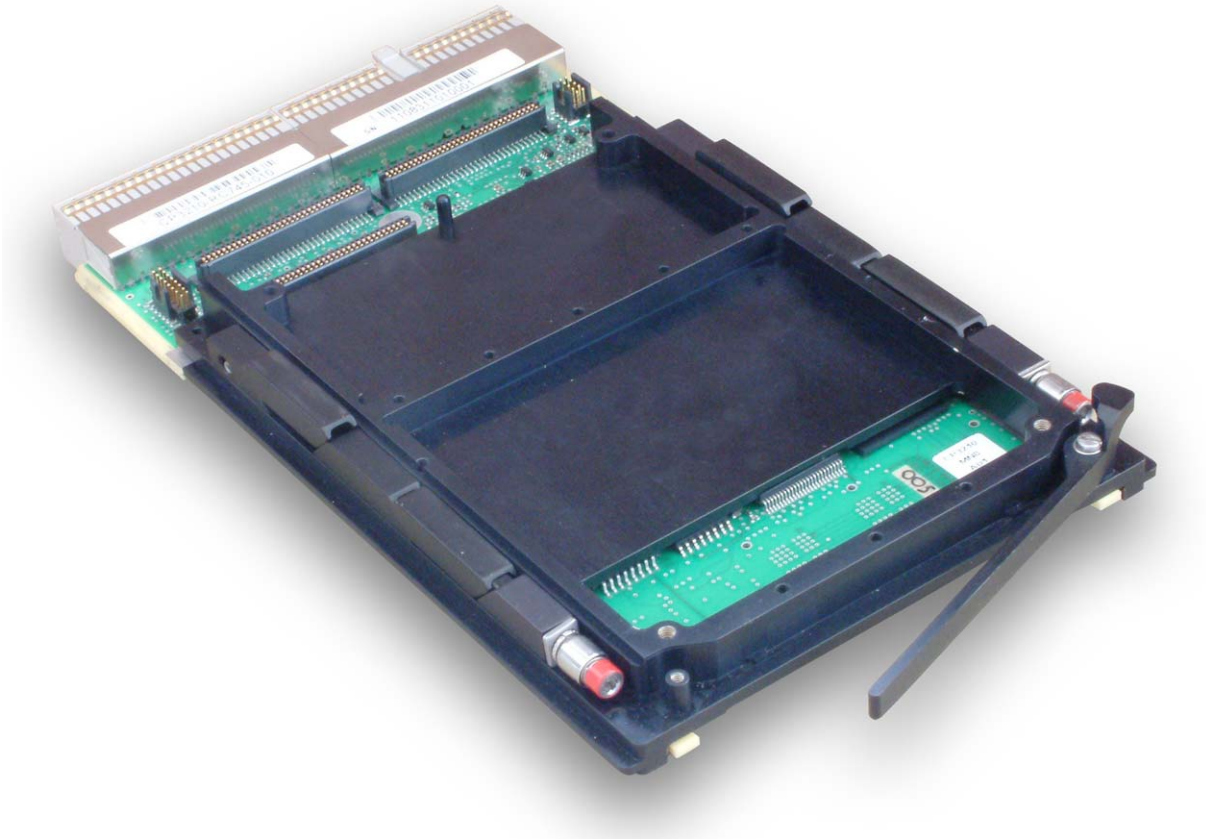


U-Boot User Manual for CP3210 Boards

SD.DT.F31-2e — March 2009

kontron

Revision History

Publication Title:	U-Boot User Manual	
Doc. ID:	SD.DT.F31-2e	
Rev.	Brief Description of Changes	Date of Issue
2e	Chapter 5 - Secure and Non-Secure Update Chapter 9 - New commands: dhcp and update	03-2009

Proprietary Note

This document contains information proprietary to Kontron. It may not be copied or transmitted by any means, disclosed to others, or stored in any retrieval system or media without the prior written consent of Kontron or one of its authorized agents.

The information contained in this document is, to the best of our knowledge, entirely correct. However, Kontron cannot accept liability for any inaccuracies or the consequences thereof, or for any liability arising from the use or application of any circuit, product, or example shown in this document.

Kontron reserves the right to change, modify, or improve this document or the product described herein, as seen fit by Kontron without further notice.

Trademarks

This document may include names, company logos and trademarks, which are registered trademarks and, therefore, proprietary to their respective owners.

Environmental Protection Statement

This product has been manufactured to satisfy environmental protection requirements where possible. Many of the components used (structural parts, printed circuit boards, connectors, batteries, etc.) are capable of being recycled.

Final disposition of this product after its service life must be accomplished in accordance with applicable country, state, or local laws or regulations.

Environmental protection is a high priority with Kontron.

Kontron follows the DEEE/WEEE directive.

You are encouraged to return our products for proper disposal.

The Waste Electrical and Electronic Equipment (WEEE) Directive aims to:

- reduce waste arising from electrical and electronic equipment (EEE)
- make producers of EEE responsible for the environmental impact of their products, especially when they become waste
- encourage separate collection and subsequent treatment, reuse, recovery, recycling and sound environmental disposal of EEE
- improve the environmental performance of all those involved during the lifecycle of EEE

Conventions

This guide uses several types of notice: Note, Caution, ESD.

Note: this notice calls attention to important features or instructions.


Caution: this notice alert you to system damage, loss of data, or risk of personal injury.


ESD: This banner indicates an Electrostatic Sensitive Device.

All numbers are expressed in decimal, except addresses and memory or register data, which are expressed in hexadecimal. The prefix `0x' shows a hexadecimal number, following the `C' programming language convention.

The multipliers `k', `M' and `G' have their conventional scientific and engineering meanings of $*10^3$, $*10^6$ and $*10^9$ respectively. The only exception to this is in the description of the size of memory areas, when `K', `M' and `G' mean $*2^{10}$, $*2^{20}$ and $*2^{30}$ respectively.

When describing transfer rates, `k' `M' and `G' mean $*10^3$, $*10^6$ and $*10^9$ *not $*2^{10}$ $*2^{20}$ and $*2^{30}$.*

In PowerPC terminology, multiple bit fields are numbered from 0 to n, where 0 is the MSB and n is the LSB. PCI and CompactPCI terminology follows the more familiar convention that bit 0 is the LSB and n is the MSB.

Signal names ending with an asterisk (*) or a hash (#) denote active low signals; all other signals are active high.

Signal names follow the PICMG 2.0 R3.0 CompactPCI Specification and the PCI Local Bus 2.3 Specification.

For Your Safety

Your new Kontron product was developed and tested carefully to provide all features necessary to ensure its compliance with electrical safety requirements. It was also designed for a long fault-free life. However, the life expectancy of your product can be drastically reduced by improper treatment during unpacking and installation. Therefore, in the interest of your own safety and of the correct operation of your new Kontron product, you are requested to conform with the following guidelines.

High Voltage Safety Instructions


Warning!
All operations on this device must be carried out by sufficiently skilled personnel only.


Caution, Electric Shock!
Before installing a not hot-swappable Kontron product into a system always ensure that your mains power is switched off. This applies also to the installation of piggybacks. Serious electrical shock hazards can exist during all installation, repair and maintenance operations with this product. Therefore, always unplug the power cable and any other cables which provide external voltages before performing work.

Special Handling and Unpacking Instructions

ESD Sensitive Device!
Electronic boards and their components are sensitive to static electricity. Therefore, care must be taken during all handling operations and inspections of this product, in order to ensure product integrity at all times

Do not handle this product out of its protective enclosure while it is not used for operational purposes unless it is otherwise protected.

Whenever possible, unpack or pack this product only at EOS/ESD safe work stations. Where a safe work station is not guaranteed, it is important for the user to be electrically discharged before touching the product with his/her hands or tools. This is most easily done by touching a metal part of your system housing.

It is particularly important to observe standard anti-static precautions when changing piggybacks, ROM devices, jumper settings etc. If the product contains batteries for RTC or memory backup, ensure that the board is not placed on conductive surfaces, including anti-static plastics or sponges. They can cause short circuits and damage the batteries or conductive circuits on the board.

Personal Injury

Be careful while handling the board, because of the cutting edges of the CPU heatsink.

Do not touch the CPU heatsink or the ruggedizer while removing the board from a rack because it can get very hot.

Do not place the board on any surface or in any form of storage container until the board and its heatsink have cooled down to room temperature.

General Instructions on Usage

In order to maintain Kontron's product warranty, this product must not be altered or modified in any way. Changes or modifications to the device, which are not explicitly approved by Kontron and described in this manual or received from Kontron's Technical Support as a special handling instruction, will void your warranty.

This device should only be installed in or connected to systems that fulfill all necessary technical and specific environmental requirements. This applies also to the operational temperature range of the specific board version, which must not be exceeded. If batteries are present, their temperature restrictions must be taken into account.

In performing all necessary installation and application operations, please follow only the instructions supplied by the present manual.

Keep all the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the board, please re-pack it as nearly as possible in the manner in which it was delivered.

Special care is necessary when handling or unpacking the product. Please consult the special handling and unpacking instruction on the previous page of this manual.

Table Of Contents

Chapter 1 - U-Boot Overview 1
1.1 General Purpose 1
1.2 U-Boot Project 1
1.3 Associated Documentation 2
1.4 U-Boot for CP3210 2
1.4.1 Overview 2
1.4.2 U-Boot Mapping on System Flash 3
Chapter 2 - U-Boot in User Mode 5
Chapter 3 - U-Boot in Rescue Mode 7
Chapter 4 - Environment Parameters 9
4.1 Print the Current Environment Parameters 9
4.2 Set the Ethernet Parameters 10
4.3 Ethernet Protocol Supported 11
4.4 Set the Load Address 11
4.5 Additional Parameters 11
4.6 Set DRAM Size and Offset in Peripheral Slot 12
4.7 Create and Run User Parameters 13
4.8 Set the CPU Frequency 13
Chapter 5 - Update U-Boot Firmware 14
5.1 Secure Update 15
5.1.1 From the User Mode 15
5.1.2 From the Rescue Mode 15
5.2 Non-secure Update 16
5.2.1 From the User Mode 16
5.2.2 From the Rescue Mode 16
Chapter 6 - Power-on self-test (POST) 17
6.1 POST Available on CP3210 17
6.2 Help of POST 19
6.3 Default Test at System Start Up 21
6.4 Remove a Test from the ROM List 21
6.5 Remove a Test from the RAM List 21
6.6 Add a Test to the ROM List 22
6.7 Add a Test to the RAM List 22
6.8 Run a Test in the U-Boot Command Line 22

6.9 POST Diagnostics 23
6.10 NVSRAM Mapping for POST 24

Chapter 7 - Booting VxWorks 27
7.1 Print/Set VxWorks Parameters under U-Boot 28
7.2 Boot VxWorks from Ethernet under U-Boot 30
7.3 Boot VxWorks from Flash System under U-Boot 33
7.4 Burn and Execute a Bootrom 37
7.5 Boot VxWorks from Ethernet under Bootrom 39
7.6 Boot VxWorks from Flash User under Bootrom 42

Chapter 8 - Booting Linux 45
8.1 Boot Linux-NFS 45
8.2 Boot Linux-RAMDISK 46

Chapter 9 - U-Boot Command Line Interface 47
9.1 Commands Summary 47
9.1.1 Proprietary Qualified Commands 47
9.1.2 Standard Qualified Commands 47
9.1.3 Standard Delivered Commands 49
9.2 Proprietary Commands 50
9.2.1 diag - perform board diagnostics 50
9.2.2 dhcp - invoke DHCP client to obtain IP/boot parameters 50
9.2.3 factory - print boards parameters 55
9.2.4 hreset - perform a hardware reset of the board 55
9.2.5 swfreq - print/set the system board frequency 56
9.2.6 update - update U-Boot firmware 56
9.2.7 vxworks - print/set/default VxWorks bootrom parameter 57
9.3 Information Commands 59
9.3.1 bdinfo - print Board Info structure 59
9.3.2 coninfo - print console devices and information 59
9.3.3 flinfo - print FLASH memory information 60
9.3.4 iminfo - print header information for application image 60
9.3.5 imls - list all images found in flash 60
9.3.6 help - print online help 60
9.3.7 regppc - print register information of the PowerPC 61
9.4 Memory Commands 62
9.4.1 base - print or set address offset 62
9.4.2 crc32 - checksum calculation 62
9.4.3 cmp - memory compare 62
9.4.4 cp - memory copy 63
9.4.5 md - memory display 63
9.4.6 mm - memory modify (auto-incrementing) 63
9.4.7 mw - memory write (fill) 63
9.4.8 nm - memory modify (constant address) 64

9.4.9 loop - infinite loop on address range . . . 64
9.5 Flash Memory Commands . . . 65
9.5.1 cp - memory command . . . 65
9.5.2 erase - erase FLASH memory . . . 65
9.5.3 flinfo - print FLASH memory information . . . 65
9.5.4 protect - enable or disable FLASH write protect . . . 66
9.6 Execution Control Commands . . . 67
9.6.1 autoscr - run script from memory . . . 67
9.6.2 bootelf - boot from an ELF image in memory . . . 67
9.6.3 bootm - boot application image from memory . . . 67
9.6.4 bootvx - boot VxWorks from an ELF image . . . 68
9.6.5 go - start application at address 'addr' . . . 69
9.7 Network Commands . . . 70
9.7.1 bootp - boot image via network using BOOTP/TFTP protocol . . . 70
9.7.2 loadb - load binary file over serial line (kermit mode) . . . 70
9.7.3 loads - load S-Record file over serial line . . . 71
9.7.4 loady - load binary file over serial line (ymodem mode) . . . 71
9.7.5 nfs - boot image via network using NFS protocol . . . 71
9.7.6 ping - send ICMP ECHO_REQUEST to network host . . . 71
9.7.7 rarpboot - boot image via network using RARP/TFTP protocol . . . 71
9.7.8 tftpboot - boot image via network using TFTP protocol . . . 71
9.8 Environment Variables Commands . . . 72
9.8.1 askenv - get environment variables from stdin . . . 72
9.8.2 printenv - print environment variables . . . 72
9.8.3 saveenv - save environment variables to persistent storage . . . 72
9.8.4 setenv - set environement variables . . . 73
9.8.5 run - run commands in an environment variable . . . 73
9.8.6 bootd - boot default, i.e., run 'bootcmd' . . . 73
9.9 Filesystem Support . . . 74
9.9.1 fsinfo - print information about filesystems . . . 74
9.9.2 fsload - load binary file from a filesystem image . . . 74
9.9.3 ls - list files in a directory (default /) . . . 74
9.10 Special Commands . . . 75
9.10.1 dcache - enable or disable data cache . . . 75
9.10.2 EEPROM sub-system . . . 75
9.10.3 icache - enable or disable instruction cache . . . 75
9.10.4 icrc32 - checksum calculation . . . 75
9.10.5 iloop - infinite loop on range address . . . 75
9.10.6 imd - I2C memory display . . . 75
9.10.7 imm - I2C memory modify (auto-incrementing) . . . 76
9.10.8 imw - I2C memory write (fill) . . . 76
9.10.9 inm - I2C memory modify (constant address) . . . 76
9.10.10 iprobe - probe to discover valid I2C chip addresses . . . 76
9.10.11 pci - list and access PCI configuration space . . . 77
9.11 Miscellaneous Commands . . . 78
9.11.1 date - get/set/reset date & time . . . 78
9.11.2 dtt - Digital Thermometer and Thermostat . . . 78
9.11.3 echo - echo args to console . . . 78

9.11.4 itest - return true/false on integer compare 78
9.11.5 reset - perform reset of the CPU 78
9.11.6 sleep - delay execution for some time 79
9.11.7 version - print monitor version 79
9.11.8 ? - alias for 'help' 79

Chapter 1 - U-Boot Overview

Functional changes that differ from previous version of the document are identified by a vertical bar in the margin.

1.1 General Purpose

This document describes the firmware and the Power-on self-test (POST) available on the Kontron CP3210 board.

The firmware used on the CP3210 is **U-Boot 1.1.5 - ID 09062**

The CP3210 U-Boot firmware includes commands to:

- Display and modify the memories and registers (Flash, SRAM, DDR SDRAM, Registers, ...)
- Access the devices
- Boot Operating Systems (VxWorks, Linux, ...)
- Run the Power-on self-test (POST)
- Configure the boot of the board

1.2 U-Boot Project

The U-Boot project is fully described on the web site: http://www.u-boot.sourceforge.net

