ADS U-boot User's Manual

ADS document # 110010-40011

Applied Data Systems

www.applieddata.net

10260 Old Columbia Road
Columbia MD, 21046 USA
301-490-4007

August 11, 2005

ADS U-boot Bootloader Manual

1. Overview

This document is the bootloader user manual for ADS single board computers that run the U-boot bootloader. These boards use U-boot to boot and run the Linux operating system. If your ADS board does not run U-boot, please see the manual for your supported bootloader (either the XScale/PXA or SA1110 bootloader).

U-boot is an open-source, multi-platform bootloader, with ports for many different embedded platforms. U-boot supports interactive commands, environment variables, command scripting, and booting from external media (PCMCIA/CF, MMC). You can find more information about U-boot and download its source code from U-boot's Sourceforge home page at http://u-boot.sourceforge.net/

This is not an exhaustive manual for U-boot. If you are looking for a detailed reference manual or command index, you should consult the main U-boot manual [1] at http://www.denx.de/twiki/bin/view/DULG/Manual.

To get started, you will need a serial terminal program to view the output of U-boot. By default, U-boot sends it's output the the "debug" port on your board. The manual for your board should indicate which serial port is the debug port. Windows usually comes with a serial terminal program called "HyperTerminal"; many Linux distributions ship with either the minicom or the kermit serial terminal program installed. ADS U-boot uses a baud rate of 38400, 8 data bits and no stop bits.

2. Booting Linux on ADS boards with U-boot

This section presents some examples for booting various types of root filesystems on your ADS board. U-boot uses external media, such as compact flash or MMC cards, to load Linux images (kernels, ramdisks, etc.) into either RAM or on-board flash. External media must be formatted with a FAT12 or FAT16 filesystem to be used with U-boot. Compact flash cards can be used either in a compact flash slot or in a PCMCIA slot, with a CF-to-PCMCIA adapter. [2]

Note:
Most CF/MMC cards come formatted with supported FAT filesystems, so if you have a new card, try it before you re-format it. The FAT requirement is only for U-boot; there are many formats that work fine once Linux is running.

2.1 The ADS U-boot boot scripts

When booting from external media (CF or MMC card), you will need to place several U-boot scripts on your card to control the behavior of U-boot. The scripts ADS provides split U-boot's configuration into smaller sections, making it easier to control how your board boots.

[1] The DULG manual is targeted at a specific PowerPC board, but the majority of the document also applies to other boards
[2] If your board is equipped with both a PCMCIA and a CF slot, only the PCMCIA slot (slot 0) can be used with U-boot

- start.txt

  The main script which U-boot loads and runs when booting from external media.

- root_config.txt

  This file controls the type and location of the kernel and root filesystem. Examples of root filesystems include ext2 ramdisks loaded from external media, JFFS2 filesystems programmed into on-board flash and ext3 filesystems stored on a Microdrive or USB hard drive.

  This file also controls whether a kernel and root filesystem will be booted directly from external media or programmed into on-board flash.

- hw_config.txt

  Configures machine-specific hardware registers (e.g., for framebuffer bit depth).

  This file is optional and is only required if you need to change the default register settings.

- tty_config.txt

  Configures the console, serial TTYs and baud rate.

  This file is optional and is only required if you need to change the default serial login or console settings.

If you wish to modify the booting behavior of U-boot, other than simply altering tty or root settings, you should consider changing the value of the 'bootcmd' environment variable or editing the start.txt file to meet your needs. Consult the main U-boot manual for the commands and syntax used by U-boot.

Under most situations, the provided start.txt file should meet your needs.

3. Booting from external media

This section describes how to boot your system from external media *without* modifying the on-board flash. When booting from external media, U-boot loads a Linux kernel directly from external media into RAM instead of loading it from the on-board flash chips.

ADS systems can be booted from compact flash cards in either a compact flash slot or a PCMCIA slot with an adapter. [3] Systems with an MMC slot can also be booted from MMC cards.

When booting from external media, you have many options for your root filesystem, including ramdisks, NFS servers and filesystems on a USB hard drive or Microdrive^{TM}.

