Freescale Semiconductor
User Guide

Document Number:IMX6LXUGSAI
Rev L3.10.17_1.0.0-ga, 05/2014

i.MX 6 SABRE-AI Linux User's Guide

Contents

1 About This Book

This document explains how to build and install the Freescale Linux BSP, where BSP stands for Board Support Package, on the i.MX 6 SABRE-AI platform.

All steps needed to get the i.MX 6 SABRE-AI platform running are detailed, including board dip switch settings, and instructions on configuring and using the U-Boot boot loader.

1.1 Audience

This information 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.

1.2 References

i.MX 6 SABRE-AI Linux Reference Manual.

1.3 Supported HW SoC/Board

Supported HW SoC/Board

- i.MX 6Quad SABRE-AI Platform
- i.MX 6DualLite SABRE-AI Platform

2 Introduction

The i.MX 6 SABRE-AI Linux BSP is a collection of binary, source code, and support files that can be used to create U-Boot boot loader, Linux kernel image, and a root file system for i.MX 6 SABRE-AI development systems For the steps on how to run and configure Yocto Project rootfs, see the *Freescale Yocto Project User's Guide* (document IMXLXYOCTOUG).

2.1 Boot Loader

The i.MX 6 SABRE-AI Linux delivery package contains the following U-Boot boot loader binary:

i.MX 6Quad SABRE-AI:

- u-boot-imx6qsabreauto_sd.imx (SD/MMC)
- u-boot-imx6qsabreauto_spi-nor.imx (SPI-NOR)
- u-boot-imx6qsabreauto_eim-nor.imx (Parallel NOR)
- u-boot-imx6qsabreauto_nand.imx (NAND)
- u-boot-imx6qsabreauto_sata.imx (SATA)

i.MX 6DualLite SABRE-AI:

- u-boot-imx6dlsabreauto.imx (SD/MMC)
- u-boot-imx6dlsabreauto_spi-nor.imx (SPI-NOR)
- u-boot-imx6dlsabreauto_eim-nor.imx (Parallel NOR)
- u-boot-imx6dlsabreauto_nand.imx (NAND)

i.MX 6Solo SABRE-AI:

- u-boot-imx6solosabreauto.imx (SD/MMC)
- u-boot-imx6solosabreauto_spi-nor.imx (SPI-NOR)
- u-boot-imx6solosabreauto_eim-nor.imx (Parallel NOR)
- u-boot-imx6solosabreauto_nand.imx (NAND)

Bootloaders are provided to support: SD/MMC, SPI-NOR, NAND, Parallel NOR.

2.2 Linux Kernel Image

This Freescale i.MX BSP contains a pre-built kernel image based on the 3.10.17 version of the Linux kernel and the device tree files associated with each of the platforms.. The i.MX 6 SABRE-AI Linux delivery package contains the following Linux kernel binary and associated device tree binaries:

The kernel can be built with different defconfigs as listed in local.conf specified by FSL_KERNEL_DEFCONFIG. For more details, see *Freescale Yocto Project User's Guide*. A manufacturing tools kernel is built by using the imx_v7_mfg_defconfig while the default kernel is built by using the imx_v7_defconfig.

i.MX 6Quad SABRE-AI:

- uImage_imx_v7_defconfig
- uImage-imx6q-sabreauto.dtb
- uImage-imx6q-sabreauto-ecspi.dtb

- uImage-imx6q-sabreauto-gpmi-weim.dtb
- uImage-3.10.17-r0-imx6q-sabreauto-flexcan1.dtb

i.MX 6DualLite SABRE-AI, i.MX 6Solo SABRE-AI:

- uImage_imx_v7_defconfig
- uImage-imx6dl-sabreauto.dtb
- uImage-imx6dl-sabreauto-ecspi.dtb
- uImage-imx6dl-sabreauto-gpmi-weim.dtb
- uImage-imx6dl-sabreauto-flexcan1.dtb

2.3 QT Root File System

The root file system package provides busybox, common libraries, and other fundamental elements.

The i.MX 6 SABRE-AI BSP package contains the following rootfs file systems:

- fsl-image-dfb-imx6qdlsolo.ext3
- fsl-image-fb-imx6qdlsolo.ext3
- fsl-image-weston-imx6qdlsolo.ext3
- fsl-image-x11-imx6qdlsolo.sdcard

The file system includes Freescale specific libraries and QT GUI. It can be mounted as NFS, or its contents can be stored on a boot media such as Secure Digital (SD) card.

