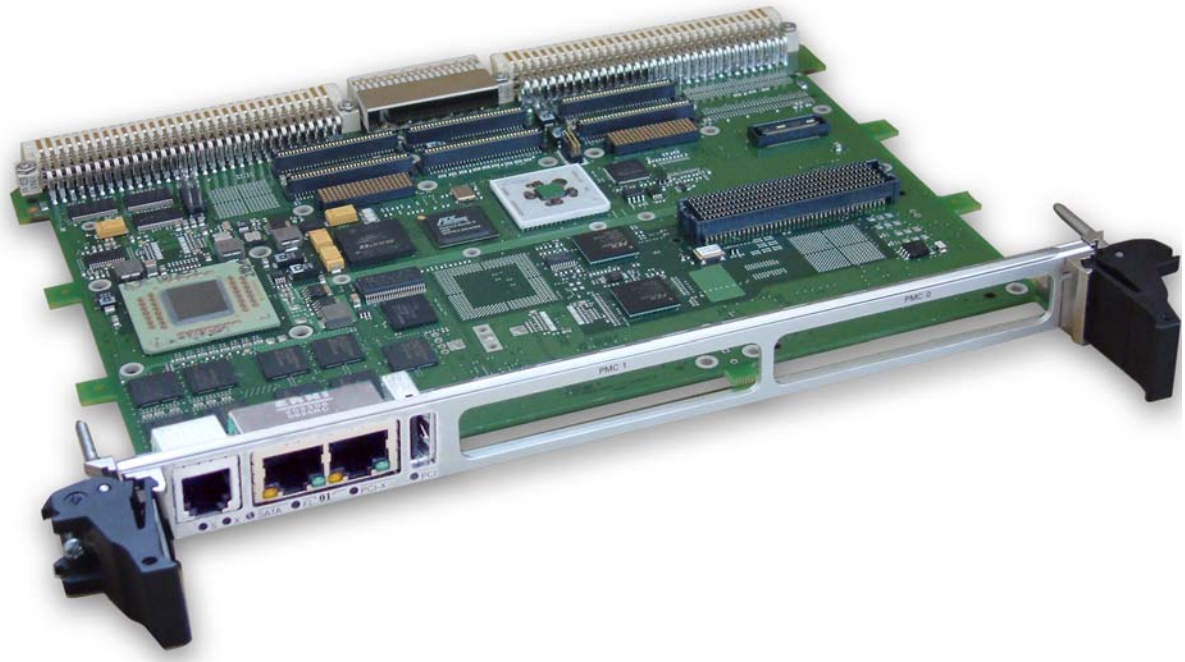
kontron

» VM6250 «



U-Boot User Manual

SD.DT.F36-2e - February 2010

Revision History

Publication Title:	VM6250 U-Boot User Manual	
Doc. ID:	SD.DT.F36-2e	
Rev.	Brief Description of Changes	Date of Issue
2e	Changes releated to ID 10047 Remove section Additional Information	02-2010
1e	Changes releated to ID 10019	01-2010
0e	Initial Version	10-2009

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.

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 General Purpose

This document describes the firmware available on the Kontron VM6250 board.

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

The firmware used on the VM6250 board is based on U-Boot 1.3.3 from U-Boot Project and the last Kontron release for VM6250 board is:

U-Boot 1.3.3 ID 10047.

The VM6250 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 Build-In Test (PBIT)
- Configure the boot of the board

1.2 U-Boot Project

The U-Boot project is fully described on the web site: http://www.denx.de/wiki/U-Boot

The U-Boot running on the VM6250 is compiled with the ELDK 4.2 tools environment using a GNU cross environment built for target=powerpc-linux on host=i686-pc-linux-gnu, located at the following URL: http://www.denx.de/en/view/Software/WebHome - Embedded_Linux_Development_Kit.

1.3 Associated Documentation

Kontron Documentation

- VM6250 6U VME SBC User's Guide CA.DT.A65
- VM6250 Hardware Release Notes CA.DT.A66
- VM6250 PBIT User's Guide SD.DT.F35
- ALMA2f - PCI 32-66 MHz / VME 2eSST FPGA Bridge User Manual CI.DT.A00

DENX Software Engineering Documentation

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

1.4 U-Boot for VM6250

1.4.1 Overview

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 VM6250 configuration.

Refer to Chapter 2 “U-Boot in User Mode" page 6, or Chapter 3 “U-Boot in Rescue Mode" page 7 for detailed information.

Power-on Built-In Test (PBIT) are available on the VM6250 if customer has ordered this option. Refer to document SD.DT.F35 for detailed information on PBIT.

