<version>.zip` file contains bootloader binaries for the i.MX family boards. The following table lists the binary files for i.MX51 EVK boards:

Binary Name	Description
mx51_TO2_3stack_redboot.bin	Redboot binary for i.MX51 3-Stack board which support MMC/SD and SPI nor boot.
mx51_TO2_3stack_redboot-no-padding.bin	Redboot binary for i.MX51 3-Stack 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_mx51.pdf`.

The default `imx51/u-boot-3ds.bin` in the release package supports NAND and MMC/SD boot for MX51 3-Stack board.

1.2 Linux Kernel image

This Freescale i.MX BSP contains the Freescale Linux 2.6.28 3-Stack kernel, driver source code, and a pre-built kernel image. You can obtain the i.MX51 3-Stack kernel image from the following location: `imx51/zImage and imx51/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.MX51 3-Stack BSP package contains the following default rootfs:

- `imx51/rootfs.ext2.gz`

rootfs.ext2.gz file system includes Freescale specific libraries and gnome GUI. It can be mounted as NFS or the source of the storage of rootfs.

- `imx51/rootfs.jffs2`

`rootfs.jffs2` file system includes Freescale specific libraries and gnome GUI. It's generated for Micron MT29F32G08QAA NAND MLC flash (erase block size: 512K, page size: 4k).

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 Linux Host

Setting up the Linux Host includes installing Linux OS and setting up TFTP/NFS server

2.1.1 Install Linux OS using Linux Builder

Install a Linux distribution such as Fedora 4/5, RedHat or ubuntu 9.04 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
```

4. Install the **tftp** server.
5. Install the **nfs** server.
6. 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
```

7. 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
}
```

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

```
vi /etc/exports
```

add this line in the file: `/tools/rootfs *(rw,sync,no_root_squash)`

9. 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".

Meanwhile, to run LTIB, some host packages are needed. If any error which is related to host package is raised, please install host package firstly.

1. Install the LTIB package not as root:

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

This command installs LTIB to your directory.

10. Build LTIB:

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

Select platform to "Freescale iMX reference boards" and exit, saving the changes. At the next menu, select platform type as "imx51" and package profile. Exit and save

changes. Please note only “Min profile” and “FSL gnome release packages” are tested by default.

In next screen, the following menu option is shown:

```
|   | --- Choose your bootloader                                  |   |
  |   |     bootloader (u-boot)  --->                           |   |
  |   | --- Choose your board                                   |   |
  |   |     board (mx51_3stack)  --->
```

The menu option “Choose your board” is only for uboot bootloader building. It builds different bootloaders for i.MX51 3DS and EVK with different machine ID.

Then run the following command:

```
./ltib
```

When complete, you can obtain the kernel image from `rootfs/boot/zImage`.

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
cp imx51/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
```

