DIMM RZ/A1H Linux BSP

Software Manual

Rev01 / 31.07.2014

emtrion GmbH

Revision: **01 / 31.07.2014**

Rev	Date/Signature	Changes
1	30.07.14/sk	--

1 Table of contents

2 Introduction

Welcome to emtrions RZ Linux board support package. This short manual will give you a startup with our BSP. It will describe how to setup the build environment and how to build your own image. Further it will present how to boot your own image and how to program it on the device.

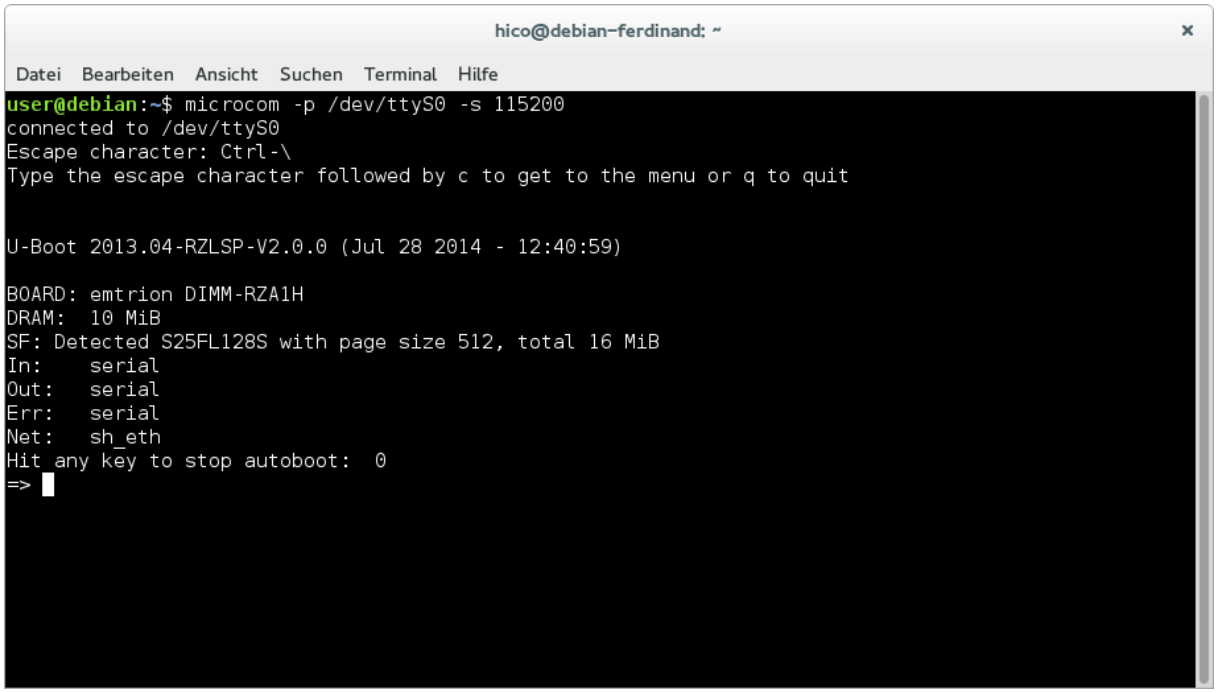
It is assumed that users of emtrion Linux developer kits are already familiar with Linux. General Linux and programming knowledge are out of the scope of this document. emtrion will gladly assist you in acquiring this knowledge. If you are interested in training courses or getting support, please contact the emtrion sales department.

This guide will show you how to do the start up of the developer kit and setup Yocto to build a Root File-System.