3.1 Booting a ramdisk from CF or MMC

To boot a Linux using a ramdisk as the root filesystem, follow these steps:

1. Place a kernel (zImage) and ramdisk (ramdisk.gz) on your compact flash or MMC card
2. Place the U-boot start.txt on your card
3. Create a root_config.txt on your card with the following contents:

[3] If your board is equipped with both a PCMCIA and a CF slot, only the PCMCIA slot (slot 0) can be used with U-boot

```
images_to_dram='zImage ramdisk.gz'
setenv root_args rw root=/dev/ram initrd=${INITRD},5m ramdisk_size=12288
setenv boot_linux 'bootm zImage_buffer'
```

4. Insert your card into the board and apply power to boot your system.

Note:
It is not necessary to program a zImage and ramdisk.gz to flash before booting it. This type of root filesystem can be booted directly from a CF or MMC card.

3.2 Booting a full distribution from a USB hard drive

When running the full Debian GNU/Linux distribution, your ADS board becomes a complete native development environment, with compilers, editors and debuggers. You also have access to the Debian package system "apt", which provides thousands of pre-compiled software packages ready to be installed on your ADS board.

Since U-boot cannot actually boot (load a kernel, etc.) from a USB storage device, you must prepare a CF or MMC card to boot from as well as a USB hard drive for the root filesystem.

Partition you USB hard drive as follows:

1. Linux swap - Linux will use this partition as virtual memory. A size of 128 to 256 MB is recommended.
2. Linux EXT3 - This is the root filesystem. You can use the remaining space on your USB hard drive for this partition, but very large partition sizes will take a long time to check if a filesystem check is required.

Place the zImage and the U-boot scripts on your external media and set the following environment variables in root_config.txt:

```
images_to_dram='zImage'
setenv root_args root=/dev/sda2 rootfstype=ext3 noinitrd rootdelay=5
setenv boot_linux 'bootm zImage_buffer'
```

Since a USB hard drive requires no initial ramdisk (initrd) to be used as a root filesystem, only the kernel (zImage) is listed in images_to_flash.

Note:
If your USB hard disk takes a longer time to become ready for Linux to use it, it may be necessary to increase the 'rootdelay' parameter to give your hard drive more time to settle before Linux mounts it.

Note:
In this example, since the mtd_args have not been set in the root_config.txt, the default MTD partitioning will be used.

3.3 Booting a full distribution from a Microdrive

Microdrives are small rotating-disk hard drives in a compact flash form-factor. Microdrives offer a large amount of storage (up to 4GB at the time of writing) at a reasonable cost and can be used to run a full

GNU/Linux environment on your ADS board. Since they employ a rotating-disk hard drive instead of flash memory, they are also suitable for repetitive use patterns such as software compilation and swap partitions (virtual memory). [4]

Since a Microdrive is seen by Linux as a PCMCIA device, you must use an *initial ramdisk* or *initrd* to load the PCMCIA card manager (cardmgr) before mounting the root filesystem. Such an initrd is provided with the Full/Traditional Debian root filesystem for ADS boards.

Partition your Microdrive as follows:

1. FAT12 or FAT16 bootloader partition - U-boot will read the kernel and other files from this partition. About 10 MB will be needed.
2. Linux swap - Linux will use this partition as virtual memory. A size of 64 to 128 MB is recommended.
3. EXT3 - This is the root filesystem and should use the remaining space on the Microdrive.

Note:
ADS provides a Microdrive installer script with the full ADS Debian distribution tarball that will properly partition and format a Microdrive. If you use this script, you will only need to copy the U-boot scripts onto the first partition.

Place the file initrd.gz, along with the zImage and the U-boot scripts, on the first (FAT) partition of your Microdrive.

Place the following in your root_config.txt:

```
images_to_dram='zImage initrd.gz'
setenv root_args root=/dev/hda3 initrd=${INITRD},3m ramdisk_size=6500
setenv boot_linux 'bootm zImage_buffer'
```

Note:
When using an EXT2 initrd with a 2.6 kernel, do not include a "rootfstype=" parameter, as it will confuse the kernel if the initrd and the root filesystem types are different.

