AM335x U-Boot User's Guide

AM335x U-Boot User's Guide

Linux PSP

U-Boot

In AM335x the ROM code serves as the bootstrap loader, sometimes referred to as the Initial Program Loader (IPL) or the Primary Program Loader (PPL). The booting is completed in two consecutive stages by U-Boot [1] binaries. *The binary for the 1st U-Boot stage is referred to as the Secondary Program Loader (SPL) or the MLO. The binary for the 2nd U-Boot stage is simply referred to as U-Boot.* SPL is a non-interactive loader and is a specially built version of U-Boot. It is built concurrently when building U-Boot.

The ROM code can load the SPL image from any of the following devices

- **Memory devices non XIP (NAND/SDMMC)**

The image should have the Image header. The image header is of length 8 byte which has the load address(Entry point) and the size of the image to be copied. RBL would copy the image, whose size is given by the length field in the image header, from the device and loads into the internal memory address specified in the load address field of Image header.

- **Peripheral devices (UART)**

RBL loads the image to the internal memory address 0x402f0400 and executes it. No Image Header present.

Two stage U-Boot design

This section gives an overview of the two stage U-Boot approach adopted for AM335X.

The size of the internal RAM in AM335X is 128KB out of which 18KB at the end is used by the ROM code. Also, 1 KB at the start (0x402f0000 - 0x402f0400) is secure and it cannot be accessed This places a limit of 109KB on the size of the U-Boot binary which the ROM code can transfer to the internal RAM and use as an initial stack before initialization of DRAM.

Since it is not possible to squeeze in all the functionality that is normally expected from U-Boot in < 110KB (after setting aside some space for stack, heap etc) a two stage approach has been adopted. Initial stage initalize only the required boot devices (NAND, MMC, I2C etc); 2nd full stage initall all other devices (ethernet, timers, clocks etc). The 1st binary is generated MLO and the 2nd stage is generated as u-boot.img.

NOTE
*When using memory boot (NAND) a header needs to be attached to the SPL binary indicating the load address and the size of the image. SPI boot additionally requires endian conversion before flashing the image.

- When using peripheral boot (UART) there can be no header as the load address is fixed.

Updated Toolchain

Starting with Sitara Linux SDK 6.0 the location of the toolchain has changed and for non ARM 9 devices a new Linaro based toolchain will be used. Details about the change in toolchain location can be found here [2]. Also details about the switch to Linaro can be found here [3].

AM18x users are not affected by the switch to Linaro. Therefore, any references to the Linaro toolchain prefix "**arm-linux-gnueabihf-**" should be replaced with "**arm-arago-linux-gnueabi-**".

Building U-Boot

Prerequisite

GNU toolchain for ARM processors from Arago is recommended. Arago Toolchain can be found in the linux-devkit directory of the SDK here [4]

NOTE
Below steps assumes that the release package is extracted inside directory represented as $AM335x-PSP-DIR

Change to the base of the U-Boot directory.

```
$ cd ./AM335x-LINUX-PSP-MM.mm.pp.bb/src/u-boot/u-boot-MM.mm.pp.bb
```

Building into a separate object directory with the "O=" parameter to make is strongly recommended.

Commands

```
$ [ -d ./am335x ] && rm -rf ./am335x
$ make O=am335x CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm am335x_evm
```

This will generate two binaries in the am335x directory, MLO and u-boot.img along with other intermediate binaries that may be needed in some cases (see below).

Host configuration

Serial port configuration

Connect a serial cable from the serial port of the EVM (serial port is next to the power switch) to the COM port on either the Windows machine or Linux host depending on where you'll be running the serial terminal software.

For correct operation the serial terminal software should be configured with the following settings:

```
*Baud rate: 115,200
*Data bits: 8
*Parity: None
*Stop bits: 1
*Flow control: None
```

NOTE
If Teraterm is being used, ensure that the latest version (4.67 is the latest version as of writing this user guide) of Teraterm is installed. The implementation of the Kermit protocol in Teraterm is not reliable in older versions. The latest version of Teraterm 4.67 can be downloaded from here [5]. Recent Teraterm updates causes slow Binary transfer over UART. In such cases, use Windows in-built HyperTerminal application.

