OpenRD-Client User Guide

Revision: v1.0

Last Update: 04-May-2009

einfochips | The Solutions People

www.einfochips.com

Contents

Figures

Revision History

#	Revision	Date	Description
1	1.0	04-May-2009	First version created

About This Manual

This manual provides an overview of the OpenRD-Client reference design based on Marvell® 88F6281 SoC. It provides details of the hardware and associated software for the OpenRD-Client.

Abbreviation and Acronyms

MTD	Memory Technology Device
POST	Power On Self Test
GbE	Gigabit Ethernet
SoC	System-on-Chip

Document Convention

Following are document convention followed:

Command	Indicates command to be executed on Linux Shell
Note	Indicates important note to be taken into consideration

Path Convention

Paths mentioned in this user guide are generated using zip packages provided on OpenRD Development Kit DVD:

<dvd-home>/openrd-devkit-v1.0/openrd_uboot/ ➔ openrd-devkit-v1.0/openrd_uboot.zip

<dvd-home>/openrd-devkit-v1.0/openrd_lsp/ ➔ openrd-devkit-v1.0/openrd_lsp.zip

<dvd-home>/openrd-devkit-v1.0/openrd_openocd/ ➔ openrd-devkit-v1.0/openrd_openocd.zip

<dvd-home>/openrd-devkit-v1.0/openrd_hardware/ ➔ openrd-devkit-v1.0/openrd_hardware.zip

<dvd-home>/openrd-devkit-v1.0/openrd_host_swsupportpackage/ ➔ openrd-devkit-v1.0/openrd_host_swsupportpackage.zip

<dvd-home>/openrd-devkit-v1.0/openrd_usbrecovery/ ➔ openrd-devkit-v1.0/openrd_usbrecovery.zip

<dvd-home>/openrd-devkit-v1.0/openrd_filesystem/ ➔ openrd-devkit-v1.0/openrd_filesystem.zip

<dvd-home>/openrd-devkit-v1.0/openrd_documentation/ ➔ openrd-devkit-v1.0/openrd_documentation.zip

<dvd-home>/openrd-devkit-v1.0/openrd_filesystem-ubuntu/ ➔ openrd-devkit-v1.0/openrd_filesystem-ubuntu.zip

1. Overview

The OpenRD-Client is powered by Marvell's Kirkwood series™ 88F6281 SoC, featuring Marvell Sheeva™ CPU technology. The OpenRD-Client is a compact, scalable, low power, RoHS compliant and fan less reference design, allowing one to leverage open source development for high end applications. The reference design is equipped with full range of interfaces available in the Kirkwood Series™ of SoCs. The key interfaces include Gigabit Ethernet, USB 2.0, SATA 2.0, eSATA, SD Memory Card/SDIO, UART, SMBus, TDM (Optional), Mini USB and JTAG port for debugging. In addition, the OpenRD-Client extends its capabilities with a 2D-GPU (Graphics Processing Unit) with a VGA Connector and Audio In/Out.

The OpenRD-Client design is targeted for wide range of applications supporting bulk storage, faster connectivity, higher throughput and performance at lower power. For example, an "Embedded Client System" solution can be readily assembled with Gigabit Ethernet connectivity, an onboard 2.5" SATA connector for hard disk and an optional enclosure.

1.1 Key Features

CPU • High performance Marvell® Sheeva™ @ 1.2 GHz • 16/16KB I/D 4 way set associative L1 caches • Unified 256KB 4-way associated L2 cache • 32 bit and 16 bit RISC architecture, Compliant with ARMv5TE • Supports both big & little endian mode • Includes MMU to support virtual memory	**On Board Memory** • 512 MB DDR2-800 SDRAM (128MB x4 chips) • 512 MB NAND Flash
Power Requirement • 12 volts power supply	**Power Monitoring** • SMBus
Storage • SD Card • SATA & eSATA Ports for HDD and Optical Storage Devices	**On Board Graphic** • VGA support up to 1280x1024 at 60 Hz • VGA Memory size of 64MB
On Board I/O • 7 USB 2.0 Port with integrated PHYs • 2 GbE Ports with GMII and RGMII support • SATA & eSATA Ports with integrated Marvell 3 Gbps SATA PHYs • UART with RS232/RS485 interface • USB interface with Debug supports JTAG & Serial Console • Audio In/Out Interface • MODBUS support • Header Connector for TDM (Optional)	**Operating System** • Linux 2.6.22.18 • Fedora Core 8 File System

1.2 OpenRD-Client Graphical Overview

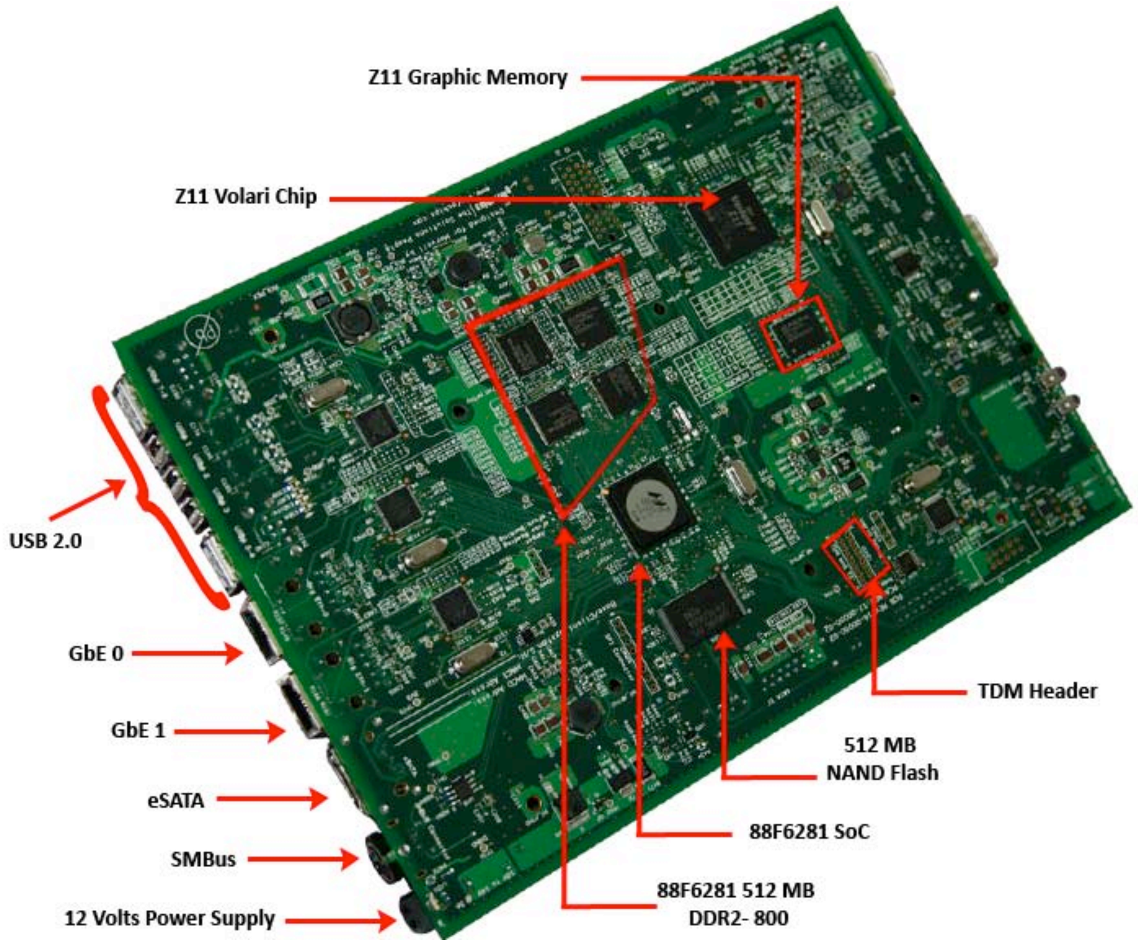


Figure 1 OpenRD-Client Top View

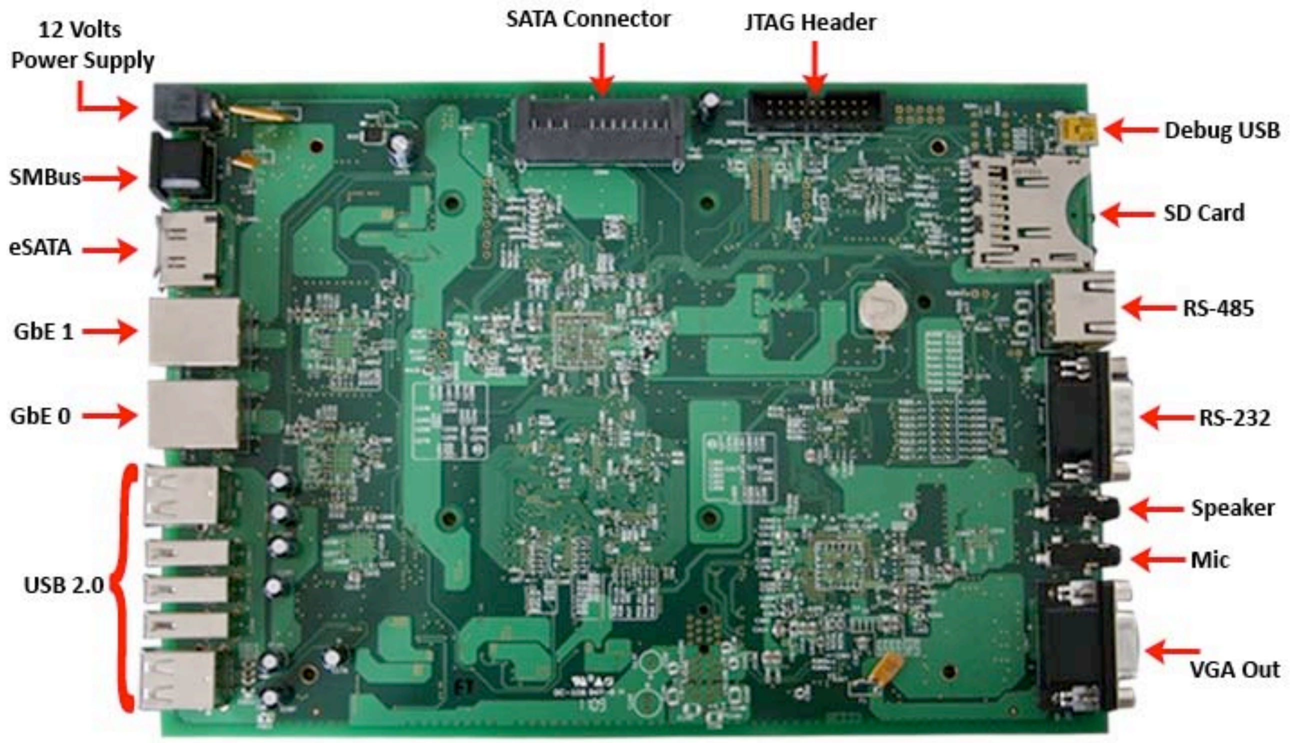


Figure 2 OpenRD-Client Bottom View

1.3 Hardware Overview

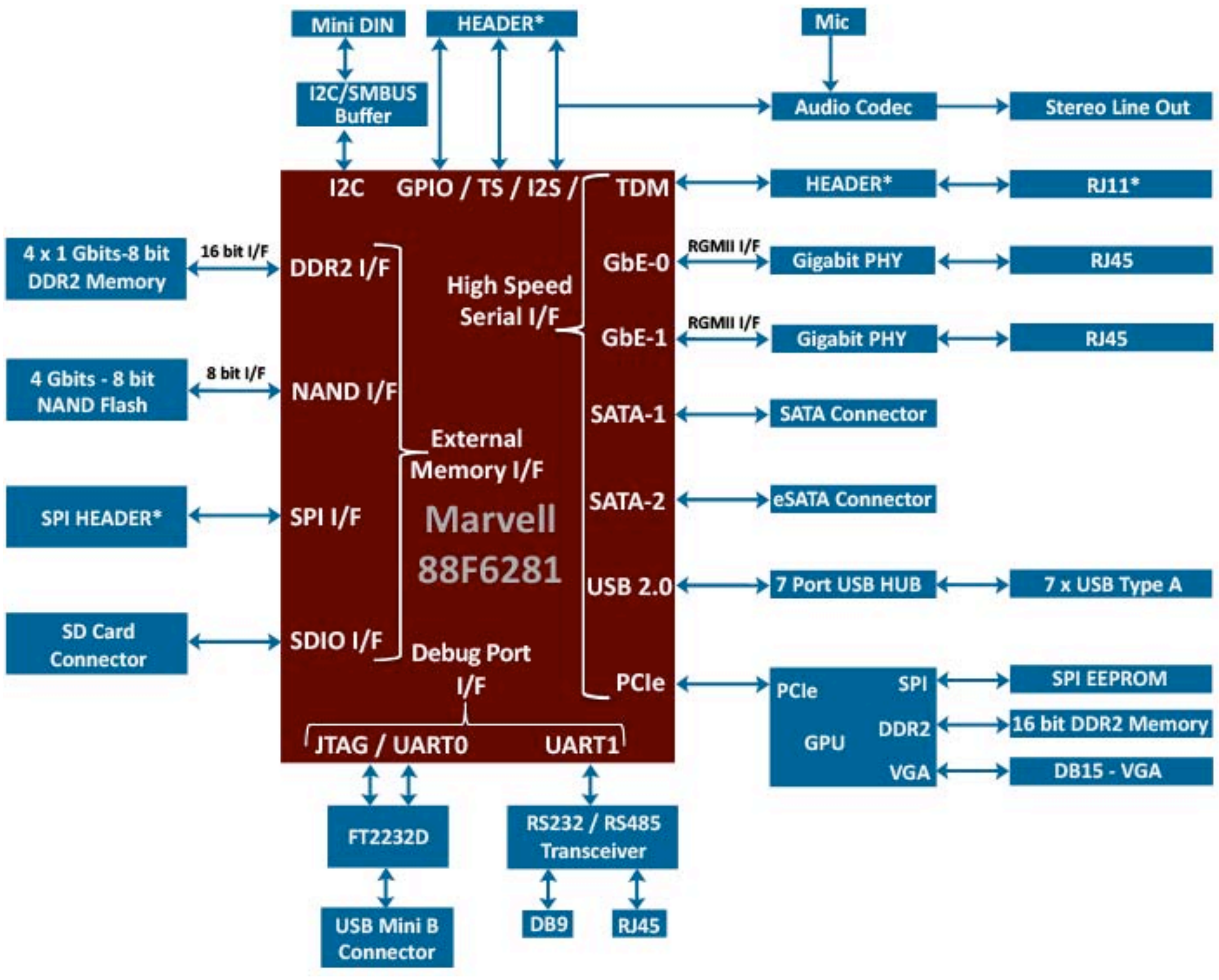


Figure 3 OpenRD-Client: Hardware Overview

*** indicates optional**

1.3.1 External Interfaces

DDR2 Memory

The 88F6281 SoC supports up to 4 banks of memory, each bank can support up to a maximum of 512MB address space.

The OpenRD-Client is designed to provide 512 MB of DRAM. The DDR2 banks are accessible through a DDR2 interface and using CS0# and CS1# chip select decoder feature of the 88F6281 DDR2 interface.

DDR2 memory interface runs at 400 MHZ clock frequency and double data rate at 800 MHz.

NAND Memory

The 88F6281 uses 4Gb density, 8 bit single NAND Flash device. The NAND flash device is accessible through glueless NAND interface controller built in the 88F6281 SoC. The interface controller can support bootstrap sequence accesses on page 0 from the device.

SD Card

The OpenRD-Client provides interface and connector for SD memory card.

USB 2.0

The 88F6281 contains a Universal Serial Bus 2.0 port, which includes an embedded USB 2.0 PHY. The SoC's USB interface can support either Host or Device mode. The OpenRD-Client has been designed to support only USB HOST mode. The OpenRD-Client provides total 7 USB 2.0 ports. Each USB 2.0 port features 480 Mbps, 12Mbps and 1.5Mbps data rate, bit stuff error detection, NRZ bit stuffing and built in FS/HS termination signaling. Each USB port can provide up to 500mA of current with maximum limit of 3500mA for 7 USB ports.

SATA & eSATA

The 88F6281 has two integrated 3 Gbps SATA ports. OpenRD-Client provides one internal SATA and one external SATA (eSATA) Port. HDD and Optical Storage Devices can be connected to these ports. These ports are ideal to provide storage interface for NAS, media servers and RAID applications.