The U-Boot running on the CP3210 is compiled with the ELDK 4.1 tools environment. The following trace shows the **gcc** version:

```
ppc_7xx-gcc (GCC) 4.0.0 (DENX ELDK 4.1 4.0.0)
Copyright (C) 2005 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The ELDK is a free download environment tools which is provide by DENX (http://www.denx.de).

1.3 Associated Documentation

Kontron Documentation

- CA.DT.A57 . . . . . CP3210 Single PowerPC 750FX User's Guide
- CA.DT.A58 . . . . . CP3210 Hardware Release Notes
- SD.DT.F30 . . . . . Release Notes BSP CP3210 Workbench 2.4 / VxWorks 6.2

DENX Software Engineering Documentation

- Documentation available at . . . . . http://www.denx.de/en/Documents/WebHome

1.4 U-Boot for CP3210

1.4.1 Overview

- U-Boot for CP3210 is available both for Peripheral and System slots boards.
- The U-Boot Mapping on System Flash is fully described in section 1.4.2 'U-Boot Mapping on System Flash" page 3.
- The U-Boot firmware may be used in **Rescue Mode** or **User Mode**, depending on the CP3210 configuration. Refer to Chapter 2 “U-Boot in User Mode" page 5, or Chapter 3 “U-Boot in Rescue Mode" page 7 for detailed information.
- Power-on self-test (POST) are available on the CP3210. Refer to Chapter 6 “Power-on self-test (POST)" page 17 for detailed information on POST.

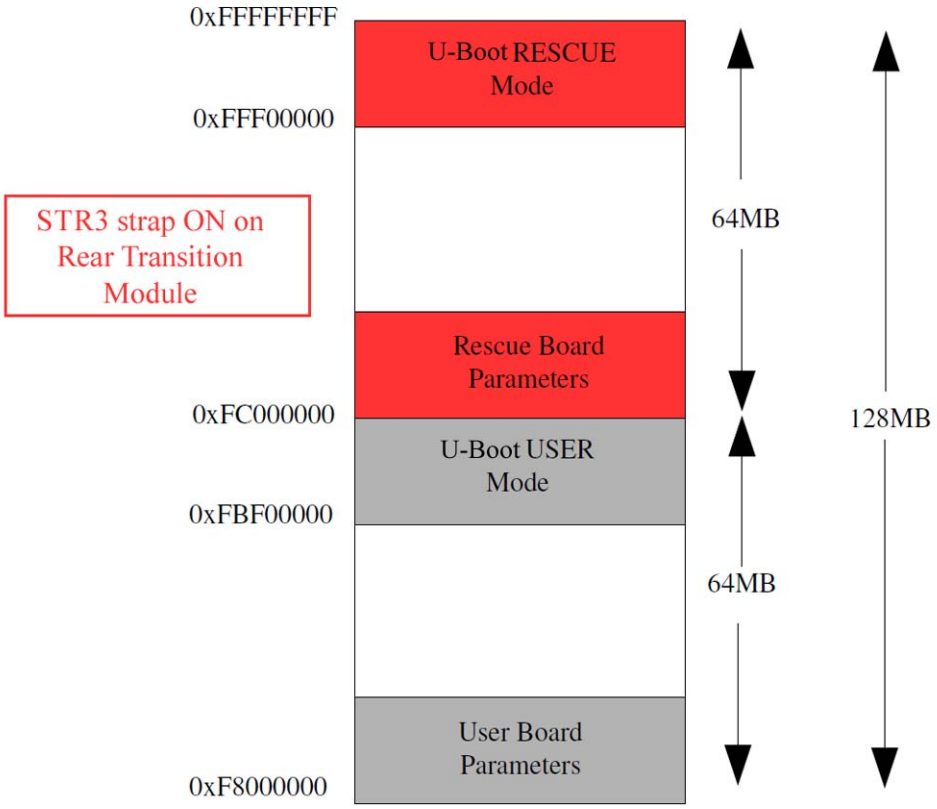
For a standard configuration CP3210 at 733 MHz, the boot time:
- without running any power-on self-test is **~ 9s**
- including the default power-on self-test run is **~11 s**

1.4.2 U-Boot Mapping on System Flash

The CP3210 has a 128 MB system flash. The system flash is split in two areas by an hardware strap (STR3)

- When the strap STR3 on the Rear Transition Module (RTM) is on, the board runs in **RESCUE Mode**.
- When the strap STR3 on the Rear Transition Module (RTM) is off, the board runs in **USER Mode**.

Flash System Mapping in RESCUE Mode



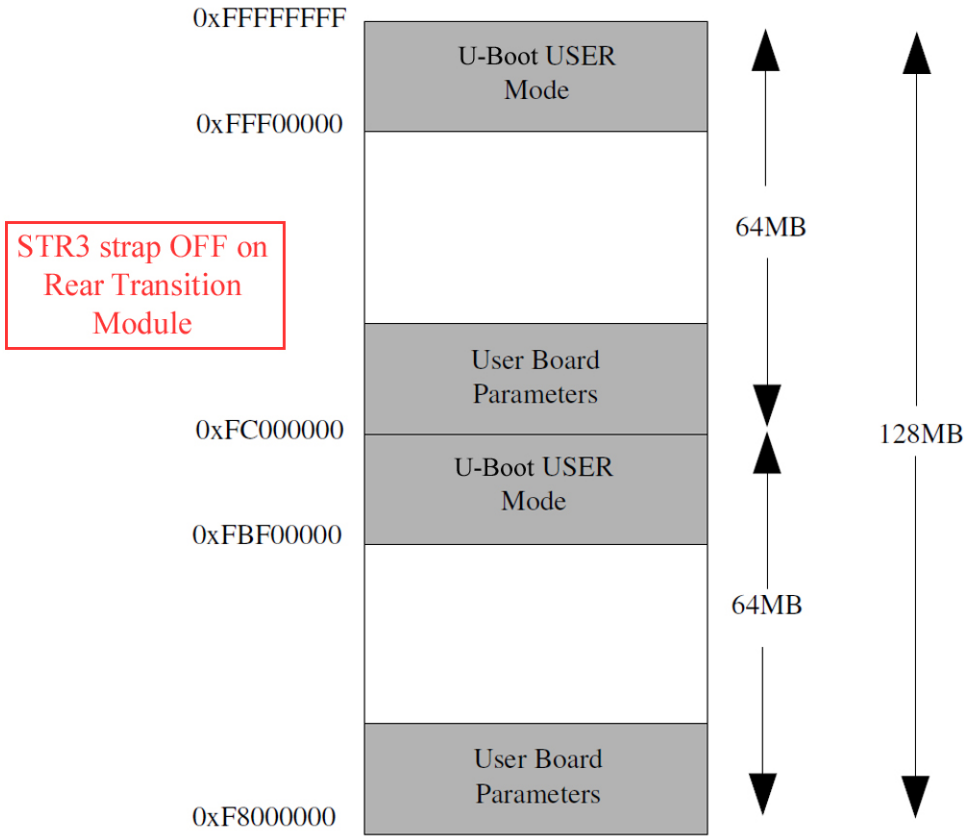
In RESCUE Mode, the board boots on the U-Boot RESCUE Mode binary code. In this case, the 128 MB of system flash is viewed entirely by the user. By the way, the user can access both the U-Boot RESCUE Mode and the U-Boot USER Mode areas.

This mode is useful when the user wants to update the U-Boot USER Mode binary code.

The U-BOOT RESCUE Mode firmware is a secure firmware and must not be erased.

When the board boots in RESCUE Mode, the U-Boot prompt displayed is '**`CP3210-RESCUE >`**'.

Flash System Mapping in USER Mode



In USER Mode, the board boots on the U-Boot USER Mode binary code. In this case, the 128 MB of system flash is viewed entirely by the user. But the upper 64 MB part is mirrored to the lower 64 MB part of the system flash .

Our advice is to use this mode by default. If the U-Boot USER Mode binary code is damaged, the user can restart the board in RESCUE Mode to restore a valid U-Boot USER Mode binary code.

When the board boots in USER Mode, U-Boot prompt displayed is '**`CP3210 >`**'.

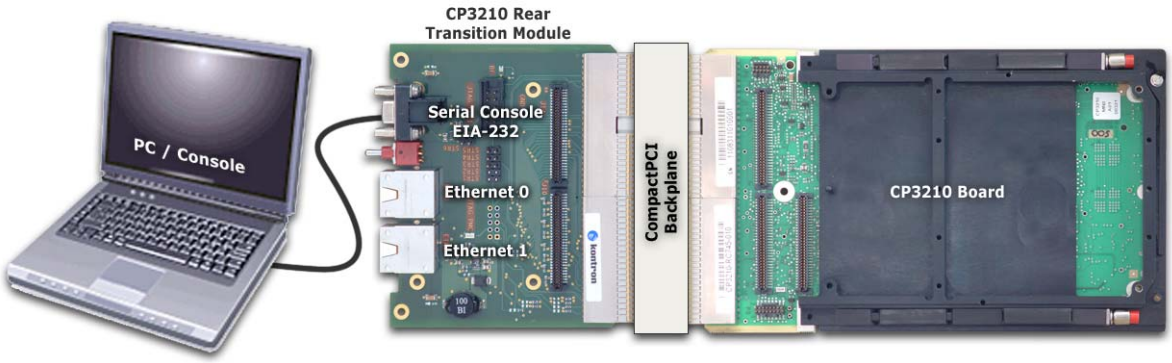
The User and Rescue parameters areas in the system flash must not be erased. These memory regions size is 2 MB each. It is recommended not to write into these specific areas.

Chapter 2 - U-Boot in User Mode

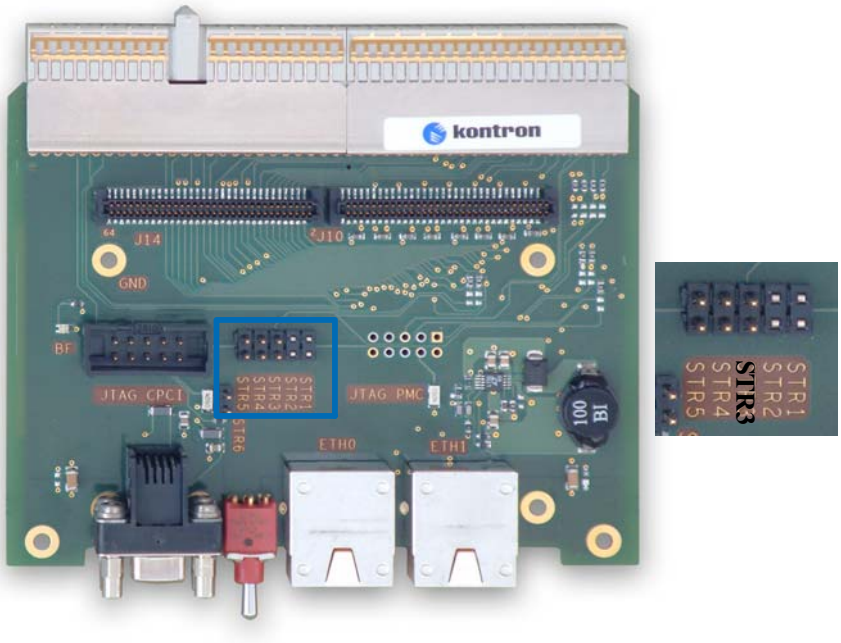
Hardware configuration needed to boot a CP3210 board in system slot:

- CP3210 Board
- Rugged Rack
- Rear Transition Module (STR3 strap OFF on the RTM)
- Serial Cable
- Ethernet Cables

On the following figure, you can find the system configuration:



On the following figure, you can see the STR3 strap location on the RTM:



Traces displayed when the CP3210 is booted up through the serial console:

```
U-Boot 1.1.5 (Sep 10 2008 - 16:33:05) KONTRON MODULAR COMPUTERS (1.0/08283)

CPU: 750FX v2.3 @ 732.875 MHz
BOARD: CP3210
S/N : 1108311010006
DTT: 1 is 60 C
Checking POST configuration
DRAM: Memory dfcdl init Done
DRAM: -- DIMM1 has 1 banks
ECC Initialization of Bank 0: PASS
CAS Latency = 2 tRP = 5 tRAS = 8 tRCD=5
Total SDRAM memory is 512 MB

POST "mem_data" (cpu0,slow,simple,haltonfail) PASSED
POST "mem_addr" (cpu0,slow,simple,haltonfail) PASSED

SD (DDR) RAM
BANK0: base - 0x00000000 size - 512M bytes

CPU's PCI 0 windows
        IO: base - 0x20000000 size - 32M bytes
MEMORY 0: base - 0x22000000 size - 32M bytes

CPU's PCI 1 windows
        IO: base - 0x24000000 size - 32M bytes
MEMORY 0: base - 0x26000000 size - 32M bytes

DEVICES
DEV 0: base - 0xc0000000 size - 256M bytes width - 32 bits - USER FLASH
DEV 1: base - 0x28000000 size - 1M bytes width - 32 bits - EPLD
DEV 2: base - 0x28100000 size - 1M bytes width - 8 bits - NVRAM / RTC
DEV 3: base - 0x28200000 size - 1M bytes width - 32 bits - NOT USE
 BOOT: base - 0xf8000000 size - 128M bytes width - 16 bits
FLASH: [131072kB@f8000000] [256MB@c0000000] 384 MB

PCI BUSES
        - PCI 0 : PMC
        - PCI 1 : System Slot

PCI 0 bus mode: Conventional PCI
PCI 1 bus mode: Conventional PCI
In:     serial
Out:    serial
Err:    serial
Ethernet dfcdl init: Done
Net: mv_enet0 [PRIME], mv_enet1

POST "mem_data" (cpu0,slow,simple,promptonfail) PASSED
POST "mem_addr" (cpu0,slow,simple,promptonfail) PASSED
POST "userflash" (cpu0,fast,simple) PASSED
POST "temp_sensors" (cpu0,fast,simple) PASSED

CP3210 >
```

Several information are given by U-Boot through the serial EIA-232 interface.

The mapping of the board is visible. It informs the user about the PCI mapping, the DDR SDRAM mapping, the flashes mapping and the flashes detection .

There is also an information on POST tests results: PASSED or FAILED.

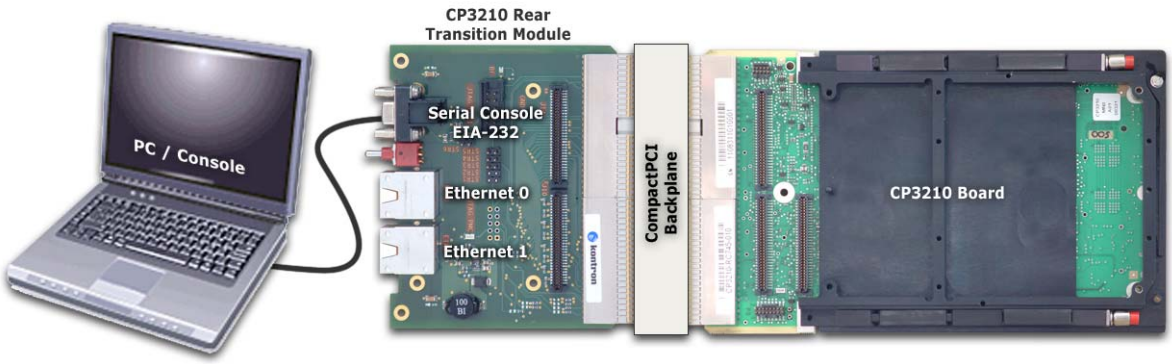
The U-Boot initialization of the board ends when the prompt '**`CP3210 >`**' appears on the serial console.

Chapter 3 - U-Boot in Rescue Mode

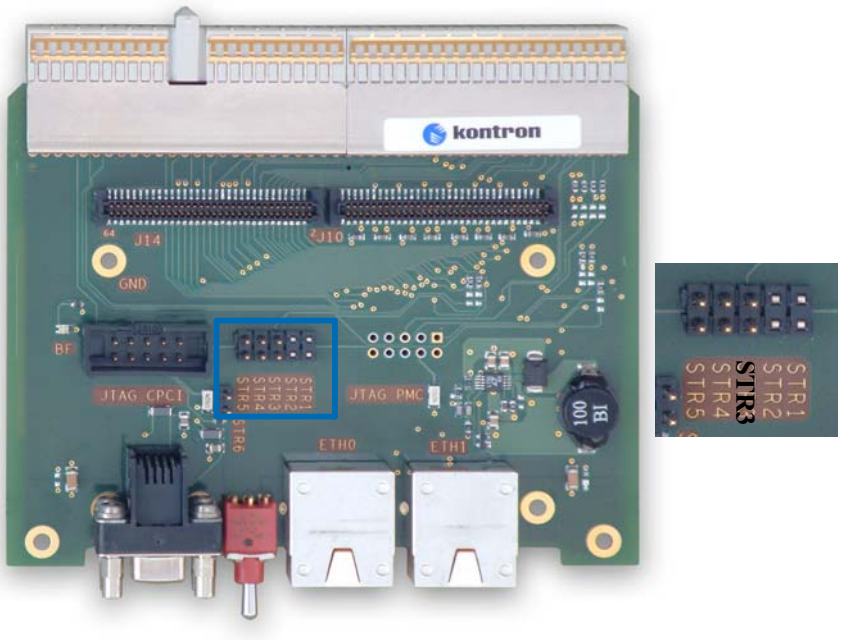
Hardware configuration needed to boot a CP3210 in system slot:

- CP3210 Board
- Rugged Rack
- Rear Transition Module (STR3 strap ON on the RTM)
- Serial Cable
- Ethernet Cables

On the following figure you can find the system configuration :



On the following figure, you can see the STR3 strap location on the RTM:



Traces displayed when the CP3210 is booted up:

```
U-Boot 1.1.5 (Sep 10 2008 - 16:33:05) KONTRON MODULAR COMPUTERS (1.0/08283)

CPU: 750FX v2.3 @ 732.875 MHz
BOARD: CP3210
S/N : 1108311010001
DTT: 1 is 25 C
Checking POST configuration
DRAM: Memory dfcdl init Done
DRAM: -- DIMM1 has 1 banks
ECC Initialization of Bank 0: PASS
CAS Latency = 2 tRP = 5 tRAS = 8 tRCD=5
Total SDRAM memory is 512 MB

POST "mem_data" (cpu0,slow,simple,haltonfail) PASSED
POST "mem_addr" (cpu0,slow,simple,haltonfail) PASSED

SD (DDR) RAM
BANK0: base - 0x00000000 size - 512M bytes

CPU's PCI 0 windows
        IO: base - 0x20000000 size - 32M bytes
MEMORY 0: base  - 0x22000000  size - 32M bytes

CPU's PCI 1 windows
        IO: base - 0x24000000 size - 32M bytes
MEMORY 0: base  - 0x26000000 size - 32M bytes

DEVICES
DEV 0: base - 0xc0000000 size - 256M bytes width - 32 bits - USER FLASH
DEV 1: base - 0x28000000 size - 1M bytes   width - 32 bits - EPLD
DEV 2: base - 0x28100000 size - 1M bytes   width -  8 bits - NVRAM / RTC
DEV 3: base - 0x28200000 size - 1M bytes   width - 32 bits - NOT USE
BOOT : base - 0xf8000000 size - 128M bytes width - 16 bits
FLASH: [131072kB@f8000000] [256MB@c0000000] 384 MB

PCI BUSES
        - PCI 0 : PMC
        - PCI 1 : System Slot

PCI 0 bus mode: Conventional PCI
PCI 1 bus mode: Conventional PCI
In:     serial
Out:    serial
Err:    serial
Ethernet dfcdl init: Done
Net: mv_enet0 [PRIME], mv_enet1

POST "mem_data" (cpu0,slow,simple,promptonfail) PASSED
POST "mem_addr" (cpu0,slow,simple,promptonfail) PASSED
POST "userflash" (cpu0,fast,simple) PASSED
POST "temp_sensors" (cpu0,fast,simple) PASSED
CP3210-RESCUE >
```

Several information are given by U-Boot through the serial EIA-232 interface.

The mapping of the board is visible. It informs the user about the PCI mapping, the DDR SDRAM mapping, the flashes mapping and the flashes detection .There is also an information on POST tests results: PASSED or FAILED. The board finished to be initialized by U-Boot when the prompt '**CP3210-RESCUE >**' appears on the serial console.

This mode is useful when the code in the User part of the system flash is broken. This mode is able to rebuild the User part of the system flash : U-Boot binary, user parameters, board parameters.

Chapter 4 - Environment Parameters

To print and set the environment variables, we recommend to use following commands under the U-Boot User Mode. These commands are also described in Chapter 9 “U-Boot Command Line Interface" page 47.

4.1 Print the Current Environment Parameters

The U-Boot is delivered on the CP3210 with default parameters. These parameters are displayed by entering the following command:

```
CP3210 > printenv
```

Example of **printenv** command outputs:

```
CP3210 > printenv
loads_echo=0
rootpath=/mnt/yellow_dog_mini
pcidelay=0
burn_uboot= tftp 200000 uboot/minotaure/uboot_rescue.bin;protect off all;era FFF00000
FFFFFFFF; cp 200000 FFF00000 0x40000;
burn_uboot_user= tftp 200000 uboot/minotaure/uboot_user.bin;protect off all;era FBF00000
FBFFFFFF; cp 200000 FBF00000 0x40000;
bootargs_root=root=/dev/nfs rw
bootargs_end=:::DB64460:egiga0:none
ethprime=mv_enet0
standalone=fsload 0x400000 uImage;setenv bootargs $(bootargs) root=/dev/mtdblock/0 rw ip=$
(ipaddr):$(serverip)$(bootargs_end); bootm 0x400000;
boot_linux=tftp 0x800000 uImagePEC7;tftp 0xA00000 uRamdisk7xx;setenv bootargs
console=ttyS0,9600 root=/dev/ram0 rw ip=$(ipaddr):$(serverip)$(bootargs_end);bootm 0x800000
0xA00000
bootlinux_nfs=tftp 0x800000 uImagePEC7;setenv bootargs root=/dev/nfs rw
nfsroot=192.168.0.1:/home/opt/ELDK_41/ppc_7xx ip=$(ipaddr):$(serverip)$(bootargs_end);bootm
0x800000
bootlinux_nfs_egiga1=setenv bootargs_end :::DB64460:egiga1:none;run bootlinux_nfs
ethact=mv_enet0
bootdelay=3
baudrate=115200
bootVx=tftp 0x2000000 vxWorks;bootvx
bootargs=mgi(0,0)pcbist2:/home/bsa/WindRiver/vxworks-6.2/target/config/cp3210/vxWorks
e=192.168.0.2:FFFFFF00 h=192.168.0.1 u=bsa f=0x8
filesize=67e20
fileaddr=2000000
ipaddr=192.168.0.2
serverip=192.168.0.1
burn_bootrom=tftp 0x2000000 bootrom.bin;erase 0xFFE00000 0xFFEFFFFF;cp 0x2000000 0xFFE00000
0x40000
bootBrom=go ffe00108
loadaddr=0xffe00108
bootcmd=bootvx
stdin=serial
stdout=serial
stderr=serial
Environment size: 1453/262140 bytes
CP3210 >
```

4.2 Set the Ethernet Parameters

The CP3210 has two Ethernet interfaces that the user needs to initialize before using them.

These interfaces are called under U-Boot as follow:

- Ethernet number 0 is named : `mv_enet0`
- Ethernet number 1 is named : `mv_enet1`

Following Ethernet parameters are displayed using the **`printenv`** command are:

```
ipaddr      : IP address of the CP3210
serverip    : IP address of the server
ethprime    : First ethernet using previous parameters
ethact      : Interface in use by default
bootcmd     : Start command(s) automatically after a delay defined by bootdelay parameter
bootdelay   : Define a boot delay in seconds before running the bootcmd parameter
```

For example,

> to set up following parameters:

CP3210 IP address	:	192.93.161.150
Server IP address	:	192.93.161.147
Prime Ethernet :	:	eth0 (name under U-Boot: **mv_enet0**)
Ethernet in action	:	eth0 (name under U-Boot: **mv_enet0**)

> enter following commands on the CP3210 board:

```
CP3210 > setenv ipaddr 192.93.161.150
CP3210 > setenv serverip 192.93.161.147
CP3210 > setenv ethprime mv_enet0
CP3210 > setenv ethact mv_enet0
CP3210 >
```

At this point, all parameters are valid and saved in the DRAM U-Boot area. But, these parameters will be lost if the board is switched off.

So, to definitely store these parameters in a non-volatile memory, the user needs to type the following command:

```
CP3210 > saveenv
Saving Environment to Flash...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
CP3210 >
```

All saved parameters in the non-volatile memory will be restored at the next startup of the board.

To check that the parameters are properly set, try a basic **ping** command under U-Boot by typing:

```
CP3210  > ping 192.93.161.147
Ethernet status port 0: Link up, Full Duplex, Speed 1 Gbps
Using mv_enet0 device
host 192.93.161.147 is alive
CP3210  >
```

4.3 Ethernet Protocol Supported

U-Boot on the CP3210 supports two protocols to download images/files into the board as a client:

- tftp Trivial Transfer Protocol
- nfs Network File System

4.4 Set the Load Address

Before downloading any images/files, the CP3210 needs to be configured as suggested in the section 4.2 “Set the Ethernet Parameters" page 10.

The **loadaddr** variable is used to store in the DDR-SDRAM the address where the image/file downloaded will be stored.

You are advised to set this address to **0x2000000** (32 MB). If the image is larger than 128 MB, set the load address to **0x10000000**.

To set this address to **0x2000000**, type the following command:

```
CP3210  > setenv loadaddr 0x2000000
CP3210 >
```

Don't forget to save these parameters using the **saveenv** command:

```
CP3210  > saveenv
Saving Environment to Flash...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
CP3210 >
```

4.5 Additional Parameters

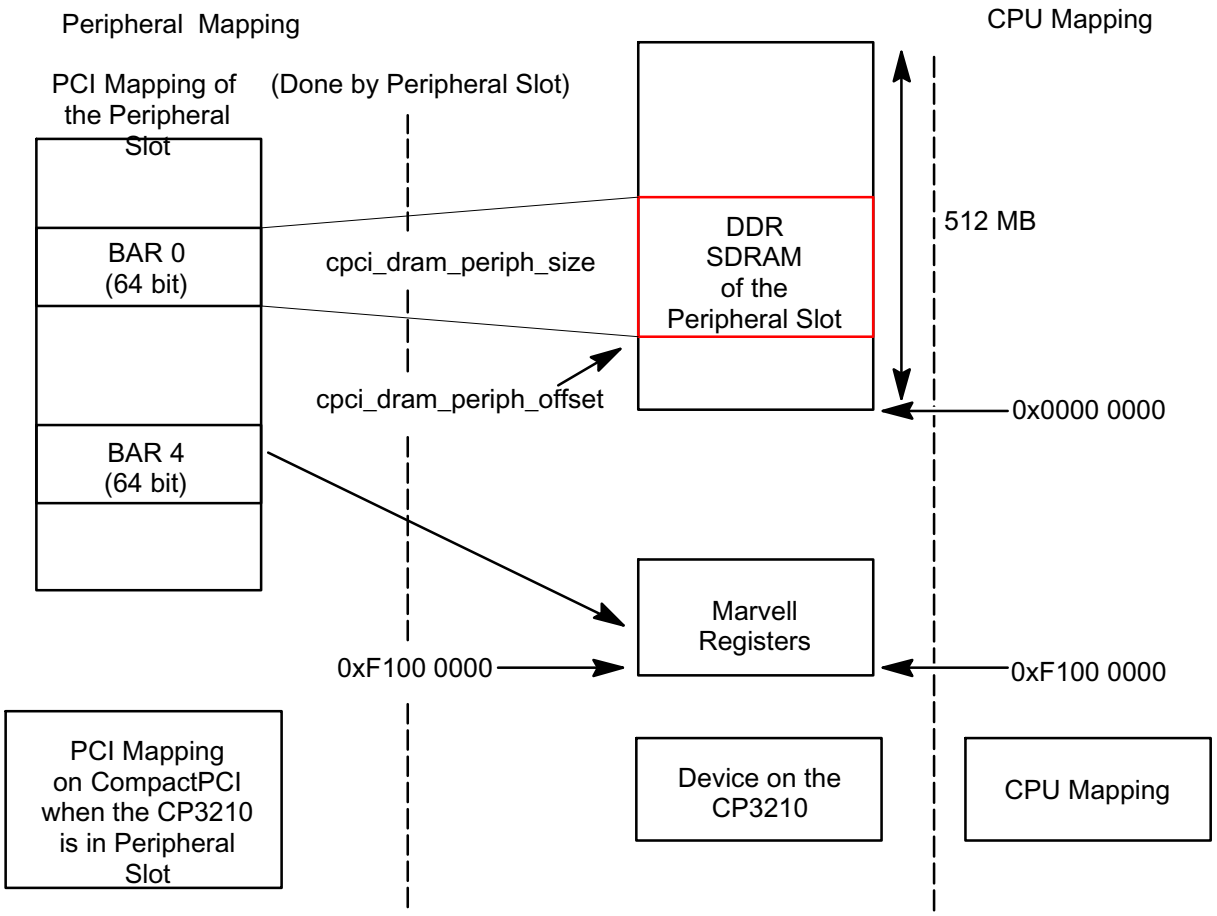
Additional useful parameters:

```
baudrate                : Baudrate of the console serial line
pcidelay                : Delay before scanning the pci interfaces in millisecond
cpci_dram_periph_size   : Dram size seen on the CPCI when the board is in the peripheral slot
cpci_dram_periph_offset : Offset address from 0x00000000
bootargs                : This parameter is a string given to the Operating System (Linux/VxWorks)
```

Default value in the firmware:

```
baudrate                : 115 200 Bauds
pcidelay                : 0 ms
cpci_dram_periph_size   : 0x02000000 (32MB)
cpci_dram_periph_offset : 0x00000000
bootargs                : mgi(0,0) sunblade:/home/test/vxWorksDec h=192.168.0.1 e=192.168.0.2:FFFFFF00
u=test pw=testfab f=0x8 ( vxWorks bootstring )
```

4.6 Set DRAM Size and Offset in Peripheral Slot



4.7 Create and Run User Parameters

U-Boot firmware on the CP3210 allows the user to create his own parameters using the U-Boot **setenv** command .

For example:

- to download an executable file “example.txt" from the network using the **tftp** protocol and to execute the associated code
- to create a user parameter named “flash_file", using the procedure described below:


The network parameters are supposed to be set as described in the sections 4.2 “Set the Ethernet Parameters" and 4.4 “Set the Load Address".

```
CP3210 > setenv flash_file 'tftp 0x2000000 example.txt;go 0x2000000'
CP3210 > saveenv
CP3210 >
```


The character "**;**" separates a command to another on the same line.

At this point the user parameter named **flash_file** is stored in the CP3210 non-volatile memory. To execute this parameter, type the following command:

```
CP3210 > run flash_file
CP3210  >
```

4.8 Set the CPU Frequency

The CP3210 U-Boot is able to switch the CPU bus frequency of the board and so the CPU internal frequency using the **swfreq** command

The CPU bus frequency can be switched from 100 MHz to 133 MHz and vice versa. The CPU frequency automatically switches from 700 MHz to 733 MHz.

This U-Boot command is available only in Rescue Mode.
If the user wants to set the CPU bus frequency at 100 MHz and 700 MHz CPU frequency type:

```
CP3210-RESCUE > swfreq set 100
CP3210-RESCUE >
```

If the user wants to set the CPU bus frequency at 133 MHz and 733 MHz CPU frequency type:

```
CP3210-RESCUE > swfreq set 133
CP3210-RESCUE >
```

The new frequencies are set only at the next power-on of the CP3210.

Chapter 5 - Update U-Boot Firmware

The CP3210 U-Boot user firmware can be updated:

- by itself (**User Mode**)
- by the CP3210 U-Boot Rescue Firmware (**Rescue Mode**).

  The second option (**Rescue Mode**) is commonly used for emergency update, if the CP3210 U-Boot User Firmware doesn't boot due to some bad manipulations.

Depending on the ID of the U-Boot firmware running on the board, the update of the U-Boot firmware can be launched as:

- a Secure Update (ID greater or equal to **09062**)
  - in User Mode refer to section 5.1.1 page 15
  - in Rescue Mode refer to section 5.1.2 page 15
- a Non-Secure Update (ID lower than **09062**)
  - in User Mode refer to section 5.2.1 page 16
  - in Rescue Mode refer to section 5.2.2 page 16

The Secure Update method is the recommended method. Before updating the firmware file, the type of the file to be downloaded is checked, and a checksum is done after the download.

The network parameters need to be set by user, as described in section 4.2 “Set the Ethernet Parameters” page 10 before updating the U-Boot user firmware.

5.1 Secure Update

If the firmware ID which is currently running on the board is greater or equal to 09062, then the user can proceed the following command. Otherwise, refer to section 5.2 'Non-Secure Update" page 16.

5.1.1 From the User Mode

For this secure update in user mode, the user needs to download previously the firmware binary file through the protocol supported by the firmware which are TFTP and NFS. Then, he can invoke the update command:

```
CP3210 > tftp 200000 uboot/minotaure/ID09062/uboot_user_ID09062.bin
Ethernet status port 0: Link up, Full Duplex, Speed 100 Mbps
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.56
Filename 'uboot/minotaure/ID09062/uboot_user_ID09062.bin'.
Load address: 0x200000
Loading: ##################################################
done
Bytes transferred = 263200 (40420 hex)

CP3210 > update user 200000

Image found @0x00200000 is U-Boot firmware
Do you really want to flash it ? (yes/no) : yes
Un-Protect Flash Bank # 1
Un-Protect Flash Bank # 2
.....Erased 8 sectors
Copy to Flash... 0%[====================] 100%
done
Update is done
CP3210 >
```

The update command checks if the binary image matches a U-Boot image and proceeds a crc checksum.

5.1.2 From the Rescue Mode

For this secure update in rescue mode, the user needs to download previously the firmware binary file through the protocol supported by the firmware which are TFTP and NFS. Then, he can invoke the update command:

```
CP3210-RESCUE > tftp 200000
uboot/minotaure/ID09062/uboot_rescue_ID09062.bin
Ethernet status port 0: Link up, Full Duplex, Speed 100 Mbps
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.56
 Filename 'uboot/minotaure/ID09062/uboot_user_ID09062.bin'.
Load address: 0x200000
  Loading: ##################################################
done
Bytes transferred = 263200 (40420 hex)

CP3210-RESCUE > update rescue 200000

Image found @0x00200000 is U-Boot firmware
Do you really want to flash it ? (yes/no) : yes
Un-Protect Flash Bank # 1
Un-Protect Flash Bank # 2
.....Erased 8 sectors
Copy to Flash... 0%[====================] 100%
done
Update is done
CP3210-RESCUE >
```

The update command checks if the binary files matches a U-Boot image and proceeds a crc checksum.

5.2 Non-secure Update

This non-secure update method is available for all the firmware IDs. Nevertheless, the method is dangerous and must be used with extreme caution.

5.2.1 From the User Mode

The U-Boot firmware use the TFTP protocol to download the binary file.

For example, to update the CP3210 firmware with the file **uboot_cp3210_xxxxx.bin** from the User Mode:

```
CP3210 > tftp 200000 uboot_cp3210_xxxxx.bin
CP3210 > protect off all
CP3210 > era FBF00000 FBFFFFFF
CP3210 > cp 200000 FBF00000 0x40000
CP3210 >
```

5.2.2 From the Rescue Mode

The U-Boot firmware use the TFTP protocol to download the binary file.

For example, to update the CP3210 firmware with the file **uboot_cp3210_xxxxx.bin** from the Rescue Mode:

```
CP3210 RESCUE > tftp 200000 uboot_cp3210_xxxxx.bin
CP3210-RESCUE > protect off all
CP3210-RESCUE > era FBF00000 FBFFFFFF
CP3210-RESCUE > cp 200000 FBF00000 0x40000
CP3210-RESCUE >
```

Chapter 6 - Power-on self-test (POST)

The POST on the CP3210 can be automatically started when the board boots up or through the command line **diag**.

6.1 POST Available on CP3210

To see the tests available on the board, type the following command:

```
CP3210 > diag
```

The similar trace is displayed:

```
CP3210 > diag
POSTs configured to run from command line :
mem_data (0) - Checks Memory/ECC data lines
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_addr (1) - Checks Memory/ECC address lines
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_pattern1 (2) - Checks Memory/ECC using pattern 0x00000000
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_pattern2 (3) - Checks Memory/ECC using pattern 0xFFFFFFFF
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_pattern3 (4) - Checks Memory/ECC using pattern 0x55555555
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_pattern4 (5) - Checks Memory/ECC using pattern 0xAAAAAAAA
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_bitflip (6) - Checks Memory/ECC using bit-flip pattern ((1 << (offset % 32))
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_addrpattern (7) - Checks Memory/ECC using address pattern (offset)
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
mem_addrpattern2 (8) - Checks Memory/ECC using address pattern (~offset)
capabilities : cpu0,slow/fast,simple
run mode 1 : cpu0,slow,simple
rtc (20) - Checks the RTC is running.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple
userflash (23) - Checks the presence of the user flash.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple
temp_sensors (31) - Checks if all temperature sensors are detected.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple
nvsram (40) - Checks the content of the NVsRAM device.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple
```

```
ethernet0 (50) - Checks PHY Internal Loopback on Ethernet interfaces 0.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple
ethernet1 (51) - Checks PHY Internal Loopback on Ethernet interfaces 1.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple
userflash_bus (95) - Checks user flash address/data bus.
capabilities : cpu0,fast,simple
run mode 1 : cpu0,fast,simple

Other POSTs available but not yet configured :
 None

Use 'help diag' to get more info.

CP3210 >
```

This previous trace shows the different tests available on the CP3210 U-Boot firmware.

For the Memory DDR-SDRAM:

- mem_data
- mem_addr
- mem_pattern1
- mem_pattern2
- mem_pattern3
- mem_pattern4
- mem_bitflip
- mem_addrpattern
- mem_addrpattern2

For the Real Time Clock (RTC):

- rtc

For the User flash:

- userflash
- userflash_bus

For the temperature sensor:

- temp_sensors

For the NVSRAM:

- nvsram

For the Ethernet Interfaces:

- ethernet0
- ethernet1

6.2 Help of POST

The on-line help of the **diag** command is available under the U-Boot firmware CP3210.

➤ Trace of the **help diag** command:

```
CP3210 > help diag
----- Usage -----
Print list of POSTs and infos about them :

  diag [rom|ram|command] [<postname>|<postnum> ...]
    rom             : Display POST(s) that are run automatically from ROM
                      (before relocation to RAM)
    ram             : Display POST(s) that are run automatically from RAM
                      (after relocation to RAM)
    command         : Display POST(s) that are run from command line
                      (default) (diag command)

    <postname>|<postnum> ...
                      list of POST(s) to display. All if the list is empty.
                      POST(s) are referenced using their name or their number.

Run POST(s) from command line :

  diag run [loop <count>] [<postname>|<postnum> ...]

    loop <count>
                      run POST(s) <count> times instead of once
    <postname>|<postnum> ...
                      list of POST(s) to run. All if the list is empty.
                      POST(s) are referenced using their name or their number.

Print POST(s) status :

  diag stat [<postname>|<postnum> ...]

    <postname>|<postnum> ...
                      list of POST(s) to display. All if the list is empty.
                      POST(s) are referenced using their name or their number.

Clear POST(s) status :

  diag [rom|ram|command] clrstat|clrallstat [<postname>|<postnum> ...]
    rom             : Clear status for POST(s) that are run automatically from ROM
                      (before relocation to RAM)
    ram             : Clear status for POST(s) that are run automatically from ROM
                      (after relocation to RAM)
    command         : Clear status for POST(s) that are run from command line
    (default)  (diag command)
    clrstat         : Reset status to NOTRUN
    clrallstat      : Reset status to NOTRUN and clear the FAILED at least
                      once flag
    <postname>|<postnum> ...
                      list of POST(s) to clear. All if the list is empty.
                      POST(s) are referenced using their name or their number.
```

```
Restore default POST configuration :

  diag default

Configure POST(s) :

  diag [rom|ram|command] cfg <cfgarg> ... <postname>|<postnum> ...
    rom             : Configure POST(s) that are run automatically from ROM
                      (before relocation to RAM)
    ram             : Configure POST(s) that are run automatically from ROM
                      (after relocation to RAM)
    command         : Configure POST(s) that are run from command line
     (default) (diag command)
  <cfgarg>          : Configure one or several POST(s).

                      <cfgarg> is either :
                      - "delete" to delete POST(s) from the list of configured POSTs
                      - "default" to configure POST(s) with a default run mode
                      - a comma separated list of runflags defining a POST
                        run mode; for example : cpu0,cpu1,fast,complex.

                      valid runflags are :
                      - "CPU" flags (can be mixed together)
                        cpu0            : run on CPU0
                        cpu1            : run on CPU1
                      - "SPEED" flags (can NOT be mixed)
                        slow            : run in slow mode (full testing)
                        fast            : run in fast mode (fast testing)
                      - "CONFIG" flags (can NOT be mixed)
                        simple          : run in simple mode (no external hardware)
                        complex                 : run in complex mode (needs external hardware)
                      - "HALT" flags (can NOT be mixed)
                        haltonfail              : halt immediately (hang) if test fails
                        promptonfail    : halt at U-Boot prompt (no OS) if test fails
                      - "RESET" flags (can be mixed together)
                        normalreset     : run after a normal reset
                        poweronreset    : run after a power-on reset
                        watchdogreset   : run after a watchdog reset
                        allresets               : run after all resets listed above
                        testreset               : run after a special reset generated by a POST

                      cfg with its argument can be repeated several times to
                      define several run modes for POST(s) in the run list. POST is
                      then run several times in different modes.

  <postname>|<postnum> ...

                      list of POST(s) to configure.
                      POST(s) are referenced using their name or their number.

CP3210 >
```

The same POST may be run from the ROM and/or from the RAM. The main difference between a test run from the ROM or the RAM is the test coverage. The test coverage of the POST run from the ROM is more extensive.

6.3 Default Test at System Start Up

Once the U-Boot firmware initialized the Marvell Disco III host bridge, the CPU and the DDR SDRAM interface, some basics POST are launched automatically. The following trace is displayed at system start up on the serial console:

```
U-Boot 1.1.5 (Sep 10 2008 - 16:33:05) KONTRON MODULAR COMPUTERS (1.0/08283)

CPU: 750FX v2.3 @ 732.875 MHz
BOARD: CP3210
S/N : 1108311010001
DTT: 1 is 25 C
Checking POST configuration
DRAM: Memory dfcdl init Done
DRAM: -- DIMM1 has 1 banks
ECC Initialization of Bank 0: PASS
CAS Latency = 2 tRP = 5 tRAS = 8 tRCD=5
Total SDRAM memory is 512 MB
```

The next trace represents the tests executed in ROM.