3 Building the Linux Platform

Information found here explains how to set up the Yocto Project build environment, install and build set the rootfs for NFS, and set up the host environment.

Note that not all of the steps are required for every boot mode. The only required steps are in the *Freescale Yocto Project User's Guide*.

3.1 Setting Up the Linux Host

See the "Freescale Yocto User Guide" included in the release package to set up the Linux host server.

3.2 How to Enable Solo Emulation on i.MX 6 SABRE-AI Board

Solo emulation can be enabled on the i.MX 6 SABRE-AI board. This is achieved by using a specific U-Boot configuration on the bootloader build process.

When this Solo emulation is enabled on the i.MX 6 SABRE-AI platform, the capabilities of the i.MX 6DualLite change as follows:

- One CPU enabled
- 32-bit data bus on DDR RAM
- 1 GB of RAM

To build U-Boot for an i.MX 6Solo, use the following command

```
MACHINE=imx6solosabreauto bitbake u-boot-imx
```

4 How to Boot the i.MX 6 SABRE-AI Board

The boot modes of the i.MX 6 SABRE-AI board are controlled by the boot configuration DIP switches on the board. To locate the boot configuration switches, see SABRE-AI Quick Start Guide (SABREAI_IMX6_QSG). The following sections list basic boot setup configurations only.

4.1 How to Enter Serial Download Mode for MFGTool

Table below shows the boot switch settings which are used to enter serial download mode for MFGTool. If bootimage is not validated in boot media, system will enter serial download mode.

Table 1. Boot switch setup for MFGTool

Switch	D1	D2	D3	D4
S3	OFF	ON	OFF	OFF

4.2 How to Boot From SD Card from Slot3

The following table shows the dip settings for SD boot on CPU board:

Table 2. Boot switch setup for SD boot on CPU board

Switch	D1	D2	D3	D4	D5	D6	D7	D8
S1	X	X	X	OFF	ON	X	X	X
S2	X	OFF	ON	OFF	-	-	-	-
S3	OFF	OFF	ON	OFF	-	-	-	-

The following table shows the dip settings for MMC boot on CPU board:

Table 3. Boot switch setup for MMC boot on CPU board

Switch	D1	D2	D3	D4	D5	D6	D7	D8
S1	X	X	X	OFF	ON	X	X	X
S2	X	ON	ON	OFF	-	-	-	-
S3	OFF	OFF	ON	OFF	-	-	-	-

4.3 How to Boot From NAND

The following table shows the dip settings for NAND boot:

Table 4. Boot switch setup for NAND

Switch	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
S1	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
S2	OFF	OFF	OFF	ON	-	-	-	-	-	-
S3	OFF	OFF	ON	OFF	-	-	-	-	-	-

4.4 How to Boot from SPI-NOR

The table below shows the boot switch settings to boot from SPI-NOR.

Table 5. Boot Switch Setup for SPI-NOR Boot

Switch	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
S1	X	X	X	X	X	X	X	X	X	X
S2	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
S3	OFF	OFF	ON	OFF	-	-	-	-	-	-

NOTE

In order to boot from SPI NOR, jumper J3 must be set between pins 2 and 3.

4.5 How to Boot from EIM (Parallel) NOR

The following table shows the dip settings for NOR boot:

Table 6. Boot switch setup for EIM NOR

Switch	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
S1	X	X	X	OFF	OFF	ON	X	X	X	X
S2	X	OFF	OFF	OFF	-	-	-	-	-	-
S3	OFF	OFF	ON	OFF	-	-	-	-		

NOTE

SPI and EIM NOR have pin conflicts. Both cannot be used for the same configuration. The default U-Boot configuration is set to SPI NOR.

5 Flash Memory Map

This chapter describes the software layout in MMC/SD cards.

This information may be useful for understanding subsequent sections about image download.

5.1 MMC/SD/SATA Memory Map

The MMC/SD/SATA scheme is different from the NAND and NOR flash which are deployed in the BSP software. The MMC/SD/SATA must keep the first sector (512 bytes) as the MBR (Master Boot Record) in order to use MMC/SD as the rootfs.

Upon boot up, the MBR is executed to look up the partition table to determine which partition to use for booting. The bootloader should be after the MBR. The kernel image and rootfs may be stored at any address after bootloader.