Target configuration

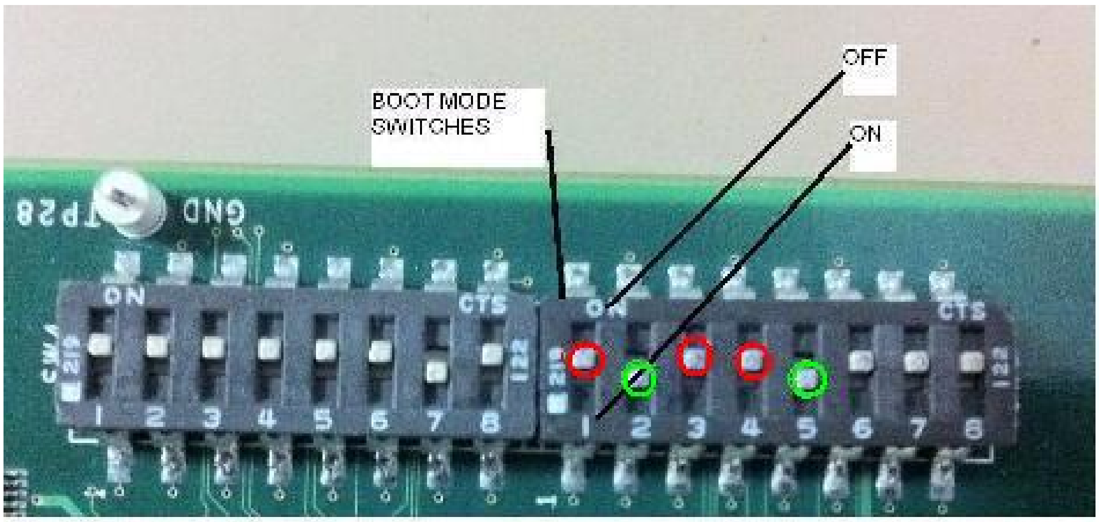
Boot Switch Settings

This option is only available on Am335x EVM. Switch SW3 is for selecting the boot modes. Also, separate DIP switch (SW8) is provided to select various profiles on EVMs.

The picture below shows the boot mode configuration switch SW3 on the AM335X EVM. **RED**: circle shows **OFF** and **GREEN** circles shows **ON** switches.

IMPORTANT
ON is labeled on the wrong side of SW3 boot mode switch.

NOTE
The bootmode setting in this picture is for NAND boot.NAND boot corresponds to (SW3 5:1) 10010.



- Make sure that the EVM boot switch settings are set to required boot mode and then power on the board.

NAND

In order to boot from the NAND flash, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	OFF	ON	OFF	OFF	ON

SPI

In order to boot from the SPI flash, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	OFF	ON	ON	OFF	ON

USB

In order to boot from the USBmode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	ON	OFF	ON	OFF

UART

In order to boot from the UART mode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	OFF	OFF	OFF	OFF

SD

In order to boot from the SD card, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	ON	ON	OFF	ON

CPSW Ethernet

In order to boot from the CPSW ethernet mode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5	6	7	8
Position	ON	ON	ON	ON	OFF	INDEPENDENT	OFF	ON

NOTE
The setting of switch SW3:[7:6] is because the EVM uses RGMII mode. For more details please refer to the TRM.

NOTE
Due to heavy pin-muxing, boot device is selectively available on selected AM335x EVMs & profiles. Details about the availability of the peripherals on different Profiles can be found from the EVM reference manual [6].

Flashing U-Boot with CCS

NOTE
Both the stages of U-Boot need to be flashed on the same media.

The tools are provided in the PSP release to write SPL & U-Boot on to the NAND flash(for NAND boot)

Refer to AM335x Flashing Tools Guide wiki page for instructions on how to flash the pre-built (or compiled) binary to NAND flash (or the recompiled one) with the help of the NAND flash writer.

