i.MX28 EVK Linux
User's Guide

Document Number: MX28EVKLUG
Rev. L2.6.35_1.1.0
02/2013

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
www.freescale.com/support

About This Book

This document explains how to build and install the Freescale Linux board support package (BSP) to the i.MX28 EVK board.

Audience

This document is intended for software, hardware, and system engineers who are planning to use the product. This document is also useful for those who want to know more about the product.

Organization

This document contains the following chapters:

Chapter 1
Introduction

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

1.1 Flash Bootloader

When the i.MX28 is reset, it executes the Read-Only Memory (ROM). There is no alternative—no other code is permitted to handle the reset exception. The ROM reads the boot mode pins to detect the boot source (USB, SD/MMC, NAND Flash, and so on) and negotiates with that source in a device-dependent way to retrieve a "boot stream." A boot stream is an executable collection of bytes in the Safe Boot (SB) format.

1.2 Boot Stream

The boot stream is an important concept for i.MX28.

The Linux release provides two boot stream images as follows:

- Linux kernel boot stream
- U-Boot boot stream

See Chapter 4 for details about creating these boot stream images.

NOTE

i.MX28 supports U-Boot but does not support RedBoot.

A boot stream contains instructions that cause it to function as an extended bootloader for the ROM. Such a boot stream starts with a `Load` command that instructs the ROM to copy the executable into memory. A final `Jump` command instructs the ROM to transfer control to the executable that is loaded, which in this case can be kernel boot stream or U-Boot boot stream.

Another important command is `Call`. This command instructs the ROM to make a function call to a given address and continue processing the boot stream when the control returns. A `Call` command is usually preceded by a `Load` command that copies into memory the function to be called. Collectively, the `Load` command, the associated executable, and the `Call` command are referred to as a bootlet.

Figure 1-1 shows a boot stream that contains two bootlets, followed by the main executable.

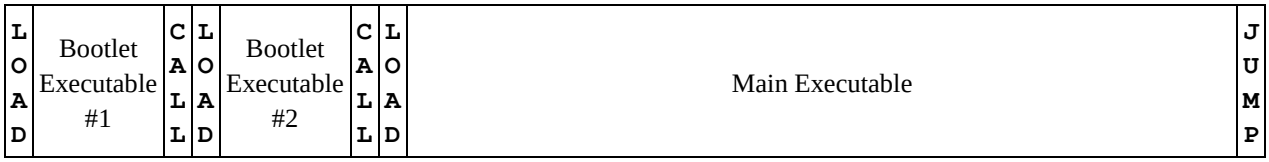

Figure 1-1. i.MX28 Boot Stream Outline

Each bootlet is an executable that is built separately, for a specific purpose, and may or may not know anything about the bootlets that precede or follow it.

The boot stream can instruct the ROM to call any number of executables before the final `Jump` command, depending on the system needs. The i.MX28 Linux BSP boot streams contain the following bootlets:

- power_prep— This bootlet configures the power supply.
- boot_prep— This bootlet configures the clocks and SDRAM.
- linux_prep— This bootlet prepares to boot Linux.

Figure 1-2 shows a kernel boot stream constructed with the i.MX28 Linux BSP.

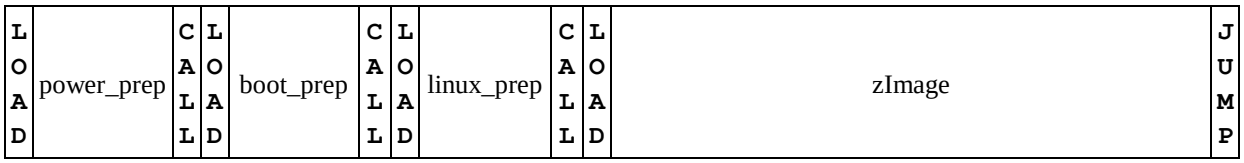

Figure 1-2. Example of i.MX28 Boot Stream Loading Linux Kernel

Figure 1-3 shows a U-Boot boot stream.

Figure 1-3. Example of i.MX28 Boot Stream Loading U-Boot

See Chapter 4 for details about creating a boot stream image.