Gigabit Ethernet (GbE)

The 88F6281 has two built-in GbE controllers that can support up to four different modes of operation. These modes are RGMII, MII, MMII and GMII. Each port is fully IEEE 802.3 compliant 10/100/1000 Mb MAC. Built in MAC of 88F6281 SoC and 88E1116 GbE PHY chip provides complete solution for Gigabit Ethernet connectivity. The OpenRD-Client supports two RGMII GbE port. These ports are ideal for network interface, ISCSI RAID storage etc.

Audio

The OpenRD-Client interfaces I2S port of 88F6281 with CS42L51-CNZ audio CODEC chip on board. The OpenRD-Client board provides two audio connectors, one for mono input from microphone and another for stereo quality audio output.

VGA

The OpenRD-Client has XGI's Volari-Z11, 2D Graphics processor on board The OpenRD-Client is designed to support up to 1280x1024 display resolution at 60 Hz. For better performance, the Volari chip has been supported with 512 Mb (64MB) of additional frame buffer DDR2 Memory.

UART

The 88F6281 has two built-in UART interfaces: UART0 and UART1. UART0 has been used as debug console along with FT2232D chip and it is available at mini USB connector (CON22). UART0 is default mode for debug console display. UART1 can be available either at RS232 connector (CON15) or RS485 connector (CON16). User may note that CON16 is RJ45 type of connector which is used for RS485 interface.

MODBUS

The OpenRD-Client provides full duplex RS485 transceiver on board to support Industrial MODBUS interface over connector CON16.

SPI interface (Optional header)

The OpenRD-Client provides an optional header (CON26) for SPI interface. SPI EEPROM or any other peripheral, which requires SPI interfacing, can be connected to this interface.

1.3.2 Power and System Clocks

The OpenRD-Client is budgeted for 36W (12V/3A) D.C. power supply. Average power consumption is around 7.2W. Maximum power consumption measured with all 7 USB ports, HDD and Ethernet operational is 29.9W. In idle state system draws power of around 4.8W.

The OpenRD-Client provides required clock reference to different functionalities of the board. The OpenRD-Client board features: 25MHz master clock source for 88F6281 SoC, 24MHz for 7 port USB hub, 25MHz for 88E1116R GbE PHY device, 14.31MHz for Graphics processor Z11, 6MHz for FT2232D chip and 32.768 KHz clock source for RTC oscillator of 88F6281.

The 88F6281 derive internal clocks from on board 25MHz clock source as per the configuration of strapping pins at power ON. The on-chip PLL frequency multiplier and the prescaler can generate programmable CPU clock between 800 MHz to 1200 MHz. The 88F6281 generates 400MHz clock source for DDR2 interface. It also generates clock sources for interfaces such as Gigabit Ethernet, PCIe, I2S Audio, SDIO, SPI and TWSI.

1.3.3 Board Debug interface and Manual Reset

The OpenRD-Client features two interface options for debugging. One is a standard 20-pin box header (CON25) located at the bottom side of PCB. Second is a mini USB connection featured by FDT2322D device.

The 20-pin header enables direct connection of an ICE to the JTAG inputs to the 88F6281 SoC. This is commonly interfaced through a Lauterbach probe. The second option is via a mini USB connection, it is a cost effective interface for the 88F6281 JTAG via mini USB cable. It utilizes an Open On-chip Debug codes (Open OCD), in which it’s been widely used by the open source codes Linux community to develop debug capability and flash capability. You can find more details by performing a web search under the “OpenOCD” or “FDT2322D” key subject.

Pressing reset switch SW2 can manually reset OpenRD-Client.

1.4 Package Contents

1. OpenRD-Client System Board
2. DVD including Software Packages, Installers, Schematics, Layout files, User Guide, Quick Start Guide, Brochure

3. 12V/3A Power Adapter
4. Mini USB cable
5. CAT6 cable
6. Enclosure (Optional)

1.5 Technical Specifications

Board Size	8.82" x 6.25" x 0.063"
Operating Temperature	0 C to 50 C
Storage Temperature	-40 C to 85 C
Power	12V/3A (36W)
Onboard Interfaces	2 x Gigabit Ethernet
	7 x USB
	1 x VGA
	1 x Audio (In/Out)
	1 x SD
	1 x SMBus
	1 x RS232
	1 x RS485
	1 x Debug (JTAG, RS-232 for console)
	1 x eSATA
	1 x SATA (2.5" HDD Connector)
Optional Enclosure Size	8.6" x 6.5" x 1.2"

2. Getting Started with the OpenRD-Client

2.1 Setup Requirements

1. 12V/3A DC Power supply
2. CAT5/CAT6 cable for network connection
3. One VGA monitor, USB keyboard and USB mouse connected to OpenRD-Client system.
4. USB cable (type A to mini B) for the system console.
5. One MS windows/Linux machine with at least 1 USB port to connect USB cable as listed above in point # 4. For further details on setting up Linux/Windows Host with appropriate drivers, please refer Appendix A.

2.2 Hardware Setup

Following procedure explains hardware setup for OpenRD-Client System

1. Connect CAT6/CAT5 cable to GbE 0 or GbE 1 port of the system (Figure 4). Make sure the other end of cable is connected to Ethernet hub/switch/router.

2. Connect USB keyboard and USB mouse to USB ports of the system. Also connect the power supply cable. (Figure 4)

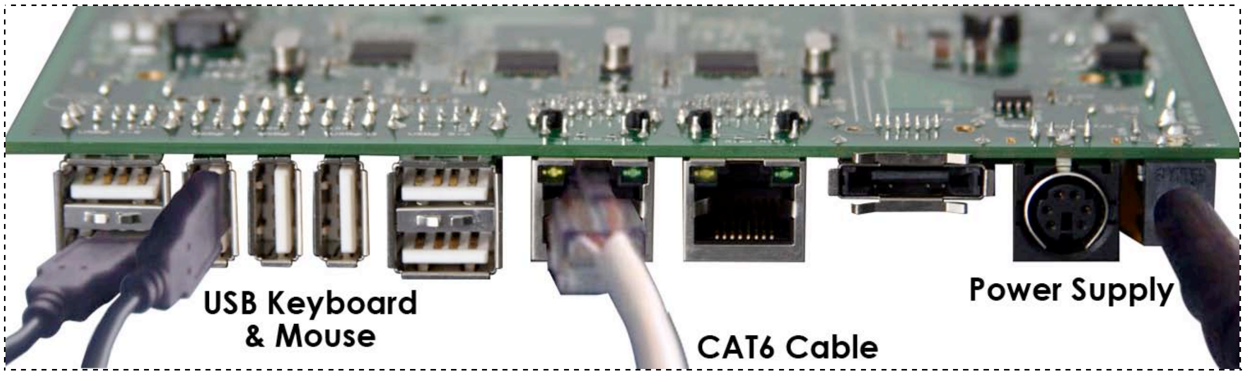


Figure 4 OpenRD-Client - USB, GbE0, and Power Supply

3. Connect VGA monitor to the VGA port of the system. (Figure 5)

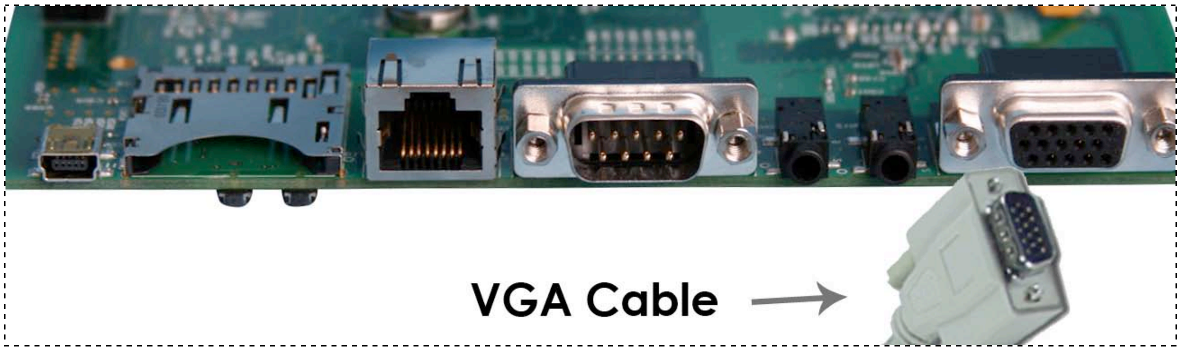


Figure 5 OpenRD-Client – VGA

4. Power on the system.

5. Please wait till the OpenRD-Client system boots up. It will display Linux boot up messages on the monitor.
6. Once system boots up, it will display log-in screen on the monitor.
7. Provide username: ‘**root**’ and password ‘**nosoup4u**’.

2.3 Default Network Configurations

By default, the OpenRD-Client will have following network configurations.

For GbE0
IP Address: 192.168.1.1
Netmask: 255.255.255.0

For GbE1
IP Address: 192.168.2.1
Netmask: 255.255.255.0

2.4 Default Login Information

By default, the OpenRD-Client has following user/pass.

Username: **admin**, Password: **openrd**
Username: **root,** Password: **nosoup4u**

2.5 Linux Logo on VGA Monitor Screen

Once you power on the OpenRD-Client, Linux will start booting and you will get following penguin logo on the VGA screen, provided that you have connected VGA monitor to your board

Figure 6 Linux Logo on VGA Monitor

3. Building OpenRD Software using source code

Pre-requisite

1. Make sure to setup GCC Toolchain, mkimage as per Appendix C.

3.1 Configuring and building U-boot

This section provides the procedure to configure the u-boot 1.1.4 with the 0001-OpenRD-Support-Uboot.patch for the OpenRD platform based on Kirkwood™ 6281(A0).

1. On a Linux machine, go to the 'home' directory and create a directory **'openrd'**.

/home# mkdir openrd

2. Copy u-boot-1.1.4 source file and patch for OpenRD in /home/openrd folder from <dvd-home>/openrd-devkit-v1.0/openrd_uboot/source/: ***u-boot-1.1.4.tar.gz, 0001-OpenRD-Support-Uboot.patch.gz***

3. Expand the source in the /home/openrd folder

~/openrd # tar –zxvf u-boot-1.1.4.tar.gz
~/openrd # gunzip 0001-OpenRD-Support-Uboot.patch.gz

4. Change directory to u-boot-1.1.4

~/openrd # cd u-boot-1.1.4

5. Apply OpenRD support patch to u-boot-1.1.4

~/openrd/u-boot-1.1.4 # patch –p1 < ../0001-OpenRD-Support-Uboot.patch

6. Do a make mrproper for a total clean build.

~/openrd/u-boot-1.1.4 # make mrproper

7. Configure u-boot for OpenRD

~/openrd/u-boot-1.1.4 # make OpenRD88f6281a_config NBOOT=1 LE=1

8. Create u-boot image

~/openrd/linux-2.6.22.18 # make

9. Rename it as u-boot.bin.openrd

~/openrd/linux-2.6.22.18 # cp u-boot-OpenRD88f6281a_400rd_nand.bin /home/openrd/uboot.bin.openrd

3.2 Configuring and building Linux Kernel using LSP

This section provides the procedure to configure the Linux kernel 2.6.22.18 with the 0001-openrd-support-LSP.patch for the OpenRD platform based on Kirkwood™ 6281(A0).

1. On a Linux machine, go to the 'home' directory and create a directory 'openrd'. **Ignore this**

step, if /home/openrd directory is already created.

/home# mkdir openrd

2. Copy the combined sources of the Linux kernel and patch for OpenRD in /home/openrd folder from <dvd-home>/openrd-devkit-v1.0/openrd_lsp/source/: ***linux-2.6.22.18.tar.gz, 0001-OpenRD-Support-LSP.patch.gz***

3. Expand the source in the /home/openrd folder

~/openrd # tar –zxvf linux-2.6.22.18.tar.gz
~/openrd # gunzip 0001-OpenRD-Support-LSP.patch.gz

4. Change directory to linux-2.6.22.18

~/openrd # cd linux-2.6.22.18

5. Apply OpenRD support patch to linux-2.6.22.18

~/openrd/linux-2.6.22.18 # patch –p1 < ../0001-OpenRD-Support-LSP.patch

6. Do a make mrproper for a total clean build.

~/openrd/linux-2.6.22.18 # make mrproper

7. Configure linux-2.6.22.18 for OpenRD

~/openrd/linux-2.6.22.18 # make mv88f6281_defconfig

8. Create a uImage

~/openrd/linux-2.6.22.18 # make uImage

9. Rename it as uImage.openrd

~/openrd/linux-2.6.22.18 # cp arch/arm/boot/uImage /home/openrd/uImage.openrd

3.3 Building File Systems

3.3.1 Building JFFS2 file system image

Pre-requisites

a. On the linux host, copy fc8.release.x11.openrd.nand.src.tar.gz to /home/openrd. This can be obtained from the CD/DVD, which is shipped along with the OpenRD-Client unit.

   Path of file system in CD/DVD:
   openrd-devkit-v1.0/openrd_filesystem/source/fc8.release.x11.openrd.nand.src.tar.gz

b. Install latest mtd-utils package on Linux host. If mtd-utils is not installed, following steps must be followed for installation.

   Download the rpm for host machine distribution from
   http://rpmfind.net/linux/rpm2html/search.php?query=mtd-utils

   After downloading rpm, install it as:

```
# rpm –ivh <mtd-utils-rpmname>
```

To check, if you already have mkfs.jffs2 installed or not, use following command:

```
-sh-3.2# find . -name mkfs.jffs2
```

OR

For Linux host distributions such as ubuntu, type following command on Linux host (**Note:** internet connection is required to carry out this step)

```
# sudo apt-get install mtd-utils
```

OR

Refer Appendix F to install pre-compiled mtd-utils binaries.

1. Extract Fedora Core 8 release file system source (**Note: run this command with root user**)

   ```
   # cd /home/openrd
   /home/openrd # tar zxvf fc8.release.x11.openrd.nand.src.tar.gz
   ```

   Make sure that /home/openrd/fc8.release.x11.openrd.nand.src/ is created on successful execution of this command.

2. Build JFFS2 file system image.

   ```
   # mkfs.jffs2 –l –n --pad –e 0x20000 –r /home/openrd/fc8.release.x11.openrd.nand.src/ –o
   fc8.release.x11.openrd.nand.jffs2
   ```

3.3.2 Building UBI file system image

Pre-requisites

a. On the linux host, copy fc8.release.x11.openrd.nand.src.tar.gz to /home/openrd. This can be obtained from the CD/DVD, which is shipped along with the OpenRD-Client unit.

   Path of file system in CD/DVD:
   openrd-devkit-v1.0/openrd_filesystem/source/fc8.release.x11.openrd.nand.src.tar.gz

b. Obtain mtd-utils v1.2.0 from http://git.infradead.org/?p=mtd-utils.git;a=summary and install lzo2 library from http://www.oberhumer.com/opensource/lzo/

   Following steps need to be followed for installing ubifs-tools using mtd-utils

   ```
   /home/openrd # tar zxvf mtd-utils.tar.gz

   /home/openrd # cd mtd-utils/mkfs.ubifs

   /home/openrd/mtd-utils/mkfs.ubifs # make

   /home/openrd/mtd-utils/mkfs.ubifs # make install

   /home/openrd/mtd-utils/mkfs.ubifs # cd ../ubi-utils/
   ```

```
/home/openrd/mtd-utils/ubi-utils # make

/home/openrd/mtd-utils/ubi-utils # make install

/home/openrd/mtd-utils/ubi-utils # cd ../..

/home/openrd # rm –rf mtd-utils
```

OR

Refer Appendix F to install pre-compiled mtd-utils binaries.