3.4 Booting a full distribution from an NFS server

If you have an NFS server available, you can run the full Debian GNU/Linux distribution by mounting your NFS server as the root filesystem. This will give you all the benefits of running a full distribution without requiring a USB hard drive or Microdrive.

Place the following in your root_config.txt:

```
images_to_dram='zImage'
setenv root_args root=/dev/nfs nfsroot=NFS_server_IP:/path/to/root ip=dhcp noinitrd
setenv boot_linux 'bootm zImage_buffer'
```

Note:
When setting up an NFS server for any diskless system (such as this), you must export the filesystem with the 'no_root_squash' option. Since this option is a potential security risk to the NFS server, you should take appropriate security precautions on your network.

[4] Flash memory cells have a limited number of write/erase cycles and are not suitable for use as swap space

3.5 Booting a kernel from external media using flash as the root filesystem

You can use an existing flash filesystem (a JFFS2, CRAMFS, etc. that you have already programmed into on-board flash) as your root even when loading the kernel from external media. This is especially useful for testing a new kernel as you do not need to reprogram your flash.

To boot your board in this fashion, use the same root_config.txt that you used when programming the flash filesystem, [5] but replace the images_to_flash line with the line:

```
images_to_dram='zImage'
```

It is not necessary to have the flash filesystem image on the external media. Only the kernel and the U-boot files are needed.

4. Programming a kernel and root filesystem into on-board flash

4.1 How installation works

U-boot installs a kernel and root filesystem into on-board flash by erasing the flash partition and writing the image into the flash chips. The flash programming process is driven by the same start.txt file used to boot from external media. Once your board's flash has been programmed with a kernel and root filesystem, your board will boot that system whenever it is booted without external media.

Note:
After programming your flash, you should immediately remove the external media and delete or rename the start.txt file so you don't accidentally reprogram your system later on.

4.2 Programming a kernel and ramdisk

When your ADS board has a kernel and ramdisk programmed into on-board flash, U-boot boots the system by loading the kernel zImage and the *compressed* ramdisk into system memory (RAM) and executing the Linux kernel. Linux then uncompresses the ramdisk, frees the memory consumed by the compressed copy and mounts the ramdisk as the root filesystem.

To program a kernel and ramdisk into on-board flash, follow these steps:

1. Place a kernel (zImage) and ramdisk (ramdisk.gz) on your CF or MMC card.
2. Place the U-boot start.txt on your card
3. Create a root_config.txt on your card with the following contents:

[5] See section 4

```
images_to_flash='zImage ramdisk.gz'
setenv root_args root=/dev/ram initrd=${INITRD},5m ramdisk_size=12288 rootfstype=ext2
setenv mtd_args mtdparts=flash0:${FLBLOCKSZ}(boot),${FLBLOCKSZ}(bootvars),2048k(zImage),5m(ramdisk.gz),-(flashfs1)
```

4. Insert your card into the board and apply power to boot your system.

Note:
The final partition, flashfs1, is not programmed with any image in this example. You can use this partition from within Linux, however, by erasing it and mounting it as jffs2, as in this example:

```
eraseall /dev/mtd4
mount -t jffs2 /dev/mtdblock4 /mnt/flash
```

The mount command or the first write to the filesystem may take a few minutes to complete as the filesystem is created/checked.

4.3 Programming a kernel and JFFS2 image

JFFS2 is the second version of the journaling flash filesystem for Linux. JFFS2 filesystems are accessed directly from the on-board flash and do not consume large amounts of memory (DRAM) like ramdisks. JFFS2 filesystems are writable and persistent allowing you to make permanent changes to the filesystem stored in your on-board flash.

To program a kernel and JFFS2 filesystem into on-board flash, follow these steps:

1. Place a kernel (zImage) and JFFS2 image (flashfs1) on your CF or MMC card
2. Place the U-boot start.txt on your card
3. Create a root_config.txt on your card with the following contents:

```
images_to_flash='zImage flashfs1'
setenv root_args root=/dev/mtdblock3 noinitrd rootfstype=jffs2
setenv mtd_args mtdparts=flash0:${FLBLOCKSZ}(boot),${FLBLOCKSZ}(bootvars),2048k(zImage),-(flashfs1)
```

4. Insert your card into the board and apply power to boot your system.

4.4 Programming a kernel and CRAMFS image

CRAMFS is a compressed read-only filesystem for flash. Like JFFS2, it is accessed directly from the on-board flash, but the contents of a CRAMFS filesystem cannot be modified. This is a useful feature, however, if you need to be absolutely sure that your filesystem cannot be corrupted or altered.