1.3 Linux Kernel and Driver

The Freescale BSP contains the Freescale Linux 2.6.35 kernel, driver source code, and prebuilt kernel images. The following prebuilt kernel images are available under the imx28 directory:

- zImage (kernel zImage)

- uImage (kernel uImage)
- imx28_linux.sb (kernel image combined with HAB-disabled boot stream)
- imx28_ivt_linux.sb (kernel image combined with HAB-enabled boot stream)
- imx28_uboot.sb (U-Boot image combined with HAB-disabled boot stream)
- imx28_ivt_uboot.sb (U-Boot image combined with HAB-enabled boot stream)

NOTE

If the HAB_DISABLE bit, HW_OCOTP_ROM7:0x8002C210:bit11, is 1, then use boot streams without the name “ivt.”

If the HAB_DISABLE bit, HW_OCOTP_ROM7:0x8002C210:bit11, is 0, then use boot streams with the name “ivt.”

1.4 Root File System

The root file system package provides busybox, common libraries, and other fundamental elements. The Linux BSP contains the following root file system images under the imx28 directory:

- rootfs.jffs2 (root file system JFFS2 image)
- rootfs.ext2.gz (root file system EXT2 image)

Chapter 2
Building the Linux Platform

This chapter explains how to set up the build environment, install and build the Linux Target Image Builder (LTIB), set up rootfs for the Network file system (NFS), and set up the host environment.

2.1 Setting Up the Build Environment

Setting up the build environment includes installing Linux operating system (OS) and LTIB.

2.1.1 Install Linux OS using Linux Builder

Install a Linux distribution, such as Fedora 4/5, RedHat or Ubuntu 9.04, 9.10, or 10.04 on a system.

2.2 Installing and Building LTIB

To install and build LTIB, perform the following steps:

NOTE

In some Linux systems, the following procedure must be performed 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 selected directory.

2. Build LTIB:

   ```
   cd <your LTIB directory>

   ./ltib
   ```

3. Select the platform as “Freescale iMX reference boards” and exit after saving the changes. In the next menu, select “mx28” as platform type and select a package profile. Profiles min, test, and fsl gnome have been tested. Exit after saving changes.

4. Then run the following command:

```
./ltib
```

When complete, LTIB will have produced the following in subdirectories of ./ltib:

- The kernel images: rootfs/boot/uImage and rootfs/boot/zImage
- The SB files of bootlets and kernel images: rootfs/boot/imx28_linux.sb, rootfs/boot/imx28_ivt_linux.sb
- The SB files of bootlets and U-Boot images: rootfs/boot/imx28_uboot.sb, rootfs/boot/imx28_ivt_linux.sb
- The JFFS2 rootfs image: rootfs.jffs2

NOTE

If you want an EXT2 file system, execute `./ltib -c`, and change the option under the LTIB “Target Image Generation” menu from “JFFS2” to “EXT2.” After rebuilding, the EXT2 rootfs image can be found in `rootfs.ext2.gz`.

2.3 Choosing rootfs over NFS

2.3.1 Setting Network Parameters

The network parameters must be set up in LTIB to boot using the NFS. To set these parameters, execute:

```
./ltib -c
```

Set the network parameters in the following LTIB path:

Package List→

```
noinitrd console=ttyAM0,115200 fec_mac=00:08:02:6B:A3:1A root=/dev/nfs
nfsroot=10.193.100.213:/data/rootfs_home/rootfs_mx28 rw ip=dhcp rootwait gpmi
```

There are two places in the BSP to get the rootfs for NFS:

- Use the EXT2 format rootfs package already provided in the distribution; or
- Use the rootfs created after building the kernel

2.3.2 Using rootfs in the Distribution

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

```
mkdir /mnt/rootfs

mkdir /tools

cp imx28/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
```

2.3.3 Using the rootfs created after Building the Kernel

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

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

2.4 Setting Up the Linux Host

To set up the Linux host system, perform the following steps:

1. Turn off the firewall to enable the TFTP to work:

   ```
   iptables -F
   ```

   or, type the following command on the command line:

   ```
   setup
   ```

2. Install the TFTP server.
3. Install the NFS server.
4. Create the `tftboot` directory.

   The kernel images (such as the zImage kernel image) and other files that need to be uploaded by TFTP are stored in the `tftboot` 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 the Linux host machine:

   ```
   vi /etc/exports
   ```

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

8. Restart the NFS and TFTP servers on the host:

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

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

9. Copy `zImage` and `rootfs.jffs2` in the release package or LTIB to the `tftp` directory.

   ```
   cp imx28/zImage /tftpboot

   cp imx28/rootfs.jffs2 /tftpboot
   ```

   or,

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

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

NOTE

These instructions specify using an NFS server. Some Linux systems use `nfsserver` instead of `nfs`. These instructions are applicable to both server types.