After flashing the 2 stages, make sure boot mode is set to NAND and power on the board.

Boot Modes

NAND

NOTE
*The following sub-sections illustrate the usage of NAND specific commands on AM335X EVM.

- Refer to EVM Switch Settings section for more info on enabling/disabling different boot devices.

This section gives an *overview* of the NAND support in U-Boot. It also describe how to store the kernel image, RAMDISK or the UBIFS filesystem to NAND so as to have a network-free boot right from powering on the board to getting the kernel up and running.

Overview

Micron NAND parts (page size 2KB, block size 128KB) are supported on AM335XEVM platforms.

NAND Layout

The NAND part on the EVM has been configured in the following manner. The addresses mentioned here are used in the subsequent NAND related commands.

```
+------------+-->0x00000000-> SPL start         (SPL copy on 1st block)
|            |
|            |-->0x0001FFFF-> SPL end
|            |-->0x00020000-> SPL.backup1 start (SPL copy on 2nd block)
|            |
|            |-->0x0003FFFF-> SPL.backup1 end
|            |-->0x00040000-> SPL.backup2 start (SPL copy on 3rd block)
|            |
|            |-->0x0005FFFF-> SPL.backup2 end
|            |-->0x00060000-> SPL.backup3 start (SPL copy on 4th block)
|            |
|            |-->0x0007FFFF-> SPL.backup3 end
|            |-->0x00080000-> U-Boot start
|            |
|            |-->0x002BFFFF-> U-Boot end
|            |-->0x00260000-> ENV start
|            |
|            |
|            |-->0x0027FFFF-> ENV end
|            |-->0x00280000-> Linux Kernel start
|            |
|            |
|            |
|            |
|            |-->0x0077FFFF-> Linux Kernel end
|            |-->0x00780000-> File system start
|            |
|            |
|            |
|            |
|            |
|            |
|            |
|            |
|            |
|            |
|            |
|            |
+------------+-->0x10000000-> NAND end (Free end)
```

Writing to NAND

To write len bytes of data from a memory buffer located at addr to the NAND block offset:

```
U-Boot# nand write <addr> <offset> <len>
```

NOTE
*Offset & len fields should be in align with 0x800 (2048) bytes. On writing 3000 (0xbb8) bytes, len field can be aligned to 0x1000 ie 4096 bytes. Offset field should be aligned to page start address, multiple of 2048 bytes.

If a bad block is encountered during the write operation, it is skipped and the write operation continues from next 'good' block.

For example, to write 0x40000 bytes from memory buffer at address 0x80000000 to NAND - starting at block 32 (offset 0x400000):

```
U-Boot# nand write 0x80000000 0x400000 0x40000
```

Reading from NAND

To read len bytes of data from NAND block at a particular offset to the memory buffer in DDR located at addr:

```
U-Boot# nand read <addr> <offset> <len>
```

If a bad block is encountered during the read operation, it is skipped and the read operation continues from next 'good' block.

For example, to read 0x40000 bytes from NAND - starting at block 32 (offset 0x400000) to memory buffer at address 0x80000000:

```
U-Boot# nand read 0x80000000 0x400000 0x40000
```

Marking a bad block

Some of the blocks in the NAND may get corrupted over a period of time. In such cases you should explicitly mark such blocks as bad so that the image that you are writing to NAND does not end up getting corrupted.

To forcefully mark a block as bad:

```
U-Boot# nand markbad <offset>
```

For example, to mark block 32 (assuming erase block size of 128Kbytes) as bad block - offset = blocknum * 128 * 1024:

```
U-Boot# nand markbad 0x400000
```

Viewing bad blocks

To view the list of bad blocks:

```
U-Boot# nand bad
```

NOTE
*The user marked bad blocks can be viewed by using this command only after a reset.

Erasing NAND

To erase NAND blocks in a particular the address range or using block numbers:

```
U-Boot# nand erase <start offset addr> <len>
```

NOTE
*start offset addr & len fields should align to 0x20000 (64*2048) bytes, i.e.to block size 128KB.