3 Device Start Up

Connect the developer kit to the serial port attached to the virtual machine and to your network. Now you can open a serial terminal application of your choice using the following **Serial Communication Settings:**

- **Baudrate: 115,2k**
- **Data bits: 8**
- **Parity: none**
- **Stop bits: 1**
- **Flow control: none**



1: Serial terminal showing U-Boot prompt

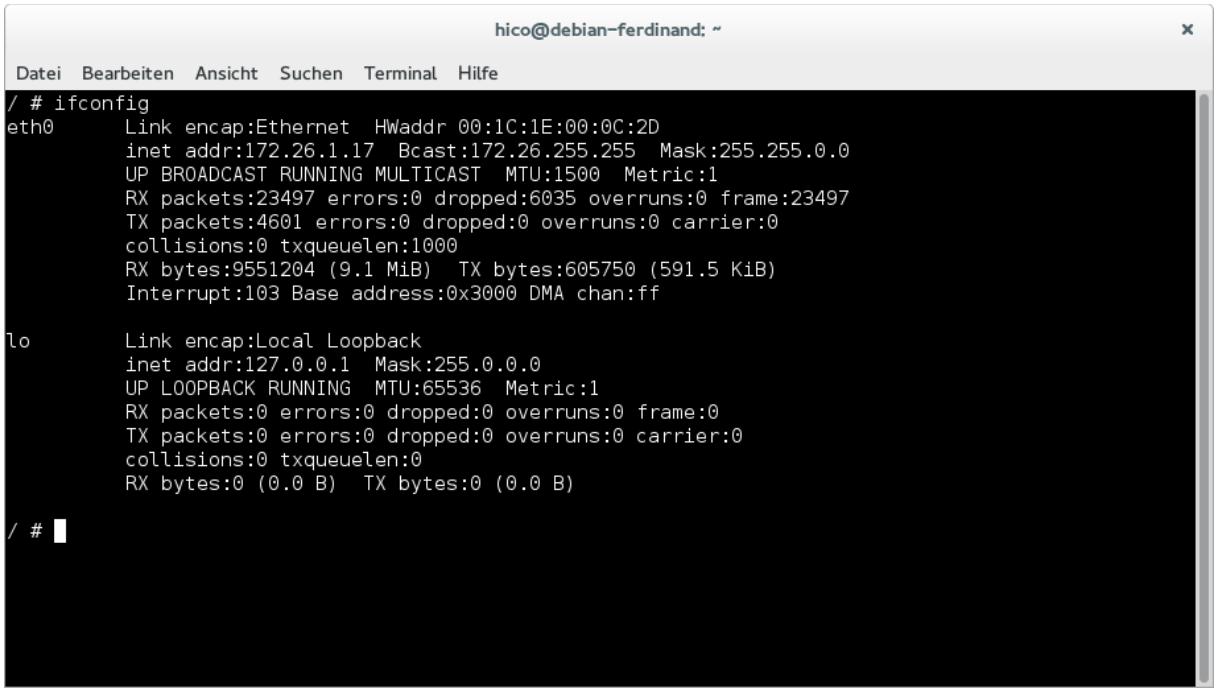
You may now power on the developer kit. You should see it booting U-Boot and Linux from Flash.

3.1 Device Network Setup

If your network offers a DHCP server the developer kit should have already obtained an IP-Address. If not you can trigger the use of it by using following command in a terminal:

```
/ # udhcpc eth0
```

You can check the result with the command **ifconfig**.



2: ifconfig output

If the setup is not correct or DHCP is not available you have to manually configure the network interface. This can be down by editing /etc/network/interfaces. Replace

```
iface eth0 inet dhcp
```

by

```
iface eth0 inet static
   address 10.1.2.3
   netmask 255.255.255.0
   gateway 10.1.1.1
```

And bring up the interface using

```
/ # ifup eth0
```

4 Setting up the Yocto build environment

To build the root file system for the module emtrion uses a Yocto/OpenEmbedded build environment . For general information about Yocto/OpenEmbedded please have a look at the following official resources:

- https://www.yoctoproject.org/documentation
- http://www.openembedded.org/wiki/OpenEmbedded-Core

emtrion delivers a ZIP-Archive including all necessary files to setup a Yocto/OpenEmbedded build environment on your machine. The following sections will describe how to setup the Yocto-build environment using the provided ZIP-Archive. These installation instructions assume that your are working on a Debian Machine, but the installation is analogously possible on other Linux-Distributions.