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 imx51/zImage /tftpboot
```

or

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

Chapter 3 Board Dip switch setup

The boot modes of i.MX51 3-Stack board are controlled by the dip switches on the CPU board and debug boards.

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

Table 3.1-2 CPU board switch setup for ATK downloading

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

Ensure J28 in the personality board is open when using ATK.

3.2 Dip switch setup for boot modes

Table 3.2-1 Debug board SW5-SW10 switch setup for boot

Switch	SW5	SW6	SW7	SW8	SW9	SW10
Internal Boot	OFF	OFF	OFF	OFF	OFF	OFF

Table 3.2-2 CPU board switch setup for NAND boot

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

Table 3.2-3 CPU board switch setup for MMC/SD boot

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

Chapter 4 Flash memory map

This chapter introduces the software layout in MMC/SD cards. It can be useful to understand the later sections about image download.

4.1 MMC/SD flash memory map

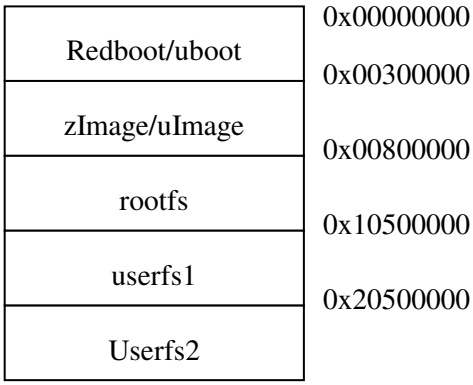
MMC/SD flash scheme is something different from NAND and NOR flash which are deployed in the BSP software. MMC/SD flash must keep the first sector (512 bytes) as MBR (Master Boot Record) in order to use MMC/SD as the rootfs. At boot up, MBR is executed to look up the partition table to determine which 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.