NOTE

A Windows TFTP program “tftp.zip” is available in the LTIB release package `Common/` folder. This TFTP program can be

installed in the Windows OS to provide a Windows TFTP server for downloading images.

2.5 Building the Manufacturing Firmware

See Section 2.2, "Installing and Building LTIB," to set up the LTIB environment.

1. Configure the firmware build profile as follows:

   ```
   ./ltib --selectype
   ```

2. Choose the following:

   --- Choose the platform type

   Selection (**imx28**) --->

   --- Choose the packages profile

   Selection (**mfg firmware profile**) --->

After LTIB completes the build, updater.sb and updater_ivt.sb will be created.

2.6 Building Kernel to Support Suspend-To-RAM Mode

In suspend-to-RAM mode, only DRAM is in self-refresh mode and is powered by external circuit; ARM core, on-chip modules, and PLL are off. After entering suspend-to-RAM mode, only power key can resume, then system starts a code boot. To enter suspend-to-RAM, run the following command:

```
echo mem > /sys/power/state
```

2.6.1 Hardware Rework

Hardware rework is needed to test suspend-to-RAM mode with MX28 EVK board.

1. Remove R31.
2. Connect an external 1.8V supply to the DDR2. This can be done by connecting the supply to the pad on R31 near the DDR2.

2.6.2 Enable Suspend-To-RAM

Suspend-to-RAM is disabled by default and can be enabled in kernel configure. To run the suspend-to-RAM, Linux kernel boot stream must be used.

System Type ---> Freescale i.MXS implementations ---> support MX28 suspend-to RAM feature

2.7 Configuring Power Source

This release adds power source configuration to enhance the performance of the system that has only VDD 5V or only battery. If the build is configured for VDD 5V only, the system will not power on if the system has battery but does not have VDD 5V.

To configure the power source, NO_DCDC_BATT_SOURCE and NO_VDD5V_SOURCE can be defined in power_prep.c of i.MX28 bootlet. The steps to configure the power source are as follows:

1. Generate the bootlet source code:

   ```
   ./ltib -p boot_stream.spec -m prep // generate bootlet source code
   ```

2. In imx-bootlets-src-10.12.01/power_prep/power_prep.c:

   ```
   #define NO_DCDC_BATT_SOURCE   // enable VDD 5V only
   ```

   Or,

   ```
   #define NO_VDD5V_SOURCE       //enable battery-only
   ```

3. Build the bootlet code:

   ```
   ./ltib -p boot_stream.spec -f       //build bootlet code
   ```

Chapter 3
Configuring the Target Hardware

This chapter details all hardware-specific configuration required to prepare the i.MX28 EVK development board for using with Linux.

3.1 External Cabling

Perform the following steps to set up external cabling:

1. Plug the Linux host straight serial console cable into the UART DSUB9 connector on the i.MX28 EVK.
2. Plug the Linux host USB A to mini-B USB cable into the mini-B USB connector on the i.MX28 EVK.

3.2 Board Configuration

The EVK board uses DIP switch S2 to select boot mode. Bits B0, B1, B2, and B3 are labeled on the board silkscreen. Table 1 gives the boot mode values.

Table 1. Boot Mode Values

B3	B2	B1	B0	Boot Mode
0	0	0	0	USB0
0	1	0	0	GPMI (NAND)
1	0	0	1	SSP0(SD0)
1	0	1	0	SSP1(SD1)

See *Hardware User Manual* for detailed EVK board configuration.

NOTE

The i.MX28 EVK board needs hardware rework for booting from SD1 (see *EVK Hardware User Guide* for details).

Chapter 4
Creating Boot Stream Image

The i.MX28 system-on-chip (SoC) contains built-in ROM firmware, which is capable of loading and executing binary images in special format from different locations, including an MMC/SD card and NAND Flash. Such a binary image is called a “boot stream” and it consists of a number of small bootable images (bootlets) and instructions for the ROM firmware to handle these bootlets (for example, load a bootlet to on-chip RAM and run it).