For a VM6250 board @1 GHz with 2 GB of memory, the boot time:

- without running any power-on self-test is 6 seconds
- Refer to SD.DT.F35 documentation for boot time including PBIT execution

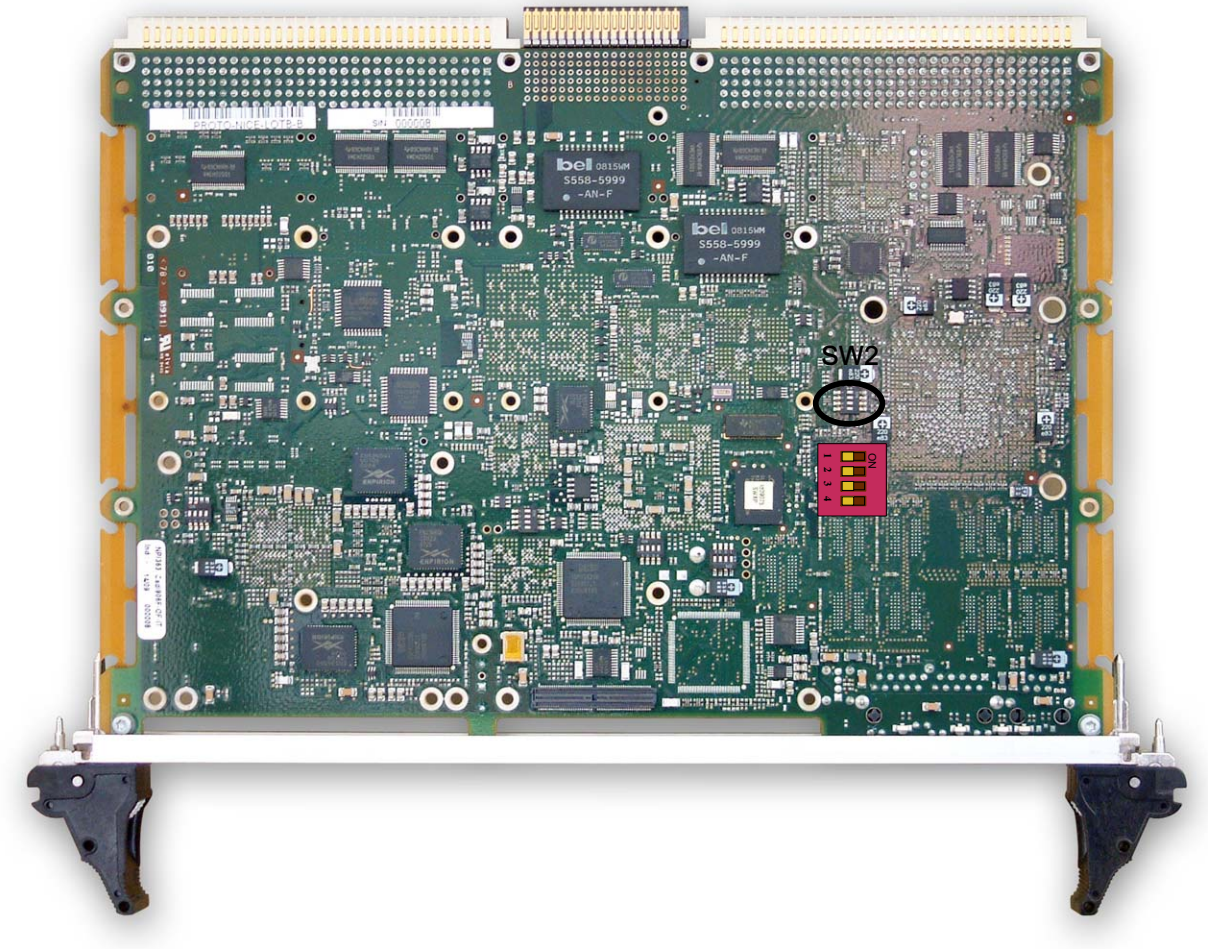
1.4.2 U-Boot Mapping on System Flash

The VM6250 has two 4 MB system flash x16 width.

The boot system flash is selected with the DIP Switch SW2, location 2 (SW2_2: Boot Flash Selection) on the board (Refer to CA.DT.A65 documentation).