MBR can be generated through “fdisk” command when creating partitions in MMC/SD cards in the Linux Host server.

4.2 NAND flash memory map

NAND flash scheme is configured statically by software. It can be adjusted according to different requirements. Figure 4.2-1 illustrates default NAND flash map on i.MX51 3-3satck platform.

Figure 4.2-1 NAND flash memory scheme



Chapter 5 Downloading images using ATK

This chapter explains how to install the ATK tool, download Redboot, u-boot to the MMC/SD card.

5.1 Installing the ATK Tools

Download the ATK tool Version 1.68 install package from the Freescale release web site or from Freescale support. Then follow ATK user guide to install new packages.

5.2 MMC/SD

5.2.1 Download RedBoot to MMC/SD

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

1. Set the dips as serial download mode by referring to Section Dip switch setup for ATK downloading
2. Power on the board. Configure the ATK Options as Figure 5.2-1. Select “Flash Tool”.
3. Program Redboot by selecting options as Figure 5.2-2 if no MBR on the MMC/SD. If MBR is created in the card already which means the rootfs has been programmed to MMC/SD, program no-padding redboot binary with the offset 0x400 by selecting options as Figure 5.2-3. Please double check this point before upgrading bootloader. Otherwise, the rootfs has to be re-programmed sometimes if MBR is broken.
4. Click **Program** to download.

Figure 5.2-1 i.MX51 ATK configuration

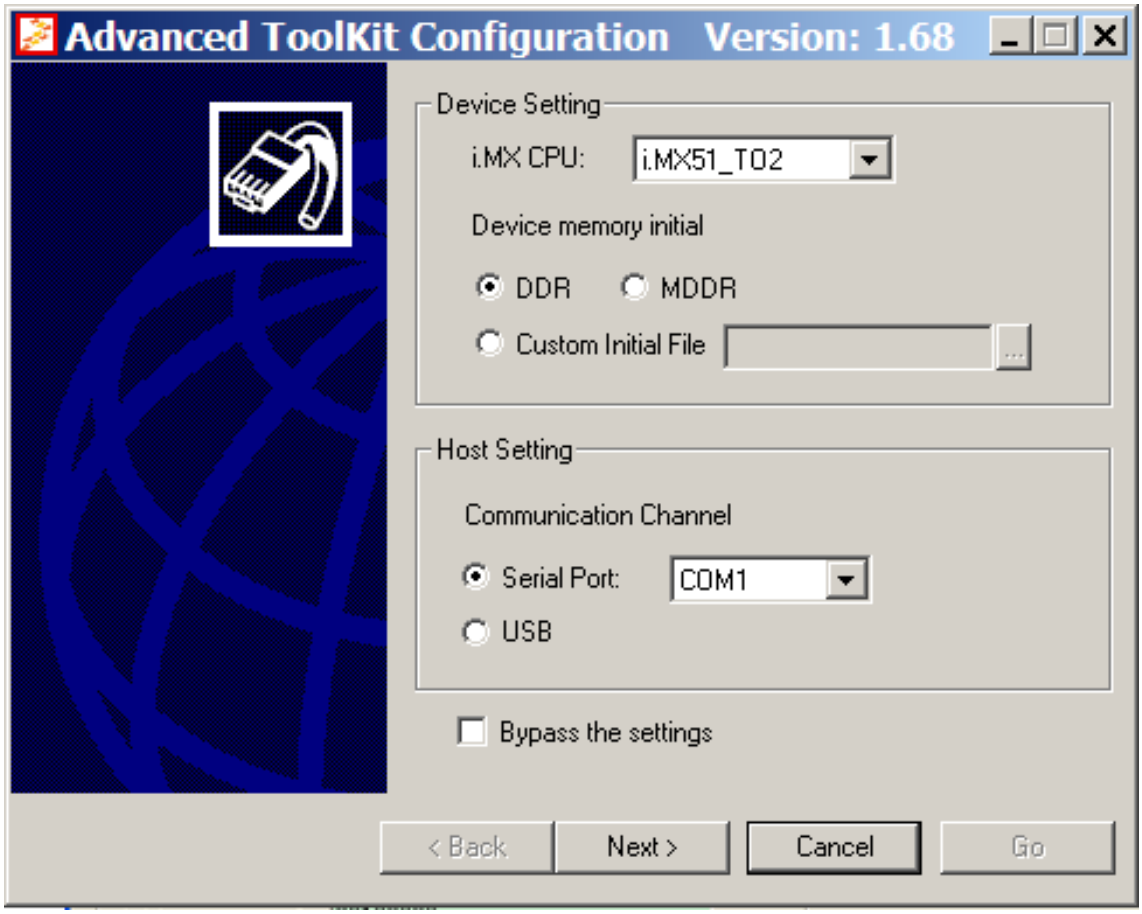


Figure 5.2-2 i.MX51 Programming Redboot Image to MMC/SD Flash

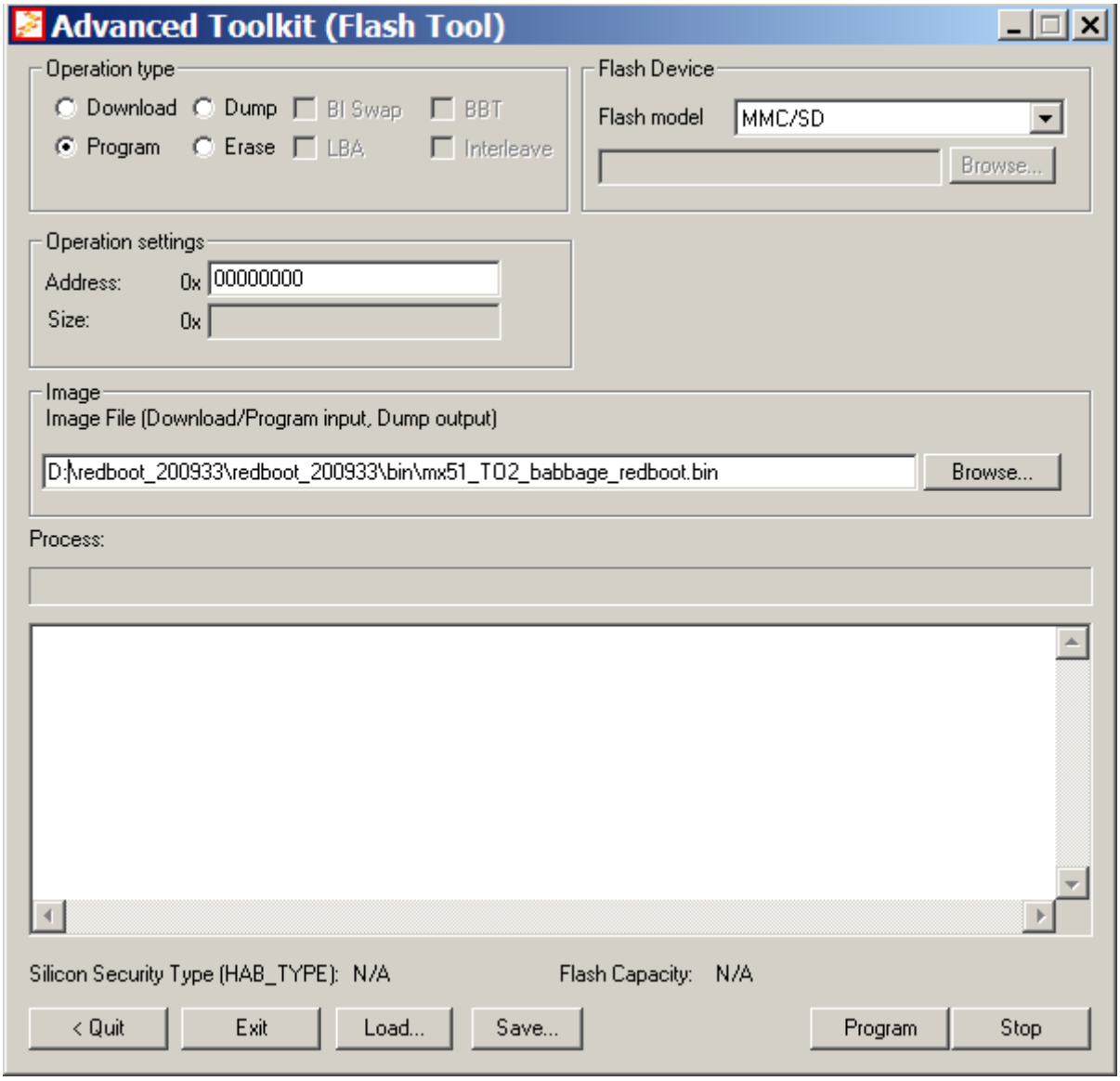
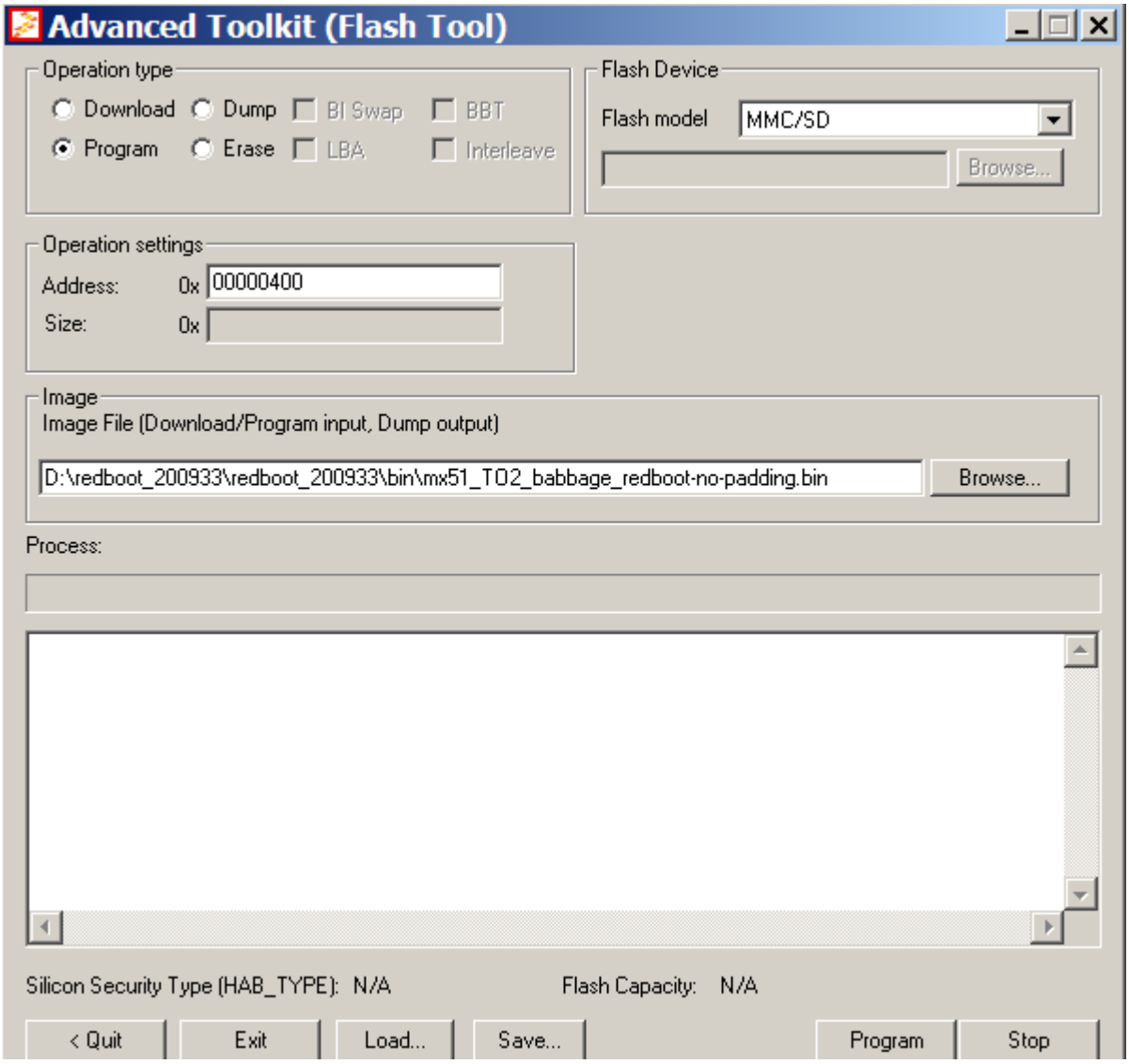


Figure 5.2-3 i.MX51 Programming no-padding Redboot to MMC/SD Flash



5.2.2 Download uBoot to MMC/SD

To download the uboot bootloader to MMC/SD card, See Section Download RedBoot to MMC/SD. Change download image file name and set the address as “0x0”.

5.3 NAND flash

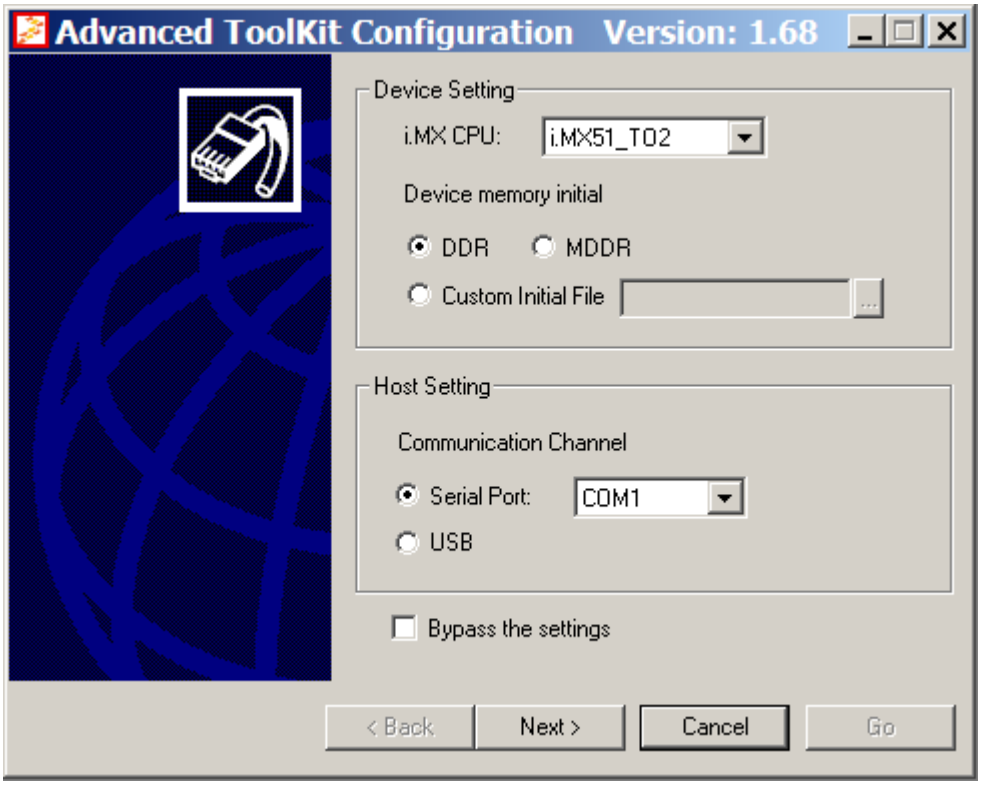
5.3.1 Erasing the NAND Flash

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:

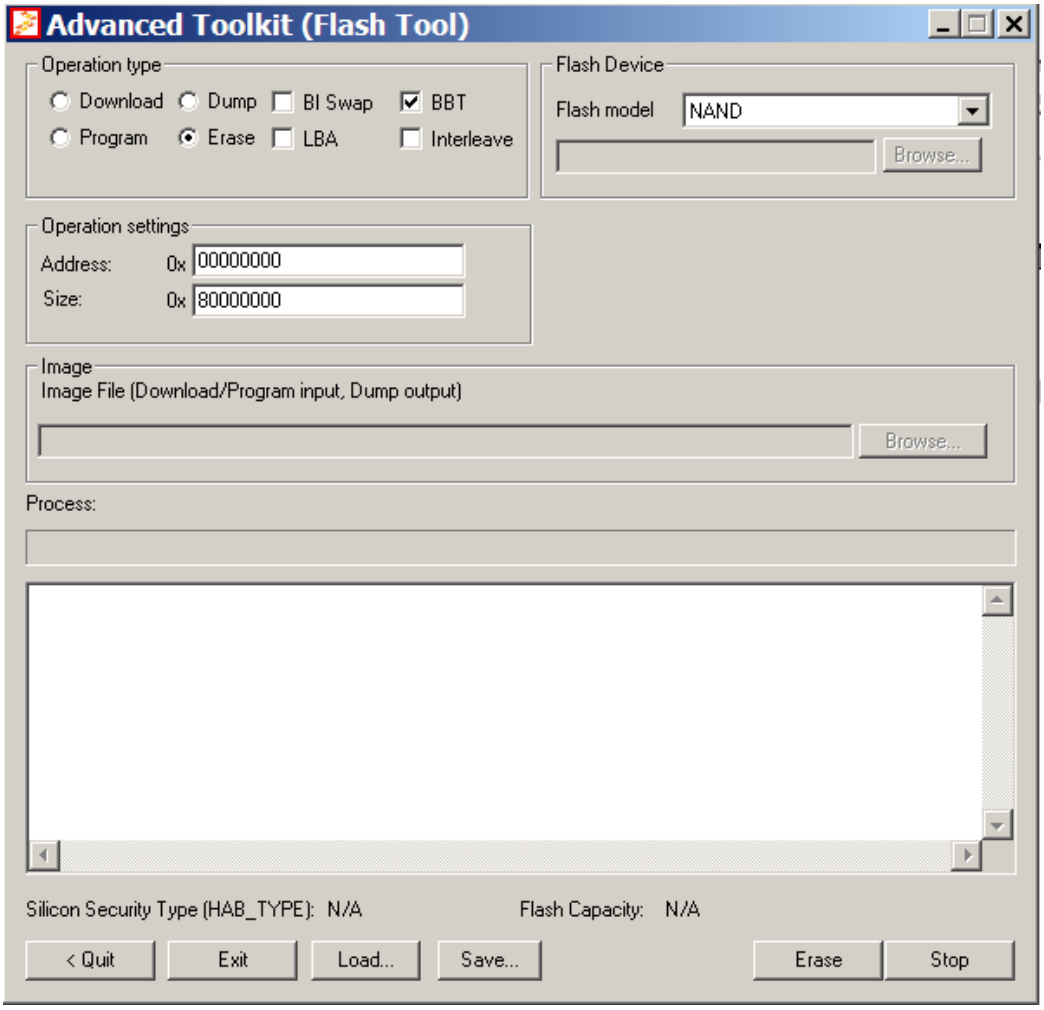
1. Set the dips as serial download mode by referring to Section Dip switch setup for ATK downloading
2. Power on i.MX51 3stack board and Run the ATK, and then select the options for your platform, for **Device memory initial** (**DDR** or **MDDR**). Check which DDR is used on your board firstly, then select "DDR" for DDR2 CPU board. Select "MDDR" for the MDDR board. Figure 5.3-1 illustrates ATK configuration for i.MX51 TO2 3stack.