First all dependencies for the Yocto Project should be installed. On Debian

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo \
    gcc-multilib build-essential chrpath libsdl1.2-dev xterm
```

will install the dependencies. For other distributions have a look at http://www.openembedded.org/wiki/Getting_started.

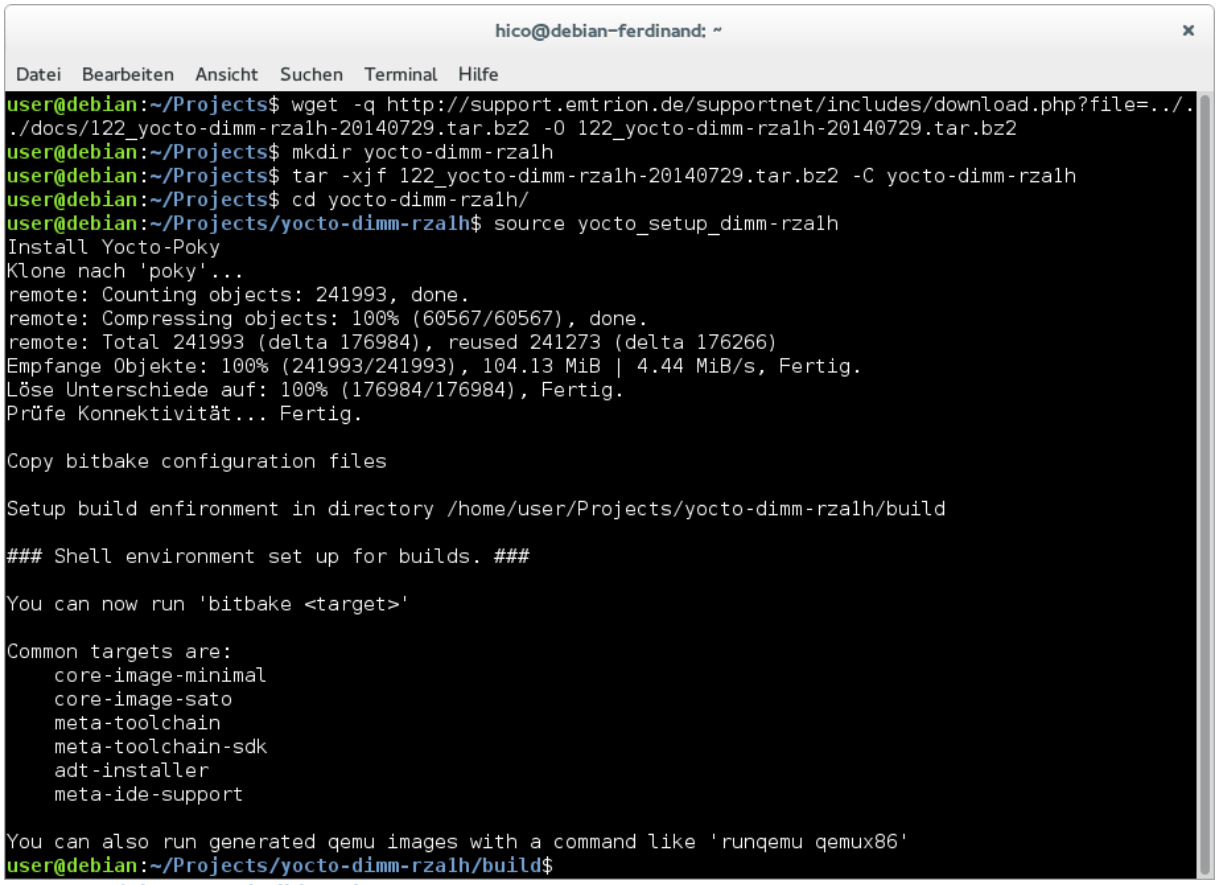
Then Download the latest Version of the Yocto-Layer from emtrions Support Website: http://support.emtrion.de/supportnet/index.php?page=software&setCPU=53 and unpack it to a directory of your choice.

To setup the build environment open a terminal and change path to the unpacked BSP. Then source the included setup script.

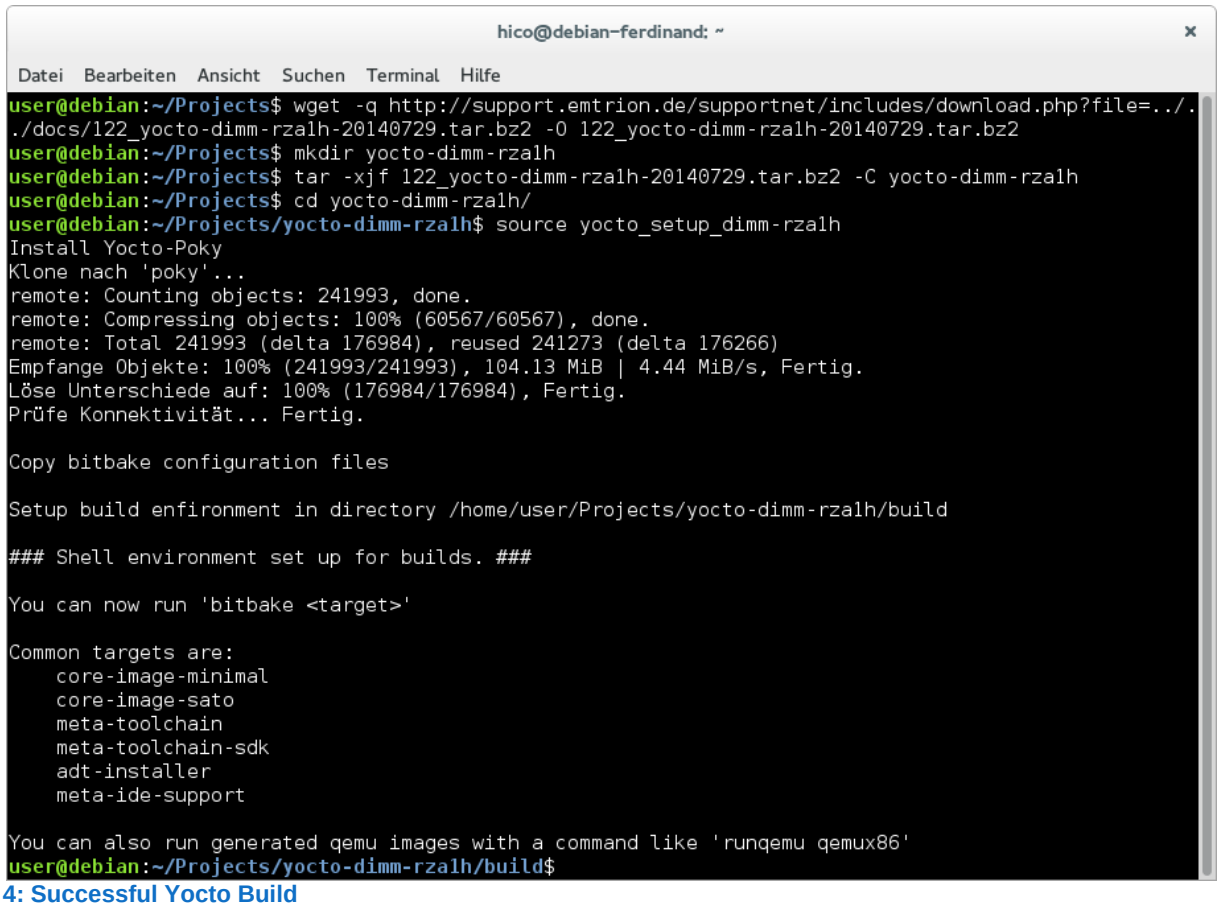
```
cd <YOUR_DIRECTORY>
source yocto_setup_dimm-rza1h
```

This will download the required Yocto Packages and checkout the correct version. Furthermore it will copy some configuration files and setup the build environment. To build the image shipped with your kit just run