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

5.2 NAND Flash Memory Map

NAND flash scheme is configured by kernel command line.

For example:

```
mtdparts=gpmi-nand:16m(boot),16m(kernel),16m(dtb),-(rootfs)
```

5.3 Parallel NOR Flash Memory Map

There is only one partition by default.

Add more partitions with kernel command line. such as:

```
        mtdparts=8000000.nor:1m(boot),-(rootfs)
```

5.4 SPI-NOR Flash Memory Map

SPI-NOR schema can be configured by the kernel command line.

For example:

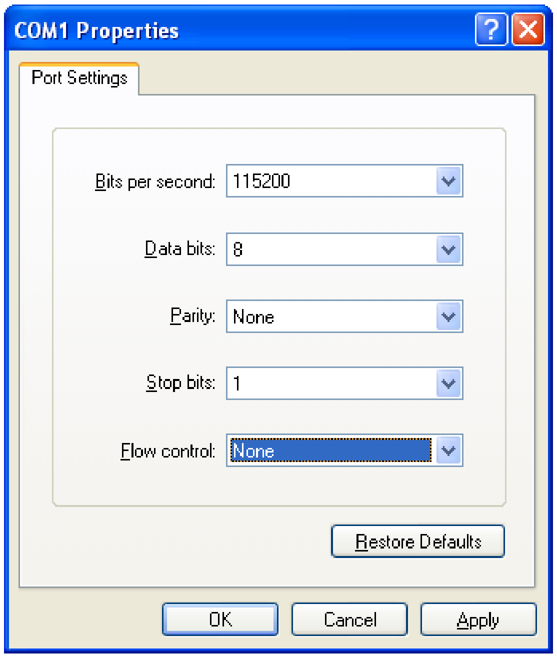
```
spi-nor mtdparts=spi32766.0:768k(boot),8k(env),128k(dtb),-(kernel)
```

6 Download Images by Bootloader or NFS

6.1 Setup Terminal

The i.MX 6 SABRE-AI board can communicate with a host server (Windows or Linux) using the serial cable. Common serial communication programs such as HyperTerminal, Tera Term, or PuTTY can be used. The example below describes the serial terminal setup using HyperTerminal on a Windows host:

1. Connect the target and the Windows PC using a serial cable.
2. Open HyperTerminal on the Windows PC and select the settings as shown in the figure below.



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

Figure 1. HyperTerminal Settings for Terminal Setup

6.2 Download Images by U-Boot

The following section describes how to download images by U-Boot.

6.2.1 MMC/SD on SD3(J14 CPU Board)

To display the U-Boot prompt, press any key before the value of the U-Boot environment variable, "bootdelay", decreases and before it times out. The default setting is 1 second.

To flash the U-Boot, refer to Chapter 7 "Using a Linux Host to Set Up an SD/MMC Card".

1. To clean up the environment variables stored on MMC/SD to their defaults, type the following in the U-Boot console:

```
U-Boot > env default -f -a
U-Boot > save
U-Boot > reset
```

2. Configure the U-Boot environment for network communications. Below is an example. The lines preceding with the "#" character are comments and have no effect.

```
U-Boot > setenv serverip <your tftpserver ip>
U-Boot > setenv bootfile <your kernel uImage name on the tftp server>
U-Boot > setenv fdt_file <your dtb image name on the tftp server>
### The user can set fake MAC address via ethaddr enviroment if the MAC address is not
fused
```

```
U-Boot > setenv ethaddr 00:01:02:03:04:05
U-Boot > save
```

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

```
U-Boot > dhcp
```

4. Query the information about MMC/SD card in slot 3.

```
U-Boot >mmc dev 0
U-Boot >mmcinfo
```

5. Check the usage of "mmc" command. The "blk#" is equal to "<the offset of read/write>/<block length of the card>". The "cnt" is equal to "<the size of read/write>/<block length of the card>".

```
U-Boot > help mmc
mmc - MMC sub system

Usage:
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition]
mmc list - lists available devices
```

6. Program the kernel uImage located in RAM at ${loadaddr} into the microSD. For example the command to write the image with the size 0x800000 from ${loadaddr} to the offset of 0x100000 of the microSD card. Refer to the following examples for the definition of the mmc Parameters.

```
blk# = (microSD Offset)/(SD block length) = 0x100000/0x200 = 0x800

cnt = (image Size)/(SD block length) = 0x800000/0x200 = 0x4000
```