This commands skips bad blocks (both factory and user marked) encountered within the specified range.

For example, to erase blocks 32 through 34:

```
U-Boot# nand erase 0x00400000 0x40000
```

NAND ECC algorithm selection

NAND flash memory, although cheap, suffers from problems like bit flipping which lead to data corruption. However by making use of some error correction coding (ECC) techniques it is possible to work around this problem.

Prior to the AM335xPSP_04.06.00.09-rc2 release, for the data stored in NAND flash, U-Boot supports following NAND ECC schemes

1. S/W ECC (Hamming code)
2. H/W ECC (Hamming code, BCH8)

Starting with the 04.06.00.09-rc2 release only BCH8 is supported, and is used by default in all cases. No user interaction is required to select the algorithm and locations where the **nandecc** command are required can be seen by looking at the history of this page.

BCH Flash OOB Layout

For any ECC scheme we need to add some extra data while writing so as to detect and correct (if possible) the errors introduced by the NAND part. In case of BCH scheme some bytes are needed to store the ECC related info.

The section of NAND memory where addition info like ECC data is stored is referred to as Out Of Band or OOB section.

The first 2 bytes are used for Bad block marker – 0xFFFF => Good block

The next ‘N’ bytes is used for BCH bytes

N = B * <Number of 512-byte sectors in a page>

1. B = 8 bytes per 512 byte sector in BCH4
2. B = 14 bytes per 512 byte sector in BCH8
3. B = 26 bytes per 512 byte sector in BCH16

So for a 2k page-size NAND flash with 64-byte OOB size, we will use BCH8. This will consume 2 + (14*4) = 58 bytes out of 64 bytes available.

The NAND flash part used in EVM does not have enough spare area to support BCH16.

ECC Schemes and their context of usage

ECC type	Usage
S/W ECC	Not used
H/W ECC - Hamming Code	Should use this scheme only for flashing the U-Boot ENV variables.
H/W ECC – BCH8	Should use this scheme while flashing any image/binary other than the U-Boot ENV variables.

To select ECC algorithm for NAND:

```
U-Boot# nandecc [sw | hw <hw_type>]
```

Usage:

sw - Set software ECC for NAND hw <hw_type> - Set hardware ECC for NAND <hw_type> - 0 for Hamming code 1 for bch4 2 for bch8 3 for bch16 Currently we support only Software, Hamming Code and BCH8. We do not support BCH4 and BCH16

ECC schemes usage table

ECC schemes usage table

Component	Default ECC scheme used by the component	ECC scheme to be used to flash the component	ECC schemes supported by the component
SPL	BCH8	BCH8	BCH8
U-boot	Hamming	BCH8	Hamming/BCH8
Linux	BCH8	BCH8	BCH8
File System	NA	BCH8	NA
Environment variables	NA	Hamming	NA

Flashing Kernel

TFTP the kernel uImage to DDR.

```
U-Boot# tftp 0x82000000 <kernel_image>
```

Now flash the kernel image to NAND at the appropriate offset (refer to NAND layout section for the offsets)

```
U-Boot# nand erase 0x00280000 0x00500000
U-Boot# nand write 0x82000000 0x00280000 0x500000
```

NOTE
*Image_size should be aligned to page size of 2048 (0x800) bytes

UBIFS file system flashing

In AM335X, UBIFS file system is used in NAND flash as it is next generation flash file system.

1. Creating and Flashing of UBIFS file system image is described over here

NOTE
In case of AM335x, file system partition is starting from 0x780000. So flashing offset for file system in U-Boot is 0x780000 and from Linux MTD partition number 7 should used for flashing file file system.

UART

This section describes how to use UART boot mode using TeraTerm.

Boot Over UART

NOTE
*The release package does not contain the binary for UART boot. Please follow the steps mentioned here for compiling u-boot and use the *spl/u-boot-spl.bin* file that is produced.