1. Extract Fedora Core 8 release file system source (**Note: run this command with root user; ignore step, if file system source is already extracted**)

   ```
   # cd /home/openrd
   /home/openrd # tar zxvf fc8.release.x11.openrd.nand.src.tar.gz
   ```

   Make sure that /home/openrd/fc8.release.x11.openrd.nand.src/ is created on successful execution of this command,

2. Create shell script named *make_ubifs.sh* with following content in /home/openrd directory:

   ```
   #!/bin/sh

   PWD=`pwd`
   TARGET_DIR=${PWD}
   ROOTFS=${PWD}/fc8.release.x11.openrd.nand.src/
   UBIFS_IMG=${TARGET_DIR}/fc8.release.x11.openrd.nand.ubifs.img
   UBINIZECFG=${TARGET_DIR}/ubinize.cfg
   UBI_IMG=${TARGET_DIR}/fc8.release.x11.openrd.nand.ubi.img
   SUBPAGE_SIZE=2048

   rm ${UBI_IMG}
   rm ${UBIFS_IMG}
   rm ${UBINIZECFG}

   mkfs.ubifs -g 1 -v -r ${ROOTFS} -m 2KiB -e 124KiB -c 4000 -o ${UBIFS_IMG} -x zlib
   touch ${UBINIZECFG}
   echo "
   [ubifs]
   mode=ubi
   image=${UBIFS_IMG}
   vol_id=0
   vol_size=480MiB
   vol_type=dynamic
   vol_name=rootfs
   vol_flags=autoresize
   " > ${UBINIZECFG}

   cat ${UBINIZECFG}

   ls -lh ${UBIFS_IMG}

   ubinize -v -o ${UBI_IMG} -m 2KiB -p 128KiB -s ${SUBPAGE_SIZE} -O
   ${SUBPAGE_SIZE} ${UBINIZECFG}
   ```

```
rm ${UBINIZECFG}
rm ${UBIFS_IMG}
```

3. Building UBIFS image

```
# chmod 755 ./make_ubifs.sh
# sh ./make_ubifs.sh
```

   On successful execution of above command, *fc8.release.x11.openrd.nand.ubi.img* file will be created in /home/openrd. This is UBIFS to be flashed on NAND.

4. Writing OpenRD Software to NAND flash

Pre-requisites

Following images are used in the procedures described below:

1. **u-boot.bin.openrd** – u-boot for OpenRD (obtain from <dvd-home>/openrd-devkit-v1.0/openrd_uboot/binary/)
2. **uImage.openrd** – Compressed Linux kernel for OpenRD (obtain from <dvd-home>/openrd-devkit-v1.0/openrd_lsp/binary/)
3. **fc8.release.x11.openrd.nand.jffs2** – Fedora Core 8 file system for OpenRD (obtain from <dvd-home>/openrd-devkit-v1.0/openrd_filesystem/binary/)
4. **fc8.release.x11.openrd.nand.ubi.img** – Fedora Core 8 file system for OpenRD (obtain from <dvd-home>/openrd-devkit-v1.0/openrd_filesystem)
5. Make sure to setup NFS root file system as per **Appendix E**.
6. Make sure to setup TFTP server as per **Appendix B**.

4.1 Writing U-boot

Pre-requisites

Obtain OpenOCD for OpenRD from DVD. Path for openocd.binaries.libftdi.tar.gz package is <dvd-home>/openrd-devkit-v1.0/openrd_openocd/binary/.

OpenRD board (client/base) connected to PC/Host with FTDI programmed for Channel A as CPU FIFO.

Steps

/home/openrd # tar zxvf openocd.binaries.libftdi.tar.gz

/home/openrd # cd openocd.binaries.libftdi/bin

/home/openrd/openocd.binaries.libftdi/bin # cp /home/openrd/u-boot-1.1.4/u-boot uboot.elf

/home/openrd/openocd.binaries.libftdi/bin # cp /home/openrd/u-boot-1.1.4/u-boot-OpenRD88f6281a_400rd_nand.bin uboot.bin

Give power cycle on OpenRD board before executing following command:

/home/openrd/openocd.binaries.libftdi/bin # ./openocd –f target/board/openrd.cfg –c init -c openrd_reflash_uboot

Notes:

1. Above command must be executed using root privilege. Execution of this command may take between 46-200 seconds.

2. After receiving following message using openocd, press Ctrl+C to quit; reset OpenRD manually using SW2 switch or power cycle. Remove USB mini B connector from OpenRD-Client board and again connect it.

   ```
   NAND flash device 'NAND 512MiB 3,3V 8-bit' found
   ```

```
successfully erased blocks 0 to 4 on NAND flash device 'NAND
512MiB 3,3V 8-bit'
wrote file uboot.bin to NAND flash 0 up to offset 0x00073000 in
42.612942s
Info:   JTAG tap: feroceon.cpu tap/device found: 0x20a023d3
(Manufacturer: 0x1e9, Part: 0x0a02, Version: 0x2)
Info:   JTAG Tap/device matched
```

3. Sample output of above command:

/home/openrd/openocd.binaries.libftdi/bin # openocd -f target/board/openrd.cfg -c init -c openrd_reflash_uboot

```
Open On-Chip Debugger 1.0 (2009-02-27-10:14) svn:unknown


BUGS? Read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS


$URL$
jtag_speed: 1
dcc downloads are enabled
Info:   JTAG tap: feroceon.cpu tap/device found: 0x20a023d3
(Manufacturer: 0x1e9, Part: 0x0a02, Version: 0x2)
Info:   JTAG Tap/device matched
Error:  unknown EmbeddedICE version (comms ctrl: 0x00000018)
Warning:no telnet port specified, using default port 4444
Warning:no gdb port specified, using default port 3333
Warning:no tcl port specified, using default port 6666
target state: halted
target halted in ARM state due to debug-request, current mode:
Supervisor
cpsr: 0x000000d3 pc: 0xffff0000
MMU: disabled, D-Cache: disabled, I-Cache: disabled
0 0 1 0: 00052078
NAND flash device 'NAND 512MiB 3,3V 8-bit' found
successfully erased blocks 0 to 4 on NAND flash device 'NAND
512MiB 3,3V
8-bit'
wrote file uboot.bin to NAND flash 0 up to offset 0x00073000 in
42.612942s
Info:   JTAG tap: feroceon.cpu tap/device found: 0x20a023d3
(Manufacturer: 0x1e9, Part: 0x0a02, Version: 0x2)
Info:   JTAG Tap/device matched
```

4.2 Writing Linux Kernel

This section shows the steps to write Linux Kernel onto the NAND flash on the KW-6281(A0) based OpenRD board. On the console of the debug board follow the steps below to write the image to the NAND flash. Initially boot the debug board from NFS location where uImage and jffs2 images are stored.

uImage.openrd can be obtained from <dvd-home>/openrd-devkit-v1.0/openrd_lsp/binary/ from DVD provided with OpenRD. Or, user may build uImage using source code provided on DVD at <dvd-home>/openrd-devkit-v1.0/openrd_lsp/source/.

Note: Following instructions stand valid only if used with NFS Root File System provided with OpenRD release

1. At the Linux prompt, check for the NAND flash partition. You should see 2 partitions after issuing the command below.

```
-sh-3.2# cat /proc/mtd
dev:    size         erasesize  name
mtd0: 00400000 00020000 "uImage"
mtd1: 1fb00000 00020000 "rootfs"
```

2. Confirm that the NAND erase and write binaries are included. These should be found in the 'usr/sbin' directory on giving the commands below.

```
-sh-3.2# find . -name flash_eraseall
-sh-3.2# find . -name nandwrite
```

3. Erase Linux Kernel partition on the NAND.

```
-sh-3.2# flash_eraseall -j  /dev/mtd0
Erasing 128 Kibyte @ 3e0000 -- 96 % complete. Cleanmarker written at 3e0000.
```

4. Write Linux Kernel to NAND

```
-sh-3.2# nandwrite –p   /dev/mtd0 uImage.openrd
Writing data to block 0
Writing data to block 20000
Writing data to block 40000
Writing data to block 60000
Writing data to block 80000
Writing data to block a0000
Writing data to block c0000
Writing data to block e0000
Writing data to block 100000
Writing data to block 120000
Writing data to block 140000
Writing data to block 160000
Writing data to block 180000
Writing data to block 1a0000
Writing data to block 1c0000
Writing data to block 1e0000
Writing data to block 200000
Writing data to block 220000
Writing data to block 240000
```

4.3 Writing File Systems

4.3.1 Writing JFFS2 file system image

This section shows the steps to copy the Fedora Core 8 file system jffs2 image onto the NAND flash on the KW-6281(A0) based OpenRD board. The Fedora Core 8 file system can then be updated and various packages can be installed.

On the console of the debug board follow the steps below to write the image to the NAND flash. Initially boot the debug board from NFS location where uImage and jffs2 images are stored.

fc8.release.x11.openrd.nand.jffs2 can be obtained from <dvd-home>/openrd-devkitv1.0/openrd_filesystem/binary/ on DVD shipped with OpenRD. Or, user may generate JFFS2 image from source provided on DVD at <dvd-home>/openrd-devkit-v1.0/openrd_filesystem/source/fc8.release.x11.openrd.nand.src.tar.gz.

Note: Following instructions stand valid only if used with NFS Root File System provided with OpenRD release

1. At the Linux prompt, check for the NAND flash partition. You should see 2 partitions after issuing the command below.

```
-sh-3.2# cat /proc/mtd
dev:    size         erasesize  name
mtd0: 00400000 00020000 "uImage"
mtd1: 1fb00000 00020000 "rootfs"
```

2. Confirm that the NAND erase and write binaries are included. These should be found in the ‘usr/sbin’ directory on giving the commands below.

```
-sh-3.2# find . -name flash_eraseall
-sh-3.2# find . -name nandwrite
```

3. Erase file system partition on the NAND.

```
-sh-3.2# flash_eraseall -j  /dev/mtd1
Erasing 128 Kibyte @ 15300000 -- 66 % complete. Cleanmarker written at 15300000.8
Kibyte @ f800000 -- 48 % complete. Cleanma.Skipping bad block at 0x15320000
Erasing 128 Kibyte @ 1b240000 -- 85 % complete. Cleanmarker written at 1b240000.
Skipping bad block at 0x1b260000
Erasing 128 Kibyte @ 1d7a0000 -- 93 % complete. Cleanmarker written at 1d7a0000.
Skipping bad block at 0x1d7c0000
Erasing 128 Kibyte @ 1fae0000 -- 99 % complete. Cleanmarker written at 1fae0000.
-sh-3.2#
```

Here, /dev/mtd1 is configured as rootfs partition using $(console) environment variable on u-boot.

5. Write the filesystem jffs2 image onto the NAND flash. Note the number of blocks that are used by the data. This is the length of the image that will be needed in the boot arguments (similarly you can choose to utilize the NAND flash space up to 506 MB) while editing the bootargs command in U-Boot.

```
-sh-3.2# nandwrite  -p –q /dev/mtd1 fc8.release.x11.openrd.nand.jffs2
```

Note: This will take some time as the time taken for flashing File system depends on size of jffs2 image

6. Create a mount directory and mount the jffs2 image on the NAND flash mtdblock1.

```
-sh-3.2# mkdir mnt1
-sh-3.2#
-sh-3.2#
-sh-3.2# mount -t jffs2  /dev/mtdblock1  /mnt1
-sh-3.2# umount /mnt1
```

7. Restart the system and enter the U-Boot prompt by stopping the auto boot,

8. Booting Linux Kernel, JFFS2 file system from NAND

Following command must be applied at u-boot prompt (Marvell>>)

```
Marvell>> setenv console 'console=ttyS0,115200
mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs) rw'

Marvell>> setenv bootargs $(console) root=/dev/mtdblock1 fb=xgifb

Marvell>> setenv bootcmd 'nand read.e 0x800000 0x100000 0x400000; bootm
0x800000'

Marvell>> saveenv
```

9. Reboot the system and the system should boot from NAND flash. Login prompt will only be received on GUI; i.e. no login prompt will be available on serial console.

```
Marvell>> reset
```

10. The login and password to access the prompt are:
    a. Login – root, Password – nosoup4u (**root can not be used to login using GUI**)
    b. Login – admin, Password - openrd

4.3.2 Writing UBIFS image

This section shows the steps to copy the Fedora Core 8 file system ubifs image onto the NAND flash on the KW-6281(A0) based OpenRD board. The Fedora Core 8 file system can then be updated and various packages can be installed.

On the console of the debug board follow the steps below to write the image to the NAND flash. Initially boot the debug board from NFS location where uImage and ubifs images are stored.

fc8.release.x11.openrd.nand.ubi.img can be obtained from <dvd-home>/openrd-devkit-v1.0/openrd_filesystem/binary/ on DVD shipped with OpenRD. Or, user may generate UBIFS from source provided on DVD at <dvd-home>/openrd-devkit-v1.0/openrd_filesystem/source/ fc8.release.x11.openrd.nand.src.tar.gz.
Note: Following instructions stand valid only if used with NFS Root File System provided with OpenRD release.

1. At the Linux prompt, check for the NAND flash partition. You should see **2 partitions** after issuing the command below.

```
-sh-3.2# cat /proc/mtd
dev:    size        erasesize  name
mtd0: 00400000 00020000 "uImage"
mtd1: 1fb00000 00020000 "rootfs"
```

2. Confirm that the NAND erase and write binaries are included. These should be found in the ‘usr/sbin’ directory on giving the commands below.

```
-sh-3.2# find . -name flash_eraseall
-sh-3.2# find . -name nandwrite
```

3. Erase file system partition on the NAND.

```
-sh-3.2# flash_eraseall /dev/mtd1
Erasing 128 Kibyte @ 15300000 -- 66 % complete. Cleanmarker written at 15300000.8
Kibyte @ f800000 -- 48 % complete. Cleanma.Skipping bad block at 0x15320000
Erasing 128 Kibyte @ 1b240000 -- 85 % complete. Cleanmarker written at 1b240000.
Skipping bad block at 0x1b260000
Erasing 128 Kibyte @ 1d7a0000 -- 93 % complete. Cleanmarker written at 1d7a0000.
Skipping bad block at 0x1d7c0000
Erasing 128 Kibyte @ 1fae0000 -- 99 % complete. Cleanmarker written at 1fae0000.
-sh-3.2#
```

Here, /dev/mtd1 is configured as rootfs partition using $(console) environment variable on u-boot.

1. Write the file system ubifs image onto the NAND.

```
# ubiformat /dev/mtd1 -f fc8.release.x11.openrd.nand.ubi.img -s 2048 –O 2048
ubiformat: mtd1 (NAND), size 531628032 bytes (507.0 MiB), 131072 eraseblocks of
131072 bytes (128.0 KiB), min. I/O size 2048slibscan: scanning eraseblock 4055 -- 100
% complete
ubiformat: 4053 eraseblocks are supposedly empty
ubiformat: bad eraseblocks: 2713, 3475, 3774
ubiformat: flashing eraseblock 3497 -- 100 % complete
ubiformat: formatting eraseblock 4055 -- 100 % complete
-sh-3.2#
```

Note: This will take some time as the time taken for flashing File system depends on size of ubifs image

2. Restart the system and press any key to stop on the U-Boot prompt

```
# reboot
```

3. Booting Linux Kernel, UBI file system from NAND

Following command must be applied at u-boot prompt (Marvell>>)

```
Marvell>> setenv console 'console=ttyS0,115200
mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs) rw'

Marvell>> setenv bootargs $(console) ubi.mtd=1,2048 root=ubi0:rootfs rootfstype=ubifs
fb=xgifb

Marvell>> setenv bootcmd 'nand read.e 0x800000 0x100000 0x400000; bootm
0x800000'

Marvell>> saveenv
```

Note: Be careful while giving “saveenv” command. “saveenv” will save the parameters into NAND. From next reboot onwards, board will use the saved environment variables.

4. Reboot the system and the system should boot from NAND flash.

```
Marvell>> reset
```

5. The login and password to access the prompt are:

a. Login – root, Password – nosoup4u (**root can not be used to login using GUI**)
b. Login – admin, Password - openrd

5. On-Board Software

5.1 Flash Memory Organization

The NAND Flash memory used for 88F6281 SoC is organized as 262,144 pages by 2112 x 8 columns and the Spare is organized as 64x8 columns. The write and read operation are executed on a page basis, while the erase operation is executed on a block basis. Bit erase operation on the NAND Flash device is not supported. Erasable Size is 256KB blocks. The 88F6281 SoC internal boot ROM initializes the DDR2 memory according to the parameter located in the image extended header running at 0xFFFF0000. It copies the U-boot image from the NAND flash into the DRAM at location 0xF0000000 accordingly to the imager header, which is located on the first page of the NAND flash at offset 0.