   This example assumes the kernel image is equal to 0x800000. If the kernel image exceeds 0x800000, increase the image length. After issuing the tftp command, filesize U-Boot environment variable is set with the number of bytes transferred. This can be checked to determine the correct size needed for the calculation. Use U-Boot command printenv to see the value.

```
U-Boot >  mmc dev 0
### suppose kernel uImage less than 8M
U-Boot > mmc write ${loadaddr} 0x800 0x4000
```

7. Program the dtb file located in RAM at ${fdt_addr} into the microSD.

```
U-Boot > tftpboot ${fdt_addr} ${fdt_file}
```

8. Boot up the system through rootfs in SD card via HannStar LVDS:

```
### For LVDS0 connection
U-Boot > setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/
mmcblk0p2 rootwait rw video=mxcfb0:dev=ldb,LDB-XGA,if=RGB666'
### For LVDS1 connection
U-Boot > setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/
mmcblk0p2 rootwait rw video=mxcfb0:dev=ldb,LDB-XGA,if=RGB666 ldb=sin1'
U-Boot >setenv bootcmd_mmc 'run bootargs_base bootargs_mmc;mmc dev 0;mmc
read ${loadaddr} 0x800 0x4000;mmc read ${fdt_addr} 0x5000 0x800;bootm ${loadaddr} - $
{fdt_addr}'
U-Boot > setenv bootcmd 'run bootcmd_mmc'
U-Boot > saveenv
```

6.3 U-Boot Configurations

The U-Boot "print" command can be used to check environment variable values.

The "setenv" command can be used to set environment variable values. See the U-Boot user guide for details.

6.4 Use i.MX 6 SABRE-AI Board as Host Server to Create rootfs

Linux provides multiple methods to program images to the storage device. This section describes how to use the i.MX 6 SABRE-AI as Linux Host server to create the rootfs on MMC/SD cardor SATA device. The example below is SD card. Device file node name needs to be changed for SATA device.

1. Boot from NFS or other storage. Check partitions information:

```
root@freescale ~$ cat /proc/partitions
```

2. To create a partition in MMC/SD Slot 3, use the fdisk command in the Linux console:

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

3. As described in Flash Memory Map, the rootfs partition should be located after kernel image; the first 0x800000 bytes can be reserved for MBR, bootloader, and kernel sections. From the log shown above, the Units of current MMC/SD card is 32768 bytes. The beginning cylinder of the first partition can be set as "0x300000/32768 = 96." The last cylinder can be set according to the rootfs size. Create a new partition by typing:

```
         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
default value 124368
         Command (m for help): w
         The partition table has been altered!
         Calling ioctl() to re-read mmcblk0 partition table
          p1
```

4. Format the MMC/SD partitions as types ext3 or ext4 type. For example, to use ext3:

```
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.
```

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

```
mount -t ext3 -o loop /fsl-image-x11-sdk-imx6qsabresdrootfs.ext3 /mnt/cdrom
cd /mnt
mkdir mmcblk0p1
mount -t ext3 /dev/mmcblk0p1/mnt/mmcblk0p2/
cp -af /mnt/cdrom/* /mnt/mmcblk0p1/
umount /mnt/mmcblk0p1
umount /mnt/cdrom
```

6. Type sync to write the contents to MMC/SD.
7. Type poweroff to power down the system. Follow the instructions in Running the Image on Target to boot the image from MMC/SD card.

NOTE

v2013.04 U-Boot by default supports loading the kernel image and DTB file from the sd/mmc vfat partition by using fatload command. To use it, please do the following:

1. Format the sd/mmc card first partition(for example 32M) with vfat filesystem.
2. Copy uImage and the DTB file into the VFAT partition after you mount the VFAT partition into your host PC.
3. Make sure that the uImage and DTB file name sync with the file name pointed by the U-Boot environment variables: fdt_file and uImage. Use print command under U-Boot to display these two environment variables. For example:

```
print fdt_file uimage
```

4. U-Boot can load the kernel image and DTB file from your VFAT partition automatically when you boot from the sd/mmc card.