1. Switch ON EVM with switch settings for UART boot. When “CCCC” characters appear on TeraTerm window, from the File Menu select Transfer --> XMODEM --> Send (1K mode)
2. Select “u-boot-spl.bin” for the transfer
3. After image is successfully downloaded, the ROM code will boot it.
4. When “CCCC” characters appear on TeraTerm window, from the File Menu select Transfer --> YMODEM --> Send (1K mode)
5. Select “u-boot.img” for the transfer
6. After image is successfully downloaded, U-Boot will boot it.
7. Hit enter and get to u-boot prompt “U-Boot# ”

Flashing images to NAND in UART boot mode

Boot using UART boot mode as here

After the U-Boot prompt **U-Boot#** comes up, the images for the 1st stage and 2nd stage can be flashed to NAND for persistent storage.

Flashing SPL to NAND in UART boot mode

Flash SPL (**MLO**) to NAND by executing the following commands:

```
U-Boot# loadb 0x82000000
```

- From TeraTerm Menu click “**File -> Transfer -> Kermit -> Send**”.
- Select the 1st stage u-boot image “**MLO**” and click “OPEN” button
- Wait for download to complete and then run following commands in u-boot prompt

```
U-Boot# nand erase 0x0 0x20000
U-Boot# nand write 0x82000000 0x0 0x20000
```

If no error messages are displayed the SPL of NAND boot has been successfully transferred to NAND.

Flashing U-Boot to NAND in UART boot mode

Flash the 2nd stage U-Boot (**u-boot.img**) to NAND by executing the following commands:

```
U-Boot# loadb 0x82000000
```

- From TeraTerm Menu click “**File -> Transfer -> Kermit -> Send**”.
- Select the 2nd stage u-boot image “**u-boot.img**” and click “OPEN” button
- Wait for download to complete and then run following commands in U-Boot prompt

```
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write 0x82000000 0x80000 0x40000
```

If no error messages are displayed the U-boot of NAND boot has been successfully transferred to NAND.

SD (Secured Digital card)

This section gives an overview of the SD support in U-Boot

Read and execute uImage from SD card

Please note that the following commands are being executed from the 2nd stage. List files on a FAT32 formatted SD card

```
U-Boot# mmc rescan
U-Boot# fatls mmc 0
```

Booting kernel image from the SD card

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 uImage
U-Boot# bootm 0x82000000
```

Read and execute u-boot from SD card

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.bin
U-Boot# go 0x82000000
```

Setting Up Boot Environment on SD Card

This section describes steps to be followed to create a standalone power-on bootable system on SD card.

Prerequisites Ensure that following is available:

- A Linux host with fdisk, sfdisk, mkfs.ext3 and mkfs.vfat utilities is available
- Copy images MLO, u-boot.img, uImage, nfs.tar.gz and mksd-am335x.sh from release package to a directory on this Linux machine. For subsequent description, we will assume these files are copied to /home/am335x. Please refer "Package Contents" section in AM335x PSP User's Guide for location of these files.
- Empty SD card (at least 256MB, preferably 4GB SDHC)
- A SD memory card reader/programmer to copy files from Linux Host

NOTE
*The SD Boot has some specific restriction about the format of the 1st partition and copying the MLO image.

- So it is recommended that the supplied script mksd-am335x.sh is used for creating partitions and copying files.

Steps

- Connect the SD memory card using Memory Card reader to the Linux Host
- Note the name allotted for this device. Type "dmesg". The SD/MMC card name should show up near the end, usually something like "SDC" (/dev/sdc) or "SDD" (/dev/sdd).
- Navigate to the /home/am335x directory where all the mentioned files are copied
- Ensure that the script mksd-am335x.sh has executable permissions
- This scripts expects arguments in the following format

./mksd-am335x.sh <sd-device-name> <sd-1st-stage-bootloader> <sd-2nd-stage-bootloader> <kernel-uImage> <filesystem>

NOTE
that the user will require root/sudo permissions

- In our example, we run the following command

sudo ./mksd-am335x.sh /dev/sdd MLO u-boot.img uImage nfs.tar.gz

- You will be asked about data getting overwritten, confirm it and the files along with filesystem will be copied to SD card
- Note that this script will create two primary partitions:
  - 1st partition is formatted as FAT32 containing MLO, u-boot.img, uImage files
  - 2nd partition is formatted as ext3 where the filesystem is extracted in root

Boot using SD card

Once the SD card has been setup as described in the previous section make sure the switch setting are set for SD boot mode and then plug in the SD card in the MMC/SD card slot on the EVM.