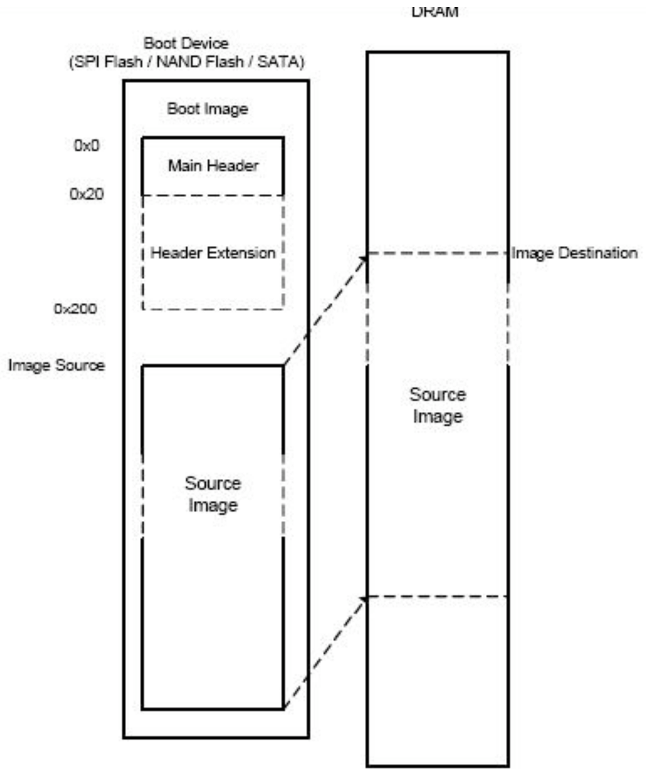


Figure 7 Flash Memory Organization

5.2 Flash Write Access

The flash mapping for the image layout as follows:

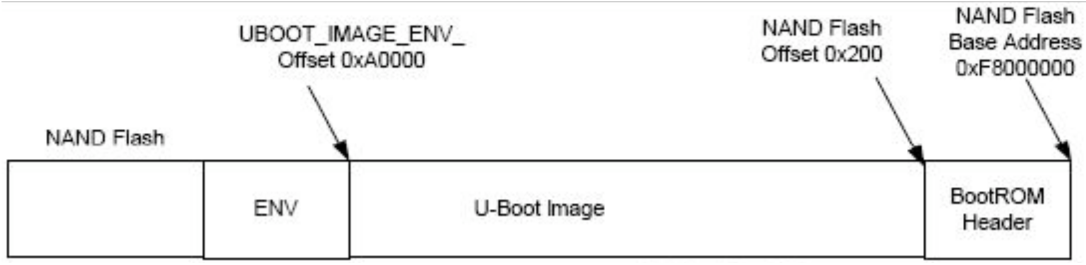


Figure 8 Flash Write Access

The boot software a.k.a. U-Boot starts at the reset after it asserted low for 20 ms, and then the POR circuit is triggered. The SysRSTn stays asserted for additional 300us after the power and clocks are stable. The NAND flash performs a boot sequence during the additional 300us time to prepare page 0 ready to be read, upon completing the above sequence, the internal CPU reset is de-asserted, and CPU starts to execute boot code form the NAND device. When the reset button is pressed, the POR circuit is triggered and the whole process is started all over again.

Since the OpenRD-Client board is booting from NAND Flash device, there are no optional strapping pin to support booting from other devices.
The boot sequence upon power on reset is as follows:

- U-Boot image is resided in the NAND FLASH – page 0
- CPU jumps to reset vector at 0xFFFF0000
- CPU executed vector code at 0xFFF90000
- CPU internal boot loader initializes register 0xF1000000
- CPU internal boot loader initializes register DDRAM 0xF0000000
- CPU internal boot loader copied Flash image from 0xF0000000 to DRAM location 0xFFF90000
- CPU running codes from 0XFFF90000 location
- CPU performed system POST and display U-Boot prompt

5.3 Functional Test Software

The functional Test SW is part of the boot software that allows testing of the 88F6281 SoC device. From U-Boot Monitor mode, commands are extended with extra functionality, which can be useful for running diagnostic and benchmarks. There are two types of Functional Test Software; POST and manufacturing diagnostic test. POST testing is expected to be an application on top of Boot loader performing sanity checks to ensure that peripherals critical to OpenRD-Client are checked every time platform is powered on. Manufacturing Diagnostics is expected to be application on top of Boot loader performing functional checks to ensure that peripherals on OpenRD-Client are usable postproduction; such diagnostics will help contract manufacturer to perform quality control on OpenRD-Client at the time of production from functional standpoints using software. Prior to loading Boot loader, contract manufacturer shall perform power supply checks as well as JTAG tests to the extent possible. In addition, contract manufacturer will have to populate OpenRD-Client as per HW assembly guidelines and connect peripheral connectors essential for Manufacturing Diagnostics tests. POST flow chart test is illustrated as follows:

U-boot source code has both – POST and Manufacturing Diagnostics implemented. To turn off, manufacturing diagnostics set run_diag environment variable to “no” on u-boot prompt as follows:

Steps to turn off Manufacturing Diagnostics

Marvell>> setenv run_diag no
Marvell>> saveenv

Steps to turn on Manufacturing Diagnostics

Marvell>> setenv run_diag yes
Marvell>> saveenv

For successful execution of Manufacturing Diagnostics, UART loopback connector must be connected on UART1 connector on OpenRD. Once manufacturing diagnostics is turned off in u-boot, during every boot up, POST will be carried out.

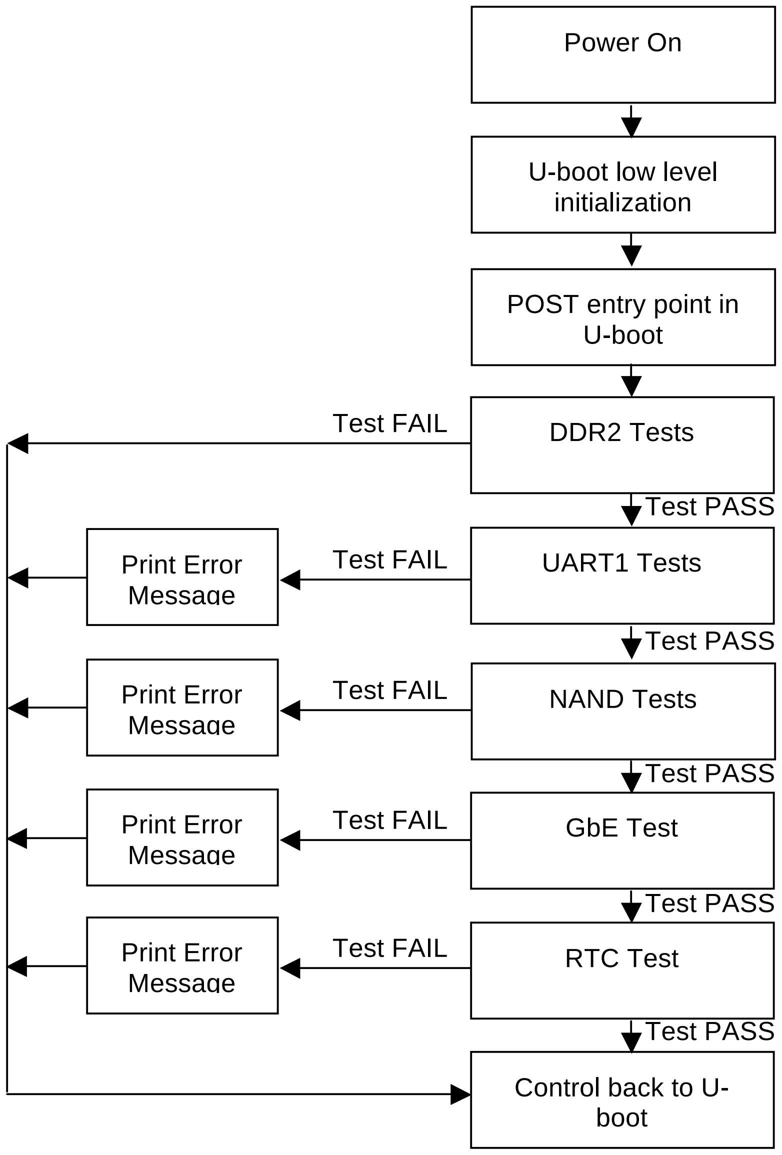


Figure 9 Functional Test Software

POST test will perform following checks:

- DDR2 Tests (data bus test, address bus test)
- NAND Tests (nand detection test, nand bad block detection test)
- UART1 Tests (UART1 internal loopback test)
- RTC (RTC date/time test)
- GbE Test (GbE link detection test)

5.4 Manufacturing Diagnostics Screen Mockup (Success case)

```
         __  __                      _ _
        |  \/  | __ _ _ ____   _____| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 24 2009 - 13:56:42) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

Checking for BootROM Routine Errors

No. of BootROM routine retries: 8
NAND: Nand ECC error


Running diagnostics ...

      DDR2 data bus test                               PASSED

      DDR2 address bus test                            PASSED

      DDR2 device test                                 PASSED

      UART 1 internal loopback test on baudrate   9600 PASSED
      UART 1 internal loopback test on baudrate  19200 PASSED
      UART 1 internal loopback test on baudrate 115200 PASSED

      UART 1 external loopback test on baudrate   9600 PASSED
      UART 1 external loopback test on baudrate  19200 PASSED
      UART 1 external loopback test on baudrate 115200 PASSED

      Device: 0, Size: 512 MB, Page Size: 2 KB, Block Size: 128 KB
      NAND detection test                              PASSED

      Bad Block: 15820000
      Bad Block: 1b760000
      Bad Block: 1dcc0000
      NAND bad-block detection test                    PASSED

      Speed: 1000 Mbps, Duplex: Full, Link: up
      GbE0 link detect test                            PASSED

      Speed: 1000 Mbps, Duplex: Full, Link: up
      GbE1 link detect test                            PASSED

      RTC test                                         PASSED

Diag completed

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>>
```

5.5 Manufacturing Diagnostics Screen Mockup (Failure case)

```
         __  __                      _ _
        |  \/  | __ _ _ ____   _____| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 24 2009 - 13:56:42) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB
```

```
Checking for BootROM Routine Errors

No. of BootROM routine retries: 8
NAND: Nand ECC error


Running diagnostics ...

        DDR2 data bus test                              PASSED

        DDR2 address bus test                           PASSED

        DDR2 device test                                PASSED

        UART 1 internal loopback test on baudrate   9600 PASSED
        UART 1 internal loopback test on baudrate  19200 PASSED
        UART 1 internal loopback test on baudrate 115200 PASSED

        UART 1 external loopback test on baudrate   9600 TIMEOUT

Diag FAILED

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>>
```

5.6 POST (Power On Self Test) Screen Mockup (Success case)

```
         __  __                      _ _
        |  \/  | __ _ _ ____   _____| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 24 2009 - 13:56:42) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
```

```
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

Checking for BootROM Routine Errors

No. of BootROM routine retries: 8
NAND: Nand ECC error

Running POST...

        DDR2 data bus test                              PASSED

        DDR2 address bus test                           PASSED

        UART 1 internal loopback test on baudrate   9600 PASSED
        UART 1 internal loopback test on baudrate  19200 PASSED
        UART 1 internal loopback test on baudrate 115200 PASSED

        Device: 0, Size: 512 MB, Page Size: 2 KB, Block Size: 128 KB
        NAND detection test                             PASSED

        Bad Block: 15820000
        Bad Block: 1b760000
        Bad Block: 1dcc0000
        NAND bad-block detection test                   PASSED

        RTC test                                        PASSED

6/6 tests PASSED
POST completed

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>>
```

5.7 POST (Power On Self Test) Screen Mockup (Failure case)

```
         __  __                      _ _
        |  \/  | __ _ _ ____   _____| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 24 2009 - 13:56:42) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

Checking for BootROM Routine Errors

No. of BootROM routine retries: 8
NAND: Nand ECC error

Running POST...

        DDR2 data bus test                              PASSED

        DDR2 address bus test                           PASSED

        UART 1 internal loopback test on baudrate   9600 PASSED
        UART 1 internal loopback test on baudrate  19200 PASSED
        UART 1 internal loopback test on baudrate 115200 PASSED

        Device: 0, Size: 512 MB, Page Size: 2 KB, Block Size: 128 KB
        NAND detection test                             PASSED

        Bad Block: 15820000
        Bad Block: 1b760000
        Bad Block: 1dcc0000
        NAND bad-block detection test                   PASSED

        RTC test                                        FAILED

5/6 tests PASSED
POST completed
```

```
CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>>
```

In addition to running Manufacturing Diagnostics or POST during boot up; it’s also possible to run these test using U-boot command line as follows:

Command syntax:

Marvell>> mv_diag <testname>

Sample output:

```
Marvell>> mv_diag
Available tests are:
 data_bus
 addr_bus
 device
 uart_int
 uart_ext
 nand_detect
 nand_bad_block
 nand_rw
 gbe_link
 rtc
```

User can run selective tests using command line specified above. For e.g. to execute rtc test using command line:

```
Marvell>> mv_diag rtc
              RTC test                                          PASSED
Marvell>>
```

6. Prepare SATA HDD with Fedora Core 8 ARM File System

In order to prepare SATA disk with Fedora 8 ARM port File System, please follow below mentioned steps:

Steps

1. Connect SATA HDD on Linux box with SATA HDD supported and partition it using **fdisk** as follows:
   - First partition (/dev/sda1) of 25GB or more for fedora linux (Ext3, Id:83)
   - Second partition (/dev/sda2) of 1MB for Swap (Swap Id: 82)
   - Third partition (/dev/sda3) of remaining space.
2. Create ext3 file system on partition 1 and 3. Create swap on partition 2

   ```
   # mkfs.ext3 /dev/sda1
   # mkfs.ext3 /dev/sda3
   # mkswap /dev/sda2
   ```
3. Mount /dev/sda1 to a mount point; for e.g. /mnt

   ```
   # mount /dev/sda1 /mnt
   ```
4. Untar “fc8.release.x11.openrd.hdd.tar.gz” on /dev/sda1 using mount point /mnt. This will extract Fedora 8 ARM port file system to /mnt

   ```
   # cd /mnt
   # tar zxvf <dvd-home>/openrd_filesystem/hdd/fc8.release.x11.openrd.hdd.tar.gz
   ```

   On successful execution of above commands, connect SATA HDD prepared with Fedora Core 8 ARM File System to SATA connector on the bottom side of OpenRD board.
5. Booting Linux Kernel from NAND, FC8 file system from SATA

   Following command must be applied at u-boot prompt (Marvell>>)

   ```
   Marvell>> setenv console 'console=ttyS0,115200
   mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs) rw'

   Marvell>> setenv bootargs $(console) root=/dev/sda1 fb=xgifb

   Marvell>> setenv bootcmd 'nand read.e 0x800000 0x100000 0x400000; bootm
   0x800000'

   Marvell>> saveenv
   ```

7. Ubuntu 9.04 Jaunty File System

7.1 Building File Systems

7.1.1 Building JFFS2 file system image

Pre-requisites

b. On the linux host, copy ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz to /home/openrd. This can be obtained from the CD/DVD, which is shipped along with the OpenRD-Client unit.

   Path of file system in CD/DVD:
   openrd-devkit-v1.0/openrd_filesystem-ubuntu/source/ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz

c. Install latest mtd-utils package on Linux host. If mtd-utils is not installed, following steps must be followed for installation.

   Download the rpm for host machine distribution from
   http://rpmfind.net/linux/rpm2html/search.php?query=mtd-utils

   After downloading rpm, install it as:

   ```
   # rpm –ivh <mtd-utils-rpmname>
   ```

   To check, if you already have mkfs.jffs2 installed or not, use following command:

   ```
   -sh-3.2# find . -name mkfs.jffs2
   ```

   OR

   For Linux host distributions such as ubuntu, type following command on Linux host (**Note:** internet connection is required to carry out this step)

   ```
   # sudo apt-get install mtd-utils
   ```

   OR

   Refer Appendix F to install pre-compiled mtd-utils binaries.

3. Extract ubuntu-9.04 jaunty release file system source (**Note: run this command with root user**)

   ```
   # cd /home/openrd
   /home/openrd # tar zxvf ubuntu-9.04.jaunty.release.openrd.src.tar.gz
   ```

   Make sure that /home/openrd/ubuntu-9.04.jaunty.release.x11.openrd.src/ is created on successful execution of this command,

4. Build JFFS2 file system image.

   ```
   # mkfs.jffs2 –l –n --pad –e 0x20000 –r /home/openrd/ubuntu-9.04.jaunty.release.x11.openrd.src/ –o ubuntu-9.04.release.x11.openrd.jffs2
   ```

7.1.2 Building UBI file system image

Pre-requisites

b. On the linux host, copy ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz to /home/openrd. This can be obtained from the CD/DVD, which is shipped along with the OpenRD-Client unit.

   Path of file system in CD/DVD:
   openrd-devkit-v1.0/openrd_filesystem-ubuntu/source/ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz

c. Obtain mtd-utils v1.2.0 from http://git.infradead.org/?p=mtd-utils.git;a=summary and install lzo2 library from http://www.oberhumer.com/opensource/lzo/

   Following steps need to be followed for installing ubifs-tools using mtd-utils

```
/home/openrd # tar zxvf mtd-utils.tar.gz

/home/openrd # cd mtd-utils/mkfs.ubifs

/home/openrd/mtd-utils/mkfs.ubifs # make

/home/openrd/mtd-utils/mkfs.ubifs # make install

/home/openrd/mtd-utils/mkfs.ubifs # cd ../ubi-utils/

/home/openrd/mtd-utils/ubi-utils # make

/home/openrd/mtd-utils/ubi-utils # make install

/home/openrd/mtd-utils/ubi-utils # cd ../..

/home/openrd # rm –rf mtd-utils
```

   OR

   Refer Appendix F to install pre-compiled mtd-utils binaries.