Figure 5.3-1 i.MX51 TO2 ATK Configuration



3. Click **Next**.
4. Select Flash Tool, and then click **Go**.
5. Select the options for your platform, as follows:

Figure 5.3-2 i.MX51 Erasing NAND Flash



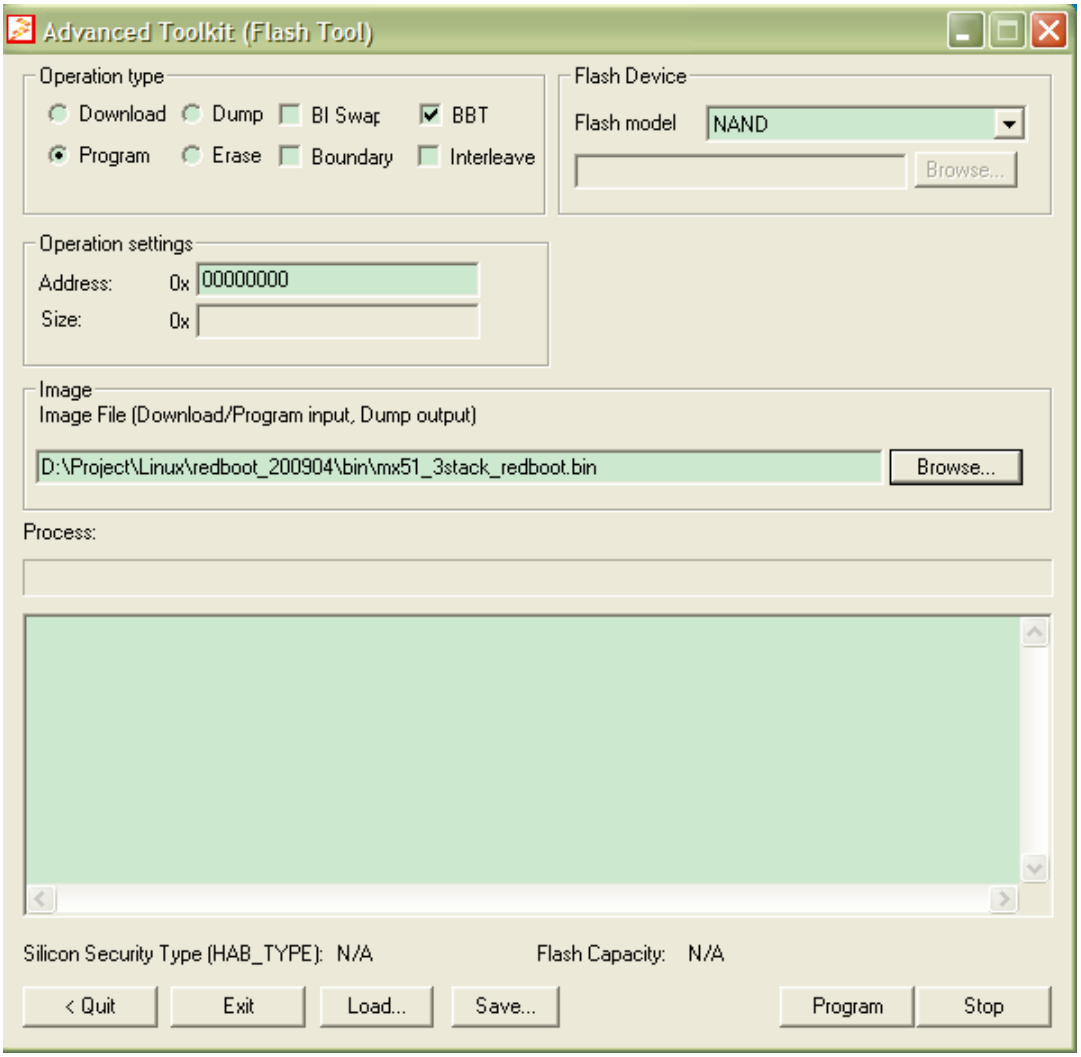
6. Click **Erase** to erase the flash.

5.3.2 Download RedBoot/uBoot Bootloader

To download the RedBoot/uBoot bootloader, use these steps:

1. Execute the steps 1-4 described in section 5.3.1
2. Program RedBoot by selecting the options for your platform, as follows:

Figure 5.3-3 i.MX51 Programming RedBoot to NAND Flash



3. Click **Browse** to select the bootloader.
4. Click **Program** to flash.

Chapter 6 Download images by bootloader or NFS

6.1 Setup Terminal

The i.MX51 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:

1. Connect the target and the Windows PC via a serial cable.
2. Open HyperTerminal on the Windows PC, and select the settings shown in Figure 6.2-1.

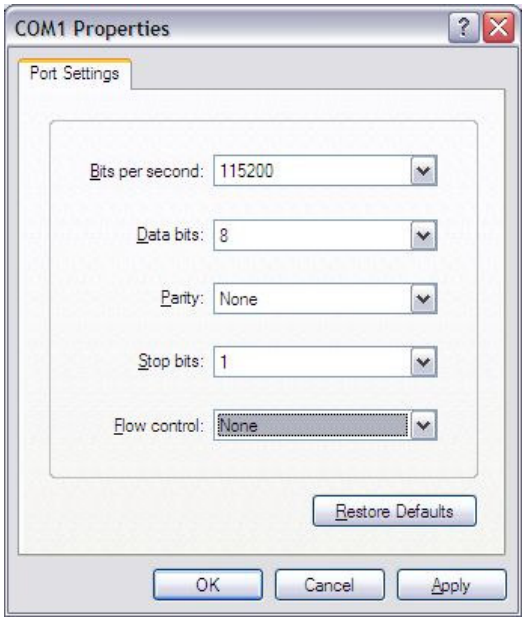


Figure 6.2-1 HyperTerminal Settings

3. Power up the target by setting up the right dips. Wait until the system enters bootloader prompt.

6.2 Download by Redboot

6.2.1 NAND flash

1. Power on the board from NAND by following the dip setting in Section Dip switch setup for boot modes.

2. Initialize and erase flash