Since CRAMFS is read-only (unlike ramdisks and JFFS2), you will need to combine it with either JFFS2 or a ramdisk to provide space for programs/users to write files. For example, the ADS Debian CRAMFS distribution also uses a JFFS2 image for persistent storage. You will have problems if you try to use a completely read-only filesystem.

To program a kernel, a CRAMFS filesystem and a JFFS2 filesystem into on-board flash, follow these steps:

1. Place a kernel (zImage), CRAMFS image (cramfs.img) and JFFS2 image (flashfs2) on your CF or MMC card
2. Place the U-boot start.txt on your card
3. Create a root_config.txt on your card with the following contents:

```
images_to_flash='zImage cramfs.img flashfs2'
setenv root_args rw root=/dev/mtdblock3 noinitrd rootfstype=cramfs
setenv mtd_args mtdparts=flash0:${FLBLOCKSZ}(boot),${FLBLOCKSZ}(bootvars),2048k(zImage),14848k(cramfs.img),-(flashfs2
```

4. Insert your card into the board and apply power to boot your system.

5. Important concepts in U-boot

5.1 Using environment variables

U-boot stores its collection of settings commands and scripts in "environment variables". These environment variables are stored in the second flash partition, named 'bootvars'.

On boot, U-boot automatically attempts to load the environment variables from flash. If U-boot cannot load it's environment variables (if the partition is corrupt or erased, for instance), you will see the following error message:

```
*** Warning - bad CRC, using default environment
```

and U-boot will fall back to it's compiled-in default environment. To prevent this message from occurring again, type `saveenv` to save the environment into flash.

5.2 Restoring the default environment variables

If you need to restore U-boot's default environment, you can remove all of the saved environment variables by typing:

```
run eraseenv
```

at the U-boot prompt and resetting your board. When U-boot restarts, you will see the above error message and U-boot will use it's compiled-in defaults. Again, you should save these defaults to on-board flash by typing:

```
saveenv
```

after restarting U-boot.

Warning:
Restoring the default environment will destroy any changes made to the environment variables, including configuration information needed to boot filesystems installed in on-board flash. You should capture the output of the `printenv` command to back-up your environment if you make changes.

5.3 Scripts versus commands

Being an interactive bootloader, U-boot has several built-in commands that you can run from U-boot's command prompt. [6] You can see a list of U-boot commands by typing `help` at the command prompt. More detailed information about built-in U-boot commands can be found in the official U-boot manual. Built-in commands are run by simply typing the name of the command at the prompt.

In addition to the built-in commands, ADS U-boot also supports scripts, which are complex collections of commands stored in environment variables or text files on external media. [7] For more information on using scripts, see chapter 9 of this manual.

6. Boot and install scripts

Boot scripts are files that are loaded by U-boot from CF or MMC cards. They contain U-boot commands to set variables, perform operations and execute other scripts to control U-boot's behavior when it starts up. Refer to the main U-boot manual for information about U-boot commands.

6.1 start.txt

On boot, ADS U-boot looks for the file start.txt [8] on external media (CF or MMC, if present on your board) and, if found, loads and executes it. Start.txt instructs U-boot to look for several other files which you can modify to control how your ADS board boots Linux (root filesystems, hardware registers, etc.) Section 6.5 describes U-boot's behavior without the start.txt file.

Below is a list of the files start.txt uses to boot your system. In most circumstances, changes to these files should be all that is necessary to make your board boot the way you want it to. You should not need to make any changes to start.txt for typical operation.

6.2 root_config.txt

The root_config.txt file is used to tell U-boot and Linux about the layout of your root filesystem. The configuration in root_config.txt is also used to build the Linux kernel command line.