```
bitbake core-image-minimal
```



3: Setup of the Yocto build environment



4: Successful Yocto Build

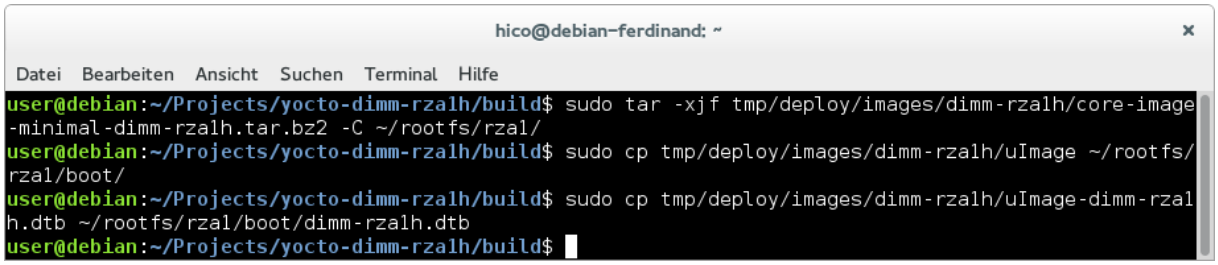
5 How to use a Network File System (NFS) as Root File System

For development purposes it is convenient to boot via network using NFS. The root file system of the device is then a subdirectory in your development machine. For detailed information about NFS setup in Debian you can read this wiki page https://wiki.debian.org/NFSServerSetup .

5.1 Export Yocto Root File System

To be able to boot your own root file system, you have to copy the root file system, kernel and device tree to the exported folder. The following commands show how to setup the export dir after a successful Yocto build:

```
sudo tar -xjf <YOUR_BUILD_DIRECTORY>/tmp/deploy/images/dimm-rza1h/core-image-minimal-dimm-rza1h.tar.bz2 -C <EXPORTED_DIRECTORY>
sudo cp <YOUR_BUILD_DIRECTORY>/tmp/deploy/images/dimm-rza1h/uImage <EXPORTED_DIRECTORY>/boot/
sudo cp <YOUR_BUILD_DIRECTORY>/tmp/deploy/images/dimm-rza1h/uImage-dimm-rza1h.dtb <EXPORTED_DIRECTORY>/boot/dimm-rza1h.dtb
```



5: Setup of the Yocto build environment

You can download a current root file system as a tar ball from our support pages (www.support.emtrion.de). This tar ball can be extracted to the exported directory and then you can boot from it.

The exact boot loader commands can be found in the chapter Booting.

6 U-Boot

The basic task of U-Boot is to load the operating system from bulk memory into RAM and then start the kernel. You can also use it to initiate an update of the kernel, the root file system and of U-Boot itself. Furthermore you can configure directly from the medium the operating system is to be booted from, for example MMC, NFS or a serial terminal.

6.1 Basic U-Boot operation

To work with U-Boot, first use a terminal program to connect to the serial line of the board. As soon as the U-Boot prompt appears in the terminal, U-Boot is ready to receive commands. To interrupt the automatic boot mechanism press a keyboard key while using the terminal. The general U-Boot documentation can be found here: http://www.denx.de/wiki/U-Boot/Documentation

U-Boot has a set of environment variables which are used to store information needed for booting the operating system. Variables can contain information such as IP addresses, but they can also contain a whole script of actions to perform sequentially. The following commands explain the basic handling of environment variables:

U-Boot command	Explanation
printenv [variable]	This shows the value of the specified variable. If no variable is specified, the whole environment is shown.
setenv [variable] [value]	Set a variable to a specific value. If no value is specified, the variable gets deleted.
saveenv	Make your changes permanent, so they remain after power off or reboot.

1: U-Boot Commands for handling of environment variables

6.1.1 Booting

The default boot device in U-Boot is determined by the variable “bootcmd”. If you want to set up one of the following boot options as a default you have to set “bootcmd” to the command mentioned below.

Boot from on-board flash

This is the default boot option configured when you receive the developer kit from emtrion. To start it manually simply use this command:

```
=> run flash_boot
```

Boot via network using a NFS share

Make sure you exported a directory via NFS in host machine. Of course the directory must contain an extracted Root File System for the board. You can build such a File System using Yocto as described in Section 4 "Setting up the Yocto build environment" or download a preconfigured Root File System from our support pages (http://support.emtrion.de):