4. Extract ubuntu-9.04 jaunty release file system source (**Note: run this command with root user; ignore step, if file system source is already extracted**)

```
# cd /home/openrd
/home/openrd # tar zxvf ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz
```

   Make sure that /home/openrd/ ubuntu-9.04.jaunty.release.x11.openrd.src/ is created on successful execution of this command,

5. Create shell script named *make_ubifs.sh* with following content in /home/openrd directory:

```
#!/bin/sh

PWD=`pwd`
TARGET_DIR=${PWD}
ROOTFS=${PWD}/ubuntu-9.04.jaunty.release.x11.openrd.src/
UBIFS_IMG=${TARGET_DIR}/ubuntu-9.04.jaunty.release.x11.openrd.ubifs.img
UBINIZECFG=${TARGET_DIR}/ubinize.cfg
```

```
UBI_IMG=${TARGET_DIR}/ubuntu-9.04.jaunty.release.x11.openrd.ubi.img
SUBPAGE_SIZE=2048

rm ${UBI_IMG}
rm ${UBIFS_IMG}
rm ${UBINIZECFG}

mkfs.ubifs -g 1 -v -r ${ROOTFS} -m 2KiB -e 124KiB -c 4000 -o ${UBIFS_IMG} -x zlib
touch ${UBINIZECFG}
echo "
[ubifs]
mode=ubi
image=${UBIFS_IMG}
vol_id=0
vol_size=480MiB
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
" > ${UBINIZECFG}

cat ${UBINIZECFG}

ls -lh ${UBIFS_IMG}

ubinize -v -o ${UBI_IMG} -m 2KiB -p 128KiB -s ${SUBPAGE_SIZE} -O
${SUBPAGE_SIZE} ${UBINIZECFG}

rm ${UBINIZECFG}
rm ${UBIFS_IMG}
```

6. Building UBIFS image

```
# chmod 755 ./make_ubifs.sh
# sh ./make_ubifs.sh
```

On successful execution of above command, *ubuntu-9.04.jaunty.release.x11.openrd.ubi.img* file will be created in /home/openrd. This is UBIFS to be flashed on NAND.

7.2 Writing File Systems

7.2.1 Writing JFFS2 file system image

This section shows the steps to copy the jaunty file system jffs2 image onto the NAND flash on the KW-6281(A0) based OpenRD board. The jaunty file system can then be updated and various packages can be installed. Packages are installed in the Ubuntu based jaunty file system using 'apt-get' command. Snippets of the logs of installed packages are shown in **Appendix G**.

On the console of the debug board follow the steps below to write the image to the NAND flash. Initially boot the debug board from NFS location where uImage and jffs2 images are stored.

ubuntu-9.04.jaunty.release.x11.openrd.jffs2 can be obtained from <dvd-home>/openrd-devkit-v1.0/openrd_filesystem-ubuntu/binary/ on DVD shipped with OpenRD. Or, user may generate JFFS2 image from source provided on DVD at <dvd-home>/openrd-devkit-v1.0/openrd_filesystem-ubuntu/source/ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz.

Note: Following instructions stand valid only if used with NFS Root File System provided with OpenRD release

1. At the Linux prompt, check for the NAND flash partition. You should see 2 partitions after issuing the command below.

```
-sh-3.2# cat /proc/mtd
dev:    size         erasesize  name
mtd0: 00400000 00020000 "uImage"
mtd1: 1fb00000 00020000 "rootfs"
```

2. Confirm that the NAND erase and write binaries are included. These should be found in the 'usr/sbin' directory on giving the commands below.

```
-sh-3.2# find . -name flash_eraseall
-sh-3.2# find . -name nandwrite
```

3. Erase file system partition on the NAND.

```
-sh-3.2# flash_eraseall -j  /dev/mtd1
Erasing 128 Kibyte @ 15300000 -- 66 % complete. Cleanmarker written at 15300000.8
Kibyte @ f800000 -- 48 % complete. Cleanma.Skipping bad block at 0x15320000
Erasing 128 Kibyte @ 1b240000 -- 85 % complete. Cleanmarker written at 1b240000.
Skipping bad block at 0x1b260000
Erasing 128 Kibyte @ 1d7a0000 -- 93 % complete. Cleanmarker written at 1d7a0000.
Skipping bad block at 0x1d7c0000
Erasing 128 Kibyte @ 1fae0000 -- 99 % complete. Cleanmarker written at 1fae0000.
-sh-3.2#
```

Here, /dev/mtd1 is configured as rootfs partition using $(console) environment variable on u-boot.

5. Write the filesystem jffs2 image onto the NAND flash. Note the number of blocks that are used by the data. This is the length of the image that will be needed in the boot arguments (similarly you can choose to utilize the NAND flash space up to 506 MB) while editing the bootargs command in U-Boot.

```
-sh-3.2# nandwrite  -p –q /dev/mtd1 ubuntu-9.04.jaunty.release.x11.openrd.jffs2
```

Note: This will take some time as the time taken for flashing File system depends on size of jffs2 image

6. Create a mount directory and mount the jffs2 image on the NAND flash mtdblock1.

```
-sh-3.2# mkdir mnt1
-sh-3.2#
-sh-3.2#
-sh-3.2# mount -t jffs2  /dev/mtdblock1  /mnt1
-sh-3.2# umount /mnt1
```

7. Restart the system and enter the U-Boot prompt by stopping the auto boot,

8. Booting Linux Kernel, JFFS2 file system from NAND

Following command must be applied at u-boot prompt (Marvell>>)

```
Marvell>> setenv console 'console=ttyS0,115200
mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs) rw'

Marvell>> setenv bootargs $(console) root=/dev/mtdblock1 fb=xgifb

Marvell>> setenv bootcmd 'nand read.e 0x800000 0x100000 0x400000; bootm
0x800000'

Marvell>> saveenv
```

9. Reboot the system and the system should boot from NAND flash. Login prompt will only be received on GUI; i.e. no login prompt will be available on serial console.

```
Marvell>> reset
```

10. The login and password to access the prompt are:
    a. Login – root, Password – nosoup4u (**root can not be used to login using GUI**)
    b. Login – admin, Password - openrd

7.2.2 Writing UBIFS image

This section shows the steps to copy the jaunty file system ubifs image onto the NAND flash on the KW-6281(A0) based OpenRD board. The jaunty file system can then be updated and various packages can be installed. Packages are installed in the Ubuntu based jaunty file system using ‘apt-get’ command. Snippets of the logs of installed packages are shown in **Appendix G**.

On the console of the debug board follow the steps below to write the image to the NAND flash. Initially boot the debug board from NFS location where uImage and ubifs images are stored.

ubuntu-9.04.jaunty.release.x11.openrd.ubi.img can be obtained from <dvd-home>/openrd-devkit-v1.0/openrd_filesystem-ubuntu/binary/ on DVD shipped with OpenRD. Or, user may generate UBIFS from source provided on DVD at <dvd-home>/openrd-devkit-v1.0/openrd_filesystem-ubuntu/source/ubuntu-9.04.jaunty.release.x11.openrd.src.tar.gz.

Note: Following instructions stand valid only if used with NFS Root File System provided with OpenRD release.

1. At the Linux prompt, check for the NAND flash partition. You should see **2 partitions** after issuing the command below.

```
-sh-3.2# cat /proc/mtd
dev:    size       erasesize  name
mtd0: 00400000 00020000 "uImage"
mtd1: 1fb00000 00020000 "rootfs"
```

2. Confirm that the NAND erase and write binaries are included. These should be found in the ‘usr/sbin’ directory on giving the commands below.

```
-sh-3.2# find . -name flash_eraseall
-sh-3.2# find . -name nandwrite
```

3. Erase file system partition on the NAND.

```
-sh-3.2# flash_eraseall /dev/mtd1
Erasing 128 Kibyte @ 15300000 -- 66 % complete. Cleanmarker written at 15300000.8
Kibyte @ f800000 -- 48 % complete. Cleanma.Skipping bad block at 0x15320000
Erasing 128 Kibyte @ 1b240000 -- 85 % complete. Cleanmarker written at 1b240000.
Skipping bad block at 0x1b260000
Erasing 128 Kibyte @ 1d7a0000 -- 93 % complete. Cleanmarker written at 1d7a0000.
Skipping bad block at 0x1d7c0000
Erasing 128 Kibyte @ 1fae0000 -- 99 % complete. Cleanmarker written at 1fae0000.
-sh-3.2#
```

Here, /dev/mtd1 is configured as rootfs partition using $(console) environment variable on u-boot.

6. Write the file system ubifs image onto the NAND.

```
# ubiformat /dev/mtd1 -f ubuntu-9.04.jaunty.release.x11.openrd.ubi.img -s 2048 –O 2048
ubiformat: mtd1 (NAND), size 531628032 bytes (507.0 MiB), 131072 eraseblocks of
131072 bytes (128.0 KiB), min. I/O size 2048slibscan: scanning eraseblock 4055 -- 100
% complete
ubiformat: 4053 eraseblocks are supposedly empty
ubiformat: bad eraseblocks: 2713, 3475, 3774
ubiformat: flashing eraseblock 3497 -- 100 % complete
ubiformat: formatting eraseblock 4055 -- 100 % complete
-sh-3.2#
```

Note: This will take some time as the time taken for flashing File system depends on size of ubifs image

7. Restart the system and press any key to stop on the U-Boot prompt

```
# reboot
```

8. Booting Linux Kernel, UBI file system from NAND

   Following command must be applied at u-boot prompt (Marvell>>)

```
Marvell>> setenv console 'console=ttyS0,115200
mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs) rw'

Marvell>> setenv bootargs $(console) ubi.mtd=1,2048 root=ubi0:rootfs rootfstype=ubifs
fb=xgifb

Marvell>> setenv bootcmd 'nand read.e 0x800000 0x100000 0x400000; bootm
0x800000'

Marvell>> saveenv
```

   Note: Be careful while giving “saveenv” command. “saveenv” will save the parameters into NAND. From next reboot onwards, board will use the saved environment variables.

9. Reboot the system and the system should boot from NAND flash.

```
Marvell>> reset
```

5. The login and password to access the prompt are:
   a. Login – root, Password – nosoup4u (**root can not be used to login using GUI**)
   b. Login – admin, Password – openrd

8. USB Recovery

This document provides the instruction to boot Linux on the OpenRD from U-Boot using a USB flash key connected to the USB port on the OpenRD. The provided set of instructions allows the burning of the file system on to the NAND flash.

Following images and material are needed to run this process successfully:

Hardware:
1. USB Flash key with at least 512 MB capacity and ext2 or ext3 filesystem format.

Software:
1. ***flashware.img*** – Flashware image which is a concatenation of the uImage and the minimal root filesystem using squashfs format.
2. ***uImage.openrd*** – Latest uImage that can be used to boot up the kernel on the OpenRD.
3. ***fc8.release.x11.openrd.nand.ubi.img*** – Filesystem that is to be burned onto the NAND flash.

All the software images except the filesystem ***'fc8.release.x11.openrd.nand.ubi.img'*** have been included in the openrd_usb_recovery.zip file. See USB Recovery Sample Log, USB Terms for USB Recovery and Building flashware.img sections for more information.

Steps

1. Copy the flashware image (flashware.img), uImage (uImage.openrd) to the USB key. Also, copy the file system image (fedora jffs2 or ubifs) to the USB key.

3. Stop the auto-boot to enter the U-Boot prompt. At the U-Boot prompt, set the following parameters:

Marvell>> set loadaddr 0x2000000

Marvell>> set mtd1Size 0x400000 (setting mtd1 size to 4MB)

Marvell>> set filesize 0x45DD000 <size of the flashware.img in hex bytes>

Marvell>> saveenv

Marvell>> reset

4. After reset, stop the auto-boot to enter the U-Boot prompt. Connect the USB key, containing the flashware.img, uImage.openrd and file system images, to OpenRD board.
Type the command below:

Marvell>> rcvr

5. OpenRD should then boot up from the flashware.img image in the USB key. The logs of the boot-up process are shown in Appendix A.

6. At the Linux prompt, do the following:

-sh-3.2# cat /proc/mtd (This should list all the 2 mtd partitions for the u-boot, uImage and rootfs as shown below)

```
dev:    size      erasesize  name
mtd0: 00400000 00020000 "uImage"
mtd1: 1fb00000 00020000 "rootfs"

# fdisk -l

# mount /dev/sda(USB key) /mnt

# cd /mnt

# flash_eraseall /dev/mtd0

# flash_eraseall -j /dev/mtd1

# nandwrite -p /dev/mtd0 uImage.openrd

# ubiformat /dev/mtd1 -f fc8.release.x11.openrd.nand.ubi.img -s 2048 -O 2048

# reboot
```

7. Again stop the auto-boot and enter the following at the U-Boot prompt:

```
Marvell>> setenv console 'console=ttyS0,115200
mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs) rw'

Marvell>> setenv bootargs $(console) ubi.mtd=1,2048 root=ubi0:rootfs rootfstype=ubifs fb=xgifb

Marvell>> setenv bootcmd 'nand read.e 0x800000 0x100000 0x400000; bootm 0x800000'

Marvell>> saveenv
```