[6] U-boot presents an interactive command prompt if autoboot is interrupted during the 3-second timer at startup or if autoboot fails for any reason.
[7] This is an ADS U-boot extension. See section 9.4
[8] Actually, it runs the 'bootcmd' environment variable which, by default, executes the 'start' environment variable.

Environment variables used in root_config.txt:

images_to_flash Space-separated list of files to be programmed into on-board flash. The file names should correspond to MTD partition names in the mtd_args section.

```
images_to_flash='zImage flashfs1'
```

images_to_dram Space-separated list of files to be loaded into RAM.

```
images_to_dram='zImage ramdisk.gz'
```

mtd_args Part of the kernel command line where MTD partitions are defined.

```
setenv mtd_args mtdparts=flash0:${FLBLOCKSZ}(boot),${FLBLOCKSZ}(bootvars),2048k(zImage),-(flashfs1)
```

boot_linux This is the command U-boot should run to finish booting linux. In most cases, this command should copy the kernel into RAM and run it.

```
setenv boot_linux cp zImage zImage_buffer; bootm zImage_buffer
```

If you have already copied a kernel into the zImage_buffer (e.g., from external media), then 'boot_linux' only needs to run the loaded kernel:

```
setenv boot_linux 'bootm zImage_buffer'
```

Note:
When a kernel and root filesystem are programmed to on-board flash using the 'images_to_flash' method, the environment variables are automatically saved to on-board flash, as well.

6.3 tty_config.txt

Environment variables used in tty_config.txt:

tty_args Part of the kernel command line where the console and serial ttys are configured

```
setenv tty_args console=ttyS0,38400 SERIALGETTY=ttyS0 SERIALBAUD=38400
```

The 'console' parameter determines where the Linux kernel messages will be sent. The SERIALGETTY and SERIALBAUD parameters are variables used by the init process to configure the serial login terminals once Linux is running.

You can also specify a normal keyboard/LCD tty as the console by setting

```
console=tty1,38400
```

Note:
Some hardware will require a different tty name (ttyAM0, ttyS1) to function properly.

6.4 hw_config.txt

The hw_config.txt file is used to configure hardware registers prior to booting Linux. This is most often used for setting up the framebuffer registers. These settings will be specific to the type of board you are using. See chapter 7 for more details.

Note:
The kernels for ADS Linux systems contain compiled-in default hardware settings that will be used if hw_config.txt is not present when booting. These settings usually specify a 640x480 8-bit display for a LQ64D343 LCD display.

6.5 U-boot's behavior without external media

If U-boot does not find a CF or MMC card on boot (or doesn't find the start.txt file on any inserted CF or MMC cards) U-boot's behavior is controlled solely by the environment variables saved in the bootvars flash partition. [9]

If you have programmed a kernel and root filesystem into on-board flash using the procedures outlined in section 4, U-boot's environment variables will be set to automatically boot that system after the three-second delay. If U-boot is not interrupted by activity on the debug port, Linux will boot automatically and you will not receive a U-boot command prompt.

If you wish to alter the default behavior of U-boot, you should use the `setenv` command to change the environment variables. Once you have set the environment variables the way you want them, use the `saveenv` command to store your current environment into flash.

If U-boot is not set up to boot from on-board flash, or if U-boot's environment has been erased, U-boot will abort automatic booting and present its command prompt instead. U-boot will also abort if is set up to boot a kernel from on-board flash, but flash does not contain a valid kernel.

If U-boot is set up to boot from on-board flash, but on-board flash does not contain a valid root filesystem, Linux will give a kernel panic with an error message like this:

```
Unable to mount root on <device>
Kernel panic - not syncing:  VFS: Unable to mount root fs on <device>
```

You will need to re-program your kernel and root filesystem or change the environment variables to point to a proper root filesystem in order to boot Linux properly.

7. Register settings and hw_config

U-boot uses the commands given in hw_config.txt to set up machine-specific register settings prior to booting Linux. This script replaces the functionality of the register.txt files from our legacy PXA bootloader.