```
POST "mem_data" (cpu0,slow,simple,haltonfail) PASSED
POST "mem_addr" (cpu0,slow,simple,haltonfail) PASSED
```

After the external devices initialization, the other POST are launched.

The next trace represents the tests executed in RAM.

```
POST "mem_data" (cpu0,slow,simple,promptonfail) PASSED
POST "mem_addr" (cpu0,slow,simple,promptonfail) PASSED
POST "userflash" (cpu0,fast,simple) PASSED
POST "temp_sensors" (cpu0,fast,simple) PASSED
POST "ethernet0" (cpu0,fast,simple) PASSED
POST "ethernet1" (cpu0,fast,simple) PASSED
```

6.4 Remove a Test from the ROM List

The user can remove some tests from the default list of tests executed in ROM using the **diag** command.

For example, to remove the test name **`mem_addr`** executed in ROM part, enter:

```
CP3210 > diag rom cfg delete mem_addr
CP3210 >
```

When the board will restart, the test **`mem_addr`** will not be launched.

6.5 Remove a Test from the RAM List

The user can remove some tests from the default list of tests executed in RAM.

For example to remove the test name **`temp_sensors`** executed in RAM part, enter:

```
CP3210 > diag ram cfg delete temp_sensors
CP3210 >
```

When the board will restart, the test **`temp_sensors`** will not be launched.

6.6 Add a Test to the ROM List

The user can add some tests to the default list of tests executed in ROM using the diag command.

For example, to add the test name **mem_addr** executed in ROM, enter:

```
CP3210 > diag rom cfg default mem-addr
CP3210 >
```

When the board will restart, the test **mem_addr** will be automatically launched.

6.7 Add a Test to the RAM List

The user can add some tests to the default list of tests executed in RAM using the diag command.

For example, to add the test name **temp_sensors** executed in RAM part, enter:

```
CP3210 > diag ram cfg default temp-sensor
CP3210 >
```

When the board will restart, the test **temp_sensors** will be automatically launched.

6.8 Run a Test in the U-Boot Command Line

To launch a test named **mem_data** from the command line, enter following command:

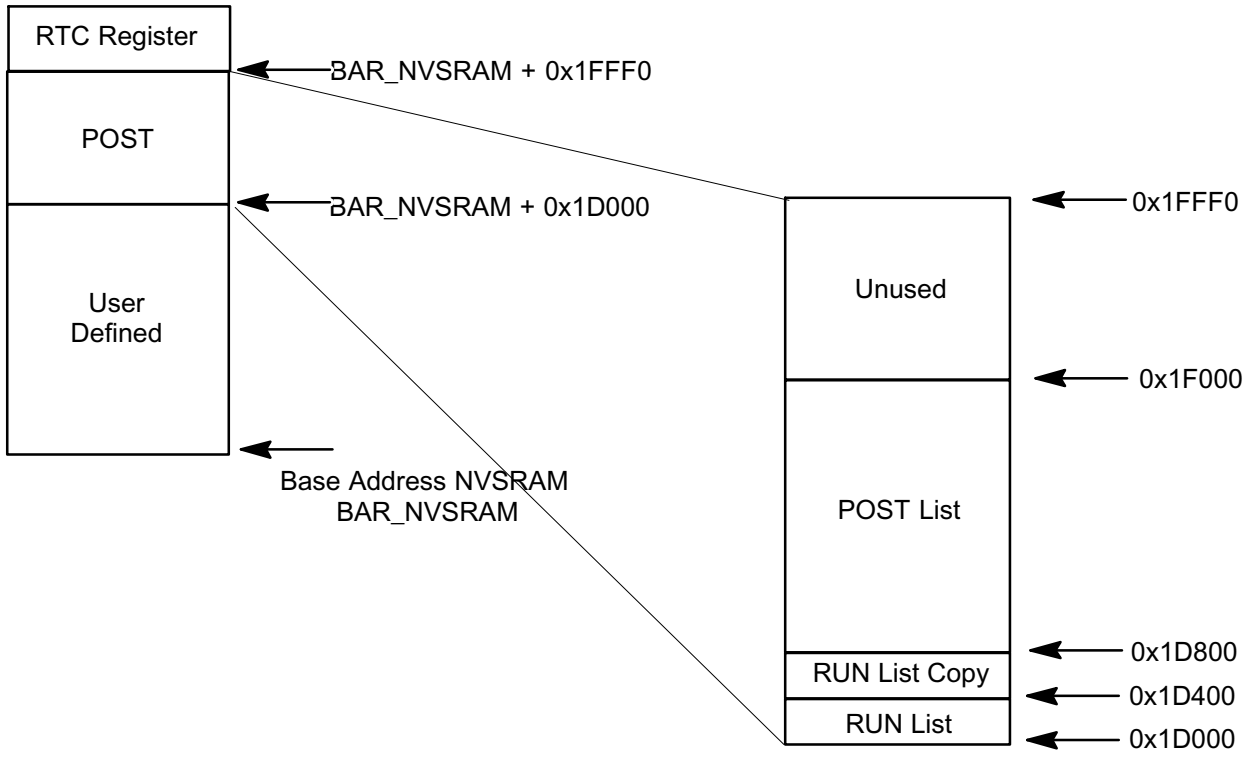
```
CP3210 > diag run mem_data
POST "mem_data" (cpu0,slow,simple) PASSED
CP3210 >
```

6.9 POST Diagnostics

The firmware displays diagnostics of test launched at system start up, by entering the following command:

```
CP3210 > diag stat
Status of POSTs configured to run automatically from ROM :
PASSED : mem_data (cpu0,slow,simple,haltonfail,allresets)
PASSED : mem_addr (cpu0,slow,simple,haltonfail,allresets)
Status of POSTs configured to run automatically from RAM :
PASSED : mem_data (cpu0,slow,simple,promptonfail,allresets)
PASSED : mem_addr (cpu0,slow,simple,promptonfail,allresets)
NOTRUN : rtc (cpu0,fast,simple,poweronreset)
PASSED : userflash (cpu0,fast,simple,allresets)
PASSED : temp_sensors (cpu0,fast,simple,allresets)
PASSED : ethernet0 (cpu0,fast,simple,allresets)
PASSED : ethernet1 (cpu0,fast,simple,allresets)
Status of POSTs configured to run from command line :
NOTRUN : mem_data (cpu0,slow,simple)
NOTRUN : mem_addr (cpu0,slow,simple)
NOTRUN : mem_pattern1 (cpu0,slow,simple)
NOTRUN : mem_pattern2 (cpu0,slow,simple)
NOTRUN : mem_pattern3 (cpu0,slow,simple)
NOTRUN : mem_pattern4 (cpu0,slow,simple)
NOTRUN : mem_bitflip (cpu0,slow,simple)
NOTRUN : mem_addrpattern (cpu0,slow,simple)
NOTRUN : mem_addrpattern2 (cpu0,slow,simple)
NOTRUN : rtc (cpu0,fast,simple)
NOTRUN : userflash (cpu0,fast,simple)
NOTRUN : temp_sensors (cpu0,fast,simple)
NOTRUN : nvsram (cpu0,fast,simple)
NOTRUN : ethernet0 (cpu0,fast,simple)
NOTRUN : ethernet1 (cpu0,fast,simple)
NOTRUN : userflash_bus (cpu0,fast,simple)
CP3210 >
```

6.10 NVSRAM Mapping for POST



➤ POST List Structure

```
# define  POST_MAX_NBFLAGS_TEST                  5

struct post_test {
    int postnum;     /* unique POST number                  */
    char cmd[20];   /* short POST name, used in commands                */
    char desc[80];  /* description of the POST             */
     int flags[POST_MAX_NBFLAGS_TEST+1];
                          /* list of flags describing the run modes supported       */
    int (*test) (int flags);        /* pointer to the test function                */
    int (*init_f) (void);           /* pointer to POST init function               */
    void (*reloc) (void);           /* pointer to POST relocation function         */
};
```

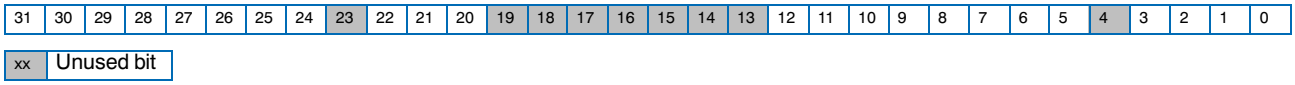
The last valid entry of the POST List area is a **post_test** structure filled with 0.

RUN List Structure

The RUN List area is a series of 32 bits words; starting with **POSTFLAG_BEGIN_LIST** and ending with **POSTFLAG_END_LIST**.

```
#define POSTFLAG_BEGIN_LIST      0x55AA1234   /* First flag of the list
-> the list is valid (only for run_list) */
#define POSTFLAG_END_LIST        0x00000000   /* Last flag of the list (only for run_list) */
```

Each 32-bit word is encoded as described bellow:



- RUNFROM subset
  - Bit 0: set to 1 if test is run from ROM (before RAM initialization)
  - Bit 1: set to 1 if the test is run from RAM (after RAM initialization)
  - Bit 2: set to 1 if the test is run from **diag** command at U-Boot prompt
- RUNON subset
  - Bit 3: set to 1 if the test is run on CPU0
- SPEED subset
  - Bit 5: set to 1 if the test is run in slow mode (better test)
  - Bit 6: set to 1 if the test is run in fast mode: faster than slow mode but may not test everything
- CONFIG subset
  - Bit 7: set to 1 if the test is run in simple mode: no need for external hardware
  - Bit 8: set to 1 if the test is run in complex mode: may need some external hardware (loopback cable for example)
- FAILED subset
  - Bit 9: set to 1 if the test halts immediately if it fails (only for RUN list)
  - Bit 10: set to 1 if the test stays at U-Boot prompt (do not boot the OS) if it fails (only for RUN list)
- RESET subset
  - Bit 11: set to 1 if the test is run after a normal reset
  - Bit 12: set to 1 if the test is run after a power-on reset
- STATUS subset
  - Bit 20: set to 1 if test has been run (only for RUN list)
  - Bit 21: set to 1 if test has passed at last run (only for RUN list)
  - Bit 22: set to 1 if test has already failed (only for RUN list)
- POSTNUM subset
  - Bit 24:31:Power-on self-test number

Each 32-bit word is encoded using following defines:

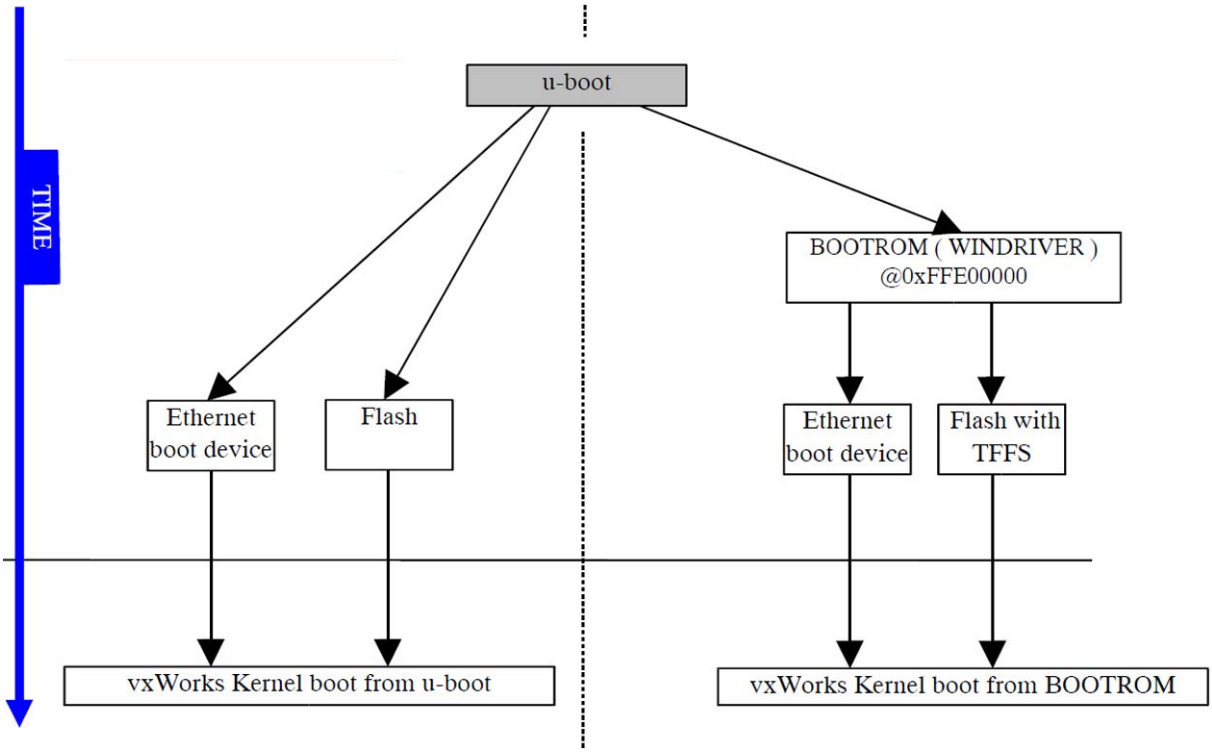
```
#define POSTFLAG_RUNFROM_ROM      0x00000001
#define POSTFLAG_RUNFROM_RAM      0x00000002
#define POSTFLAG_RUNFROM_COMMAND  0x00000004
#define POSTFLAG_RUNFROM_ALL
            (POSTFLAG_RUNFROM_ROM | POSTFLAG_RUNFROM_RAM | POSTFLAG_RUNFROM_COMMAND)
#define POSTFLAG_RUNFROM_MASK     POSTFLAG_RUNFROM_ALL

#define POSTFLAG_RUNON_CPU0       0x00000008

#define POSTFLAG_SPEED_SLOW       0x00000020
#define POSTFLAG_SPEED_FAST       0x00000040
#define POSTFLAG_SPEED_ALL        (POSTFLAG_SPEED_SLOW | POSTFLAG_SPEED_FAST)
#define POSTFLAG_SPEED_MASK       POSTFLAG_SPEED_ALL

#define POSTFLAG_CONFIG_SIMPLE    0x00000080
#define POSTFLAG_CONFIG_COMPLEX   0x00000100
#define POSTFLAG_CONFIG_ALL       (POSTFLAG_CONFIG_SIMPLE | POSTFLAG_CONFIG_COMPLEX)
#define POSTFLAG_CONFIG_MASK      POSTFLAG_CONFIG_ALL

#define POSTFLAG_FAILED_HALT      0x00000200
#define POSTFLAG_FAILED_PROMPT    0x00000400
#define POSTFLAG_FAILED_ALL       (POSTFLAG_FAILED_HALT | POSTFLAG_FAILED_PROMPT)
#define POSTFLAG_FAILED_MASK      POSTFLAG_FAILED_ALL

#define POSTFLAG_RESET_NORMAL     0x00000800
#define POSTFLAG_RESET_POWERON    0x00001000
#define POSTFLAG_RESET_WATCHDOG   0x00002000
#define POSTFLAG_RESET_ALL        (POSTFLAG_RESET_NORMAL | POSTFLAG_RESET_POWERON |
POSTFLAG_RESET_WATCHDOG)
#define POSTFLAG_RESET_TEST       0x00004000
#define POSTFLAG_RESET_MASK       (POSTFLAG_RESET_ALL | POSTFLAG_RESET_TEST)

#define POSTFLAG_TEST_RUN           0x00100000
#define POSTFLAG_TEST_LASTRUN_PASSED  0x00200000
#define POSTFLAG_TEST_ALREADY_FAILED  0x00400000  /*

#define POSTFLAG_POSTNUM_ALL      0xFF000000
#define POSTFLAG_POSTNUM_MASK     0xFF000000
#define POSTFLAG_POSTNUM_SHIFT    24          /* number of bit for right shift to extract test number (only for
run_list) */
#define POSTNUM_TO_POSTFLAG(postnum)  (((postnum) << POSTFLAG_POSTNUM_SHIFT) &
POSTFLAG_POSTNUM_MASK)
 #define POSTFLAG_TO_POSTNUM(postflag) (((postflag) & POSTFLAG_POSTNUM_MASK) >>
POSTFLAG_POSTNUM_SHIFT)
```

Chapter 7 - Booting VxWorks

Several methods to boot VxWorks Operating System on the CP3210 are available.

- The user can boot directly a VxWorks kernel from U-Boot. U-Boot can boot the kernel from the Ethernet or from the Flash.
- The user can boot a bootrom from U-Boot, and the bootrom can boot the VxWorks kernel. The bootrom can boot the kernel from the Ethernet or from the flash using TFFS file system.

The following picture details the different ways to boot VxWorks:



The VxWorks kernel boot time from U-Boot (left side of the above picture) is better (up to 20% lower) than the VxWorks kernel boot time from a bootrom (right side of the above picture).

7.1 Print/Set VxWorks Parameters under U-Boot

U-Boot firmware on CP3210 board is able to configure a bootrom bootstring. This bootstring is necessary for booting a VxWorks kernel or a bootrom.

The user can print (with a bootrom menu look), the default bootstring programmed by running:

```
CP3210 > vxworks print

boot device             : mgi
unit number             : 0
processor number        : 0
host name               : sunblade
file name               : /home/test/vxWorksDec
inet on ethernet (e)    : 192.168.0.2:FFFFFF00
host inet (h)       : 192.168.0.1
user (u)            : test
ftp password (pw)       : testfab
flags (f)           : 0x8

CP3210 >
```

The user can configure these bootstring parameters:

```
CP3210 > vxworks set

'.' = clear field; '-' = go to previous field; ^D = quit

boot device             : mgi0
processor number        : 0
host name               : sunblade
file name               : /home/test/vxWorksDec
inet on ethernet (e)    : 192.168.0.2:FFFFFF00
inet on backplane (b)   :
host inet (h)           : 192.168.0.1
gateway inet (g)        :
user (u)                : test
ftp password (pw) (blank = use rsh): testfab
flags (f)               : 0x8
target name (tn)        : cp3210
startup script (s)      :
other (o)               :

Saving bootargs in Global Board Feature parameters in flash ...
erase 0xfc060000 0xfc07ffff
.Erased 1 sectors
cp 0x0fbe92b0 0xfc060000 0x00008000
Copy to Flash... 0%[====================] 100%
done
Saving bootargs in uboot parameters in flash ...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
uboot parameters saved !
CP3210 >
```

On the previous trace, the board settings for bootrom are (See WindRiver documentation):

- Ethernet device use: mgi0 (Ethernet num 0 on CP3210 board)
- Processor num: 0
- Server Machine name: sunblade
- Kernel File: /home/test/vxWorksDec
- inet on ethernet: 192.168.0.2:FFFFFF00 (IP address : Network Mask)
- Host Inet: 192.168.0.1 (server IP address)
- user: test (name of user login for RSH connection protocol)

After setting the bootrom parameters, the bootstring generated appears in the U-Boot parameters as follow (using the **`printenv`** command) :

```
bootargs=mgi(0,0)sunblade:/home/test/vxWorksDec   e=192.168.0.2:FFFFFF00 h=192.168.0.1 u=test pw=testfab
f=0x8 tn=cp3210
```

The Ethernet address in the U-Boot parameters with the name **`ipaddress`** needs to be the same in the bootrom string in **`inet on ethernet (e)`** parameter.

7.2 Boot VxWorks from Ethernet under U-Boot

First, the user needs to configure the U-Boot parameters as explained in section 4.2 “Set the Ethernet Parameters“ page 10 and set the **loadaddr** as explained in section 4.4 “Set the Load Address" page 11.

Then, the user needs to configure the boot string parameter of the bootrom using the command **vxworks set** as explained in section 7.1 “Print/Set VxWorks Parameters under U-Boot". The trace below sums up the procedure:

```
CP3210 > setenv ipaddr 192.168.0.2
CP3210 > setenv serverip 192.168.0.1
CP3210 > setenv loadaddr 0x2000000
CP3210 > vxworks set

'.' = clear field; '-' = go to previous field; ^D = quit

boot device                     : mgi0
processor number                : 0
host name                       : sunblade pcbist2
file name                       : /home/test/vxWorksDec   /home/user/cp3210/vxWorks
inet on ethernet (e)        : 192.168.0.2:FFFFFF00
inet on backplane (b)       :
host inet (h)               : 192.168.0.1
gateway inet (g)            :
user (u)                    : test user-name
ftp password (pw) (blank = use rsh): testfab .
flags (f)                   : 0x8
target name (tn)            : cp3210
startup script (s)          :
other (o)                       :

Saving bootargs in Global Board Feature parameters in flash ...
erase 0xfc060000 0xfc07ffff
.Erased 1 sectors
cp 0x0fbe92b0 0xfc060000 0x00008000
Copy to Flash... 0%[====================] 100%
done
Saving bootargs in uboot parameters in flash ...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
uboot parameters saved !
CP3210 > saveenv
Saving Environment to Flash...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
CP3210 >
```

On the previous trace, the board settings for U-Boot are:

- IP address of the CP3210 : 192.168.0.2
- IP address of the Server : 192.168.0.1
- Load address of the download file on CP3210 : 0x2000000

On the previous trace, the board settings for bootrom are (See WindRiver documentation):

- Ethernet device use: mgi0 (Ethernet num 0 on CP3210 board)
- Processor num: 0
- Server Machine name: pcbist2
- Kernel File: /home/user/cp3210/vxWorks
- inet on ethernet: 192.168.0.2:FFFFFF00 (IP address : Network Mask)
- Host Inet: 192.168.0.1 (server IP address)
- user: user-name(name of user login for RSH connection protocol)

The file **`/home/user/cp3210/vxWorks`** on the server machine needs to be copied in the TFTP server folder. By the way the U-Boot can download the code.

At this point the board CP3210 is configured, so the kernel can be downloaded from a TFTP server. Obviously an external machine must have a TFTP server to able the CP3210 to download the VxWorks kernel image.

To run the kernel, enter following command:

```
CP3210 > tftp 0x2000000 vxWorks
Ethernet status port 0: Link up, Full Duplex, Speed 1 Gbps
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.2
Filename 'vxWorks'.
Load address: 0x2000000
Loadin          ##################################################################
                ##################################################################
                ##################################################################
                ##################################################################
                ############################################################
done
Bytes transferred = 1641148 (190abc hex)
CP3210 > bootvx
## Ethernet MAC address not copied to NV RAM
Loading .text @ 0x00200000 (1295872 bytes)
Loading .init$00 @ 0x0033c600 (16 bytes)
Loading .init$99 @ 0x0033c610 (16 bytes)
Loading .fini$00 @ 0x0033c620 (16 bytes)
Loading .fini$99 @ 0x0033c630 (16 bytes)
Loading .data @ 0x0033c640 (71504 bytes)
Clearing .bss @ 0x0034dd90 (86800 bytes)
## Using bootline (@ 0x4200): mgi(0,0)pcbist2:/home/bsa/WindRiver/vxworks-6.2/target/config/cp3210/vxWorks
e=192.168.0.2:FFFFFF00 h=192.168.0.1 u=bsa f=0x8 tn=cp3210
## Starting vxWorks at 0x00200000 ...
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase II
Ctrl env init... done.
CPU if init..... done.
TWSI init..... done.
Memory init..... done.
Int ctrl init... done.
IDMA channels... stopped.
XOR channels.... stopped.
Eth unit init... done.
```

```
Marvell Controller initialization Done.
Init Shared Memory Manager....
Addr:0x00100000 Size:0x00100000 done
Power Management Activated... done
CPU L2 cache initialization... done
Detected DRAM size 512MB
IDMA init....... done
IDMA int Ctrl... done
PCI0 init....... done
PCI1 init....... done
XOR init........ done
256KB integrated SRAM - MV_CACHE_COHER_HW_WB
Integrated SRAM init ECC Done.
Integ SRAM init... done
Init Auxiliary Clock Timer ...... Done
System Slot Board
Boot Flash 128MB detected
Main Flash 256MB detected
vxGppIntConnect: cause = 10
Attaching interface lo0... BSA : in ipCreateAdd
done
mgi0 Phy. Reset Done
mgi0 Phy. Autoneg. Done
Ethernet port #0 initialized
mgi0 Interface Loaded
mgi0 Interface Started
Attached IPv4 interface to mgi unit 0
BSA : in ipCreateAdd
BSA : in ipCreateAdd
Loading symbol table from pcbist2:/home/bsa/WindRiver/vxworks-6.2/target/config/cp3210/vxWorks.sym ...done
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
      ]]]]]]]]]]]  ]]]]     ]]]]]]]]]]       ]]              ]]]]         (R)
 ]     ]]]]]]]]]  ]]]]]]     ]]]]]]]]       ]]               ]]]]
 ]]     ]]]]]]]  ]]]]]]]]     ]]]]]] ]     ]]                ]]]]
 ]]]     ]]]]] ]    ]]]  ]     ]]]] ]]]   ]]]]]]]]]  ]]]] ]] ]]]]  ]]   ]]]]]
 ]]]]     ]]]  ]]    ]  ]]]     ]] ]]]]] ]]]]]]   ]] ]]]]]]] ]]]] ]]   ]]]]
 ]]]]]     ]  ]]]]     ]]]]]      ]]]]]]]] ]]]]   ]] ]]]]    ]]]]]]]    ]]]]
 ]]]]]]      ]]]]]     ]]]]]]    ]  ]]]]]  ]]]]   ]] ]]]]    ]]]]]]]]    ]]]]
 ]]]]]]]    ]]]]]  ]    ]]]]]]  ]    ]]]   ]]]]   ]] ]]]]    ]]]] ]]]]    ]]]]
 ]]]]]]]]  ]]]]]  ]]]    ]]]]]]]      ]     ]]]]]]]  ]]]]    ]]]]  ]]]] ]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]       Development System
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks 6.2
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]       KERNEL: WIND version 2.8
 ]]]]]]]]]]]]]]]]]]]]]]]]]]       Copyright Wind River Systems, Inc., 1984-2005

CPU: CP3210/IBM750Fx (2.3). Processor #0.
Memory Size: 0x1bfa000. BSP version 3.0/08245.
Created: Sep 11 2008, 10:48:12
ED&R Policy Mode: deployed
WDB Comm Type: WDB_COMM_NETWORK
WDB: Ready.
->
```

7.3 Boot VxWorks from Flash System under U-Boot

First, the user needs to configure the U-Boot parameters as explained in section 4.2 “Set the Ethernet Parameters“ page 10 and set the **`loadaddr`** as explained in section 4.4 “Set the Load Address" page 11.

Then, the user needs to configure the boot string parameter of the bootrom by using the command **`vxworks set`** as explained in section 7.1 “Print/Set VxWorks Parameters under U-Boot". The trace below sums up the procedure:

```
CP3210 > setenv ipaddr 192.168.0.2
CP3210 > setenv serverip 192.168.0.1
CP3210 > setenv loadaddr 0x2000000
CP3210 > vxworks set

'.' = clear field; '-' = go to previous field; ^D = quit

boot device                     : mgi0
processor number                : 0
host name                       : sunblade pcbist2
file name                       : /home/test/vxWorksDec   /home/user/cp3210/vxWorks
inet on ethernet (e)        : 192.168.0.2:FFFFFF00
inet on backplane (b)       :
host inet (h)               : 192.168.0.1
gateway inet (g)            :
user (u)                    : test user-name
ftp password (pw) (blank = use rsh): testfab .
flags (f)                   : 0x8
target name (tn)            : cp3210
startup script (s)          :
other (o)                       :

Saving bootargs in Global Board Feature parameters in flash ...
erase 0xfc060000 0xfc07ffff
.Erased 1 sectors
cp 0x0fbe92b0 0xfc060000 0x00008000
Copy to Flash... 0%[====================] 100%
done
Saving bootargs in uboot parameters in flash ...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
uboot parameters saved !
CP3210 > saveenv
Saving Environment to Flash...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
CP3210 >
```

On the previous trace, the board settings for U-Boot are:

- IP address of the CP3210: 192.168.0.2
- IP address of the Server: 192.168.0.1
- Load address of the download file on CP3210: 0x2000000

On the previous trace, the board settings for bootrom are (See WindRiver documentation):

- Ethernet device use: mgi0 (ethernet num 0 on CP3210 board)
- Processor num: 0
- Server Machine name: pcbist2
- Kernel File: /home/user/cp3210/vxWorks
- Inet on Ethernet: 192.168.0.2:FFFFFF00 (IP address: Network Mask)
- Host Inet: 192.168.0.1 (server IP address)
- User: user-name (name of user login for RSH connection protocol)

The CP3210 is configured.

Next state consists in burning the VxWorks code in the flash. We use, in following example, the System Flash address **0xFE000000** to store the VxWorks ELF (Executable Loading File) image. Trace to burn the code into the flash:

```
CP3210> tftp 0x2000000 vxWorks
Ethernet status port 0: Link up, Full Duplex, Speed 1 Gbps
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.2
Filename 'vxWorks'.
Load address: 0x2000000
Loading:        ###################################################################
                ###################################################################
                ###################################################################
                ###################################################################
                ##################################################################
done
Bytes transferred = 1641148 (190abc hex)
CP3210 > erase 0xFE000000 0xFE1FFFFF
.................Erased 16 sectors
CP3210 > cp 0x2000000 0xFE000000 0x80000
Copy to Flash... 0%[====================] 100%
done
CP3210 > setenv loadaddr 0xFE000000
CP3210 > md 0xFE000000
fe000000: 7f454c46 01020100 00000000 00000000    .ELF............
fe000010: 00020014 00000001 00200000 00000034    ......... .....4
fe000020: 0014de64 80000000 00340020 00010028    ...d.....4. ...(
fe000030: 000c0009 00000001 00000060 00200000    ...........`. ..
fe000040: 00200000 0014dd90 001630a0 00000007    . ........0.....
fe000050: 00000020 00000000 00000000 00000000    ... ............
fe000060: 7c671b78 3c404c00 38420064 90400900    |g.x<@L.8B.d.@..
fe000070: 48000019 3c200020 38210000 3821fff0    H...< . 8!..8!..
fe000080: 7ce33b78 48030e70 7fe802a6 7c000278    |.;xH..p....|..x
fe000090: 7c1043a6 7c1143a6 7c1243a6 7c1343a6    |.C.|.C.|.C.|.C.
fe0000a0: 7c5f42a6 5442843e 28028001 41820014    |_B.TB.>(...A...
fe0000b0: 28028002 4182000c 28028003 40820014    (...A...(...@...
fe0000c0: 7c1443a6 7c1543a6 7c1643a6 7c1743a6    |.C.|.C.|.C.|.C.
fe0000d0: 3c400000 7c0004ac 4c00012c 7c50fba6     <@..|...L..,|P..
fe0000e0: 4c00012c 7c0004ac 38402000 7c0004ac    L..,|...8@ .|...
fe0000f0: 7c400124 4c00012c ff80010c ff00010c    |@.$L..,........
CP3210 >
```

- The image can be a standalone kernel commonly named VxWorks.st.
- The U-Boot parameter **`loadaddr`** is set up with the address where the kernel is burnt in flash; 0xFE000000 in above example.
- These parameters can be saved permanently by using the **`saveenv`** command.

Next stage details how to startup the VxWorks code from the flash:

```
CP3210 > bootvx
## Ethernet MAC address not copied to NV RAM
Loading .text @ 0x00200000 (1295872 bytes)
Loading .init$00 @ 0x0033c600 (16 bytes)
Loading .init$99 @ 0x0033c610 (16 bytes)
Loading .fini$00 @ 0x0033c620 (16 bytes)
Loading .fini$99 @ 0x0033c630 (16 bytes)
Loading .data @ 0x0033c640 (71504 bytes)
Clearing .bss @ 0x0034dd90 (86800 bytes)
## Using bootline (@ 0x4200): mgi(0,0)pcbist2:/home/user/cp3210/vxWorks
e=192.168.0.2:FFFFFF00 h=192.168.0.1 u=bsa f=0x8 tn=cp3210
## Starting vxWorks at 0x00200000 ...
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase II
Ctrl env init... done.
CPU if init..... done.
TWSI init..... done.
Memory init..... done.
Int ctrl init... done.
IDMA channels... stopped.
XOR channels.... stopped.
Eth unit init... done.
Marvell Controller initialization Done.
Init Shared Memory Manager....
Addr:0x00100000 Size:0x00100000 done
Power Management Activated... done
CPU L2 cache initialization... done
Detected DRAM size 512MB
IDMA init..... done
IDMA int Ctrl... done
PCI0 init..... done
PCI1 init..... done
XOR init..... done
256KB integrated SRAM - MV_CACHE_COHER_HW_WB
Integrated SRAM init ECC Done.
Integ SRAM init... done
Init Auxiliary Clock Timer ..... Done
System Slot Board
Boot Flash 128MB detected
Main Flash 256MB detected
vxGppIntConnect: cause = 10
Attaching interface lo0... BSA : in ipCreateAdd
done
mgi0 Phy. Reset Done
mgi0 Phy. Autoneg. Done
Ethernet port #0 initialized
mgi0 Interface Loaded
mgi0 Interface Started
Attached IPv4 interface to mgi unit 0
BSA : in ipCreateAdd
BSA : in ipCreateAdd
```

```
Loading symbol table from pcbist2:/home/user/cp3210/vxWorks.sym ...done
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
      ]]]]]]]]]]]  ]]]]     ]]]]]]]]]]       ]]              ]]]]         (R)
 ]     ]]]]]]]]]  ]]]]]]     ]]]]]]]]       ]]               ]]]]
 ]]     ]]]]]]]  ]]]]]]]]     ]]]]]] ]     ]]                ]]]]
 ]]]     ]]]]] ]    ]]]  ]     ]]]] ]]]   ]]]]]]]]]  ]]]] ]] ]]]]  ]]   ]]]]]
 ]]]]     ]]]  ]]    ]  ]]]     ]] ]]]]] ]]]]]]   ]] ]]]]]]] ]]]] ]]   ]]]]
 ]]]]]     ]  ]]]]     ]]]]]     ]]]]]]]] ]]]]   ]] ]]]]    ]]]]]]]    ]]]]
 ]]]]]]      ]]]]]     ]]]]]]    ]  ]]]]]  ]]]]   ]] ]]]]    ]]]]]]]]    ]]]]
 ]]]]]]]    ]]]]]  ]    ]]]]]]  ]    ]]]   ]]]]   ]] ]]]]    ]]]] ]]]]    ]]]]
 ]]]]]]]]  ]]]]]  ]]]    ]]]]]]]      ]     ]]]]]]]  ]]]]    ]]]]  ]]]] ]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]       Development System
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks 6.2
 ]]]]]]]]]]]]]]]]]]]]]]]]]]       KERNEL: WIND version 2.8
 ]]]]]]]]]]]]]]]]]]]]]]]]]       Copyright Wind River Systems, Inc., 1984-2005

CPU: CP3210/IBM750Fx (2.3). Processor #0.
Memory Size: 0x1bfa000. BSP version 3.0/08245.
Created: Sep 11 2008, 10:48:12
ED&R Policy Mode: deployed
WDB Comm Type: WDB_COMM_NETWORK
WDB: Ready.
->
```

7.4 Burn and Execute a Bootrom

Configure the CP3210 as follow:

```
CP3210 > setenv ipaddr 192.168.0.2
CP3210 > setenv serverip 192.168.0.1
CP3210 > setenv loadaddr 0x2000000
CP3210 > vxworks set

'.' = clear field; '-' = go to previous field; ^D = quit

boot device                     : mgi0
processor number                : 0
host name                       : sunblade pcbist2
file name                        : /home/test/vxWorksDec   /home/user/cp3210/vxWorks
inet on ethernet (e)        : 192.168.0.2:FFFFFF00
inet on backplane (b)       :
host inet (h)               : 192.168.0.1
gateway inet (g)            :
user (u)                    : test user-name
ftp password (pw) (blank = use rsh): testfab .
flags (f)                   : 0x8
target name (tn)            : cp3210
startup script (s)          :
other (o)                       :

Saving bootargs in Global Board Feature parameters in flash ...
erase 0xfc060000 0xfc07ffff
.Erased 1 sectors
cp 0x0fbe92b0 0xfc060000 0x00008000
Copy to Flash... 0%[====================] 100%
done
Saving bootargs in uboot parameters in flash ...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
uboot parameters saved !
CP3210 > saveenv
Saving Environment to Flash...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
CP3210 >
```

To configure the U-Boot parameters, refer to section 4.2 “Set the Ethernet Parameters“ and to set the **`loadaddr`** refer to section 4.4 “Set the Load Address".

The bootrom can be generated under the WindRiver environment. The generated file named **`bootrom.bin`**. is needed by the user to compile.

This code is built to be placed at the address 0xFFE00000.

The **bootrom.bin** file needs to be downloaded into the CP3210 and burned at the address 0xFFE00000 in System Flash, as follows:.

```
CP3210 > tftp 0x2000000 bootrom.bin
Ethernet status port 0: Link up, Full Duplex, Speed 1 Gbps
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.2
Filename 'bootrom.bin'.
Load address: 0x2000000
Loading:        ##################################################################
                ####################
done
Bytes transferred = 435536 (6a550 hex)
CP3210 > erase 0xFFE00000 0xFFEFFFFF
.....Erased 8 sectors
CP3210 > cp 0x2000000 0xFFE00000 0x40000
Copy to Flash... 0%[====================] 100%
done
CP3210 > md 0xFFE00000
ffe00000: 00000000 00000000 00000000 00000000                  ................
ffe00010: 00000000 00000000 00000000 00000000                  ................
ffe00020: 00000000 00000000 00000000 00000000                  ................
ffe00030: 00000000 00000000 00000000 00000000                  ................
ffe00040: 00000000 00000000 00000000 00000000                  ................
ffe00050: 00000000 00000000 00000000 00000000                  ................
ffe00060: 00000000 00000000 00000000 00000000                  ................
ffe00070: 00000000 00000000 00000000 00000000                  ................
ffe00080: 00000000 00000000 00000000 00000000                  ................
ffe00090: 00000000 00000000 00000000 00000000                  ................
ffe000a0: 00000000 00000000 00000000 00000000                  ................
ffe000b0: 00000000 00000000 00000000 00000000                  ................
ffe000c0: 00000000 00000000 00000000 00000000                  ................
ffe000d0: 00000000 00000000 00000000 00000000                  ................
ffe000e0: 00000000 00000000 00000000 00000000                   ................
ffe000f0: 00000000 00000000 00000000 00000000                  ................
CP3210 >
ffe00100: 4c00012c 48000039 4c00012c 48000069                  L..,H..9L..,H..i
ffe00110: 436f7079 72696768 74203139 38342d31                  Copyright 1984-1
ffe00120: 39393920 57696e64 20526976 65722053                  999 Wind River S
ffe00130: 79737465 6d732c20 496e632e 39600002                  ystems, Inc.9`..
ffe00140: 3ca0f100 60a50268 3c80f200 5484803e                  <...`..h<...T..>
ffe00150: 7c802d2c 3c40f100 60420278 7c60142c                  |.-,<@..`B.x|`.,
ffe00160: 64630007 6063fe0f 7c60152c 7c0004ac dc..`c..|`.,|...
ffe00170: 48000009 7c6b1b78 7c0004ac 3c800020                  H...|k.x|...<..
ffe00180: 3884037c 3ca00020 38a50100 3cc0ffe0 8..|<.. 8...<...
ffe00190: 38c60100 7c852050 7c843214 7c8803a6                  8...|. P|.2.|...
ffe001a0: 4e800021 2c0b0002 40820144 3c800020                  N..!,...@..D<..
ffe001b0: 38840cf8 3ca00020 38a50100 3cc0ffe0  8...<.. 8...<...
ffe001c0: 38c60100 7c852050 7c843214 7c8803a6                  8...|. P|.2.|...
ffe001d0: 4e800021 3c800020 38840ac4 3ca00020                  N..!<.. 8...<..
ffe001e0: 38a50100 3cc0ffe0 38c60100 7c852050 8...<...8...|. P
ffe001f0: 7c843214 7c8803a6 4e800021 3ca00300                  |.2.|...N..!<...
CP3210 >
```

The board can start up the bootrom with a direct go to 0xFFE00100 address. The command to run the bootrom is:

```
CP3210 > go ffe00100
```

7.5 Boot VxWorks from Ethernet under Bootrom

See the trace below:

```
CP3210 > go ffe00100
## Starting application at 0xFFE00100 ...
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase I
Ctrl env init... done.
CPU if init..... done.
TWSI init....... done.
Memory init..... Jump to ROMable DRAM config...
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase II
Ctrl env init... done.
CPU if init..... done.
TWSI init....... done.
Memory init..... done.
Int ctrl init... done.
IDMA channels... stopped.
XOR channels.... stopped.
Eth unit init... done.
Marvell Controller initialization Done.
Init Shared Memory Manager....
 Addr:0x00100000 Size:0x00100000 done

CPU L2 cache initialization... done
Detected DRAM size 512MB
IDMA init....... done
IDMA int Ctrl... done
PCI0 init....... done
PCI1 init....... done
XOR init........ done
256KB integrated SRAM - MV_CACHE_COHER_HW_WB
Integrated SRAM init ECC Done.
Integ SRAM init... done
Init Auxiliary Clock Timer ..... Done
System Slot Board
Boot Flash 128MB detected
Main Flash 256MB detected
                                        VxWorks System Boot

Copyright 1984-2005 Wind River Systems, Inc.