8.1 USB Recovery Sample Log

This section provides the log of the successful recovery process of booting Linux from U-Boot using the USB flash key.

```
         __  __                      _ _
        |  \/  | __ _ _ __ __   __ ___| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 22 2009 - 20:18:28) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz
```

```
DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

Checking for BootROM Routine Errors

No. of BootROM routine retries: 8
NAND: Nand ECC error

Running POST...

        DDR2 data bus test                              PASSED

        DDR2 address bus test                           PASSED

        UART 1 internal loopback test on baudrate   9600 PASSED
        UART 1 internal loopback test on baudrate  19200 PASSED
        UART 1 internal loopback test on baudrate 115200 PASSED

        Device: 0, Size: 512 MB, Page Size: 2 KB, Block Size: 128 KB
        NAND detection test                             PASSED

        Bad Block: 08e60000
        Bad Block: 18720000
        Bad Block: 1fb60000
        NAND bad-block detection test                   PASSED

        RTC test                                        PASSED

6/6 tests PASSED
POST completed

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>>
Marvell>> setenv loadaddr 0x2000000
Marvell>> setenv mtd1Size 0x400000
Marvell>> setenv filesize 0x45DD000
Marvell>> saveenv
Saving Environment to NAND...
Erasing Nand...Writing to Nand... done
```

```
Marvell>> reset
?
         __  __                      _ _
        |  \/  | __ _ _ ____   _____| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 22 2009 - 20:18:28) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

Checking for BootROM Routine Errors

No. of BootROM routine retries: 8
NAND: Nand ECC error

Running POST...

        DDR2 data bus test                                PASSED

        DDR2 address bus test                             PASSED

        UART 1 internal loopback test on baudrate   9600 PASSED
        UART 1 internal loopback test on baudrate  19200 PASSED
        UART 1 internal loopback test on baudrate 115200 PASSED

        Device: 0, Size: 512 MB, Page Size: 2 KB, Block Size: 128 KB
        NAND detection test                               PASSED

        Bad Block: 08e60000
        Bad Block: 18720000
        Bad Block: 1fb60000
        NAND bad-block detection test                     PASSED

        RTC test                                          PASSED

6/6 tests PASSED
POST completed
```

```
CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>> rcvr
USB:   scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
Trying to load image from USB flash drive using FAT FS
reading /flashware.img

** Unable to read "/flashware.img" from usb 0:1 **
Trying to load image from USB flash drive using ext2 FS partition 0
Failed to mount ext2 filesystem...
** Bad ext2 partition or disk - usb 0:0 **
Trying to load image from USB flash drive using ext2 FS partition 1
....
..............
.......................................................................
.......................................................................
........................
.......................................................................
.......................................................................
........................
.......................................................................
.......................................................................
........................
.......................................................................
.......................................................................
........................

73256960 bytes read
Update bootcmd

bootcmd: setenv bootargs $(console) root=/dev/ram0 rootfstype=squashfs
initrd=0x2400000,0x41dd000 ramdisk_size=67444 recovery=usb
serverip=0.0.0.0; bootm 0x2000000;
Booting the image (@ 0x2000000)...
## Booting image at 02000000 ...
   Image Name:   Linux-2.6.22.18
   Created:      2009-04-22  15:01:37 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2405764 Bytes =  2.3 MB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
OK

Starting kernel ...
```

```
Uncompressing
Linux.......................................................................
............................................................................
..........
Linux version 2.6.22.18 (dhaval@devbox) (gcc version 4.2.1) #1 Wed Apr
22 20:31:28 IST 2009
CPU: ARM926EJ-S [56251311] revision 1 (ARMv5TE), cr=00053177
Machine: Feroceon-KW
Using UBoot passing parameters structure
Memory policy: ECC disabled, Data cache writeback
CPU0: D VIVT write-back cache
CPU0: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU0: D cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
Built 1 zonelists.  Total pages: 130048
Kernel command line: console=ttyS0,115200 root=/dev/ram0
rootfstype=squashfs initrd=0x2400000,0x41dd000 ramdisk_size=67444
recovery=usb serverip=0.0.0.0
PID hash table entries: 2048 (order: 11, 8192 bytes)
Console: colour dummy device 80x30
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 256MB 256MB 0MB 0MB = 512MB total
Memory: 447360KB available (4376K code, 365K data, 140K init)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
NET: Registered protocol family 16

CPU Interface
-------------
SDRAM_CS0 ....base 00000000, size 256MB
SDRAM_CS1 ....base 10000000, size 256MB
SDRAM_CS2 ....disable
SDRAM_CS3 ....disable
PEX0_MEM ....base e8000000, size 128MB
PEX0_IO ....base f2000000, size   1MB
INTER_REGS ....base f1000000, size   1MB
NFLASH_CS ....base fa000000, size   2MB
SPI_CS ....base f4000000, size  16MB
BOOT_ROM_CS ....no such
DEV_BOOTCS ....no such
CRYPT_ENG ....base f0000000, size   2MB

  Marvell Development Board (LSP Version KW_LSP_4.2.7_patch2)-- OpenRD-
Client  Soc: 88F6281 A0 LE

 Detected Tclk 200000000 and SysClk 400000000
MV Buttons Device Load
Marvell USB EHCI Host controller #0: c65fa600
PEX0 interface detected Link X1
PCI: bus0: Fast back to back transfers disabled
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
Time: kw_clocksource clocksource has been installed.
NET: Registered protocol family 2
IP route cache hash table entries: 16384 (order: 4, 65536 bytes)
```

```
TCP established hash table entries: 65536 (order: 7, 524288 bytes)
TCP bind hash table entries: 65536 (order: 6, 262144 bytes)
TCP: Hash tables configured (established 65536 bind 65536)
TCP reno registered
checking if image is initramfs...it isn't (bad gzip magic numbers);
looks like an initrd
Freeing initrd memory: 67444K
RTC registered
Use the XOR engines (acceleration) for enhancing the following
functions:
  o RAID 5 Xor calculation
  o kernel memcpy
  o kenrel memzero
Number of XOR engines to use: 4
cesadev_init(c0011674)
mvCesaInit: sessions=640, queue=64, pSram=f0000000
Warning: TS unit is powered off.
MV Buttons Driver Load
squashfs: version 3.3 (2007/10/31) Phillip Lougher
squashfs: LZMA suppport for slax.org by jro
NTFS driver 2.1.28 [Flags: R/W].
JFFS2 version 2.2. (NAND) Â© 2001-2006 Red Hat, Inc.
io scheduler noop registered
io scheduler anticipatory registered (default)
XGIfb: Options <NULL>
XGIfb: Relocate IO address: 1000 [00001030]
XGIfb: Enable PCI device
XGIfb: Video ROM usage disabled

XGIfb: Enable PCI device
XGIfb: Video ROM usage disabled
XGIfb: SR14=0 DramSzie 200000 ChannelNum 1
XGIfb: Framebuffer at 0xe8000000, mapped to 0xe0c00000, size 2048k
XGIfb: MMIO at 0xec000000, mapped to 0xe0880000, size 256k
XGIfb: XGIInitNew() ...12345678910111215171818118218319202122232425OK
XGIfb: Memory heap starting at 4096K
XGIfb: Using MMIO queue mode
XGIfb: No or unknown bridge type detected
XGIfb: Default mode is 1024x768x16 (60Hz)
Console: switching to colour frame buffer device 128x48
XGIfb: Installed XGIFB_GET_INFO ioctl (80046ef8)
fb0:  frame buffer device, Version 0.8.01
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing
disabled
serial8250.0: ttyS0 at MMIO 0xf1012000 (irq = 33) is a 16550A
serial8250.0: ttyS1 at MMIO 0xf1012100 (irq = 34) is a 16550A
RAMDISK driver initialized: 20 RAM disks of 67444K size 4096 blocksize
Loading Marvell Ethernet Driver:
  o Cached descriptors in DRAM
  o DRAM SW cache-coherency
  o Single RX Queue support - ETH_DEF_RXQ=0
  o Single TX Queue support - ETH_DEF_TXQ=0
  o TCP segmentation offload enabled
  o Receive checksum offload enabled
  o Transmit checksum offload enabled
  o Network Fast Processing (Routing) supported
  o Driver ERROR statistics enabled
  o Driver INFO statistics enabled
  o Proc tool API enabled
  o Rx descripors: q0=128
  o Tx descripors: q0=532
  o Loading network interface(s):
    o eth0, ifindex = 1, GbE port = 0
    o eth1, ifindex = 2, GbE port = 1

mvFpRuleDb (dc3c0000): 16384 entries, 65536 bytes
Intel(R) PRO/1000 Network Driver - version 7.3.20-k2-NAPI
Copyright (c) 1999-2006 Intel Corporation.
e100: Intel(R) PRO/100 Network Driver, 3.5.17-k4-NAPI
e100: Copyright(c) 1999-2006 Intel Corporation
Integrated Sata device found
scsi0 : Marvell SCSI to SATA adapter
scsi1 : Marvell SCSI to SATA adapter
NFTL driver: nftlcore.c $Revision: 1.98 $, nftlmount.c $Revision: 1.41
$
NAND device: Manufacturer ID: 0xec, Chip ID: 0xdc (Samsung NAND 512MiB
3,3V 8-bit)
Scanning device for bad blocks
Bad eraseblock 1139 at 0x08e60000
Bad eraseblock 3129 at 0x18720000
Bad eraseblock 4059 at 0x1fb60000
Using static partition definition
Creating 3 MTD partitions on "nand_mtd":
0x00000000-0x00100000 : "u-boot"
0x00100000-0x00300000 : "uImage"
0x00300000-0x20000000 : "root"
ehci_marvell ehci_marvell.70059: Marvell Orion EHCI
ehci_marvell ehci_marvell.70059: new USB bus registered, assigned bus
number 1
ehci_marvell ehci_marvell.70059: irq 19, io base 0xf1050100
ehci_marvell ehci_marvell.70059: new USB bus registered, assigned bus
number 1
ehci_marvell ehci_marvell.70059: irq 19, io base 0xf1050100
ehci_marvell ehci_marvell.70059: USB 2.0 started, EHCI 1.00, driver 10
Dec 2004
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
USB Universal Host Controller Interface driver v3.0
usb 1-1: new high speed USB device using ehci_marvell and address 2
usb 1-1: configuration #1 chosen from 1 choice
hub 1-1:1.0: USB hub found
hub 1-1:1.0: 7 ports detected
usb 1-1.1: new high speed USB device using ehci_marvell and address 3
usb 1-1.1: configuration #1 chosen from 1 choice
usbcore: registered new interface driver usblp
drivers/usb/class/usblp.c: v0.13: USB Printer Device Class driver
Initializing USB Mass Storage driver...
scsi2 : SCSI emulation for USB Mass Storage devices
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
mice: PS/2 mouse device common for all mice
i2c /dev entries driver
Linux telephony interface: v1.00
Marvell Telephony Driver:
```

```
Warning Tdm is Powered Off
md: linear personality registered for level -1
md: raid0 personality registered for level 0
md: raid1 personality registered for level 1
raid6: int32x1     98 MB/s
raid6: int32x2    114 MB/s
raid6: int32x4    123 MB/s
raid6: int32x8    111 MB/s
raid6: using algorithm int32x4 (123 MB/s)
md: raid6 personality registered for level 6
md: raid5 personality registered for level 5
md: raid4 personality registered for level 4
raid5: measuring checksumming speed
   arm4regs  :  1089.600 MB/sec
   8regs     :   758.800 MB/sec
   32regs    :   904.400 MB/sec
raid5: using function: arm4regs (1089.600 MB/sec)
device-mapper: ioctl: 4.11.0-ioctl (2006-10-12) initialised: dm-
devel@redhat.com
dm_crypt using the OCF package.
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
mvsdmmc: irq =28 start f1090000
mvsdmmc: irq_detect=93
usbcore: registered new interface driver usbhid
drivers/hid/usbhid/hid-core.c: v2.6:USB HID core driver
Advanced Linux Sound Architecture Driver Version 1.0.14 (Thu May 31
09:03:25 2007 UTC).
ALSA device list:
  #0: Marvell mv88fx_snd ALSA driver
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
RAMDISK: squashfs filesystem found at block 0
RAMDISK: Loading 67444KiB [1 disk] into ram disk... done.
VFS: Mounted root (squashfs filesystem) readonly.
Freeing init memory: 140K
init started: BusyBox v1.7.0 (2008-02-26 19:25:17 IST)
starting pid 305, tty '': '/etc/init.d/rcS'
starting pid 307, tty '': '/bin/sh'
-sh-3.2# scsi 2:0:0:0: Direct-Access     JetFlash TS2GJFV30        8.07
PQ: 0 ANSI: 2
sd 2:0:0:0: [sda] 4014078 512-byte hardware sectors (2055 MB)
sd 2:0:0:0: [sda] Write Protect is off
sd 2:0:0:0: [sda] Assuming drive cache: write through
sd 2:0:0:0: [sda] 4014078 512-byte hardware sectors (2055 MB)
sd 2:0:0:0: [sda] Write Protect is off
sd 2:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 2:0:0:0: [sda] Attached SCSI removable disk
sd 2:0:0:0: Attached scsi generic sg0 type 0

-sh-3.2#
```

8.2 U-Boot Terms for USB Recovery

The U-Boot sources provided in the openrd_uboot.zip file includes the necessary modification that enables the USB recovery procedure. It is important to know certain terms that can help you debug the USB recovery procedure if it does not run successfully.

It is important to check the following parameters in the U-Boot if the USB recovery procedure fails:

a) ***mtd1Size*** – This is the size of the mtd partition ‘1’. This is the NAND partition where the uImage is stored after writing it to the NAND flash.
b) ***filesize*** – This is the parameter where the file size of the flashware.img is defined in hex.
c) ***loadaddr*** – It is the load address for the u-boot to load the flashware.img in RAM. By default, it is set to 0x2000000.

Following commands are used at the u-boot prompt to check the above variables to confirm they are correct.

Marvell>> print mtd1Size

Marvell>> print loadaddr

Marvell>> print filesize

Below is the log of the U-Boot prompt using the above commands:

```
         __  __                      _ _
        |  \/  | __ _ _ ____   _____| | |
        | |\/| |/ _` | '__\ \ / / _ \ | |
        | |  | | (_| | |   \ V /  __/ | |
        |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
 ** MARVELL BOARD: OpenRD-Client LE

U-Boot 1.1.4 (Apr 22 2009 - 20:18:28) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

Checking for BootROM Routine Errors
```

```
No. of BootROM routine retries: 8
NAND: Nand ECC error

Running POST...

        DDR2 data bus test                                PASSED

        DDR2 address bus test                             PASSED

        UART 1 internal loopback test on baudrate   9600 PASSED
        UART 1 internal loopback test on baudrate  19200 PASSED
        UART 1 internal loopback test on baudrate 115200 PASSED

        Device: 0, Size: 512 MB, Page Size: 2 KB, Block Size: 128 KB
        NAND detection test                               PASSED

        Bad Block: 08e60000
        Bad Block: 18720000
        Bad Block: 1fb60000
        NAND bad-block detection test                     PASSED

        RTC test                                          PASSED

6/6 tests PASSED
POST completed

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

Module 0 is AUDIO
Module 1 is RGMII

USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>> rcvr
USB:   scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
Trying to load image from USB flash drive using FAT FS
reading /flashware.img

** Unable to read "/flashware.img" from usb 0:1 **
Trying to load image from USB flash drive using ext2 FS partition 0
Failed to mount ext2 filesystem...
** Bad ext2 partition or disk - usb 0:0 **
Trying to load image from USB flash drive using ext2 FS partition 1
....
..............
.......................................................................
.......................................................................
........................
.......................................................................
.......................................................................
........................
```

```
......................................................................
......................................................................
........................
......................................................................
......................................................................
........................

73256960 bytes read
Update bootcmd
```

8.3 Building flashware.img

This section provides the steps to create the *flashware.img* for the USB recovery process. Please make sure to build a uImage with the following support enabled as default:

1. squashfs support
2. initrd support
3. RAMDISK support with the following
a. (20) Default number of RAM disks
b. (4096) Default RAM disk size (kbytes)
c. (4096) Default RAM disk block size (bytes) (NEW)

You can enable the support for the above features in the uImage by using ***'make menuconfig'*** command after making the *'mv88f6281_defconfig'* configuration file, while building the kernel and the LSP. Refer to section for information in configuring and building the kernel and
LSP for OpenRD.

Once the uImage is created with the initrd, RAMDISK and squashfs support, do the following on the Linux machine.

1. Find the size of the uImage. Usually, the uImage file is around 2.3 MB.

# ls –al uImage.openrd

2. For the recovery procedure, the mtd partition for uImage is set to be 0x400000, 4MB or 4194304 bytes. Hence we need to create a uImage.img file which is 4MB in size. Hence we need to use a pad file called uimage.pad. The size of the uImage.pad file is obtained by using the following equation:

size = *(*4194304 *bytes) – <size of uImage>*

3. On the linux host, use the following command to create a uImage.pad file.

*# dd if=/dev/zero of=uImage.pad bs=(**size**) count=1*

4. Create the uImage.img file using the cat command as below.

# cat uimage.openrd uImage.pad >> uImage.img

5. Use the rootfsv1.0 from the openrd_host_swsupportpackage/linux/rootfs.tar.gz to create a squashfs image 'rootfs.img'. The rootfs.img can be created by using the command as below.

# mksquashfs rootfsv1.0 rootfs.img –b 4096

```
Parallel mksquashfs: Using 1 processor
Creating little endian 3.1 filesystem on rootfs.img, block size 4096.
[===============================================================
=================================================== ] 40467/40707  99%
Exportable Little endian filesystem, data block size 4096, compressed data, compressed
metadata, compressed fragments, duplicates are removed
Filesystem size 67443.06 Kbytes (65.86 Mbytes)
        44.81% of uncompressed filesystem size (150522.31 Kbytes)
Inode table size 139624 bytes (136.35 Kbytes)
        38.53% of uncompressed inode table size (362384 bytes)
Directory table size 68147 bytes (66.55 Kbytes)
        58.31% of uncompressed directory table size (116878 bytes)
Number of duplicate files found 18
Number of inodes 7346
Number of files 5187
Number of fragments 1751
Number of symbolic links  322
Number of device nodes 1596
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 241
Number of uids 2
        dhaval (500)
        root (0)
Number of gids 1
        unknown (1000)
```

6. Concatenate the uImage.img and rootfs.img image files to get flashware.img.

```
# cat uImage.img rootfs.img >> flashware.img
```

9. Interfaces/Peripherals Not Tested

The interfaces provided on header are not tested.

10. Known Issues

1. Writing u-boot using OpenOCD causes u-boot POST code to print. This is due to limitation of OpenOCD.

   ```
   No. of BootROM routine retries: 8
   NAND: Nand ECC error
   ```

   Workaround:
   Re-write u-boot using u-boot; this error will be removed. U-boot can be reflashed using U-boot using following procedure:

   Following procedure assumes tftp server setup on 192.168.1.2/255.255.255.0, user will need to change ipaddr, netmask and serverip environment variables according to network configurations to which OpenRD is connected.

   Marvell>> setenv ipaddr 192.168.1.1
   Marvell>> setenv netmask 255.255.255.0
   Marvell>> setenv serverip 192.168.1.2
   Marvell>> bubt u-boot.bin.openrd

2. GUI performance using Fedora Core 8, Ubuntu 9.04 Jaunty file systems is slow at times.
3. Compiling u-boot, and Linux Kernel generate multiple warning; these warnings can be safely ignored.
4. '**top**' command on Ubuntu 9.04 Jaunty file system shows > 50% CPU utilization after login using GUI. This is determined to be bug with top command.