For each register setting, a command is created that uses the U-boot command `mw` to set a word at a particular address to a specific value. Since the commands are stored in environment variables, they can be

[9] If your bootvars partition is erased, the compiled-in defaults will be used for the environment.

given meaningful variable names. The command 'hw_config' is set to run all of the commands for configuring the hardware registers. If there are too many individual `mw` commands, [10] you will need to add subcommands, as shown in the example below. U-boot will execute the command 'hw_config' prior to loading and running Linux.

Example lines for hw_config.txt:

```
setenv message echo hw_config.txt example only - do not use
setenv vidclkdiv mw 0x80930084 0x0000c205
setenv pixelmode mw 0x80030054 0x0000140c
setenv scrnlines mw 0x80030030 0x000001E0
setenv vlinestep mw 0x80030038 0x00000140
setenv hclktotal mw 0x80030010 0x0000031f
setenv hactstrtstop mw 0x80030018 0x001F029F
#  too many to fit on a single command/line
setenv hw1 run message vidclkdiv pixelmode scrnlines vlinestep
setenv hw2 run hclktotal hactstrtstop
#  run all of them
setenv hw_config run hw1 hw2
```

Using this format for hw_config.txt allows you to easily split up the file into multiple lines and keep the file easy to read.

8. Advanced U-boot configuration

8.1 How to reprogram (upgrade) U-boot

The default U-boot environment shipped with ADS systems contains several scripts that can be used to re-program the U-boot bootloader to a new version. To re-program U-boot, follow these steps:

1. Place the new u-boot.bin file on a CF or MMC card.

   Warning:
   The u-boot.bin file *MUST* contain a valid version of U-boot for your ADS board. If u-boot.bin is not a valid bootloader or contains bugs, using it can render your board inoperable. If you have downloaded an updated u-boot.bin from the ADS support web site, double-check to be sure you have downloaded the proper file for your model board. Also, you should use md5 to verify that the file downloaded correctly.

2. Insert the card into your board, connect a serial terminal and boot your system.

3. After applying power (but before the three-second timer expires), press the Enter key to receive the U-boot command prompt.

4. If you are using a CF card (either in a CF slot or a PCMCIA slot), type the command:

   ```
   run p_reprogram_uboot
   ```

   If you are using an MMC card, type the command

   ```
   run mmc_reprogram_uboot
   ```

[10] U-boot allows a maximum of 16 arguments per command

Warning:
Do not remove power from your board while U-boot is being programmed. Power loss during a bootloader update may render your board unbootable. If this happens you may have to return the board to ADS for reprogramming.

5. Once programming has completed, you will see the following message:

```
Copy to Flash...  done
```

   You should now type `reset` to boot the newly-installed U-boot.

8.2 How to make U-boot secure

To make U-boot more secure, there are several changes you can make to the environment variables to disable booting from external media and interactive booting.

To prevent U-boot from allowing activity on the serial console from interrupting the automatic boot process, remove the autoboot delay by running:

```
setenv bootdelay 0
```

Warning:
This will prevent you from getting a U-boot prompt. It will be necessary to erase U-boot's environment from within Linux to get a U-boot prompt.

To prevent U-boot from reading external media at startup, you should change the 'start' environment variable to boot your system only from on-board flash. The 'start' variable stores the commands U-boot runs during automatic boot. To set a minimal automatic boot command, type the following at the U-boot command prompt:

```
setenv start 'test -n ${hw_config} && run hw_config; test -n ${boot_linux} && run boot_linux'
```

This start command would only read the 'hw_config' and 'boot_linux' commands from the saved environment and not from external media. These commands would also need to be stored in the environment.

Note:
You must run the saveenv command to save your environment changes to flash or they will be lost when you reboot.

8.3 How to make U-boot quiet

Setting the environment variable 'silent' to any value (i.e., having this variable present) will prevent U-boot from sending any output to the debug port:

```
setenv silent 1
saveenv
```

When silent mode is enabled, you can still get a U-boot command prompt by pressing Enter during the autoboot countdown after applying power to the board. The countdown will not be visible but the system still gives you a chance to interrupt the boot process.

To disable silent mode, clear the 'silent' variable by typing:

```
setenv silent
saveenv
```

When the silent variable is set, U-boot will set the Linux console to null (empty, console=) on the command line automatically. If you wish to silence the serial gettys, as well, you should disable (comment out) the serial login gettys in the file /etc/inittab.