Flashing images to NAND in SD boot

Boot using SD boot mode as here.

After the 2nd stage prompt **U-Boot#** comes up, the images for the 1st stage and 2nd stage can be flashed to NAND for persistent storage.

Flashing SPL to NAND in SD boot

Flash SPL (**MLO**) to NAND by executing the following commands:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 MLO
```

```
U-Boot# nand erase 0x0 0x20000
U-Boot# nand write 0x82000000 0x0 0x20000
```

If no error messages are displayed the SPL of NAND boot has been successfully transferred to NAND.

Flashing U-Boot to NAND in SD boot

Flash the 2nd stage U-Boot (**u-boot.img**) to NAND by executing the following commands:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.img
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write 0x82000000 0x80000 0x40000
```

If no error messages are displayed the U-boot of NAND boot has been successfully transferred to NAND.

Setting U-Boot environment using uEnv.txt

U-Boot environment variables can be modified using a plain text file named uEnv.txt. The scripts can be used to modify and even over-ride the various parameters like bootargs, TFTP serverip etc. If a command named uenvcmd is defined in the file it will be executed. uEnv.txt can be loaded from SD card and tftp server.

Example text file named *uEnv.txt*

IMPORTANT
*The uEnv.txt file should be in ***unix*** format. Also make sure that there is ***an empty line*** at the end of the file.

```
bootargs=console=ttyO0,115200n8 root=/dev/mmcblk0p2 mem=128M rootwait
bootcmd=mmc rescan; fatload mmc 0 0x82000000 uImage; bootm 0x82000000
uenvcmd=boot
```

The file *uEnv.txt* is automatically loaded from SD if the bootcmd is run. It can be loaded and put into the environment manually.

Making use pre-existing uEnv on SD card

uEnv.txt on an SD card can be used to override the env settings saved on a persistent storage like NAND by making use of the following commands

```
 U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x81000000 uEnv.txt
U-Boot# env import -t 0x81000000 $filesize
U-Boot# boot
```

SPI

Note

- This feature is not supported prior to PSP 04.06.00.08 (and AMSDK 05.05.00.00).
- This feature changed kernel location and filenames with PSP 04.06.00.09 (and AMSDK 05.06.00.00).
- The release package does not contain the binary for SPI boot. Please follow the steps mentioned here for compiling u-boot and use the *MLO.spi* and *u-boot.bin* files that are produced.
- Following those same instructions but building for **am335x_evm_spiboot** rather than **am335x_evm** will result in binaries that will use the SPI flash for environment rather than NAND.

In this example we initially boot from an SD card and use that to transfer the files to write to SPI flash. If loading via other methods, modify the commands below.

1. Switch ON EVM with switch settings such that SPI is present and boot via non-SPI. See SPI boot for more information.
2. Write it to SPI memory by executing the following commands:

```
U-Boot# sf probe 0
U-Boot# sf erase 0 +E0000
U-Boot# mmc rescan
U-Boot# fatload mmc 0 ${loadaddr} MLO.byteswap
U-Boot# sf write ${loadaddr} 0 ${filesize}
U-Boot# fatload mmc 0 ${loadaddr} u-boot.img
U-Boot# sf write ${loadaddr} 0x80000 ${filesize}
```

CPSW Ethernet

Note

- This feature is not supported prior to PSP 04.06.00.08 (and AMSDK 05.05.00.00).
- The release package does not contain the binary for CPSW ethernet boot. Please follow the steps mentioned here for compiling u-boot and use the *spl/u-boot-spl.bin* and *u-boot.img* files that are produced.