11. Power Supply Notice

Power supply requirement for the OpenRD-Client is D.C 12V/3A.

12. Further Reading

eInfochips Web store : http://www.einfochips.com/marvell

Downloads:
http://www.marvell.com/products/embedded_processors/developer/kirkwood/openrd.jsp

Appendix A

Configuring the Console

Connect the USB to mini-USB console cable in the mini-USB connector on the OpenRD.

1. Preparing Minicom as the Board Console (Linux)

Pre-requisite

1. Make sure ftdi_sio.ko on Linux Host is a loadable module. FTDI driver for Linux is merged with stock Linux Kernel for all kernels above v2.6.10 (stock kernel can be obtained from www.kernel.org) have FTDI driver built-in. If it's already loaded, unload it using following command:

   # rmmod ftdi_sio

2. Re-load ftdi_si.ko on Linux Host using OpenRD VID/PID; and power on the system

   # modprobe ftdi_sio vendor=0x0403 product=0x9e90

   If driver are loaded properly, giving following command should list two USB devices – namely /dev/ttyUSB0, /dev/ttyUSB1.

   *# ls –l /dev/ttyUSB**

Below are the configuration settings for minicom:

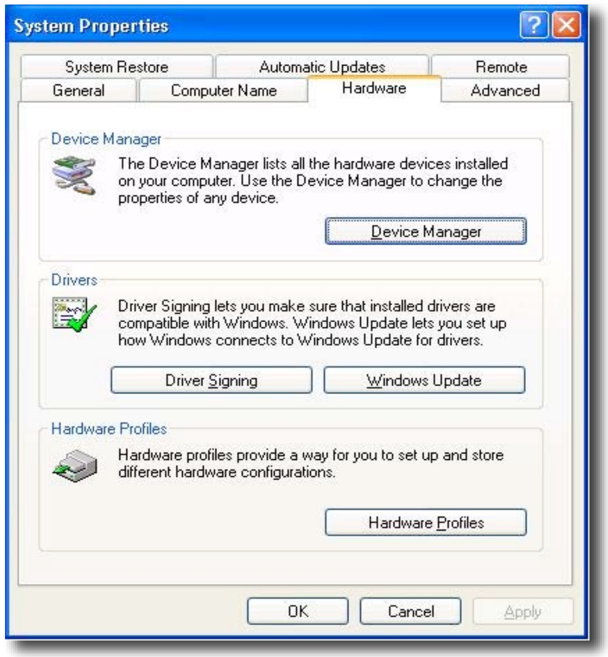
1. Login as root.
2. Execute minicom -s.
3. From the menu select Serial port setup.
4. Set Serial device field to /dev/ttyUSB1. (Assuming development board connected to USB port)
5. Set Bps/Par/Bits field to 115200 8N1.
6. Set Hardware Flow Control to No.
7. Set Software Flow Control to No.
8. Select Exit.
9. Re-run minicom, using command – *minicom -o*

2. Preparing a Terminal as the Board Console (Windows)

Use the CDM 2.04.14_OpenRD package at *../openrd_host_swsupportpackage/windows/openrd-ftdi-driver.zip* to install the mini-USB-to-USB debug console driver. Below are the configuration settings for Terminal:

1. On Windows Host, Check 'Ports' from the 'Device Manager'. It should display two USB serial ports with different COM port number in the list. The higher COM port number of this is used for console. Steps to verify this:

   a. Right click on '**My Computer**'.
   b. Select '**Properties**' option.
   c. Select '**Hardware**' tab.



The Device Manager lists all the hardware devices installed on your computer. Use the Device Manager to change the properties of any device.

Driver Signing lets you make sure that installed drivers are compatible with Windows. Windows Update lets you set up how Windows connects to Windows Update for drivers.

Hardware profiles provide a way for you to set up and store different hardware configurations.

Figure 10 System Properties

2. Click on '**Device Manager**' button.

3. Expand '**Ports (COM & LPT)**' label as shown below. As seen below COM4 is higher number; and thus, COM4 should be used as console.

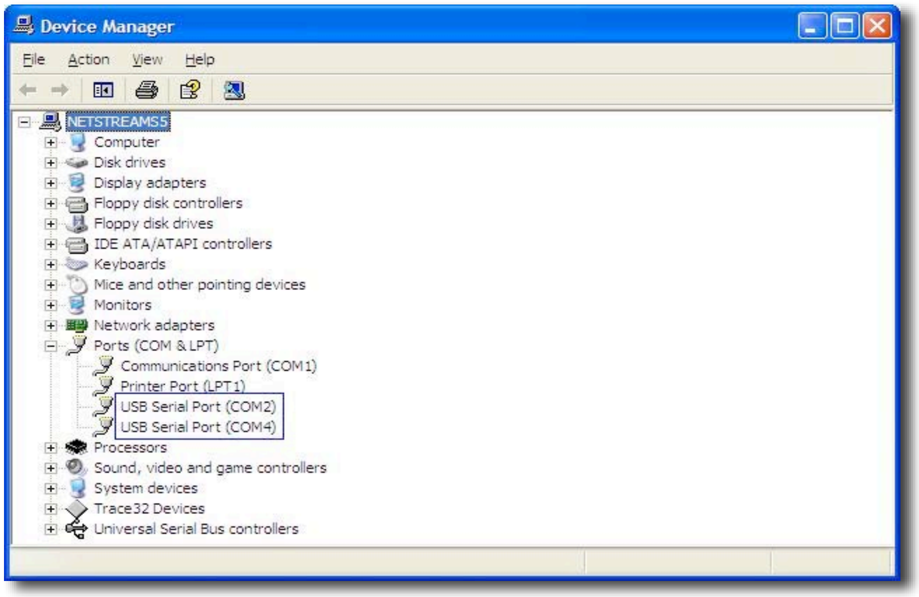
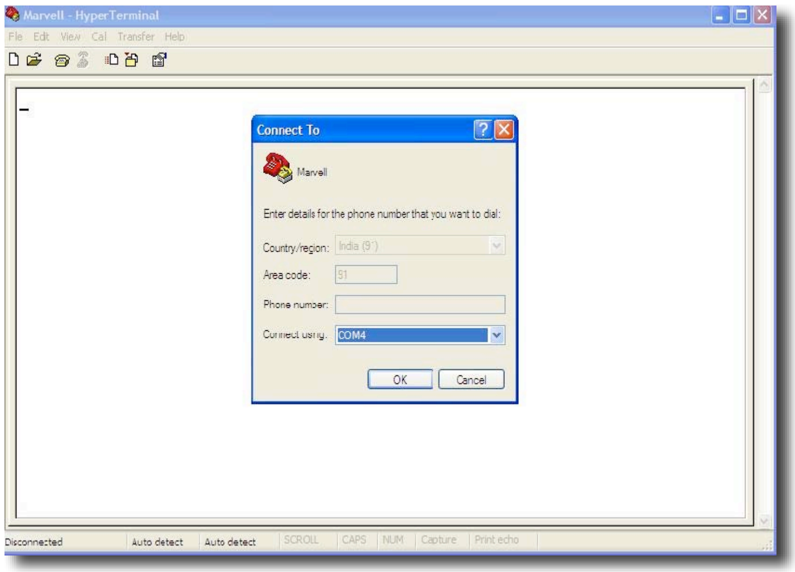


Figure 11 Device Manager

4. Launch Hyper Terminal application. Select COM port for console as indicated by previous point.



Connect To

Marvell

Enter details for the phone number that you want to dial:

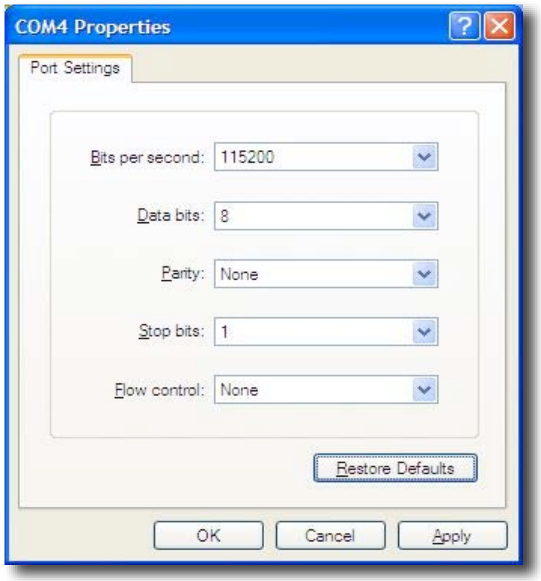
Country/region: India (91)

Area code: 91

Phone number:

Connect using: COM4

Figure 12 Hyper Terminal



COM4 Properties

Bits per second: 115200

Data bits: 8

Parity: None

Stop bits: 1

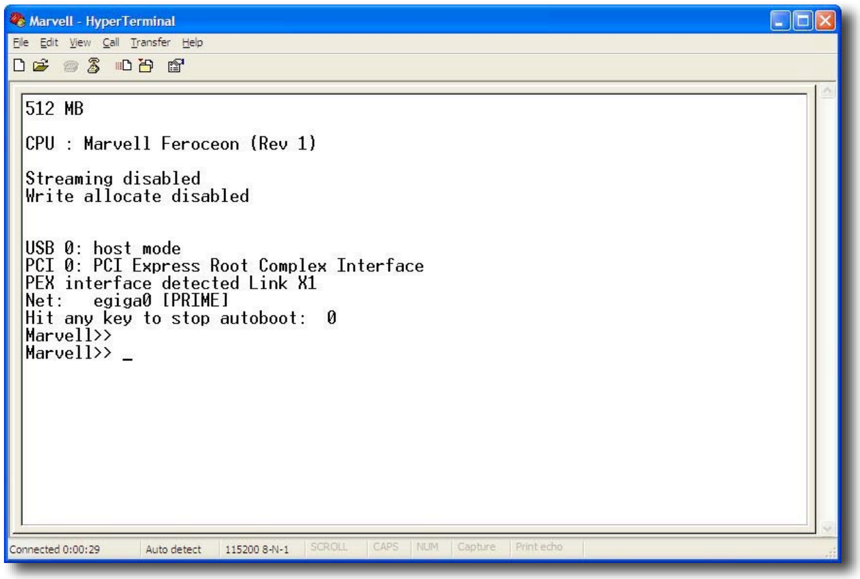
Flow control: None

Figure 13 COM Properties

5. Set the Configure properties as follows:
   - Bits per sec field to *115200*
   - Data bits to 8
   - Parity to *None*
   - Stop bit to *1*
   - Flow Control to *None*

6. Now click on 'Call' button from toolbar. This will help OpenRD-Client to show console to the Hyper Terminal. The board will boot itself and after boot up, it will provide Linux prompt.

(Note: If you are not getting any message/prints on hyper terminal, press “Disconnect” button from toolbar, press reset switch on board and press “Call” button again).



```
512 MB

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled


USB 0: host mode
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME]
Hit any key to stop autoboot:  0
Marvell>>
Marvell>> _
```

Figure 14 U-Boot prompt on Console

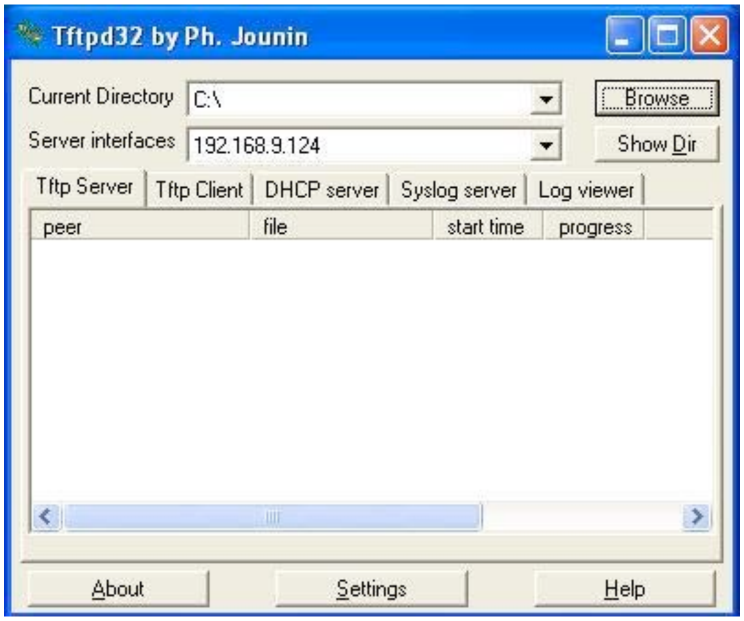
Appendix B

Configuring TFTP server/daemon

On Windows Host

Install Tftpd32 from http://tftpd32.jounin.net/tftpd32_download.html

Run Tftpd32, click on browse for locating tftproot directory i.e. directory where uImage.openrd, u-boot.bin.openrd are located.



Current Directory C:\
Server interfaces 192.168.9.124

Figure 15 Configuring TFTP Server on Host PC

On Linux Host

Pre-requisite

- **xinetd** daemon installed on Linux Host, if tftp-server is not running as standalone daemon.

Steps

Download the rpm for host machine distribution from http://rpmfind.net/linux/rpm2html/search.php?query=tftp-server

After downloading rpm, install it as:

# rpm –ivh <tftp-server-rpmname>

OR

For Linux host distributions such as ubuntu, type following command on Linux host (**Note:** internet connection is required to carry out this step)

# sudo apt-get install tftp-server

Appendix C

GCC cross-compiler, mkimage

Follow the instructions to install the gcc cross-compiler on the Linux host system. Path for the gcc.tar.bz2 package is <dvd-home>/openrd-devkit-v1.0*/openrd_host_swsupportpackage/linux/.* **Make sure to run all command using root user.**

1. Copy gcc.tar.bz2 to the /home/openrd directory on Linux host

2. Go to the /home/openrd directory. Create directory named “toolchain”

```
# cd /home/openrd

/home/openrd # mkdir toolchain

/home/openrd # cd toolchain

/home/openrd/toolchain #
```

3. Untar the gcc package

```
/home/openrd/toolchain # tar -xjvf /home/openrd/gcc.tar.bz2
```

4. Add the gcc path to the working directory

```
/home/openrd/toolchain # export PATH=/home/openrd/toolchain/gcc/bin:$PATH

/home/openrd/toolchain # cd /home/openrd/
```

5. Copy “mkimage” binary located to HOST Linux machine in the “/usr/bin” directory.

```
/home/openrd/toolchain # cp <dvd-home>/openrd-devkit-v1.0/openrd_uboot/tools/mkimage
/usr/bin/
```

Appendix D

Configuring NFS on Linux Host

This section provides the information on the settings needed on the Linux host to configure NFS.

Create NFS root directory. **Make sure following commands are applied using root user.** Path for rootfs.tar.gz package is
<dvd-home>/openrd-devkit- v1.0*/openrd_host_swsupportpackage/linux/.*

```
/home/openrd # mkdir nfs
/home/openrd # cd nfs
/home/openrd # tar zxvf rootfs.tar.gz
/home/openrd # mv rootfsv1.0 rootfs
```

1. Add the following lines to the /etc/exports script and save.

```
/home/openrd/nfs/rootfs *(rw,sync,no_root_squash)
/tftpboot/ *(rw,sync,no_root_squash)
```

2. Restart the ‘NFS’ server.

```
/home/openrd # /etc/init.d/nfs restart
Shutting down NFS mountd:   [ OK ]
Shutting down NFS daemon:   [ OK ]
Shutting down NFS quotas:   [ OK ]
Shutting down NFS services:   [ OK ]
Starting NFS services:    [ OK ]
Starting NFS quotas:     [ OK ]
Starting NFS daemon:    [ OK ]
Starting NFS mountd:    [ OK ]
/home/openrd #
```

After restarting NFS, list NFS exports using following commands:

```
/home/openrd # nfs-export --list
```

Appendix E

Boot OpenRD from NFS

This section provides the information on the settings needed on the Linux host and the OpenRD U-Boot to boot OpenRD from NFS.