```
=> setenv serverip          [ip-address of virtual machine]
=> setenv nfsroot           [path to the root file system]
# use this command if you want net_boot as defalut boot option
=> setenv bootcmd 'run net_boot'
=> saveenv
=> run net_boot
```

Now the board should boot via network using the NFS share in the virtual machine.

7 Programming the System

Generally it is possible to program the system either from U-Boot or from Linux. Since we can access many different media (such as e.g. Ethernet, USB or SD-Card) from Linux, this guide focuses on the programming from within an Linux environment.

The (write) access to the flash chip in Linux depends on the installation of mtd-uitls. If your build your root file system with the Yocto Layer provided by emtrion the mtd-utils are installed by default.

As your changing parts of the system it is strongly recommended to run the following commands only on a device booted from a NFS share. For the required commands see chapter Booting.

7.1 Partitioning of the Flash

The flash has several partitions for the different components of the software. The following table shows the partitioning:

Start (Offset)	End	Length	Linux Name	Description
0x18000000 (0x00000000)	0x18080000	0x080000	/dev/mtd0	U-Boot
0x18080000(0x00080000)	0x180c0000	0x040000	/dev/mtd1	U-Boot Environment
0x180c0000 (0x000c0000)	0x184c0000	0x400000	/dev/mtd2	Kernel
0x184c0000 (0x004c0000)	0x18500000	0x040000	/dev/mtd3	Device Tree Blob
0x18500000 (0x0050000)	END		/dev/mtd4	Root File System

2: Flash Partitioning

7.2 Update of the Root File System

To program a new root file system it is first required to format the partition for jffs2 file system. This is done using the flash_erase command with --jffs2 option. After that the jffs2 image can be written to flash using flashcp, resulting in this command sequence:

```
/ # flash_erase --jffs2 -q /dev/mtd4 0 0
/ # flashcp -v /Path/to/your/image.jffs2 /dev/mtd4
```

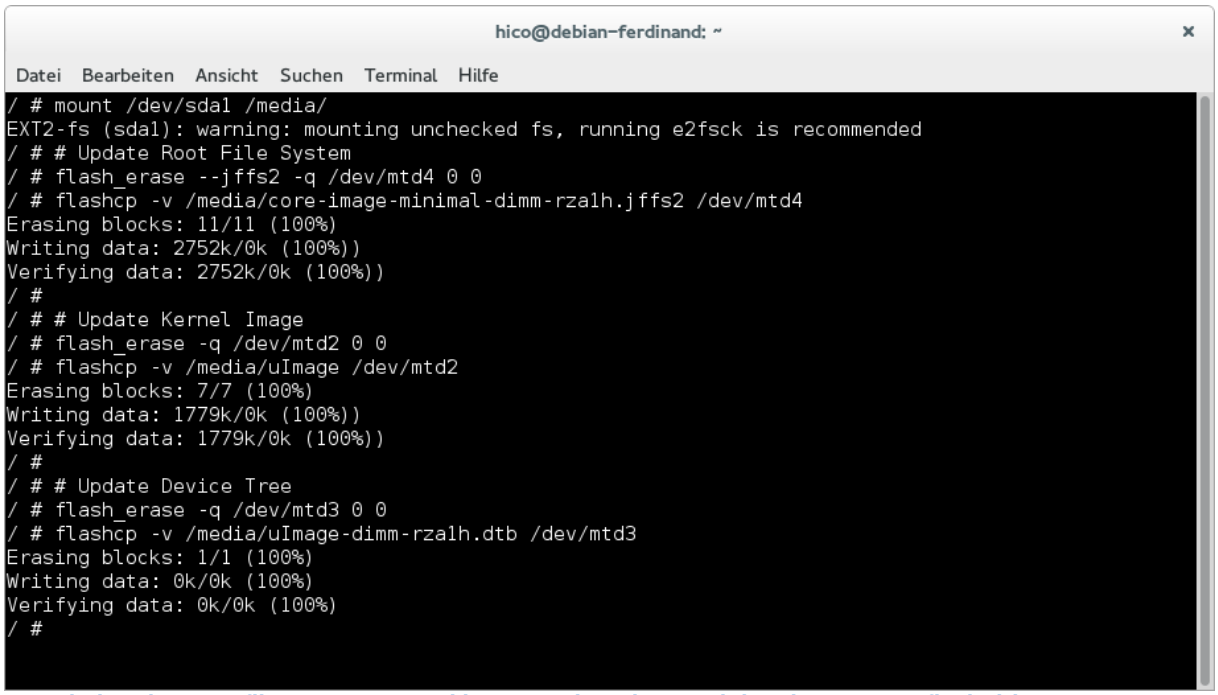
The image can be located on the mounted NFS share or any other mounted media such as a USB flash drive or a SD/MMC card.

7.3 Update of the Linux Kernel and Device Tree

To program a new Linux kernel image or a new device tree first erase the partition and then copy the updated images. The required command sequence are the following