Booting Over CPSW Ethernet

- Configure DHCPd and tftpd on your host.
- Within the dhcpd configuration, add entries to send the *u-boot-spl.bin* or *u-boot.img* files based on the vendor-class-identifier field. For ISC dhcpd an example host entry looks like this:

```
 host am335x_evm {
hardware ethernet de:ad:be:ee:ee:ef;
if substring (option vendor-class-identifier, 0, 10) = "DM814x ROM" {
  filename "u-boot-spl.bin";
} elsif substring (option vendor-class-identifier, 0, 17) = "AM335x U-Boot SPL" {
  filename "u-boot.img";
} else {
  filename "uImage-am335x";
}
}
```

- Copy the *u-boot-spl.bin* and *u-boot.img* files you have build to the directory tftpd serves files from.
- Switch ON EVM with switch settings for CPSW ethernet boot.
- Hit enter and get to u-boot prompt “U-Boot# ”

Flashing in CPSW Ethernet boot mode

It is possible to build a version of U-Boot that boots over the cpsw ethernet interface and will automatically load and execute a script file. This can be used in initial programming or restoring of boards. It follows the general principles outlined above. In this case first you must build for **am335x_evm_restore_flash** rather than **am335x_evm** when building U-Boot. Next, you will need to create the script file that is run. There are examples provided in the source tree at **doc/am335x.net-spl/debrick-nand.txt** and **doc/am335x.net-spl/debrick-spi.txt**. You should copy one of these files and modify for your exact situation. Once you have modified the script you will need to turn it into a scr file using the following command:

```
./tools/mkimage -A arm -O U-Boot -C none -T script -d <your script> debrick.scr
```

and copy the resulting debrick.scr file to the location tftpd serves files out of. For more information please see the **doc/am335x.net-spl/README** file in the source tree.

U-Boot Network configuration

In order to download the Linux kernel image from the TFTP server and for mounting NFS the network settings in U-Boot need to be configured.

When booting for the first time, U-Boot tries to fetch the MAC address from the env space. If it returns empty, it will look for MAC address from the eFuse registers in the Control module space and set the "ethaddr" variable in the env appropriately. The ethaddr can also be set using the setenv/saveenv commands. In such cases the user-set MAC address will take effect on subsequent reboot only.

To set a different MAC address use the following command

```
U-Boot# set ethaddr <random MAC address eg- 08:11:23:32:12:77>
```

NOTE
*When setting a MAC address please ensure that the LSB of the 1st byte if not 1 i.e. when setting the MAC address: **y** in x**y**:ab:cd:ef:gh:jk has to be an even number. For more info this refer to the wiki page http://en.wikipedia.org/wiki/MAC_address

In case a static ip is not available run the dhcp command to obtain the ip address from the DHCP server on the network to which the EVM is connected.

```
U-Boot# setenv serverip <tftp server in your network>
U-Boot# netmask 255.255.255.0
U-Boot# dhcp
U-Boot# saveenv
```

In case a static ip is available run the following commands

```
U-Boot# setenv ipaddr <your static ip>
U-Boot# saveenv
```

This completes the network configuration in U-Boot.

U-Boot Environment Variables

After completing the network configuration and flashing the kernel image and filesystems to flash you need to set some other parameters which are essential for booting the kernel. We make use of a number of existing helper variables that exist in the environment at present. To see what they are set to use the *printenv* command. After setting **bootargs** along with any other variables for your needs you will need to do:

```
U-Boot# saveenv
```

In all cases we make use of **optargs** to control passing in of additional arguments and **ip_method** to determine how the kernel will deal with networking PRIOR to userspace spawning init. This does not control for example if a dhcp client will be started as part of the userspace init sequence.