4.1 Setting Up the Kernel Command Line

In LTIB, run the following command, then choose “Package list,” and then set default and alternative kernel command lines under the “boot stream” option:

```
./ltib -m config
```

4.2 Building the Boot Stream Image

In LTIB, to build a new Linux kernel and U-Boot boot stream image, run the following command:

```
./ltib -p boot_stream.spec -f
```

The output boot stream images are available in `rootfs/boot/` directory. `imx28_linux.s` and `imx28_uboot.sb` are HAB-disabled images and `imx28_ivt_linux.sb` and `imx28_ivt_uboot.sb` are HAB-enabled images. By default, the i.MX28 EVK is shipped with HAB enabled.

Chapter 5
Booting the Target Hardware

This chapter explains how to boot the i.MX28 development board for the first time. Linux kernel can be booted on the i.MX28 using the following ways:

- Boot from USB
- Boot from MMC/SD card
- Boot from NAND Flash
- Boot from Ethernet (network boot)

All boot modes except network boot are supported by the i.MX28 built-in ROM firmware. The ROM code reads the boot stream image containing the Linux kernel from the first three sources. Network boot of the Linux kernel is performed by the U-Boot bootloader. U-Boot is loaded and started by the ROM firmware through the USB or MMC card or NAND Flash.

The Linux software development kit (SDK) provides the following two boot stream images:

- Linux kernel boot stream
- U-Boot boot stream

See Chapter 4 for details on how to generate a new boot stream image.

The following sections describe how to prepare and boot the Linux kernel in each boot mode.

5.1 Target Preparation

5.1.1 Setting Up Kernel Command Line

The kernel boot command line can be set in LTIB in the "boot stream" configuration menu under "Package list" → "boot stream." These command line options include a default command line and three command lines selected by key-press during system start up. If the command line configuration file has less than four command lines, then the unused entries are replaced by the following default command line string:

```
console=ttyAM0,115200
```

To select the location of the root file system, the "root" command line variable must be configured. Based on the root file system storage type, additional command line options may also need to be set:

- Root file system located on a MMC card partition:

  ```
  root=/dev/mmcblk0p[N] rw rootwait
  ```

  Where N is the number of the MMC card primary partition containing the root file system

- Root file system located on a NAND Flash (JFFS2):

  ```
  root=/dev/mtdblock1  rootfstype=jffs2
  ```

- Root file system on NFS over Ethernet link:

  ```
  ip=dhcp/off/[Target IP] root=/dev/nfs nfsroot=/tools/rootfs
  ```

  Where:

  Host IP is the IP address of Ubuntu Linux host

  Target IP is the IP address assigned to the i.MX28 development board

- ENET MAC address

  NOTE

  On the i.MX28 EVK, the MAC address is stored in OTP fuses that have been pre-programmed. If a different MAC address is to be used, then the following option can be added:

  ```
  fec_mac=xx:xx:xx:xx:xx:x
  ```

  Where:

  'xx:xx:xx:xx:xx:xx' indicates the MAC address of ENET of the EVK board

- gpmi or ssp1 selection

  ```
  gpmi/ssp1
  ```

  Where:

  gpmi: initialize gpmi (that is, NAND) interface

  ssp1: initialize ssp1 (that is, SD Card 1) interface

  Either a NAND device can be used, or an SD device on SD slot 1 can be used, but not both. This is due to pin-sharing on the i.MX28. SD slot 0 is unaffected by this choice.

There are four preset command lines that allow booting the kernel with the root file system located on SD/MMC card, NAND Flash, NFS, or RAM disk:

- Default command line for SD card (no key press during booting):

```
noinitrd console=ttyAM0,115200 root=/dev/mmcblk0p3 rw rootwait gpmi
```

- Alternative command line 1 for NAND (press KEY1 during booting):

```
noinitrd console=ttyAM0,115200 ubi.mtd=1 root=ubi0:rootfs0
rootfstype=ubifs rw gpmi
```

- Alternative command line 2 for NFS (press KEY2 during booting):

```
noinitrd console=ttyAM0,115200 fec_mac=00:08:02:6B:A3:1A root=/dev/nfs
nfsroot=10.193.100.213:/data/rootfs_home/rootfs_mx28 rw ip=dhcp rootwait
gpmi
```

- Alternative command line 3 for RAM disk (press KEY3 during booting):

```
noinitrd console=ttyAM0,115200 root=/dev/ram0 rdinit=/sbin/init
fec_mac=00:08:02:6B:A3:1A gpmi
```

5.1.2 Rebuilding the Linux Image

If the default command lines are modified, then it is necessary to rebuild the release to get the Linux kernel boot stream image with those updated command lines. In LTIB, run the following command:

```
./ltib -p boot_stream -f
```

5.1.3 Writing the Boot Stream and rootfs to a Boot Medium

This section describes how to write the boot stream and rootfs to a boot medium.