```
Update Kernel Image
/ # flash_erase /dev/mtd2 0 0
/ # flashcp -v /Path/to/your/uImage /dev/mtd2
Update Kernel Image
/ # flash_erase /dev/mtd3 0 0
/ # flashcp -v /Path/to/your/devicetree.dtb /dev/mtd3
```

The kernel image and the device tree blob can be located on the mounted NFS share or any other mounted media such as a USB flash drive or a SD/MMC card.



6: Updating the Root file system, Kernel image and Devi Tree Blob using an USB flash drive

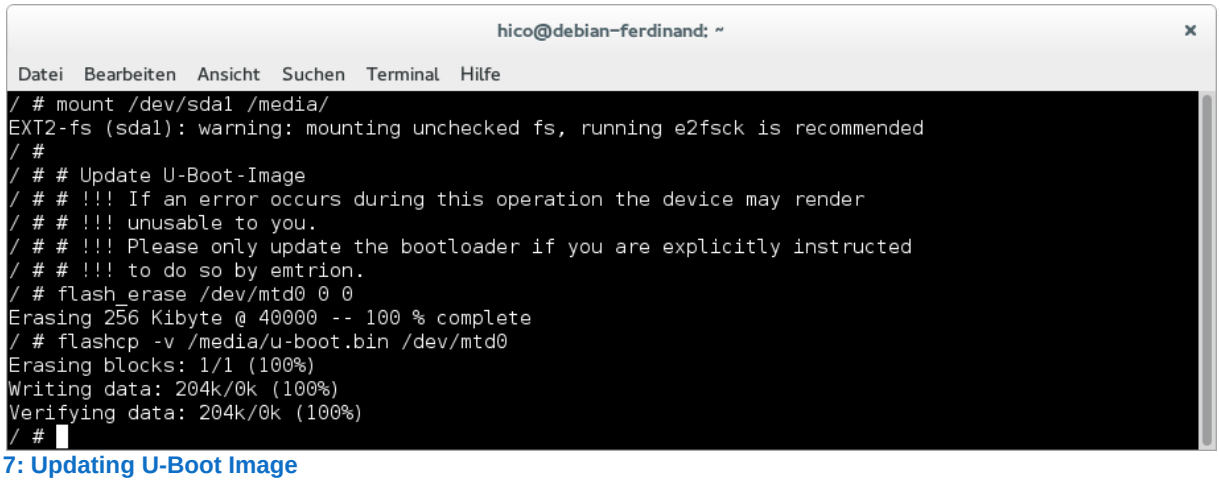
7.4 Update of U-Boot Bootloader

Attention: If the board is turned off while updating the bootloader or another error occurs, the board will be rendered unusable to you. Please only update the bootloader if you are explicitly instructed to do so by emtrion.

Erase and flash a new U-Boot image located on a mounted NFS share or any other mounted media such as a USB flash drive or a SD/MMC card using the following command sequence:

```
/ # flash_erase /dev/mtd0 0 0
/ # flashcp -v /Path/to/your/U-Boot-Image /dev/mtd0
```

To reset the U-Boot environment just erase the according flash partition using flash_erase. Doing this will also erase the assigned MAC-Address and possibly other system specific variables. Therefore backup the current environment and restore these variables!



7: Updating U-Boot Image