Environment Settings for Ramdisk

In case you are using a RAMDISK as the Linux filesystem:

```
U-Boot# setenv bootargs ${console} ${optargs} root=/dev/ram rw initrd=${loadaddr},32MB ip=${ip_method}
```

For booting from NAND:

```
U-Boot# setenv nand_src_addr 0x00280000
U-Boot# setenv nand_img_siz 0x170000
U-Boot# setenv initrd_src_addr 0x00780000
U-Boot# setenv initrd_img_siz 0x320000
U-Boot# setenv bootcmd 'nand read ${kloadaddr} ${nand_src_addr}
${nand_img_siz};nand read ${loadaddr} ${initrd_src_addr}
${initrd_img_siz};bootm ${kloadaddr}'
```

For booting from SD card:

```
U-Boot# setenv bootcmd 'mmc rescan;run mmc_load_uimage;fatload mmc ${mmc_dev} ${loadaddr} initrd.ext3.gz;bootm ${kloadaddr}'
```

NOTE
*The sizes of images mentioned in the above commands have to be modified based on the actual image size. Also, it should be aligned to sector size of the flash device used.

Environment Settings for UBIFS Filesystem

- The U-Boot environment variable **bootargs** contains arguments to be passed to the Linux kernel. The **bootargs** variable here makes use of the **nand_root** variable. The typical format of this variable for a UBIFS root filesystem is:

```
root=ubi0:<VOLUME NAME> ubi.mtd=<PARTITION_ID>,YYYY rw
```

The value of **PARTITION_ID** depends on MTD device which holds the rootfs, **YYYY** depends on the page size of the partition and **VOLUME NAME** depends on the volume name given in ubinize.cfg file while creating UBIFS image as described here . In cases where you have multiple UBI volumes, ubi0 would change to the volume with the root partition.

Once **nand_root** is set:

```
U-Boot# setenv bootcmd run nand_boot
```

Environment Settings for jffs2 Filesystem

The **bootargs** variable here makes use of the **nand_root** variable. The root file system format is jffs2:

Enabling and using the jffs2 as a root file system is describe here [7]

Environment Settings for NFS Filesystem

NOTE
*When setting a MAC address please ensure that the LSB of the 1st byte if not 1 i.e. when setting the MAC address: **y** in x**y**:ab:cd:ef:gh:jk has to be an even number. For more info this refer to the wiki page http://en.wikipedia.org/wiki/MAC_address

- If you need to use a separate server for TFTP and NFS please see how to use the **next-server** option in your DHCP server.

Modify the required variables:

```
U-Boot# print ethaddr                          <-- Check if MAC address is assigned and is unique

U-Boot# setenv ethaddr <unique-MAC-address>    <-- Set only if not present already, format uv:yy:zz:aa:bb:cc

U-Boot# setenv serverip <NFS and TFTP server-ip>

U-Boot# setenv rootpath /location/of/nfsroot/export

U-Boot# setenv bootcmd net_boot
```

Booting the kernel

In case everything went well, boot the kernel using the following command.

```
U-Boot# boot
```

Archived

Sitara Linux SDK 05.07 [8]

References

[1] http://www.denx.de/wiki/U-Boot/WebHome
[2] http://processors.wiki.ti.com/index.php/Sitara_Linux_SDK_GCC_Toolchain#Updated.C2.A0Linux-Devkit_Structure
[3] http://processors.wiki.ti.com/index.php/Sitara_Linux_SDK_GCC_Toolchain#Switch_to_Linaro
[4] http://software-dl.ti.com/dsps/dsps_public_sw/am_bu/sdk/AM335xSDK/latest/index_FDS.html
[5] http://logmett.com/index.php?/products/teraterm.html
[6] http://www.ti.com/tool/tmdxevm3358
[7] http://processors.wiki.ti.com/index.php/AM335x_JFFS2_Support_Guide
[8] http://processors.wiki.ti.com/index.php?title=AM335x_U-Boot_User%27s_Guide&oldid=151545

Article Sources and Contributors

AM335x U-Boot User's Guide *Source*: http://processors.wiki.ti.com/index.php?oldid=200805 *Contributors*: Akshay.s, Cem8101, Fcooper, Gururaja, Helene Maria, Jinleileiking, Joelagnel, Rachna, SekharNori, Trini, VaibhavBedia, X0155329

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha

Image:Bootmodeswitches.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Bootmodeswitches.jpg *License*: unknown *Contributors*: Mike Tadyshak