5.1.3.1 MMC Boot

The default kernel command line uses three primary partitions as follows:

- File Allocation Table (FAT)
- Boot stream partition of type 0x53 (OnTrack DM6 Aux3)
- Linux rootfs, such as EXT2

You can use the following steps to create such an SD/MMC card:

- Create three primary partitions on MMC card in addition to the mandatory one that contains the boot stream image. The example below shows the fdisk -1 output, which is configured using the MMC/SD card (all these steps are done in the host Linux computer):

```
fdisk -l /dev/sdb

Disk /dev/sdb: 1967 MB, 1967128576 bytes
```

```
61 heads, 62 sectors/track, 1015 cylinders

Units = cylinders of 3782 * 512 = 1936384 bytes

Disk identifier: 0x00000000

Device Boot Start End   Blocks      Id    System

/dev/sdb1   *           1         974     1840896    b  W95 FAT32

/dev/sdb2             974         977        5339+  53  OnTrack DM6 Aux3

/dev/sdb3             977        1016       74572  83  Linux
```

- Build sdimage in host Linux computer. sdimage.c is in the “uuc” package in LTIB. Use below steps to build sdimage in Linux:

```
./ltib -p uuc.spec -m prep

cd rpm/BUILD/uuc-2.6.35.3-1.1.0

gcc sdimage.c -o sdimage
```

  (Any C compiler can be used in the Linux)

- Create an MMC partition image that contains the Linux kernel boot stream in target using “sdimage:”

```
sdimage –f imx28_linux.sb –d /dev/sdb

sync
```

- Create an MMC partition image that contains the U-Boot boot stream and uImage.

```
sdimage –f imx28_uboot.sb –d /dev/sdb

sudo dd if=uImage of=/dev/sdb bs=1K seek=128

sync
```

- Clean the U-Boot environment variable to use the default:

```
dd if=/dev/zero of=zero.raw bs=1K count=127

sudo dd if=zero.raw of=/dev/sdb bs=1K seek=1

sync
```

- Format the second partition:

```
mkfs.ext2 /dev/sdb3
```

- Mount the second MMC/SD partition:

  ```
  mount /dev/sdb3 /mnt/mmc
  ```

- Copy the Linux release development root file system to the directory where the MMC partition is mounted. See Section 2.3, "Choosing rootfs over NFS," for retrieving the root file system.
- Add manual updates to the root file system in `/mnt/mmc`, if required.
- Unmount the MMC partition:

  ```
  umount /mnt/mmc
  ```

As an alternative, the Windows tools, `cfimager`, can create the partitions, and write the boot stream and rootfs:

```
cfimager.exe -a -f imx28_linux.sb -e rootfs.ext2 -d <mass storage disk, no ":", such as H>
```

To write U-Boot boot stream, kernel, and rootfs:

```
cfimager.exe -a -f imx28_uboot.sb -e rootfs.ext2 -d <mass storage disk, no ":", such as H>

cfimager.exe -raw -offset 0x20000 -f uImage -d <mass storage disk, no ":", such as H>
```

Clean the U-Boot environment variable to use the default:

```
cfimager.exe -raw -offset 0x400 -f zero.raw -d <mass storage disk, no ":", such as H>
```

Where zero.raw is 127KB binary file filled with zero. It can be created in Windows using any editor.

NOTE

The default rootfs file system released for SD card is EXT2 format. EXT2 is not a journaling file system. Any disruption to the file system while syncing can cause a file system error, such as power lost, kernel panic, and so on. To avoid such errors, either follow the normal power sequence or use an EXT3 file system. You can convert an EXT2 file system to an EXT3 file system by using the command `tune2fs -j rootfs.ext2`.

5.1.3.2 NAND Boot

A boot stream image cannot be burned to NAND Flash from the Linux host. It is necessary to first load the kernel from an MMC card or network boot. Then, after Linux is running on the board, it is possible to burn the boot stream image to NAND using the "kobs-ng" tool:

- Copy the boot stream to the root file system. For example, in the case of an NFS root, use:

  ```
  cp <where the boostream lives>/iMX28_linux.sb /tools/rootfs
  ```

- Boot the target and log into it.
- On the target, burn the boot stream image to the Flash:

  ```
  #flash_eraseall /dev/mtd0

  #kobs-ng init imx28_linux.sb
  ```

- Copy the JFFS2 image to current root file system.