6.5 Flash U-Boot on SPI-NOR

To flash U-Boot on SPI-NOR, perform the following steps:

1. Boot from SD card.
2. Set Jumper J3 to position: 2-3.
3. Fetch built-in SPI-NOR support u-boot.bin image (see note).

```
----------------
tftpboot ${loadaddr} u-boot.imx
-----------------
```

4. Flash U-Boot image in SPI-NOR.

```
-----------------
sf probe
sf erase 0 0x80000
sf write ${loadaddr} 0 0x80000
----------------
```

5. Set Boot switch.
   - S2-1 1
   - S2-2 1
   - S2-3 0

- S2-4 0
- S1-[1:10] X

6. Reboot target board.

NOTE

Build U-Boot from source with SPI-NOR configuration settings:

1. `MACHINE=<i.MX machine>-spi-nor bitbake u-boot-imx -c deploy`
   where `<i.MX machine>` can be the following machines:
   - imx6qsabreauto
   - imx6dlsabreauto
   - imx6solosabreauto
2. Copy the resulting image into your host system TFTP folder

6.6 Flash U-Boot on Parallel NOR

To flash U-Boot on Parallel NOR, perform the following steps:

1. Boot from SD card.
2. tftp U-Boot image.

   ```
   tftpboot ${loadaddr} u-boot.imx
   ```
3. Flash U-Boot image.

   ```
   cp.b ${loadaddr} 8000400 ${filesize}
   ```
4. Flash kernel.

   ```
   tftpboot ${loadaddr} uImage

   cp.b ${loadaddr} 80c0000 ${filesize}
   ```
5. Change boot switches and reboot.

   ```
   S2 all 0

   S1-6 1 others 0
   ```
6. By default, rootfs is mounted on NFS.

NOTE

Build U-Boot from source with SPI-NOR configuration settings:

1. `MACHINE=<i.MX machine>-spi-nor bitbake u-boot-imx -c deploy`
   where `<i.MX machine>` can be the following machines:
   - imx6qsabreauto
   - imx6dlsabreauto
   - imx6solosabreauto
2. Copy the resulting image into your host system TFTP folder

7 Using a Linux Host to Set Up an SD/MMC Card

Information found here describes the steps to prepare an SD/MMC card to boot up an i.MX 6 SABRE-AI board.

The Yocto Project build creates an SD card image that can be flashed directly.

7.1 Requirements

An SD/MMC card reader, like a USB card reader, is required. It will be used to transfer the boot loader and kernel images to initialize the partition table and copy the root file system. To simplify the instructions, it is assumed that a 4GB SD/MMC card is used.

Any Linux distribution can be used for the following procedure.

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, enter the command:

```
$ 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 example, the device node assigned is /dev/sdb (a block is 1kB large).

NOTE

Make sure the device node is correct for the SD/MMC card. Otherwise, it may damage your operating system or data on your computer.

7.2 Copying the Boot Loader Image

Enter the following command to copy the U-Boot image to the SD/MMC card where the U-Boot image can be any of the following:

- u-boot-imx6dlsabreauto_sd.imx
- u-boot-imx6slevk_sd.imx
- u-boot-imx6dlsabresd_sd.imx
- u-boot-imx6solosabreauto_sd.imx
- u-boot-imx6qsabreauto_sd.imx
- u-boot-imx6solosabresd_sd.imx
- u-boot-imx6qsabresd_sd.imx

```
$ sudo dd if=<U-Boot image> of=/dev/sdb bs=512 seek=2 conv=fsync
```

The first 1 KB of the SD/MMC card, that includes the partition table, will be preserved.

7.3 Copying the Kernel Image and DTB File

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

```
$ sudo dd if=uImage_imx_v7_defconfig of=/dev/sdb bs=512 seek=2048 conv=fsync
```

This will copy uImage to the media at offset 1 MB (bs x seek = 512 x 2048 = 1MB).

The following command will copy the i.MX6Q SABREAI DTB image to the SD/MMC card:

```
$ sudo dd if=uImage-imx6q-sabreauto.dtb of=/dev/sdb bs=512 seek=20480 conv=fsync
```

This will copy uImage-imx6q-sabreauto.dtb to the media at offset 10 MB (bs x seek = 512 x 20480 = 10MB).

7.4 Copying the Root File System (rootfs)

First, a partition table must be created. If a partition already exists and it is big enough for the file system you want to deploy, then you can skip this step.

To create a rootfs partition, make sure that the rootfs will not overwrite the kernel image and the dtb file is stored in the SD card. Because the dtb file is stored at 10M offset, you can place the rootfs at 20M offset.

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

NOTE

On most Linux host operating systems, SD card will be mounted automatically upon insertion. Therefore, before running fdisk, please make sure that SD card is unmounted (via 'sudo umount /dev/sdb').

Type the following parameters (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]
+20M    [starting at offset sector #40960, i.e. 20MB, 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 file system format ext3 or ext4 is a good option for removable media due to the built-in journaling.