A. Linux Host Settings

On the Linux host follow the steps below:

1. Copy the 'uImage.openrd' to the 'tftpboot' folder. **Here, it's assumed that /home/openrd/uImage.openrd is already created.**

   /home/openrd # cp –a /home/openrd/uImage.openrd/tftpboot

2. Copy file system image(s) to NFS root directory.

   For JFFS2
   # cp /home/openrd/fc8.release.x11.openrd.nand.jffs2 /home/openrd/nfs/rootfs/

   For UBIFS
   # cp /home/openrd/fc8.release.x11.openrd.nand.ubi.img /home/openrd/nfs/rootfs/

3. Copy uImage (kernel) to NFS root directory.

   # cp /home/openrd/uImage.openrd /home/openrd/nfs/rootfs/

B. OpenRD U-Boot Settings

On the OpenRD reference board follow the steps below to boot from NFS.

```
Marvell>> set ipaddr 192.168.1.1
Marvell>> set netmask 255.255.255.0
Marvell>> set serverip 192.168.1.yyy
Marvell>> set rootpath /home/openrd/nfs/rootfs
Marvell>> set image_name uImage.openrd
Marvell>> setenv console 'console=ttyS0,115200
mtdparts=nand_mtd:0x400000@0x100000(uImage),0x1fb00000@0x500000(rootfs)
rw'
Marvell>> set bootargs_root ‘root=/dev/nfs rw’
Marvell>> set bootargs_end ‘::$(netmask):DB88FXX81:eth0:none’
Marvell>> set bootcmd ‘tftpboot 0x2000000 $(image_name);setenv bootargs
$(console) $(bootargs_root) nfsroot=$(serverip):$(rootpath)
ip=$(ipaddr):$(serverip)$(bootargs_end); bootm 0x2000000’
Marvell>> saveenv
```

Note: Here, environment variable serverip must be set to Linux host having NFS, TFTP server setup.

Appendix F

MTD-Utils (UBIFS, JFFS2)

Follow the instructions to install the mtd-utils on the Linux host system. Path for the mtd-utils.tar.gz package is <dvd-home>/openrd-devkit-v1.0*/openrd_filesystem/binary/.* **Make sure to run all command using root user.**

1. Copy mtd-utils-bin.tar.gz to the /home/openrd directory on Linux host

2. Go to the /home/openrd directory. Create directory named “tools”

```
# cd /home/openrd

/home/openrd # mkdir tools

/home/openrd # cd tools

/home/openrd/tools #
```

3. Untar the mtd-utils package

```
/home/openrd/tools # tar -zxvf /home/openrd/mtd-utils-bin.tar.gz
```

4. Add the mtd-utils path to the working directory

```
/home/openrd/tools # export PATH=/home/openrd/tools:$PATH

/home/openrd/tools # cd /home/openrd/
```

Appendix G

Installing Packages in the Jaunty File System

1. Installing samba

```
root@debian:~#
root@debian:~#
root@debian:~# apt-get install samba
Reading package lists... 0%Reading package lists... 0%Reading
package lists... 1%Reading package lists... 8%Reading package
lists... 15%Reading package lists... 22%Reading package lists...
26%Reading package lists... 26%Reading package lists... 26%Reading
package lists... 26%Reading package lists... 28%Reading package
lists... 34%Reading package lists... 40%Reading package lists...
46%Reading package lists... 52%Reading package lists... 58%Reading
package lists... 65%Reading package lists... 71%Reading package
lists... 77%Reading package lists... 82%Reading package lists...
87%Reading package lists... 92%Reading package lists... 96%Reading
package lists... 97%Reading package lists... 97%Reading package
lists... 99%Reading package lists... 99%Reading package lists...
Done
Building dependency tree... 0%Building dependency tree... 0%Building
dependency tree... 0%Building dependency tree... 50%Building
dependency tree... 50%Building dependency tree... 75%Building
dependency tree
Reading state information... 0%Reading state information...
3%Reading state information... Done
samba is already the newest version.
The following packages were automatically installed and are no
longer required:
  libx11-data libxcb1 libxau6 libxdmcp6 libxcb-xlib0 libx11-6
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
root@debian:~#
root@debian:~#
root@debian:~#
```

2. Checking for Updates

```
root@debian:~#
root@debian:~#
root@debian:~# apt-get update
0% [Working]            Get:1 http://ports.ubuntu.com jaunty
Release.gpg [189B]
            0% [1 Release.gpg 0/189B 0%]
99% [Working]             Get:2 http://ports.ubuntu.com jaunty
Release [74.6kB]
             3% [2 Release 2623/74.6kB 3%]
29% [2 Release 21603/74.6kB 28%]                               99%
[Working]             99% [2 Release gpgv 74637]
99% [Working]             99% [Waiting for headers]
Get:3 http://ports.ubuntu.com jaunty/main Packages [1260kB]
                         5% [3 Packages 0/1260kB 0%]5% [3 Packages
0/1260kB 0%]5% [3 Packages 0/1260kB 0%]5% [3 Packages 0/1260kB 0%]
7% [3 Packages 20480/1260kB 1%]                              15% [3
Packages 135168/1260kB 10%]26% [3 Packages 278528/1260kB 22%]55% [3
Packages 663552/1260kB 52%]                                  86% [3
Packages 1077248/1260kB 85%]                                   99%
[Working]
214kB/s 0s99% [3 Packages bzip2 0] [Waiting for headers]
214kB/s 0s
Get:4 http://ports.ubuntu.com jaunty/restricted Packages [1221B]
99% [3 Packages bzip2 0] [Waiting for headers]
214kB/s 0s99% [3 Packages bzip2 309248] [Waiting for headers]
214kB/s 0s99% [3 Packages bzip2 635904] [Waiting for headers]
214kB/s 0s99% [3 Packages bzip2 894976] [Waiting for headers]
214kB/s 0s
Get:5 http://ports.ubuntu.com jaunty/universe Packages [4347kB]
23% [3 Packages bzip2 894976] [5 Packages 4071/4347kB 0%]
214kB/s 20s24% [3 Packages bzip2 1174528] [5 Packages 61011/4347kB
1%]         214kB/s 20s29% [3 Packages bzip2 1254400] [5 Packages
352256/4347kB 8%]        214kB/s 18s36% [3 Packages bzip2 1254400]
[5 Packages 761856/4347kB 17%]       214kB/s 16s44% [3 Packages
bzip2 1254400] [5 Packages 1179648/4347kB 27%]      214kB/s 14s51%
[3 Packages bzip2 1254400] [5 Packages 1593344/4347kB 36%]
214kB/s 12s58% [3 Packages bzip2 1254400] [5 Packages 2007040/4347kB
46%]      214kB/s 10s61% [3 Packages bzip2 1454080] [5 Packages
2185311/4347kB 50%]      214kB/s 10s65% [3 Packages bzip2 1619968]
[5 Packages 2401391/4347kB 55%]       214kB/s 9s68% [3 Packages
bzip2 1799168] [5 Packages 2569291/4347kB 59%]       415kB/s 4s71%
[3 Packages bzip2 1944576] [5 Packages 2721131/4347kB 62%]
415kB/s 3s74% [3 Packages bzip2 2115584] [5 Packages 2895872/4347kB
66%]       415kB/s 3s77% [3 Packages bzip2 2270208] [5 Packages
3094891/4347kB 71%]       415kB/s 3s81% [3 Packages bzip2 2434048]
[5 Packages 3277391/4347kB 75%]       415kB/s 2s84% [3 Packages
bzip2 2579456] [5 Packages 3493888/4347kB 80%]       415kB/s 2s88%
[3 Packages bzip2 2699264] [5 Packages 3706631/4347kB 85%]
415kB/s 1s92% [3 Packages bzip2 2765824] [5 Packages 3919872/4347kB
90%]       415kB/s 1s94% [3 Packages bzip2 2989056] [5 Packages
4049731/4347kB 93%]       415kB/s 0s97% [3 Packages bzip2 3175424]
[5 Packages 4181131/4347kB 96%]       415kB/s 0s99% [3 Packages
bzip2 3325952] [5 Packages 4346111/4347kB 99%]       415kB/s 0s
Get:6 http://ports.ubuntu.com jaunty/multiverse Packages [153kB]
97% [3 Packages bzip2 3338240] [6 Packages 0/153kB 0%]
415kB/s 0s99% [3 Packages bzip2 3469312] [6 Packages 146092/153kB
95%]         415kB/s 0s99% [3 Packages bzip2 3470336]
415kB/s 0s99% [3 Packages bzip2 3731456]…..
..
..
..
..
..            99% [6 Packages bzip2 0]                         99% [6
Packages bzip2 234496]99% [6 Packages bzip2 372736]99% [6 Packages
bzip2 372736]99% [6 Packages bzip2 372736]99% [6 Packages bzip2
372736]99% [6 Packages bzip2 372736]99% [6 Packages bzip2 487424]99%
 [6 Packages bzip2 640000]                            100% [Working]
Fetched 5836kB in 1min4s (89.8kB/s)
Reading package lists... 0%Reading package lists... 0%Reading
package lists... 1%Reading package lists... 13%Reading package
lists... 22%Reading package lists... 26%Reading package lists...
26%Reading package lists... 26%Reading package lists... 26%Reading
package lists... 29%Reading package lists... 40%Reading package
lists... 49%Reading package lists... 61%Reading package lists...
72%Reading package lists... 81%Reading package lists... 91%Reading
```

```
package lists... 97%Reading package lists... 97%Reading package
lists... 98%Reading package lists... 99%Reading package lists...
99%Reading package lists... Done
root@debian:~#
root@debian:~#
root@debian:~#
```

3. Installing vim

```
root@debian:~#
root@debian:~#
root@debian:~# apt-get install vim
Reading package lists... 0%Reading package lists... 0%Reading
package lists... 48%Reading package lists... Done
Building dependency tree... 0%Building dependency tree... 0%Building
dependency tree... 50%Building dependency tree... 50%Building
dependency tree... 79%Building dependency tree
Reading state information... 0%Reading state information...
2%Reading state information... Done
The following packages were automatically installed and are no
longer required:
  libx11-data libapr1 libxcb1 libxau6 libxdmcp6 ssl-cert libxcb-
xlib0 libx11-6
  libpq5
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 34 not upgraded.
Need to get 6656kB of archives.
After this operation, 26.7MB of additional disk space will be used.
Do you want to continue [Y/n]? y
0% [Working]            Get:1 http://ports.ubuntu.com jaunty/main
vim-runtime 2:7.2.079-1ubuntu1 [5809kB]
            0% [1 vim-runtime 0/5809kB 0%]
0% [1 vim-runtime 43483/5809kB 0%]
3% [1 vim-runtime 208463/5809kB 3%]
9% [1 vim-runtime 601203/5809kB 10%]
20% [1 vim-runtime 1357483/5809kB 23%]29% [1 vim-runtime
1967763/5809kB 33%]40% [1 vim-runtime 2668563/5809kB 45%]60% [1 vim-
runtime 4034287/5809kB 69%]68% [1 vim-runtime 4575323/5809kB 78%]
87% [Working]            Get:2 http://ports.ubuntu.com jaunty/main
vim 2:7.2.079-1ubuntu1 [847kB]
             87% [2 vim 0/847kB 0%]                      100%
[Working]             Fetched 6656kB in 5s (1298kB/s)
Selecting previously deselected package vim-runtime.
(Reading database ... 11606 files and directories currently
installed.)
Unpacking vim-runtime (from .../vim-runtime_2%3a7.2.079-
1ubuntu1_all.deb) ...
Adding `diversion of /usr/share/vim/vim72/doc/help.txt to
/usr/share/vim/vim72/doc/help.txt.vim-tiny
 by vim-runtime'
Adding `diversion of /usr/share/vim/vim72/doc/tags to
/usr/share/vim/vim72/doc/tags.vim-tiny by vim-
runtime'
```

```
Selecting previously deselected package vim.
Unpacking vim (from .../vim_2%3a7.2.079-1ubuntu1_armel.deb) ...
Setting up vim-runtime (2:7.2.079-1ubuntu1) ...
Processing /usr/share/vim/addons/doc

Setting up vim (2:7.2.079-1ubuntu1) ...

root@debian:~#
root@debian:~#
root@debian:~#
```

4. Installing gdb

```
root@debian:~#
root@debian:~#
root@debian:~# apt-get install gdb
Reading package lists... 0%Reading package lists... 100%Reading
package lists... Done
Building dependency tree... 0%Building dependency tree... 0%Building
dependency tree... 50%Building dependency tree... 50%Building
dependency tree... 80%Building dependency tree
Reading state information... 0%Reading state information...
3%Reading state information... Done
The following packages were automatically installed and are no
longer required:
  libx11-data libxcb1 libxau6 libxdmcp6 libxcb-xlib0 libx11-6
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  libexpat1
Suggested packages:
  gdb-doc
The following NEW packages will be installed:
  gdb libexpat1
0 upgraded, 2 newly installed, 0 to remove and 34 not upgraded.
Need to get 3003kB of archives.
After this operation, 6423kB of additional disk space will be used.
Do you want to continue [Y/n]? y
0% [Working]            Get:1 http://ports.ubuntu.com jaunty/main
libexpat1 2.0.1-4 [119kB]
0% [1 libexpat1 0/119kB 0%]                           1%
[1 libexpat1 43483/119kB 36%]                                Get:2
http://ports.ubuntu.com jaunty/main gdb 6.8-3ubuntu2 [2884kB]
                                4% [2 gdb 4828/2884kB 0%]
8% [2 gdb 136228/2884kB 4%]                           12% [2 gdb
269088/2884kB 9%]                             18% [2 gdb
432608/2884kB 14%]23% [2 gdb 574228/2884kB 19%]28% [2 gdb
743588/2884kB 25%]34% [2 gdb 931928/2884kB 32%]
40% [2 gdb 1102748/2884kB 38%]48% [2 gdb 1337808/2884kB 46%]54% [2
gdb 1531988/2884kB 53%]                              61% [2 gdb
1739308/2884kB 60%]                                       299kB/s
3s70% [2 gdb 2010868/2884kB 69%]
299kB/s 2s76% [2 gdb 2181688/2884kB 75%]
299kB/s 2s84% [2 gdb 2410908/2884kB 83%]
299kB/s 1s90% [2 gdb 2597788/2884kB 90%]
299kB/s 0s97% [2 gdb 2797808/2884kB 97%]
299kB/s 0s100% [Working]
299kB/s 0s
Fetched 3003kB in 8s (338kB/s)
```

```
Selecting previously deselected package libexpat1.
(Reading database ... 10231 files and directories currently
installed.)
Unpacking libexpat1 (from .../libexpat1_2.0.1-4_armel.deb) ...
Selecting previously deselected package gdb.
Unpacking gdb (from .../gdb_6.8-3ubuntu2_armel.deb) ...
Setting up libexpat1 (2.0.1-4) ...

Setting up gdb (6.8-3ubuntu2) ...

Processing triggers for libc6 ...
ldconfig deferred processing now taking place
root@debian:~#
root@debian:~#
root@debian:~#
```

5. Installing tftp

```
root@debian:~#
root@debian:~#
root@debian:~# apt-get install tftp
Reading package lists... 0%Reading package lists... 0%Reading
package lists... 48%Reading package lists... Done
Building dependency tree... 0%Building dependency tree... 0%Building
dependency tree... 50%Building dependency tree... 50%Building
dependency tree... 78%Building dependency tree
Reading state information... 0%Reading state information...
2%Reading state information... Done
The following packages were automatically installed and are no
longer required:
  libx11-data libapr1 libxcb1 libxau6 libxdmcp6 ssl-cert libxcb-
xlib0 libx11-6
  libpq5
Use 'apt-get autoremove' to remove them.
The following NEW packages will be installed:
tftp
0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.
Need to get 19.4kB of archives.
After this operation, 86.0kB of additional disk space will be used.
0% [Working]            Get:1 http://ports.ubuntu.com
jaunty/universe tftp 0.17-17ubuntu1 [19.4kB]
            0% [1 tftp 0/19.4kB 0%]                       100%
[Working]              Fetched 19.4kB in 0s (32.9kB/s)
Selecting previously deselected package tftp.
(Reading database ... 13131 files and directories currently
installed.)
Unpacking tftp (from .../tftp_0.17-17ubuntu1_armel.deb) ...
Setting up tftp (0.17-17ubuntu1) ...
root@debian:~#
root@debian:~#
root@debian:~#
```

About eInfochips

eInfochips is a worldwide leading Product Development Services company providing a full range of services including Chip/ASIC Design, Embedded System and Software Product development. eInfochips has been in business since 1994 and has developed strong expertise in system design, including hardware and software design and verification.

For additional information, please visit http://www.einfochips.com

Technical Question
Email: openrd@einfochips.com

Contact Us
Email: sales@einfochips.com

eInfochips | The Solutions People

eInfochips Limited	eInfochips, Inc.
11 A/B Chandra Colony,	1230 Midas Way,
Ellisbridge,	Suite 200,
Ahmedabad 380 006	Sunnyvale, CA 94085
Gujarat, India	USA
Tel +91-79-2656 3705	Tel +1-408-496-1882
Fax +91-79-2656 0722	Fax +1-801-650-1480