  For example, in the case of NFS root, run the following command on the Linux host. Note that the JFFS2 image must match the type of Flash device in use.

  ```
  cp rootfs.jffs2 /tools/rootfs
  ```

- On the target, erase the MTD 1 partition:

  ```
  flash_eraseall /dev/mtd1
  ```

- On the target, burn the JFFS2 image from the rootfs to the Flash:

  ```
  nandwrite /dev/mtd1 rootfs.jffs2
  ```

5.1.3.3 Network Boot

Linux kernel network boot is implemented using the U-Boot bootloader, which is part of the Linux release for Freescale i.MX28. The U-Boot boot stream is loaded from SD card or USB by sb_loader.exe tool. See Section 5.3.1, "USB Boot," and Section 5.3.3, "Network Boot," for details.

5.2 Host Preparation

This section describes preparation of the host computer.

5.2.1 Root File System on NFS Partition

See Section 2.3, "Choosing rootfs over NFS," for details.

5.3 Target Boot

This section describes how to boot the target EVK.

5.3.1 USB Boot

Perform the following steps to boot from USB:

- Select the USB boot mode (0000) on the DIP switch. See Chapter 3 for details.
- Plug in the power cord to the i.MX28 development board.
- Press the power key.
- Press KEY1/2/3 to select an alternative boot cmdline; hold the key until the bootlets have run. To use the default boot command line, do not press any key.
- After Windows recognizes the EVK as a USB HID device, run the following command in the Windows console:

```
z:\sb_loader.exe /f imx28_linux.sb

z:\sb_loader.exe /f imx28_uboot.sb
```

NOTE

Use the `ivt` boot streams if the chip is HAB enabled.

5.3.2 MMC/NAND Boot

Perform the following steps to boot the MMC/NAND:

- Select a boot mode (SD/MMC:1001, NAND: 0100) using the DIP switch. See Chapter 3 for details.
- For SD card, insert the SD card with i.MX28_linux.sb and rootfs.ext2 burned by cfimage.exe into the SD slot 0. For NAND, the boot stream and rootfs image should be burned into the Flash as described previously.
- Apply power to the i.MX28 development board.
- Press the power key.
- Press KEY1/2/3 to select an alternative boot cmdline; hold the key until the bootlets have run. To use the default boot command line, do not press any key.
- The bootlets and kernel will run.

5.3.3 Network Boot

Perform the following steps to boot from the network:

- Connect the target and host using an Ethernet 10 Base-T cable.
- Ensure that the Trivial File Transfer Protocol (TFTP) server is running on the Linux host.
- Copy the Linux kernel image to the /tftpboot directory:

```
cp rootfs/boot/uImage /tftpboot
```

- Ensure that the rootfs file is available in the ROOTFS_DIR folder.
- Insert SD card with i.MX28_uboot.sb burnt by cfimage.exe into SD slot0.
- Power on the i.MX28 development board.
- Press the power key. Run bootlets and U-Boot.
- Press the Enter key in the U-Boot serial console (for example, using the minicom) to get the U-Boot prompt.
- Set the U-Boot run-time variables as follows:

```
MX28 U-Boot > setenv bootargs 'console=ttyAM0,115200n8'

MX28 U-Boot > setenv bootcmd 'run bootcmd_net'

MX28 U-Boot > setenv bootdelay 2

MX28 U-Boot > setenv baudrate 115200

MX28 U-Boot > setenv serverip [Host IP]

MX28 U-Boot > setenv netmask 255.255.255.0

MX28 U-Boot > setenv bootfile uImage

MX28 U-Boot > setenv loadaddr 0x42000000

MX28 U-Boot > setenv nfsroot [ROOTFS_DIR]

MX28 U-Boot > setenv bootargs_nfs 'setenv bootargs ${bootargs}
root=/dev/nfs ip=dhcp nfsroot=${serverip}:${nfsroot}  fec_mac=[MAC
address] gpmi'

MX28 U-Boot > setenv bootcmd_net 'run bootargs_nfs; dhcp; bootm'

MX28 U-Boot >setenv ethaddr [MAC address]

MX28 U-Boot >saveevn
```

Where:

Host IP: IP address of the Ubuntu Linux host

ROOTFS_DIR: NFS folder path

MAC address: MAC address of Ethernet

- Load and start the Linux kernel:

```
boot
```

Chapter 6
Downloading Images using MFGTools

6.1.1 Installing MFGTools

Unzip Mfgtools-Rel-1.1.0_SDK_MX28_UPDATER.tar.gz.