```
RedBoot>  fis init
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
... Erase from 0x00100000-0x00180000:
Erase 0x00100000: .
... Program from 0x07e80000-0x07f00000 at 0x00100000:
```

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

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

4. Program the zImage into NAND flash:

```
RedBoot> fis create -f 0x300000 kernel
... Read from 0x07e80000-0x07eff000 at 0x00100000: ..
... Read from 0x07e80000-0x07eff000 at 0x00100000: ..
... Read from 0x07e80000-0x07eff000 at 0x00100000: ..
... Erase from 0x00300000-0x00500000:
Erase 0x00300000: ....
... Program from 0x00100000-0x002d4150 at 0x00300000: .....
... Erase from 0x00100000-0x00180000:
Erase 0x00100000: .
... Program from 0x07e80000-0x07f00000 at 0x00100000: ..
```

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

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

6. Program the rootfs into NAND flash:

```
RedBoot> fis create -f 0x800000 root
... Read from 0x07e80000-0x07eff000 at 0x00100000: ..
... Read from 0x07e80000-0x07eff000 at 0x00100000: ..
... Read from 0x07e80000-0x07eff000 at 0x00100000: ..
... Erase from 0x00800000-0x08100000:
Erase 0x00800000: ..................................
Erase 0x01800000: ..................................
Erase 0x02800000: ..................................
Erase 0x03800000: ..................................
Erase 0x04800000: ..................................
Erase 0x05800000: ..................................
Erase 0x06800000: ..................................
Erase 0x07800000: ..................
... Program from 0x00100000-0x07a00000 at 0x00800000: ..........
................................................................................
................................................................................
.............................................................
... Erase from 0x00100000-0x00180000:
Erase 0x00100000: .
```