CPU: CP3210/IBM750Fx (2.3)
Version: VxWorks 6.2
BSP version: 3.0/08245
Creation date: Sep 11 2008, 15:55:14
```

```
Press any key to stop auto-boot...
 0
auto-booting...


boot device                     : mgi
unit number                     : 0
processor number                : 0
host name                       : pcbist2
file name : /home/user/cp3210/vxWorks
inet on ethernet (e)            : 192.168.0.2:FFFFFF00
host inet (h)       : 192.168.0.1
user (u)            : user-name
flags (f)           : 0x8
target name (tn)    : cp3210

Attaching
interface lo0... BSA : in ipCreateAdd
done
mgi0 Phy. Reset Done
mgi0 Phy. Autoneg. Done
Ethernet port #0 initialized
mgi0 Interface Loaded
mgi0 Interface Started
muxDevLoad failed for device entry 1!
muxDevLoad failed for device entry 2!
Attached IPv4 interface to mgi unit 0
BSA : in ipCreateAdd
Loading... BSA : in ipCreateAdd
1483520
Starting at 0x200000...

mgi0 Interface Down
mgi0 Interface Stopped
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase II
Ctrl env init... done.
CPU if init..... done.
TWSI init..... done.
Memory init..... done.
Int ctrl init... done.
IDMA channels... stopped.
XOR channels.... stopped.
Eth unit init... done.
Marvell Controller initialization Done.
Init Shared Memory Manager....
 Addr:0x00100000 Size:0x00100000 done
```

```
CPU L2 cache initialization... done
Detected DRAM size 512MB
IDMA init....... done
IDMA int Ctrl... done
PCI0 init....... done
PCI1 init....... done
XOR init........ done
256KB integrated SRAM - MV_CACHE_COHER_HW_WB
Integrated SRAM init ECC Done.
Integ SRAM init... done
Init Auxiliary Clock Timer ...... Done
System Slot Board
Boot Flash 128MB detected
Main Flash 256MB detected
vxGppIntConnect: cause = 10
Attaching interface lo0... BSA : in ipCreateAdd
done
mgi0 Phy. Reset Done
mgi0 Phy. Autoneg. Done
Ethernet port #0 initialized
mgi0 Interface Loaded
mgi0 Interface Started

Loading symbol table from pcbist2:/home/user/config/cp3210/vxWorks.sym ...done

]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
      ]]]]]]]]]]]  ]]]]     ]]]]]]]]]]       ]]              ]]]]         (R)
 ]     ]]]]]]]]]  ]]]]]]     ]]]]]]]]       ]]               ]]]]
 ]]     ]]]]]]]  ]]]]]]]]     ]]]]]] ]     ]]                ]]]]
 ]]]     ]]]]] ]    ]]]  ]     ]]]] ]]]   ]]]]]]]]]  ]]]] ]] ]]]]  ]]   ]]]]]
 ]]]]     ]]]  ]]    ]  ]]]     ]] ]]]]] ]]]]]]   ]] ]]]]]]] ]]]] ]]   ]]]]
 ]]]]]     ]  ]]]]     ]]]]]      ]]]]]]]] ]]]]   ]] ]]]]    ]]]]]]]    ]]]]
 ]]]]]]      ]]]]]     ]]]]]]    ]  ]]]]]  ]]]]   ]] ]]]]    ]]]]]]]]    ]]]]
 ]]]]]]]    ]]]]]  ]    ]]]]]]  ]    ]]]   ]]]]   ]] ]]]]    ]]]] ]]]]    ]]]]
 ]]]]]]]]  ]]]]]  ]]]    ]]]]]]]      ]     ]]]]]]]  ]]]]    ]]]]  ]]]] ]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]       Development System
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks 6.2
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]       KERNEL: WIND version 2.8
 ]]]]]]]]]]]]]]]]]]]]]]]]]]       Copyright Wind River Systems, Inc., 1984-2005

CPU: CP3210/IBM750Fx (2.3). Processor #0.
Memory Size: 0x1bfa000. BSP version 3.0/08245.
Created: Sep 11 2008, 16:02:12
ED&R Policy Mode: deployed
WDB Comm Type: WDB_COMM_NETWORK
WDB: Ready.
->
```

7.6 Boot VxWorks from Flash User under Bootrom

TFFS is used as the device boot. Refer to section 7.2.2 "Boot by TFFS" in the “Release NOTES BSP CP3210 Workbench 2.4/VxWorks 6.2" (SD.DT.F30) for detailed information on bootrom parameters and TFFS installation.

See the trace below:

```
CP3210 > go ffe00100
## Starting application at 0xFFE00100 ...
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase I
Ctrl env init... done.
CPU if init..... done.
TWSI init..... done.
Memory init..... Jump to ROMable DRAM config...
Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase II
Ctrl env init... done.
CPU if init..... done.
TWSI init..... done.
Memory init..... done.
Int ctrl init... done.
IDMA channels... stopped.
XOR channels.... stopped.
Eth unit init... done.
Marvell Controller initialization Done.
Init Shared Memory Manager....
 Addr:0x00100000 Size:0x00100000 done

CPU L2 cache initialization... done
Detected DRAM size 512MB
IDMA init..... done
IDMA int Ctrl... done
PCI0 init..... done
PCI1 init..... done
XOR init..... done
256KB integrated SRAM - MV_CACHE_COHER_HW_WB
Integrated SRAM init ECC Done.
Integ SRAM init... done
Init Auxiliary Clock Timer ..... Done
System Slot Board
Boot Flash 128MB detected
Main Flash 256MB detected
                                        VxWorks System Boot


Copyright 1984-2005 Wind River Systems, Inc.
```

```
CPU: CP3210/IBM750Fx (2.3)
Version: VxWorks 6.2
BSP version: 3.0/08245
Creation date: Sep 11 2008, 15:55:14
Press any key to stop auto-boot...
 1
[VxWorks Boot]: p

boot device                 : tffs=0,0
unit number                 : 0
processor number            : 0
host name                   : pcbist2
file name                   : /tffs/vxWorks
inet on ethernet (e) : 192.168.0.2:FFFFFF00
host inet (h)        : 192.168.0.1
user (u)             : user-name
flags (f)            : 0x8
[VxWorks Boot]    : @

[VxWorks Boot]: @

boot device         : tffs=0,0
unit number         : 0
processor number : 0
host name                   : pcbist2
file name                   : /tffs/vxWorks
inet on ethernet (e) : 192.168.0.2:FFFFFF00
host inet (h)        : 192.168.0.1
user (u)             : bsa
flags (f)            : 0x8

Attaching to TFFS... usrTffsConfig: tffsName =/tffs0 fsmName =/tffs0:0 fileName =/tffs/vxWorks devName= /tffs/
done.
Loading /tffs/vxWorks...1598800
Starting at 0x200000...

Detect System clock (SysClk) 133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz
mvCtrlInit Phase II
Ctrl env init... done.
CPU if init..... done.
TWSI init..... done.
Memory init..... done.
Int ctrl init... done.
IDMA channels... stopped.
XOR channels.... stopped.
Eth unit init... done.
Marvell Controller initialization Done.
Init Shared Memory Manager....
Addr:0x00100000 Size:0x00100000 done

CPU L2 cache initialization... done
Detected DRAM size 512MB
IDMA init..... done
IDMA int Ctrl... done
PCI0 init..... done
```

```
PCI1 init....... done
XOR init........ done
256KB integrated SRAM - MV_CACHE_COHER_HW_WB
Integrated SRAM init ECC Done.
Integ SRAM init... done
Init Auxiliary Clock Timer ...... Done
System Slot Board
Boot Flash 128MB detected
Main Flash 256MB detected
Target Name: vxTarget
done
mgi0 Phy. Reset Done
mgi0 Phy. Autoneg. Done
Ethernet port #0 initialized
mgi0 Interface Loaded
mgi0 Interface Started
usrTffsConfig: tffsName =/tffs0 fsmName =/tffs0:0 fileName =/tffs/vxWorks devName= /tffs/

Loading symbol table from /tffs/vxWorks.sym ...done
 ]
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
      ]]]]]]]]]]]  ]]]]     ]]]]]]]]]]      ]]              ]]]]         (R)
 ]     ]]]]]]]]]  ]]]]]]     ]]]]]]]]      ]]               ]]]]
 ]]     ]]]]]]]  ]]]]]]]]     ]]]]]] ]    ]]                ]]]]
 ]]]     ]]]]] ]    ]]]  ]     ]]]] ]]]   ]]]]]]]]] ]]]] ]] ]]]]  ]]   ]]]]]
 ]]]]     ]]]  ]]    ]  ]]]     ]] ]]]]] ]]]]]]  ]] ]]]]]]] ]]]] ]]   ]]]]
 ]]]]]     ]  ]]]]     ]]]]]      ]]]]]]]] ]]]]  ]] ]]]]    ]]]]]]]    ]]]]
 ]]]]]]      ]]]]]     ]]]]]]    ]  ]]]]]  ]]]]  ]] ]]]]    ]]]]]]]]    ]]]]
 ]]]]]]]    ]]]]]  ]    ]]]]]]  ]    ]]]   ]]]]  ]] ]]]]    ]]]] ]]]]    ]]]]
 ]]]]]]]]  ]]]]]  ]]]    ]]]]]]]      ]     ]]]]]]] ]]]]    ]]]]  ]]]] ]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]      Development System
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]      VxWorks 6.2
 ]]]]]]]]]]]]]]]]]]]]]]]]]]      KERNEL: WIND version 2.8
 ]]]]]]]]]]]]]]]]]]]]]]]]]      Copyright Wind River Systems, Inc., 1984-2005

CPU: CP3210/IBM750Fx (2.3). Processor #0.
Memory Size: 0x1bfa000. BSP version 3.0/08245.
Created: Sep 15 2008, 18:18:47
ED&R Policy Mode: deployed
WDB Comm Type: WDB_COMM_NETWORK
WDB: Ready.
->
```

Chapter 8 - Booting Linux


Following traces are given as examples.

8.1 Boot Linux-NFS

```
CP3210 > printenv b_linux_nfs
b_linux_nfs=tftp 0x800000 uImagePEC7;setenv bootargs root=/dev/nfs rw nfsroot=192.93.161.147:/home/opt/-
ELDK_41/ppc_7xx ip=$(ipaddr):$(serverip)$(bootargs_end);bootm 0x800000
CP3210 > run b_linux_nfs
Ethernet status port 0: Link up, Half Duplex, Speed 100 Mbps
Using mv_enet0 device
TFTP from server 192.93.161.147; our IP address is 192.93.161.150
Filename 'uImagePEC7'.
Load address: 0x800000
Loading: ##################################################################
         ##################################################################
         ############################
done
Bytes transferred = 1792346 (1b595a hex)
## Booting image at 00800000 ...
   Image Name:   Linux-2.6.11-centaure_1.0
   Created:      2008-09-05  16:33:07 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    1792282 Bytes =  1.7 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
UBoot : Starting up the linux kernel
.......
.......
.......
INIT: version 2.85 booting
              Welcome to DENX Embedded Linux Environment
              Press 'I' to enter interactive startup.
Building the cache [  OK  ]
 storage network audio done[  OK  ]
.......
.......
.......
Starting system logger: [  OK  ]
Starting kernel logger: [  OK  ]
Starting portmap: [  OK  ]
Mounting NFS filesystems:  [  OK  ]
Mounting other filesystems:  [  OK  ]
Starting xinetd: [  OK  ]

DENX ELDK version 4.1 build 2007-01-19
Linux 2.6.11-centaure_1.0 on a ppc

DB64460 login:
```

8.2 Boot Linux-RAMDISK

```
CP3210 > printenv b_linux_tftp
b_linux_tftp=tftp 800000 uImagePEC7;tftp a00000 uRamdisk7xx;setenv bootargs console=ttyS0,115200 root=/dev/-
ram0 rw ip=$(ipaddr):$(serverip)$(bootargs_end);bootm 800000 a00000
CP3210 > run b_linux_tftp
Ethernet status port 0: Link up, Half Duplex, Speed 100 Mbps
Using mv_enet0 device
TFTP from server 192.93.161.147; our IP address is 192.93.161.150
Filename 'uImagePEC7'.
Load address: 0x800000
Loading: ##################################################################
         ##########################
done
Bytes transferred = 1792346 (1b595a hex)
Ethernet status port 0: Link up, Half Duplex, Speed 100 Mbps
Using mv_enet0 device
TFTP from server 192.93.161.147; our IP address is 192.93.161.150
Filename 'uRamdisk7xx'.
Load address: 0xa00000
Loading: ##################################################################
         ########################################
done
Bytes transferred = 1531127 (175cf7 hex)
## Booting image at 00800000 ...
   Image Name:   Linux-2.6.11-centaure_1.0
   Created:      2008-09-05  16:33:07 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   .......
## Loading RAMDisk Image at 00a00000 ...
   Image Name:   Simple Embedded Linux Framework
   Created:      2007-01-20  14:43:55 UTC
   Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
   .......
UBoot : Starting up the linux kernel
UBoot : cmd line (0x007fff00) : console=ttyS0,115200 root=/dev/ram0 rw
ip=192.93.161.150:192.93.161.147:::DB64460:egiga0:none
Total memory = 512MB; using 1024kB for hash table (at c0400000)
Linux version 2.6.11-centaure_1.0 (bsa@pcbist2.tctfr.thales) (gcc version 4.0.0 (DENX ELDK 4.1 4.0.0)) #4 Fri Sep 5
18:32:54 CEST 2008
.......
.......
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
      device=egiga0, addr=192.93.161.150, mask=255.255.255.0, gw=255.255.255.255,
     host=DB64460, domain=, nis-domain=(none),
     bootserver=192.93.161.147, rootserver=192.93.161.147, rootpath=
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem).
Mounted devfs on /dev
Freeing unused kernel memory: 340k init
root:~> ### Application running ...
egiga0: link up<5>, half duplex<5>, speed 100 Mbps<5>

root:~>
```

Chapter 9 - U-Boot Command Line Interface

The U-Boot implementation delivered with CP3210 contains the complete boot code infrastructure. Our Quality Insurance process has thoroughly qualified proper operation of the U-Boot commands required to operate CP3210. Such commands are indicated as **Qualified** in the following sections.

Conversely, U-Boot other commands are **Delivered** as part of the U-Boot core. They should not depend on the CP3210 hardware and have not formally be qualified by Kontron.

In case of trouble, please report issues first to the U-Boot community http://bugs.denx.de/databases/u-boot/new and then to support-kom-sa@kontron.com.

There is no formal commitment from Kontron to issue formal fixes for **Delivered** features.

9.1 Commands Summary

9.1.1 Proprietary Qualified Commands

List of U-BOOT proprietary qualified commands available only for U-Boot on CP3210 boards.

Command Name	Description	Command Type	See section
diag	perform board diagnostics	Proprietary	9.2.1 page 50
dhcp	invoke DHCP client to obtain IP/boot parameters	Proprietary	9.2.2 page 50
factory	print board parameters	Proprietary	9.2.3 page 55
hreset	perform Hardware Reset of the board	Proprietary	9.2.4 page 55
swfreq	print/set the system board frequency	Proprietary	9.2.5 page 56
update	update U-Boot firmware	Proprietary	9.2.6 page 56
vxworks	print/set/default vxworks bootrom parameter	Proprietary	9.2.7 page 57

9.1.2 Standard Qualified Commands

List of U-BOOT standard qualified commands available on CP3210 boards.

Command Name	Description	Command Type	See section
?	alias for 'help'	Miscellaneous	9.11.8 page 79
bdinfo	print Board Info structure	Information	9.3.1 page 59
bootd	boot default, i.e., run 'bootcmd'	Environment Var.	9.8.6 page 73
bootelf	boot from an ELF image in memory	Execution Control	9.6.2 page 67
bootm	boot application image from memory	Execution Control	9.6.3 page 67
bootp	boot image via network using bootp/TFTP protocol	Network	9.7.1 page 70
bootvx	boot VxWorks from an ELF image	Execution Control	9.6.4 page 68
cmp	memory compare	Memory	9.4.3 page 62
coninfo	print console devices and information	Information	9.3.2 page 59
cp	memory copy	Memory Flash Memory	9.4.4 page 63 9.5.1 page 65
crc32	checksum calculation	Memory	9.4.2 page 62
date	get/set/reset date & time	Miscellaneous	9.11.1 page 78
dcache	enable or disable data cache	Special	9.10.1 page 75
dtt	Digital Thermometer and Thermostat	Miscellaneous	9.11.2 page 78

Command Name	Description	Command Type	See section
echo	echo args to console	Miscellaneous	9.11.3 page 78
erase	erase FLASH memory	Flash Memory	9.5.2 page 65
flinfo	print FLASH memory information	Information Flash Memory	9.3.3 page 60 9.5.3 page 65
go	start application at addr 'addr'	Execution Control	9.6.5 page 69
help	print online help	Information	9.3.6 page 60
icache	enable or disable instruction cache	Special	9.10.3 page 75
icrc32	checksum calculation	Special	9.10.4 page 75
iloop	infinite loop on address range	Special	9.10.5 page 75
imd	I2C memory modify (auto-incrementing)	Special	9.10.6 page 75
iminfo	print header information for application image	Information	9.3.4 page 60
imm	I2C memory modify (auto-incrementing)	Special	9.10.7 page 76
imw	I2C memory write (fill)	Special	9.10.8 page 76
inm	I2C memory modify (constant address)	Special	9.10.9 page 76
iprobe	probe to discover valid I2C chip addresses	Special	9.10.10 page 76
loop	infinite loop on address range	Memory	9.4.9 page 64
md	memory display	Memory	9.4.5 page 63
mm	memory modify (auto-incrementing)	Memory	9.4.6 page 63
mw	memory write (fill)	Memory	9.4.7 page 63
nfs	boot image via network using NFS protocol	Network	9.7.5 page 71
nm	memory modify (constant address)	Memory	9.4.8 page 64
pci	list and access PCI configuration space	Special	9.10.11 page 77
ping	send ICMP ECHO_REQUEST to network host	Network	9.7.6 page 71
printenv	print environment variables	Environment Var.	9.8.2 page 72
protect	enable or disable FLASH write protection	Flash Memory	9.5.4 page 66
rarpboot	boot image via network using RARP/TFTP protocol	Network	9.7.7 page 71
regppc	print register information of the PowerPC	Information	9.3.7 page 61
reset	perform reset of the CPU	Miscellaneaous	9.11.5 page 78
run	run commands in an environment variable	Environment Var.	9.8.5 page 73
saveenv	save environment variables to persistent storeage	Environment Var.	9.8.3 page 72
setenv	set environment variables	Environment Var.	9.8.4 page 73
sleep	delay execution for some time	Miscellaneous	9.11.6 page 79
tftpboot	boot image via network using TFTP protocol	Network	9.7.8 page 71
version	print monitor version	Miscellaneous	9.11.7 page 79

9.1.3 Standard Delivered Commands

List of U-BOOT standard delivered commands available on CP3210 boards.

Command Name	Description	Command Type	See section
askenv	get environment variables from stdin	Environment Var.	9.8.1 page 72
autoscr	run script from memory	Execution Control	9.6.1 page 67
base	print or set address offset	Memory	9.4.1 page 62
eeprom	EEPROM sub-system	Special	9.10.2 page 75
fsinfo	print information about filesystems	Filesystem Sup.	9.9.1 page 74
fsload	load binary file from a filesystem image	Filesystem Sup.	9.9.2 page 74
imls	list all images found in flash	Information	9.3.5 page 60
itest	return true/false on integer compare	Miscellaneous	9.11.4 page 78
loadb	load binary file over serial line (kermit mode)	Network	9.7.2 page 70
loads	load s-record file over serial line	Network	9.7.3 page 71
loady	load binary file over serial line (ymodem mode)	Network	9.7.4 page 71
ls	list files in a directory (default /)	Filesystem Sup.	9.9.3 page 74

9.2 Proprietary Commands

9.2.1 diag - perform board diagnostics

This command is fully described in Chapter 6 'Power-on self-test (POST)" page 17.

9.2.2 dhcp - invoke DHCP client to obtain IP/boot parameters

The dhcp command on CP3210 implements the dhcp client protocol.

In addition, the dhcp command can be setup using extended parameters in U-Boot environment variables to support **vendor-class-identifier** (dhcp option: 60 - vendor-identifier Mode) and **dhcp-client-identifier** (dhcp option: 61 - client-identifier Mode).

By default, the dhcp request uses a **vendor-class-identifier** set to **CP3210** for the CP3210 board.

Hereafter, default dhcp request outputs from the U-Boot firmware on a CP3210 board:

```
No.     Time        Source                Destination           Protocol
Info
      1 0.000000    0.0.0.0               255.255.255.255       DHCP
DHCP Discover - Transaction ID 0xde403915

Frame 1 (297 bytes on wire, 297 bytes captured)
    Arrival Time: Mar  3, 2009 10:43:44.869789000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 297 bytes
    Capture Length: 297 bytes
    Protocols in frame: eth:ip:udp:bootp Ethernet II, Src: 00:00:de:40:36:14, Dst: ff:ff:ff:ff:ff:ff
    Destination: ff:ff:ff:ff:ff:ff (Broadcast)
    Source: 00:00:de:40:36:14 (Unigraph_40:36:14)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 0.0.0.0 (0.0.0.0), Dst Addr:
255.255.255.255 (255.255.255.255)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 283
    Identification: 0x0000 (0)
    Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 255
    Protocol: UDP (0x11)
    Header checksum: 0x7ad2 (correct)
    Source: 0.0.0.0 (0.0.0.0)
    Destination: 255.255.255.255 (255.255.255.255) User Datagram Protocol, Src Port: bootpc (68), Dst
Port: bootps (67)
    Source port: bootpc (68)
    Destination port: bootps (67)
    Length: 263
    Checksum: 0x0000 (none)
Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xde403915
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: 00:00:de:40:36:14 (Unigraph_40:36:14)
    Server host name not given
    Boot file name: vxWorks
    Magic cookie: (OK)
    Option 53: DHCP Message Type = DHCP Discover
    Option 57: Maximum DHCP Message Size = 576
   Option 60: Vendor class identifier = "CP3210"
```

In the request above, the option 60 is used, with the default value CP3210 for the **vendor-class-identifier**.

Three modes are described below:

- The default Mode
- The vendor-identifier Mode
- The client-identifier Mode

To support the dhcp request, a machine needs to have a dhcp server initialized.

In following examples, the dhcp server is running on a Linux OS (Debian Sarge 3.0). The configuration file used to configure the dhcp daemon on this OS is: **isc-dhcpd-V3.0.1**. Those examples should be tuned or adapted to the dhcp daemon and OS releases and versions.

Default Mode

- Contents of **/etc/dhcp3/dhcpd.conf** configuration file:

```
ddns-update-style none;

subnet 192.168.0.0 netmask 255.255.255.0 {
   authorative;

   pool {
      range 192.168.0.40 192.168.0.56;
   }
}
```

Do not forget to restart the **dhcpd** daemon once the configuration file **/etc/dhcp3/dhcpd.conf** has been updated.

In the Default Mode, all dhcp requests on the subnetwork 192.168.0.0 will be accepted by the linux server.

- Configuration of the CP3210:
  - When the CP3210 board is booted, configure the default bootfile that the board need to download after the dhcp request is done.

```
CP3210 > setenv bootfile 'vxWorks'
CP3210 > saveenv
```

The file **'vxWorks'** is stored in the **/tftpboot** directory of the dhcp server.

- Now the CP3210 board is configured, run the **dhcp** command:

```
CP3210 > dhcp
Ethernet status port 0: Link up, Full Duplex, Speed 100 Mbps
DHCP broadcast 1
DHCP client bound to address 192.168.0.52
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.52
Filename 'vxWorks'
Load address: 0x800000
Loading:
#################################################################
...
#################################################
done
Bytes transferred = 1595234 (185762 hex)
CP3210 >
```

vendor-identifier Mode

> Contents of /etc/dhcp3/dhcpd.conf configuration file:

```
ddns-update-style none;
class "dsu" {
    match if option vendor-class-identifier = "BOARD_1";
    filename "vxWorks";
}

subnet 192.168.0.0 netmask 255.255.255.0 {
    authorative;
    pool {
        allow members of "dsu";
        range 192.168.0.40 192.168.0.56;
    }
}
```


Do not forget to restart the dhcpd daemon once the configuration file /etc/dhcp3/dhcpd.conf has been updated.

In this example:

- the client will be affected with an IP address between 192.168.0.40 and 192.168.0.56
- the bootfile file that will be downloaded is: vxWorks
- the dhcp vendor class identifier is: BOARD_1

> Configuration of the CP3210:

When the dhcpd daemon is running, configure the board by setting the dhcp_vendor_class_id environment variable:

```
CP3210 > setenv dhcp_vendor_class_id 'BOARD_1'
CP3210 > saveenv
```


The default value of the environment variable dhcp_vendor_class_id is "CP3210".

> Now the CP3210 board is configured, run the dhcp command:

```
CP3210 > dhcp
Ethernet status port 0: Link up, Full Duplex, Speed 100 Mbps
DHCP broadcast 1
DHCP client bound to address 192.168.0.56
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.56
Filename 'vxWorks'.
Load address: 0x800000
Loading:
#################################################################
...
#################################################
done
Bytes transferred = 1595234 (185762 hex)
CP3210 >
```

client-identifier Mode

> Contents of **/etc/dhcp3/dhcpd.conf** configuration file:

```
class "dsu_client" {
      match if substring (option dhcp-client-identifier,0,5) = "LAN_1";
      filename "uImagePEC7";
   }
subnet 192.168.0.0 netmask 255.255.255.0 {
      authorative;
      pool {
         allow members of "dsu_client";
         range 192.168.0.40 192.168.0.56;
      }
}
```

Do not forget to restart the **dhcpd** daemon once the configuration file **/etc/dhcp3/dhcpd.conf** has been updated.

In this example:

- the client will be affected with an IP address between 192.168.0.40 and 192.168.0.56
- the bootfile file that will be downloaded is: uImagePEC7
- the dhcp client identifier is: LAN_1

> Configuration of the CP3210:

When the **dhcpd** daemon is running, configure the board by setting the **dhcp_client_id** environment variable:

```
CP3210 > setenv dhcp_client_id 'LAN_1'
CP3210 > saveenv
```

> Now the CP3210 board is configured, run the **dhcp** command:

```
CP3210 > dhcp
Ethernet status port 0: Link up, Full Duplex, Speed 100 Mbps
DHCP broadcast 1
DHCP client bound to address 192.168.0.54
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.54
Filename 'uImagePEC7'.
Load address: 0x800000
Loading:
#################################################################
...
#################################################
done
Bytes transferred = 1792346 (1b595a hex)
CP3210 >
```

9.2.3 factory - print boards parameters

```
 CP3210 > help factory
 factory print
  - print the board parameters
CP3210
```

Example of factory command outputs:

```
CP3210 > factory print
Ethernet Configuration :
        mgi0            : 00:00:de:40:36:10
        mgi1            : 00:00:de:40:36:12

Board Configuration :
        Board Name      : CP3210
        Serial Number   : 1108311010001
        EC Level        : 1000
        HCD Number      : 11130115
        PCB Level       : A
        Class Board     : RC
        Bridge Vers.    : MV64460-B1-BDT-I133
        CPU Version     : IBM25PPC750FX-GB1032T
        SDRAM DDR Size  : 512MB
        Sys. Flash Size : 128MB
        User Flash size : 256MB
        CPU Bus Freq.   : 132500KHz
        CPU Freq.       : 728750KHz
        CPU Range       : Upper than 600MHz
        NVRAM           : Equiped
        PMC IO          : PMC VIO 3V3
        Finition        : Immersion Tin ( ITIN )
CP3210 >
```

9.2.4 hreset - perform a hardware reset of the board

If the board is system controller in CPCI rack, the reset is also propagated to all peripheral boards in the rack.

```
CP3210 > help hreset
hreset
    - perform a hardware reset of the board
CP3210 >
```

9.2.5 swfreq - print/set the system board frequency

```
CP3210 > help swfreq
swfreq print
     - print the system board frequency
swfreq set frequency
     - set the system board frequency
       (available frequencies: 100 MHz and 133 MHz)
CP3210 >
```

Command available only in Rescue Mode.

To set the CPU bus frequency at 100 MHz (accordingly the CPU frequency to 700 MHz), enter:

```
CP3210-RESCUE > swfreq set 100
CP3210-RESCUE >
```

To set the CPU bus frequency at 133 MHz (accordingly the CPU frequency to 733 MHz), enter:

```
CP3210-RESCUE > swfreq set 133
CP3210-RESCUE >
```

9.2.6 update - update U-Boot firmware

➤ User Mode

```
CP3210 > help update
update user addr
     - update the U-Boot firmware in user mode from address addr in memory
CP3210 >
```

➤ Rescue Mode

```
CP3210-RESCUE > help update
update user addr
     - update the uboot firmware in user mode from address addr in memory
update rescue addr
      - update the uboot firmware in rescue mode from address addr in memory
CP3210-RESCUE >
```

- In User Mode, the firmware can only update the user firmware.
- In Rescue Mode, the firmware can update the rescue firmware, but also the user firmware.

9.2.7 vxworks - print/set/default VxWorks bootrom parameter

```
CP3210 > help vxworks
vxworks print
      - print vxworks bootrom parameters
vxworks set
      - set vxworks bootrom parameters
vxworks default
      - default vxworks bootrom parameters
CP3210 >
```

Example of vxworks command outputs.

➤ print vxworks bootrom parameters:

```
CP3210 > vxworks print

boot device          : mgi
unit number          : 0
processor number     : 0
host name            : pcsysinfo2
file name            : /home/user/vxWorksNEW
inet on ethernet (e) : 192.168.0.2:FFFFFF00
host inet (h)        : 192.168.0.1
user (u)             : user-name
ftp password (pw)    : user-passdd
flags (f)            : 0x8

CP3210 >
```

➤ reset vxworks bootrom parameters to their default values:

```
CP3210 > vxworks default
vxworks default
Saving bootargs in Global Board Feature parameters in flash ...
.Erased 1 sectors
Copy to Flash...  0%[====================] 100% done Saving bootargs in uboot
parameters in flash ...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash...  0%[====================] 100% done Protected 2 sectors
uboot parameters saved !

CP3210 > vxworks print

boot device          : mgi
unit number          : 0
processor number     : 0
host name            : sunblade
file name            : /home/test/vxWorksDec
inet on ethernet (e) : 192.168.0.2:FFFFFF00
host inet (h)        : 192.168.0.1
user (u)             : test
ftp password (pw)    : testfab
flags (f)            : 0x8

CP3210 >
```

➤ set vxworks bootrom parameters to specific values:

```
CP3210 > vxworks set

'.' = clear field; '-' = go to previous field; ^D = quit

boot device             : mgi0
processor number        : 0
host name               : sunblade
file name               : /home/test/vxWorksDec
inet on ethernet (e)    : 192.168.0.2:FFFFFF00
inet on backplane (b):
host inet (h)           : 192.168.0.1
gateway inet (g)        :
user (u)                : test
ftp password (pw) (blank = use rsh): testfab
flags (f)               : 0x8
target name (tn)        : cp3210
startup script (s)      :
other (o)               :

Saving bootargs in Global Board Feature parameters in flash ...
erase 0xfc060000 0xfc07ffff
.Erased 1 sectors
cp 0x0fbe92b0 0xfc060000 0x00008000
Copy to Flash... 0%[====================] 100%
done
Saving bootargs in uboot parameters in flash ...
Un-Protected 2 sectors
Erasing Flash.....Erased 2 sectors
Writing to Flash... 0%[====================] 100%
done
Protected 2 sectors
uboot parameters saved !
CP3210 >
```

9.3 Information Commands

9.3.1 bdinfo - print Board Info structure

The **bdinfo** command (short: **bdi**) prints the information that U-Boot passes about the board such as memory addresses and sizes, clock frequencies, MAC address, etc. This information is mainly needed to be passed to the O.S. kernel.

```
CP3210 > help bdinfo
bdinfo - No help available.
CP3210 >
```

The information is passed to the O.S. kernel in a **mv_bd_info** structure defined below:

```
typedef struct mv_bd_info {
  /* U-Boot standard */
  unsigned int   bi_memstart;    /* start of DRAM memory */
  unsigned int   bi_memsize;     /* size  of DRAM memory in bytes */
  unsigned int   bi_flashstart;  /* start of FLASH memory */
  unsigned int   bi_flashsize;   /* size  of FLASH memory */
  unsigned int   bi_flashoffset; /* reserved area for startup monitor */
  unsigned int   bi_sramstart;   /* start of SRAM memory */
  unsigned int   bi_sramsize;    /* size  of SRAM memory */
  unsigned int   bi_bootflags;   /* boot / reboot flag (for LynxOS) */
  unsigned int   bi_ip_addr;     /* IP Address */
  unsigned char  bi_enetaddr[6]; /* Ethernet adress */
  unsigned short bi_ethspeed;    /* Ethernet speed in Mbps */
  unsigned int   bi_intfreq;     /* Internal Freq, in MHz */
  unsigned int   bi_busfreq;     /* Bus Freq, in MHz */
  unsigned int   bi_baudrate;    /* Console Baudrate */
  unsigned char  bi_enet1addr[6];/* second onboard ethernet port */
  unsigned char    bi_enet2addr[6];/* third onboard ethernet port */
  /* Kontron Specific */
  unsigned int   bi_tct_magic;
  unsigned int   bi_uboot_id;
  unsigned int   bi_tclk; /* internal MV64460 clock */
  unsigned int   bi_mtdparts_count;
  bi_mtdparts_t  bi_mtdparts[BI_MAX_MTDPARTS_COUNT];
  unsigned int    bi_pci0_busfreq; /* PCI Bus speed, in Hz */
  char            bi_bios_version[16];
  char            bi_user_version[16];
} mv_bd_t;
```

The address of this structure in the board SDRAM is the first boot parameter given to the O.S.

9.3.2 coninfo - print console devices and information

The **coninfo** command (short: **conin**) displays information about the available console I/O devices.

```
CP3210 > help conin
coninfo
CP3210 >
```

9.3.3 flinfo - print FLASH memory information

The command **flinfo** (short: **fli**) can be used to get information about the available flash memory (see Flash Memory Commands below).

```
CP3210 > help flinfo
flinfo
    - print information for all FLASH memory banks
flinfo N
    - print information for FLASH memory bank # N

CP3210 >
```

9.3.4 iminfo - print header information for application image

iminfo (short: **imi**) is used to print the header information for images like Linux kernels or ramdisks. It prints (among other information) the image name, type and size and verifies that the CRC32 checksums stored within the image are OK.

```
CP3210 > help iminfo
iminfo addr [addr ...]
    - print header information for application image starting at
      address 'addr' in memory; this includes verification of the
      image contents (magic number, header and payload checksums)

CP3210 >
```

9.3.5 imls - list all images found in flash

```
CP3210 > help imls
imls
     - Prints information about all images found at sector
       boundaries in flash.

CP3210 >
```

9.3.6 help - print online help

The **help** command (short: **h or ?**) prints online help. Without any arguments, it prints a list of all U-Boot commands that are available in your configuration of U-Boot. You can get detailed information for a specific command by typing its name as argument to the **help** command:

```
CP3210 > help
help [command ...]
    - show help information (for 'command')
'help' prints online help for the monitor commands.

Without arguments, it prints a short usage message for all commands.

To get detailed help information for specific commands you can type
'help' with one or more command names as arguments.

CP3210 >
```

9.3.7 regppc - print register information of the PowerPC

Example of **regppc** command outputs.

```
CP3210 > regppc

=== PowerPC Registers ===

MSR  : 0x0000a030
L2CR : 0x00000000
HID0 : 0x0000c000
HID1 : 0x58000000
HID2 : 0x00000000

CP3210 >
```

9.4 Memory Commands

9.4.1 base - print or set address offset

You can use the **base** command (short: **ba**) to print or set a "base address" that is used as address offset for all memory commands; the default value of the base address is 0, so all addresses you enter are used unmodified. However, when you repeatedly have to access a certain memory region (like the internal memory of some embedded PowerPC processors) it can be very convenient to set the base address to the start of this area and then use only the offsets:

```
CP3210 > help base
base
    - print address offset for memory commands
base off
    - set address offset for memory commands to 'off'

CP3210 >
```

9.4.2 crc32 - checksum calculation

The **crc32** command (short: **crc**) can be used to caculate a CRC32 checksum over a range of memory:

```
CP3210 > crc 100004 3FC
CRC32 for 00100004 ... 001003ff ==> 3c82ba9d
CP3210 >
```

When used with 3 arguments, the command stores the calculated checksum at the given address:

```
CP3210 > crc 100004 3FC 100000
CRC32 for 00100004 ... 001003ff ==> 3c82ba9d
=>
=> md 100000 4
00100000: 3c82ba9d 6aa09851 47f3731e 0000009e    <...j..QG.s.....
CP3210 >
```

As you can see, the CRC32 checksum was not only printed, but also stored at address 0x100000.

9.4.3 cmp - memory compare

With the **cmp** command you can test of the contents of two memory areas is identical or not. The command will either test the whole area as specified by the 3rd (length) argument, or stop at the first difference.