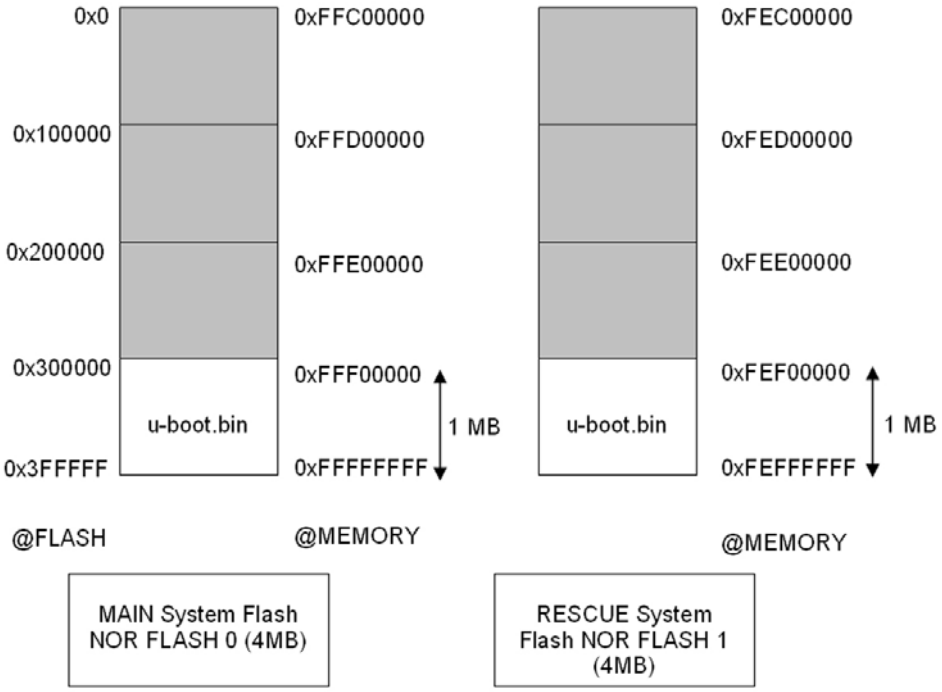
When the DIP Switch SW2_2 is set to **OFF**, the FLASH #0 is the boot system flash (MAIN Flash) and U-Boot prompt is: VM6250 =>

When the DIP switch SW2_2 is set to **ON**, the FLASH #1 is the boot system flash (RESCUE Flash) and U-Boot prompt is: VM6250-RESCUE =>



DIP Switch SW2	Function	Description
1	Reserved	Reserved
2	Boot Flash Selection	ON (0) Boot from Flash 1 OFF (1) Boot from Flash 0
3	Reserved	Reserved
4	Reserved	Reserved

System Flash Mapping



1.4.3 Memory Mapping

The following table shows the Memory mapping implemented on U-Boot:

LAW	BAT	LCS	Description	Size	Address range
0	-	CS0	NOR FLASH0	16M	0xFFFF_FFFF - 0xFF00_0000
	-	CS1	NOR FLASH1	16M	0xFEFF_FFFF - 0xFE00_0000
	-	-	unused	91M	0xFDFF_FFFF - 0xF850_0000
	-	-	Bridge PCIe/USB	32K	0xF84F_FFFF - 0xF840_8000
	-	-	Firm stack	32K	0xF840_7FFF - 0xF840_0000
	-	-	unused	3M	0xF83F_FFFF - 0xF810_0000
	-	-	CCSRBAR	1M	0xF80F_FFFF - 0xF800_0000
	-	CS2	LPC MEM (FLASH)	64M	0xF7FF_FFFF - 0xF400_0000
-	-	-	unused	64M	0xF3FF_FFFF - 0xF000_0000
1	-	CS3	Onboard Regs + LPC I/O	64M	0xEFFF_FFFF - 0xEC00_0000
	-	CS4	NvSRAM	64M	0xEBFF_FFFF - 0xE800_0000
5	0	-	PCI Express 2 I/O	64M	0xE7FF_FFFF - 0xE400_0000
4	0	-	PCI Express 1 I/O	64M	0xE3FF_FFFF - 0xE000_0000
-	-	-	unused	256M	0xDFFF_FFFF - 0xD000_0000
3	4	-	PCI Express 2 MEM	512M	0xCFFF_FFFF - 0xC000_0000
2	3	-	PCI Express 1 MEM	1G	0xBFFF_FFFF - 0x8000_0000
1	1	-	DDR-SDRAM	2G	0x7FFF_FFFF - 0x0000_0000

Legend:

- **LAW** = Local Access Window
- **BAT** = Block Address Translation
- **LCS** = Local Chip Select

Chapter 2 - U-Boot in User Mode

Connect a serial console to the VM6250 board with serial speed set to 115200 bauds and an Ethernet cable either on the front panel or via the Rear Transition Module (RTM) depending of the VM6250 board class (SA/RC).