```
$ sudo mkfs.ext3 /dev/sdb1
Or
$ sudo mkfs.ext4 /dev/sdb1
```

Copy the target file system to the partition:

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

Extract a rootfs package to certain directory: extract `fsl-image-fb-imx6qdlsolo.ext3` to /home/user/rootfs for example:

```
$ sudo mount -o loop -t ext3 fsl-image-fb-imx6qdlsolo.ext3 /home/user/rootfs
```

NOTE

The rootfs directory needs to be created manually.

Assume that the root file system files are located in /home/user/rootfs as in the previous step:

```
$ cd /home/user/rootfs
$ sudo cp -a * /home/user/mountpoint
$ sudo umount /home/user/mountpoint
$ sudo umount /home/user/rootfs
```

NOTE

Copying the file system takes several minutes depending on the size of your rootfs.

The file system content is now on the media.

8 Running the Image on Target

This chapter explains how to run an image on the target from downloaded device and NFS.

These instructions assume that you have downloaded the kernel image using the instructions in Download Images by Bootloader or NFS, or Using a Linux Host to Set Up an SD/MMC Card. If you have not setup your Serial Terminal yet, please refer to Setup Terminal .

8.1 Run the Image from NFS

To boot from NFS, do as follows:

1. Power on the board.
2. Enter the following commands in the U-Boot prompt:

```
U-Boot > setenv serverip 10.192.225.216
U-Boot > setenv bootfile uImage
U-Boot > setenv nfsroot /data/rootfs_home/rootfs_mx6
U-Boot > setenv bootargs_base 'setenv bootargs console=ttymxc0,115200 uart_at_4M'
### For LCD connection
U-Boot > setenv bootargs_nfs 'setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp
nfsroot=${serverip}:${nfsroot},v3,tcp video=mxcfb0:dev=lcd,if=RGB565'
### For EPDC connection
U-Boot > setenv bootargs_nfs 'setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp
nfsroot=${serverip}:${nfsroot},v3,tcp video=mxcepdcfb:E060SCM,bpp=16
max17135:pass=2,vcom=-2030000'
### HDMI and LCD dual display connection
U-Boot > setenv bootargs_nfs 'setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp video=mxcfb0:dev=lcd,if=RGB565'
video=mxcfb0:dev=sii902x_hdmi,1920x1080M@60,if=RGB24'
U-Boot > setenv bootcmd_net 'run bootargs_base bootargs_nfs;dhcp ${uimage};dhcp ${fdt_addr} $
{fdt_file};bootm ${loadaddr} - ${fdt_addr}'
U-Boot > setenv bootcmd 'dhcp; run bootcmd_net'
U-Boot > saveenv
```

NOTE

If MAC address has not burned into fuse, you must set MAC address to use network in U-Boot.

```
setenv ethaddr xx:xx:xx:xx:xx:xx
```

8.2 Run the Image from MMC/SD

To boot the system from MMC/SD flash follow the steps below:

1. Power on the board.
2. Assume the kernel image starts from the address 0x100000 byte (the block start address is 0x800). The kernel image size is 0x800000 byte. Enter the following commands in the U-Boot prompt:

```
U-Boot > setenv loadaddr 0x80800000
U-Boot > setenv bootargs_base 'setenv bootargs console=ttymxc0,115200'
U-Boot > setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/
mmcblk0p2 rootwait rw video=mxcfb0:dev=lcd,if=RGB565 video=mxcfb0:dev=sii902x_hdmi,
1920x1080M@60,if=RGB24'
```

```
U-Boot > setenv bootcmd_mmc 'run bootargs_base bootargs_mmc;mmc dev
0;mmc read ${loadaddr} 0x800 0x4000;mmc read ${fdt_addr} 0x5000 0x800;bootm ${loadaddr}
- ${fdt_addr}'
U-Boot > setenv bootcmd 'run bootcmd_mmc'
U-Boot > saveenv
U-Boot > run bootcmd
```

8.3 Run the Image from NAND

The following steps may be used to boot the system from NAND:

1. Power up the board.
2. Assume the kernel image starts from the address 0x1400000 byte (the block start address is 0x800). The kernel image size is less than 0x400000 byte. The rootfs is located in /dev/mtd2. Enter the following commands in the U-Boot prompt:

```
U-Boot >  setenv loadaddr 0x12000000
U-Boot > setenv fdt_addr 0x18000000
U-Boot > setenv fdt_high 0xffffffff
U-Boot > setenv bootargs_base 'setenv bootargs console=ttymxc3,115200'
U-Boot > setenv bootargs_nand 'setenv bootargs ${bootargs} ubi.mtd=3
root=ubi0:rootfs rootfstype=ubifs rootwait rw mtdparts=gpmi-nand:16m(boot),16m(kernel),
16m(dtb),-(rootfs)'
U-Boot > setenv bootcmd_nand 'run bootargs_base bootargs_nand;nand read ${loadaddr}
0x1000000 0x800000;nand read ${fdt_addr} 0x2000000 0x100000;bootm ${loadaddr} - ${fdt_addr}'
U-Boot > setenv bootcmd 'run bootcmd_nand'
U-Boot > run bootcmd
```

8.4 Run the Image from Parallel NOR

The following steps may be used to boot the system from Parallel NOR:

1. Power up the board.
2. Assume the kernel image starts from the address 0x80000 byte. In the U-Boot prompt:

```
U-Boot > setenv loadaddr 0x12000000
U-Boot > setenv fdt_addr 0x18000000
U-Boot > setenv fdt_high 0xffffffff
U-Boot > setenv bootargs_base 'setenv bootargs console=ttymxc3,115200'
U-Boot > setenv bootargs_nor 'setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp weim-nor nfsroot=${serverip}:${nfsroot},v3,tcp'
U-Boot > setenv bootcmd_nor 'run bootargs_base bootargs_nor; cp.l 0x80c0000 $
{loadaddr} 0x800000;cp.l 0x80a0000 ${fdt_addr} 0x20000;bootm ${loadaddr} - ${fdt_addr} '
U-Boot > setenv bootcmd 'run bootcmd_nor'
U-Boot > run bootcmd
```

9 How to Check Current CPU Frequency

Scaling governors are used in the Linux kernel to set the CPU frequency.

CPU frequencies can be scaled automatically depending on the system load either in response to ACPI events, or manually by userspace programs. For more information about governors, read governors.txt from http://kernel.org/doc/Documentation/cpu-freq/governors.txt

The following are some of the frequently used commands:

- To get the available scaling governors:

  ```
  cat /sys/devices/system/cpu/*/cpufreq/scaling_available_governors
  ```
- To check the current CPU frequency:

  ```
  cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_cur_freq
  ```

  Frequency will be displayed depending on the governor set.
- To check the maximum frequency:

  ```
  cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_max_freq
  ```
- To check the minimum frequency:

  ```
  cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_min_freq
  ```
- To set constant CPU frequency:
  - Set the scaling governor to userspace and set the desired frequency:

    ```
    echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
    ```

Kernel is pre-configured to support only certain frequencies. The list of frequencies currently supported can be obtained from

```
cat /sys/devices/system/cpu/cpu0/cpufreq/stats/time_in_state
```

Set always to Max Frequency:

```
echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

9.1 How to Change the Core Operating Frequency from 1 GHz to 850 MHz

In order to display the current supported frequencies use the following command:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/stats/time_in_state
```

Check whether 850 MHz is listed in the table. If it is not, then kernel will chose the closest max frequency. In this case, it will be 996 MHz.

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor

echo 852000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_set_speed
```

In order to add the supported frequency, please check the kernel source arch/arm/mach-mx6/cpu_op-mx6.c. This process involves performing a few mandatory steps.

9.2 How to Check Frequency from Registers

The following command will help to determine the default PLL1 which is used by the CPU clock:

```
/unit_tests/memtool -32 0x020C8000 1
```

For example, for 792 MHz, the _hex_ value will be '80002042', the 'div_select' divisor is specified in bits [6:0], and the PLL1 frequency is computed with the following command:

```
Frequency = OSC clk (which is 24MHz) * div_select/2.0
```

The calculation will work out to 792 (MHz).

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: IMX6LXUGSAI
Rev. L3.10.17_1.0.0-ga
05/2014