```
... Program from 0x07e80000-0x07f00000 at 0x00100000: ..
```

7. 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

- Power on the board from MMC/SD by following the dip settings in section Dip switch setup for boot modes.
- 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 0x00060000-0x00080000: .
... Program from 0x07ee0000-0x07f00000 at 0x00060000: .
```

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

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

- Program the kernel image into MMC/SD

```
RedBoot> fis create -f 0x100000 kernel
... Read from 0x07ee0000-0x07eff000 at 0x00060000: .
... Read from 0x07ee0000-0x07eff000 at 0x00060000: .
... Read from 0x07ee0000-0x07eff000 at 0x00060000: .
... Erase from 0x00100000-0x002e0000: .............
... Program from 0x00100000-0x002d1254 at 0x00100000: .
... Erase from 0x00060000-0x00080000: .
... Program from 0x07ee0000-0x07f00000 at 0x00060000: .
```

- 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"
```

- To program the rootfs to MMC/SD, See section "Use i.MX51 as Host server to create rootfs" or section "Using a Linux Host to set up an SD/MMC card".

6.2.3 Redboot configurations

Redboot "fconfig" command can be used to configure boot up scripts, tftp server address, fec mac address and so on. For example,

```
RedBoot> fconfig
Run script at boot: true
Boot script:
...
Enter script, terminate with empty line
>> fis load kernel
```

```
>> exec -c "noinitrd console=ttymxc0 root=/dev/nfs nfsroot=10.193.100.213:/data/rootfs_home/rootfs_mx51 rw
ip=dhcp"
>>
Boot script timeout (1000ms resolution): 1
Use BOOTP for network configuration: true
Default server IP address: 10.193.100.213
Board specifics: 0
Console baud rate: 115200
Set FEC network hardware address [MAC]: true
FEC network hardware address [MAC]: 0x00:0x04:0x9F:0x00:0xEA:0xC8
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Update RedBoot non-volatile configuration - continue (y/n)? y
... Read from 0x1fef0000-0x1feff000 at 0x00084000: ..
... Erase from 0x00080000-0x00094000:
... Program from 0x1fef0000-0x1ff00000 at 0x00084000: ....
```

6.3 Download by u-boot

6.3.1 NAND flash

- Power on the board from NAND by following the dip setting in Section Dip switch setup for boot modes. Then setup environment variables for network. For example,

```
BBG U-Boot > setenv serverip 10.193.100.213
BBG U-Boot > setenv ethaddr  00:04:9F:00:EA:CF
BBG U-Boot > setenv ipaddr 10.193.102.93
BBG U-Boot > saveenv
```

- Check environment variables. Ensure the right values are set.

```
BBG U-Boot > print
....
```

- Load the kernel into RAM from tftp server

```
MX51 U-Boot > setenv kernel uImage.mx51.r351
MX51 U-Boot > bootp ${loadaddr} ${kernel}
smc911x: initializing
smc911x: detected LAN9217 controller
smc911x: phy initialized
smc911x: MAC 00:04:9f:00:ea:cf
BOOTP broadcast 1
BOOTP broadcast 2
DHCP client bound to address 10.193.102.94
TFTP from server 10.193.100.213; our IP address is 10.193.102.94
Filename 'uImage.mx51.r351'.
Load address: 0x90800000
Loading: ## Warning: gatewayip needed but not set
## Warning: gatewayip needed but not set
#################################################################
      ###########################################################
done
Bytes transferred = 1803160 (1b8398 hex)
```

- Erase old kernel image and upgrade to new kernel image. Please note you must make erase/program size as page alignment. For example, the new kernel image size is 0x1d296c, you should make erase/program size as 0x1d4000. One simple method is to set reserved kernel image size as 0x200000. So you don't need to care for the actual size of the image. But the disadvantage is that much time will be spent to load dirty data. Here 0x300000 is the start address of kernel in NAND.

```
MX51 U-Boot > setenv kernel_size 0x200000
MX51 U-Boot > setenv kernel_start 0x300000
MX51 U-Boot > nand erase $(kernel_start) $(kernel_size)

NAND erase: device 0 offset 0x300000, size 0x200000
Erasing at 0x480000 -- 100% complete.
OK
MX51 U-Boot > nand write ${loadaddr} $(kernel_start) $(kernel_size)

NAND write: device 0 offset 0x300000, size 0x200000
 2097152 bytes written: OK
```

- Due to ENGR00116785 (Uboot: Timeout when downloading a big size file via tftp. Possibility: 100%), recommend to use NFS to flash rootfs now. Boot up system from NFS or other storage as the rootfs. For example,

```
MX51 U-Boot > setenv nfsroot /data/rootfs_home/rootfs_mx51
MX51 U-Boot > saveenv
MX51 U-Boot > run bootcmd_net
```

- Program rootfs to NAND via NFS:

```
root@freescale ~$ flash_eraseall /dev/mtd2
Erasing 512 Kibyte @ 10000000 -- 100% complete.
root@freescale ~$ nandwrite -p /dev/mtd2 /rootfs.jffs2
```

6.3.2 MMC/SD

- Power on the board from MMC/SD by following the dip setting in Section Dip switch setup for boot modes. Then execute:

```
MX51 U-Boot > setenv serverip 10.193.100.158
MX51 U-Boot > setenv ethaddr  00:04:9F:00:EA:CF
MX51 U-Boot > setenv ipaddr 10.193.102.93
MX51 U-Boot > saveenv
```

- Copy uImage to tftp server. Then download it to RAM:

```
MX51 U-Boot > tftpboot ${loadaddr} uImage
```

- Due to uboot limitation, the below steps can only be applied to the cards whose size less than 4G. The further uboot version will remove this limitation and support >=4G size cards.

- Program the kernel uImage into MMC/SD

```
MX51 U-Boot > mmcinit
SD card.
Vendor: Man 03 OEM SD "SD04G" Date 03/2008
Product: 1080569975
```

```
Revision: 8.0
BBG U-Boot > cp.b ${loadaddr} 0x100000 0x200000
Copy to MMC...
```

- Boot up the system through NFS

```
MX51 U-Boot > setenv bootcmd_mmc 'run bootargs_base bootargs_nfs; mmcinit;cp.b 0x100000 $
{loadaddr} 0x200000;bootm'
BBG U-Boot > setenv bootcmd 'bootcmd_mmc'
```

- To program the rootfs to MMC/SD, See section "Use i.MX51 as Host server to create rootfs" or section "Using a Linux Host to set up an SD/MMC card".

6.3.3 uboot configurations

uboot "print" command can be used to check environment variable values. "setenv" command is to set environment variable value. See uboot User Guide for the details.

6.4 Use i.MX51 as Host server to create rootfs to MMC/SD

Linux provides multiple methods to program images to the storage. For example, dd command in linux server (See Appendix). The following section demonstrates how to use MX51 as Linux Host server to program images to MMC/SD

- Boot from NFS or the other storage. Then use i.MX51 as Linux Host server to create the rootfs on MMC/SD card. Here is the example to create partition through MMC/SD Slot 0. In linux console, type "fdisk":

```
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 124368.
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: 4075 MB, 4075290624 bytes
4 heads, 16 sectors/track, 124368 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

      Device Boot      Start         End      Blocks  Id System
```

- Create a new partition by typing “n”. As mentioned, the rootfs partition should be located after kernel image. So the first 0x300000 bytes can be reserved for MBR, bootloader and kernel sections. From above log, the “Units” of current MMC/SD card is 32768 bytes. The begin cylinder of the first partition can be set as “0x300000/32768 = 96”. The last cylinder can be set according to the rootfs size.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-124368, default 1): 96
Last cylinder or +size or +sizeM or +sizeK (96-124368, default 124368): Using de
fault value 124368

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

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

- Format the MMC/SD partitions as ext3 type or ext2 type. Here takes ext3 as the example.

```
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)
248992 inodes, 994184 blocks
49709 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1019215872
31 block groups
32768 blocks per group, 32768 fragments per group
8032 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

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

```
mount -t ext2 -o loop /rootfs.ext2 /mnt/cdrom
cd /mnt
mkdir mmcblk0p1
mount -t ext3 /dev/mmcblk0p1 /mnt/mmcblk0p1/
cp -rf /mnt/cdrom/* .
umount /mnt/mmcblk0p1
umount /mnt/cdrom
```

- Type “sync” to write the contents to MMC/SD.
- Type “poweroff” to power down the system. Then follow the below section to boot the whole images from MMC/SD card.

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 using the instructions in Chapter 5 or Chapter 6.

7.1 Run the image from NFS

To boot from NFS, do as the follows:

Redboot:

```
fis load kernel
exec –c “noinitrd console=ttymxc0 root=/dev/nfsroot rootfstype=nfsroot nfsroot=<the IP address of the host machine>:/tools/rootfs rw ip=dhcp”
```

U-Boot:

```
BBG U-Boot > setenv nfsroot /data/rootfs_home/rootfs_mx51
BBG U-Boot > run bootcmd_net
```

7.2 Run the image from NAND

Redboot:

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

U-Boot:

```
MX51 U-Boot > setenv bootargs_nand 'setenv bootargs ${bootargs} root=/dev/mtdblo
ck2 ip=dhcp rootfstype=jffs2'
MX51 U-Boot > setenv bootcmd_nand 'run bootargs_base bootargs_nand;nand read ${l
oadaddr} 0x300000 0x200000;bootm'
MX51 U-Boot > setenv bootcmd 'run bootcmd_nand'
MX51 U-Boot > saveenv
Saving Environment to NAND...
Erasing Nand...
Warning: Erase size 0x00020000 smaller than one erase block 0x00080000
        Erasing 0x00080000 instead
Erasing at 0x100000 -- 100% complete.
Writing to Nand... done
```

7.3 Run the image from MMC/SD flash

To boot the whole system from MMC/SD flash by following the bellow steps:

1. Set dip switch as section Dip switch setup for boot modes. Then power on the board.
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:

```
MX51 U-Boot > setenv bootargs_mmcsd 'setenv bootargs ${bootargs} root=/dev/mmcblk0p1  ip=dhcp
rootfstype=ext3'
MX51 U-Boot > setenv bootcmd_mmcsd 'run bootargs_base bootargs_mmcsd; mmcinit;cp.b 0x100000 ${
loadaddr} 0x200000;bootm'
MX51 U-Boot > saveenv
MX51 U-Boot > run bootcmd_mmcsd
```

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.

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 Redboot image to the SD/MMC card:

```
$ sudo dd if=mx51_TO2_babbage_redboot.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=mx51_TO2_babbage_redboot-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 -n -b 0x100000 -f 0x100000 -l 0x300000 kernel
```