6.1.2 MFGTool Usage

See MFGTool documents under “Document” folder before you start to use MFGTool.

Following are some instructions for i.MX28 EVK MFGTool usage:

1. Connect USB cable from the host computer to USB port (J82) on MX28 EVK board.
2. Connect UART to the host computer for console output.
3. Open terminal program.
4. Set boot dips as “USB0 mode” (see Section 3.2, “Board Configuration”).
5. Power up board.
6. You can specify your images in two ways: First is by editing “Profiles\MX28 Linux Update\OS Firmware\ucl.xml” to modify file path or Flash operations according to your usage. The other way is by copying your files in “Profiles\MX28 Linux Update\OS Firmware\files” directory. You can replace the files inside this folder. Note that you will find updater.sb and updater_ivt.sb binaries in “Profiles\MX28 Linux Update\OS Firmware” folder; you must not replace these files.
7. Execute “MfgTool.exe” program. Select “Options -> Configuration…” menu. If you are connecting i.MX28 to MFGTool for the first time, install USB driver under “Drivers\iMX_BulkIO_Driver.”
8. Select the right USB port in the “USB Ports” sheet**.**
9. Select the right profile in the “Profiles” sheet. Type the item in “Operations.” Right-click and select “Edit…” Select “Single chip NAND” to program the images to NAND. Select “SD (without uboot)” to program images to SD card.
10. Start the download process by pressing the green Start button. You will see the progress bar as well as the current task in the notification bar as shown in Figure 6-1. When you receive an “Update Complete” message in the notification bar, press the red Stop button to finish.

11. Sometimes, manufacturing tool reports an error message when it is downloading the file system in an SD card. That can be caused by insufficient space in the SD card due to a small partition size. To fix this, edit the "Profiles\MX28 Linux Update\OS Firmware\ fdisk-u.input" file and increase the size of the partition according to your file system requirements. You can notice that the content of this file are *fdisk* inputs, so increasing the partition size increases the number before "w" letter.

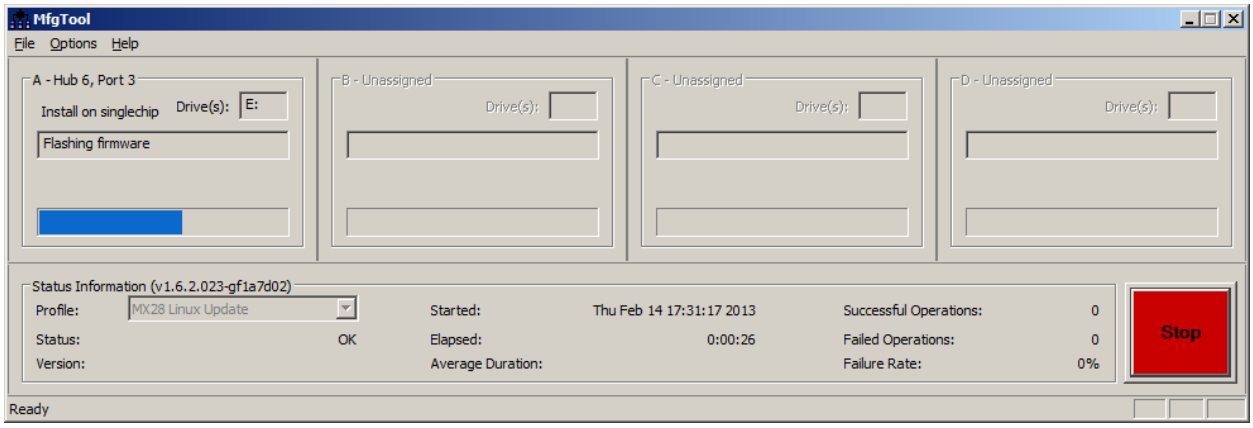

Figure 6-1. Programming NAND with Manufacturing Tool

NOTE

Mfgtools-Rel-1.1.0 supports Windows XP; it does not support Windows 7.