9. ADS-specific features and changes

9.1 Raw zImage kernels

Normally, U-boot expects to boot kernels from a special uzImage, which encapsulates the Linux kernel into a U-boot -specific format. The ADS U-boot, however, also accepts raw zImage files, as produced by a make zImage command.

9.2 Raw ASCII scripts

ADS U-boot provides the ability to run U-boot scripts in memory without having to encapsulate the script inside a uImage (using the mkimage tool is not necessary). This functionality allows the execute_txt_file script to load and execute scripts stored as ASCII text files on external media.

9.3 Setting the ethernet MAC address

On supported boards, the U-boot command `macaddr` can be used to set the ethernet MAC address. The MAC address is a 6-byte hexadecimal number that uniquely identifies each endpoint on an ethernet network/LAN. You can view your board's MAC address by typing either:

```
macaddr
```

(with no arguments) or:

```
printenv ethaddr
```

To set the MAC address, provide the address as an argument to the `macaddr` command, as in the following example:

```
$ macaddr 00:60:0c:aa:bb:cc
```

The MAC address can be entered with or without the colons between the bytes. No other type of separator is allowed.

Note:
You board should come from the factory pre-programmed with a valid, unique MAC address. You should not change your board's MAC address unless you have a valid reason for doing so.

9.4 Running scripts from text files

The default U-boot environment contains an ADS-supplied script that allows you to easily run a U-boot script that is stored as a text file on a CF or MMC card. To run a U-boot script from a text file, specify the name of the script by typing:

```
txt_file=my_script.txt
```

then run the script by typing

```
run execute_txt_file
```

You can also use this technique from within another text file script to call one script from another.

Note:
If files with the same name exist on both a CF and MMC card, the CF card will take precedence. (Use `printenv` to observe the 'execute_txt_file' variable)

9.5 Map commands

The commands `mapinfo` , `mapadd` , and `mapdel` , can be used to maintain a human-readable memory map. This map allows you to name various memory ranges, including flash regions (partitions) and memory regions (buffers).

The output of the `mapinfo` command will list all of the default regions:

```
$ mapinfo
 Name             Location   Size       Type      Source    Guard
 ----             --------   ----       ----      ------    -----
 boot             0x00000000 0x00040000 Flash     bootargs  Off
 bootvars         0x00040000 0x00040000 Flash     bootargs  Off
 zImage           0x00080000 0x00180000 Flash     bootargs  Off
 flashfs1         0x00200000 0x01e00000 Flash     bootargs  Off
 script_buffer    0xa0000800 0x00007800 DRAM      Initial   Off
 zImage_buffer    0xa0008000 0x00180000 DRAM      Initial   Off
 scratch_buffer   0xa0600000 0x01000000 DRAM      Initial   Off
```

These regions are used by many of the ADS-supplied scripts and default commands. The regions of type "Flash" correspond to MTD partitions in on-board flash, and are named according to the partition names. The regions of the "DRAM" are sections in memory and are used to hold scripts, kernel images, ramdisks and flash images that are loaded by U-boot.

The `mapdel` command deletes a named region from the map table. The only argument is the name of the region to delete.

The `mapadd` command will add a named region in either DRAM or flash (depending on the address) to the map table. The first argument is the name of the region. The second argument is the beginning address of the region. The third argument is the size of the region. The fourth argument is optional and determines whether or not U-boot will prevent writes to the region.

To create a new buffer called "foo" at address 0xa0600000, with a size of 16MiB (16777216 bytes), enter the command:

```
mapadd foo 0xa0600000 0x01000000
```

Note:
U-boot interprets most numerical values as hexadecimal values even if a leading '0x' is not supplied. A notable exception to this rule is the 'baudrate' value, which is interpreted as decimal.

9.6 ADS-supplied boot/install scripts

The files start.txt, hw_config.txt, root_config.txt, tty_config.txt are boot scripts that ADS has provided to make it easier to control the behaviour of U-boot. These scripts break down the boot process into steps and allow for simpler individual files.