```
CP3210 > help cmp
cmp [.b, .w, .l] addr1 addr2 count
    - compare memory

CP3210 >
```

Please note that the *count* argument specifies the number of data items to process, i. e. the number of long words or words or bytes to compare.

9.4.4 cp - memory copy

The **cp** is used to copy memory areas.

```
CP3210 > help cp
cp [.b, .w, .l] source target count
    - copy memory

CP3210 >
```

9.4.5 md - memory display

The **md** can be used to display memory contents both as hexadecimal and ASCII data.

```
CP3210 > help md
md [.b, .w, .l] address [# of objects]
    - memory display

CP3210 >
```

The last displayed memory address and the value of the count argument are remembered, so when you enter **md** again without arguments it will automatically continue at the next address, and use the same *count* again.

9.4.6 mm - memory modify (auto-incrementing)

```
CP3210 > help mm
mm [.b, .w, .l] address
    - memory modify, auto increment address

CP3210 >
```

The **mm** is a method to interactively modify memory contents. It will display the address and current contents and then prompt for user input. If you enter a legal hexadecimal number, this new value will be written to the address. Then the next address will be prompted. If you don't enter any value and just press ENTER, then the contents of this address will remain unchanged. The command stops as soon as you enter any data that is not a hex number (like .).

9.4.7 mw - memory write (fill)

```
CP3210 > help mw
mw [.b, .w, .l] address value [count]
    - write memory

CP3210 >
```

The **mw** is a way to initialize (fill) memory with some value. When called without a *count* argument, the value will be written only to the specified address. When used with a *count*, then a whole memory areas will be initialized with this value.

9.4.8 nm - memory modify (constant address)

The **nm** command (non-incrementing memory modify) can be used to interactively write different data several times to the same address. This can be useful for instance to access and modify device registers:

```
CP3210 > help nm
nm [.b, .w, .l] address
    - memory modify, read and keep address

CP3210 >
```

9.4.9 loop - infinite loop on address range

```
CP3210 > help loop
loop [.b, .w, .l] address number_of_objects
    - loop on a set of addresses

CP3210 >
```

The **loop** command reads in a tight loop from a range of memory. This is intended as a special form of a memory test, since this command tries to read the memory as fast as possible.

This command will never terminate. There is no way to stop it but to reset the board!

9.5 Flash Memory Commands

9.5.1 cp - memory command

```
CP3210 > help cp
cp [.b, .w, .l] source target count
    - copy memory

CP3210 >
```

The **cp** command "knows" about flash memory areas and will automatically invoke the necessary flash programming algorithm when the target area is in flash memory.

Writing to flash memory may fail when the target area has not been erased (see **erase** below), or if it is write-protected (see **protect** below).

Remember that the *count* argument specifies the number of items to copy. If you have a "length" instead (= byte count) you should use cp.b or you will have to calculate the correct number of items.

9.5.2 erase - erase FLASH memory

```
CP3210 > help era
erase start end
    - erase FLASH from addr 'start' to addr 'end'
erase start +len
    - erase FLASH from addr 'start' to the end of sect w/addr 'start'+'len'-1
erase N:SF[-SL]
    - erase sectors SF-SL in FLASH bank # N
erase bank N
    - erase FLASH bank # N
erase all
    - erase all FLASH banks

CP3210 >
```

The **erase** command (short: **era**) is used to erase the contents of one or more sectors of the flash memory. It is one of the more complex commands; the help output shows this.

9.5.3 flinfo - print FLASH memory information

The command **flinfo** (short: **fli**) can be used to get information about the available flash memory (see Flash Memory Commands below).

```
CP3210 > help flinfo
flinfo
    - print information for all FLASH memory banks
flinfo N
    - print information for FLASH memory bank # N

CP3210 >
```

9.5.4 protect - enable or disable FLASH write protect

```
CP3210 > help protect
protect on  start end
    - protect FLASH from addr 'start' to addr 'end'
protect on start +len
    - protect FLASH from addr 'start' to end of sect w/addr 'start'+'len'-1
protect on  N:SF[-SL]
    - protect sectors SF-SL in FLASH bank # N
protect on  bank N
    - protect FLASH bank # N
protect on  all
    - protect all FLASH banks
protect off start end
    - make FLASH from addr 'start' to addr 'end' writable
protect off start +len
    - make FLASH from addr 'start' to end of sect w/addr 'start'+'len'-1
wrtable
protect off N:SF[-SL]
    - make sectors SF-SL writable in FLASH bank # N
protect off bank N
    - make FLASH bank # N writable
protect off all
    - make all FLASH banks writable

CP3210 >
```

The **protect** command is another complex one. It is used to set certain parts of the flash memory to read-only mode or to make them writable again. Flash memory that is "protected" (= read-only) cannot be written (with the **cp** command) or erased (with the **erase** command). Protected areas are marked as (RO) (for "read-only") in the output of the flinfo command.

The actual level of protection depends on the flash chips used on your hardware, and on the implementation of the flash device driver for this board. In most cases U-Boot provides just a simple software-protection, i. e. it prevents you from erasing or overwriting important stuff by accident (like the U-Boot code itself or U-Boot's environment variables), but it cannot prevent you from circumventing these restrictions - a nasty user who is loading and running his own flash driver code cannot and will not be stopped by this mechanism. Also, in most cases this protection is only effective while running U-Boot, i. e. any operating system will not know about "protected" flash areas and will happily erase these if requested to do so.

9.6 Execution Control Commands

9.6.1 autoscr - run script from memory

```
CP3210 > help autoscr
autoscr [addr] - run script starting at addr - A valid autoscr header must be
present

CP3210 >
```

With the **autoscr** command you can run "shell" scripts under U-Boot: You create a U-Boot script image by simply writing the commands you want to run into a text file; then you will have to use the mkimage tool to convert this text file into a U-Boot image (using the image type script).

This image can be loaded like any other image file, and with **autoscr** you can run the commands in such an image.

9.6.2 bootelf - boot from an ELF image in memory

```
CP3210 > help bootelf
bootelf  [address] - load address of the ELF image.
CP3210 >
```

9.6.3 bootm - boot application image from memory

```
CP3210 > help bootm
bootm [addr [arg ...]]
    - boot application image stored in memory
   passing arguments 'arg ...'; when booting a Linux kernel,
   'arg' can be the address of an initrd image
   When booting a Linux kernel which requires a flat device-tree
   a third argument is required which is the address of the
   device-tree blob. To boot that kernel without an initrd image,
   use a '-' for the second argument. If you do not pass a third
   a bd_info struct will be passed instead

CP3210 >
```

The **bootm** command is used to start operating system images. From the image header it gets information about the type of the operating system, the file compression method used (if any), the load and entry point addresses, etc. The command will then load the image to the required memory address, uncompressing it on the fly if necessary. Depending on the OS it will pass the required boot arguments and start the OS at it's entry point.

The first argument to **bootm** is the memory address (in RAM, ROM or flash memory) where the image is stored, followed by optional arguments that depend on the OS.

Linux requires the flattened device tree blob to be passed at boot time, and bootm expects its third argument to be the address of the blob in memory. Second argument to **bootm** depens on whether an initrd initial ramdisk image is to be used. If the kernel should be booted without the initial ramdisk, the second argument should be given as "-", otherwise it is interpreted as the start address of initrd (in RAM, ROM or flash memory).

9.6.4 bootvx - boot VxWorks from an ELF image

```
CP3210 > help bootvx
 bootvx  [address] - load address of VxWorks ELF image.

CP3210 >
```

```
CP3210 > tftp 0x2000000 vxWorks
Ethernet status port 0: Link up, Full Duplex, Speed 1 Gbps Using mv_enet0
device TFTP from server 192.168.0.1; our IP address is 192.168.0.2 Filename
'vxWorks'.
Load address: 0x2000000
Loading: ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ########################
done
Bytes transferred = 1782036 (1b3114 hex)
CP3210 > bootvx
## Ethernet MAC address not copied to NV RAM setenv bootargs
'mgi(0,0)pcsysinfo2:/home/test/vxWorks e=192.168.0.2:FFFFFF00 h=192.168.0.1
u=user-name pw=pw-user-name f=0x8'
Loading .text @ 0x00200000 (1414112 bytes)
Loading .init$00 @ 0x003593e0 (16 bytes)
Loading .init$99 @ 0x003593f0 (16 bytes)
Loading .fini$00 @ 0x00359400 (16 bytes)
Loading .fini$99 @ 0x00359410 (16 bytes)
Loading .data @ 0x00359420 (72464 bytes)
Clearing .bss @ 0x0036af30 (89120 bytes)
## Using bootline (@ 0x4200): mgi(0,0)pcsysinfo2:/home/test/vxWorks
e=192.168.0.2:FFFFFF00 h=192.168.0.1 u=user-name pw=pw-user-name f=0x8 ##
Starting vxWorks at 0x00200000 ...
Detect System clock (SysClk)  133MHz
Detect Discovery clock (Tclk) 133MHz
Processor clock 733MHz

CP3210 >
```

9.6.5 go - start application at address 'addr'

```
CP3210 > help go
go addr [arg ...]
    - start application at address 'addr'
      passing 'arg' as arguments

CP3210 >
```

U-Boot has support for so-called standalone applications. These are programs that do not require the complex environment of an operating system to run. Instead they can be loaded and executed by U-Boot directly, utilizing U-Boot's service functions like console I/O or malloc() and free().

This can be used to dynamically load and run special extensions to U-Boot like special hardware test routines or bootstrap code to load an OS image from some filesystem.

The **go** command is used to start such standalone applications. The optional arguments are passed to the application without modification.

9.7 Network Commands

9.7.1 bootp - boot image via network using BOOTP/TFTP protocol

```
CP3210 > help bootp
bootp [loadAddress] [bootfilename]

CP3210 >
```

The user needs to configure a bootp server and a tftp server.
For the bootp server, use a bootpd daemon running on Linux (Debian Sarge 3.0 in following example) configured as follow:

In the file /etc/inetd.conf:

```
bootps      dgram udp  wait  root  /usr/sbin/bootpd
bootpd -i -t 120
```

In the file /etc/bootptap:

```
.default:\
      :td=/tftpboot:hd=/tftpboot:bf=uImage:\
      :sm=255.255.255.0:\
      :hn:to=-18000:

minotaure:ha=0000de403614:tc=.default:ip=192.168.0.10
```

When the bootp and tftp servers are configured, the user can tun the bootp command under U-Boot firmware:

```
CP3210 > bootp
Ethernet status port 0: Link up, Full Duplex, Speed 100 Mbps
BOOTP broadcast 1
Using mv_enet0 device
TFTP from server 192.168.0.1; our IP address is 192.168.0.10
Filename '/tftpboot/uImage'.
Load address: 0x800000
Loading:
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#############################################################
done
Bytes transferred = 1959130 (1de4da hex)
CP3210 >
```

9.7.2 loadb - load binary file over serial line (kermit mode)

```
CP3210 > help loadb
loadb [ off ] [ baud ]
    - load binary file over serial line with offset 'off' and baudrate 'baud'

CP3210 >
```

9.7.3 loads - load S-Record file over serial line

```
 CP3210 > help loads
loads [ off ] [ baud ]
    - load S-Record file over serial line with offset 'off' and baudrate
'baud'

CP3210 >
```

9.7.4 loady - load binary file over serial line (ymodem mode)

```
CP3210 > help loady
 loady [ off ] [ baud ]
    - load binary file over serial line with offset 'off' and baudrate 'baud'

CP3210 >
```

9.7.5 nfs - boot image via network using NFS protocol

```
 CP3210 > help nfs
nfs [loadAddress] [host ip addr:bootfilename]
CP3210 >
```

9.7.6 ping - send ICMP ECHO_REQUEST to network host

```
CP3210 > help ping
ping pingAddress
CP3210 >
```

9.7.7 rarpboot - boot image via network using RARP/TFTP protocol

```
CP3210 > help rarp
rarpboot [loadAddress] [bootfilename]
CP3210 >
```

9.7.8 tftpboot - boot image via network using TFTP protocol

```
CP3210 > help tftpboot
tftpboot [loadAddress] [[hostIPaddr:]bootfilename]

CP3210 >
```

9.8 Environment Variables Commands

9.8.1 askenv - get environment variables from stdin

```
CP3210 > help askenv
askenv name [message] [size]
    - get environment variable 'name' from stdin (max 'size' chars)
askenv name
    - get environment variable 'name' from stdin
askenv name size
    - get environment variable 'name' from stdin (max 'size' chars)
askenv name [message] size
    - display 'message' string and get environment variable 'name'from stdin
(max 'size' chars)

CP3210 >
```

9.8.2 printenv - print environment variables

```
CP3210 > help printenv
printenv
    - print values of all environment variables
printenv name ...
    - print value of environment variable 'name'

CP3210 >
```

The printenv command prints one, several or all variables of the U-Boot environment. When arguments are given, these are interpreted as the names of environment variables which will be printed with their values.

Without arguments, printenv prints all a list with all variables in the environment and their values, plus some statistics about the current usage and the total size of the memory available for the environment.

9.8.3 saveenv - save environment variables to persistent storage

```
CP3210 > help saveenv
saveenv - No help available.

CP3210 >
```

All changes you make to the U-Boot environment are made in RAM only. They are lost as soon as you reboot the system. If you want to make your changes permanent you have to use the saveenv command to write a copy of the environment settings to persistent storage, from where they are automatically loaded during startup.

9.8.4 setenv - set environement variables

```
CP3210 > help setenv
setenv name value ...
    - set environment variable 'name' to 'value ...'
setenv name
    - delete environment variable 'name'

CP3210 >
```

To modify the U-Boot environment you have to use the **setenv** command. When called with exactly one argument, it will delete any variable of that name from U-Boot's environment, if such a variable exists. Any storage occupied for such a variable will be automatically reclaimed.

A common mistake is to write

```
setenv name=value
```

instead of

```
setenv name value
```

There will be no error message, which lets you believe everything went OK, but it didn't: instead of setting the variable name to the value value you tried to delete a variable with the name name=value - this is probably not what you intended! Always remember that name and value have to be separated by space and/or tab characters!

9.8.5 run - run commands in an environment variable

```
CP3210 > help run
run var [...]
    - run the commands in the environment variable(s) 'var'

CP3210 >
```

You can use U-Boot environment variables to store commands and even sequences of commands. To execute such a command, you use the **run** command.

You can call **run** with several variables as arguments, in which case these commands will be executed in sequence.

If you execute several variables with one call to **run**, any failing command will cause "run" to terminate, i. e. the remaining variables are not executed.

9.8.6 bootd - boot default, i.e., run 'bootcmd'

The **bootd** (short: **boot**) executes the default **boot** command, i. e. what happens when you don't interrupt the initial countdown. This is a synonym for the run **bootcmd** command.

9.9 Filesystem Support

9.9.1 fsinfo - print information about filesystems

```
CP3210 > help fsinfo
 fsinfo     - print information about filesystems

CP3210 >
```

9.9.2 fsload - load binary file from a filesystem image

```
CP3210 > help fsload
 fsload [ off ] [ filename ]
    - load binary file 'filename' from flash bank with offset 'off'
CP3210 >
```

9.9.3 ls - list files in a directory (default /)

```
CP3210 > help ls
 ls [ directory ]
    - list files in a directory.

CP3210 >
```

9.10 Special Commands

9.10.1 dcache - enable or disable data cache

```
CP3210 > help dcache
dcache [on, off]
    - enable or disable data (writethrough) cache

CP3210 >
```

9.10.2 EEPROM sub-system

```
CP3210 > help eeprom
eeprom write addr off cnt
- read/write 'cnt' bytes at EEPROM offset 'off'

CP3210 >
```

9.10.3 icache - enable or disable instruction cache

```
CP3210 > help icache
icache [on, off]
- enable or disable instruction cache

CP3210 >
```

9.10.4 icrc32 - checksum calculation

```
CP3210 > help icrc32
icrc32 chip address [.0, .1, .2] count
- compute CRC32 checksum

CP3210 >
```

9.10.5 iloop - infinite loop on range address

```
CP3210 > help iloop
iloop chip address[.0, .1, .2] [# of objects]
- loop, reading a set of addresses

CP3210 >
```

9.10.6 imd - I2C memory display

```
CP3210 > help imd
imd chip address[.0, .1, .2] [# of objects]
- i2c memory display

CP3210 >
```

9.10.7 imm - I2C memory modify (auto-incrementing)

```
CP3210 > help imm
imm chip address[.0, .1, .2]
- memory modify, auto increment address

CP3210 >
```

9.10.8 imw - I2C memory write (fill)

```
CP3210 > help imw
imw chip address[.0, .1, .2] value [count]
- memory write (fill)

CP3210 >
```

9.10.9 inm - I2C memory modify (constant address)

```
CP3210 > help inm
inm chip address[.0, .1, .2]
- memory modify, read and keep address

CP3210 >
```

9.10.10 iprobe - probe to discover valid I2C chip addresses

```
CP3210 > help iprobe
iprobe
    -discover valid I2C chip addresses

CP3210 >
```

The **iprobe** command returns valid I2C chip addresses in hexadecimal format. These addresses are used as the **chip address** argument of the commands: **icr32, iloop, imd, imm, imw, inm** described in previous sections (9.10.4 to 9.10.9)

Example of **iprobe** command output:

```
CP3210 > iprobe
Valid chip addresses: 48

CP3210 >
```

9.10.11 pci - list and access PCI configuration space

```
CP3210 > help pci
pci [bus] [long]
- short or long list of PCI devices on bus 'bus'
pci header b.d.f
- show header of PCI device 'bus.device.function'
pci display[.b, .w, .l] b.d.f [address] [# of objects]
- display PCI configuration space (CFG)
pci next[.b, .w, .l] b.d.f address
- modify, read and keep CFG address
pci modify[.b, .w, .l] b.d.f address
- modify, auto increment CFG address
pci write[.b, .w, .l] b.d.f address value
- write to CFG address
CP3210 >
```

9.11 Miscellaneous Commands

9.11.1 date - get/set/reset date & time

```
CP3210 > help date
 date [MMDDhhmm[[CC]YY][.ss]]
 date reset
  - without arguments: print date & time
  - with numeric argument: set the system date & time
  - with 'reset' argument: reset the RTC

CP3210 >
```

9.11.2 dtt - Digital Thermometer and Thermostat

```
CP3210 > help dtt
 Read temperature from digital thermometer and thermostat.

CP3210 >
```

Example of **dtt** command outputs.

```
CP3210 > dtt
DTT1: 40 C

CP3210 >
```

9.11.3 echo - echo args to console

```
CP3210 > help echo
echo [args..]
    - echo args to console; \c suppresses newline

CP3210 >
```

The **echo** command echoes the arguments to the console.

9.11.4 itest - return true/false on integer compare

```
CP3210 > help itest
itest [.b, .w, .l, .s] [*]value1 <op> [*]value2
CP3210 >
```

9.11.5 reset - perform reset of the CPU

The **reset** command reboots the system.

9.11.6 sleep - delay execution for some time

```
CP3210 > help sleep
sleep N
    - delay execution for N seconds (N is _decimal_ !!!)

CP3210 >
```

The **sleep** command pauses execution for the number of seconds given as the argument.

9.11.7 version - print monitor version

You can print the version and build date of the U-Boot image running on your system using the **version** command (short: **vers**)

9.11.8 ? - alias for 'help'

You can use ? as a short form for the help command.

MAILING ADDRESS
Kontron Modular Computers S.A.S.
150 rue Marcelin Berthelot - BP 244
ZI TOULON EST
83078 TOULON CEDEX - France

TELEPHONE AND E-MAIL
+33 (0) 4 98 16 34 00
sales@kontron.com
support-kom-sa@kontron.com

For further information about other Kontron products, please visit our Internet web site: www.kontron.com.