> VM6250/SA Font Panel Overview

> Traces displayed when the VM6250 is booted up through the serial console in User mode with DIP switch SW2_2 set to **OFF**:

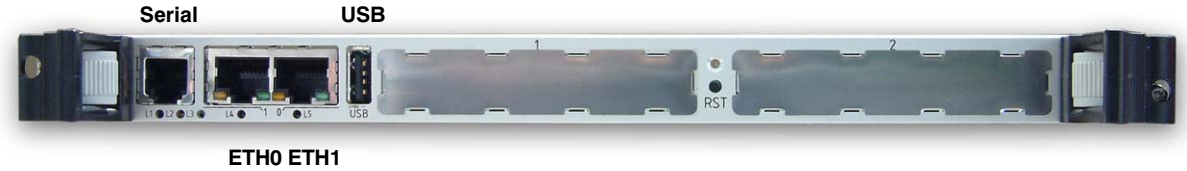
```
U-Boot 1.3.3 ID10047 for KONTRON VM6250 board

Freescale PowerPC
CPU:
    Core: E600 Core 0, Version: 0.2, (0x80040202)
    System: 8640D/8641D, Version: 2.1, (0x80900121)
    Clocks: CPU:1056 MHz, MPX: 528 MHz, DDR: 264 MHz, LBC:  66 MHz
    L2: Enabled
Board: VM6250
I2C:   ready
DRAM:          -> DDR: Init mem CTL 1 step 1 done
               -> DDR: Init mem CTL 1 step 2 done
               -> DDR: Init mem CTL 1 step 3 done
               -> DDR: 1024 MB
VPD EEPROM:   ready
FLASH:  8 MB
PCI: ready
In:    serial
Out:   serial
Err:   serial
Net:   eTSEC1, eTSEC2, eTSEC3, eTSEC4
SATA:
               SATA0 : No SATA Device Found
               SATA1 : No SATA Device Found
VM6250 =>
```

Chapter 3 - U-Boot in Rescue Mode

Connect a serial console to the VM6250 board with serial speed set to 115200 bauds and an Ethernet cable either on the front panel or via the Rear Transition Module (RTM) depending of the VM6250 board class (SA/RC).

> VM6250/SA Font Panel Overview



> Traces displayed when the VM6250 is booted up through the serial console in Rescue mode with DIP switch SW2_2 set to **ON**:

```
   U-Boot 1.3.3 ID10047 for KONTRON VM6250 board

Freescale PowerPC
CPU:
    Core: E600 Core 0, Version: 0.2, (0x80040202)
    System: 8640D/8641D, Version: 2.1, (0x80900121)
    Clocks: CPU:1056 MHz, MPX: 528 MHz, DDR: 264 MHz, LBC:  66 MHz
    L2: Enabled
Board: VM6250
I2C:   ready
DRAM:         -> DDR: Init mem CTL 1 step 1 done
              -> DDR: Init mem CTL 1 step 2 done
              -> DDR: Init mem CTL 1 step 3 done
              -> DDR: 1024 MB
Boot from U-Boot rescue image. PBIT bypassed.
VPD EEPROM:   ready
FLASH:  8 MB
PCI:  ready
In:    serial
Out:   serial
Err:   serial
Net:   eTSEC1, eTSEC2, eTSEC3, eTSEC4
SATA:
             SATA0 : No SATA Device Found
             SATA1 : No SATA Device Found
Boot from U-Boot rescue image. PBIT bypassed.
VM6250-RESCUE =>
```

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 6 “U-Boot Command Line Interface" page 17.

4.1 Print the Current Environment Parameters

The U-Boot is delivered on the VM6250 with default parameters. These parameters are displayed by running the **printenv** command:

Here under an example of display on the VM6250 board:

```
VM6250 => printenv
bootcmd=setenv bootargs root=/dev/nfs rw nfsroot=$nfsip:$rootpath
ip=$ipaddr:$serverip:$gatewayip:$netmask:$hostname:$netdev:off
console=$consoledev,$baudrate $othbootargs;tftp $loadaddr $bootfile;tftp $fdtaddr
$fdtfile;bootm $loadaddr - $fdtaddr
ramboot=setenv bootargs root=/dev/ram rw console=$consoledev,$baudrate $othbootargs;tftp
$ramdiskaddr $ramdiskfile;tftp $loadaddr $bootfile;tftp $fdtaddr $fdtfile;bootm $loadaddr
$ramdiskaddr $fdtaddr
nfsboot=setenv bootargs root=/dev/nfs rw nfsroot=$nfsip:$rootpath
ip=$ipaddr:$serverip:$gatewayip:$netmask:$hostname:$netdev:off
console=$consoledev,$baudrate $othbootargs;tftp $loadaddr $bootfile;tftp $fdtaddr
$fdtfile;bootm $loadaddr - $fdtaddr
bootdelay=10
baudrate=115200
loads_echo=1
ipaddr=192.93.167.109
serverip=192.93.167.102
rootpath=/opt/nfsroot
gatewayip=192.93.167.46
netmask=255.255.255.0
hostname=unknown
bootfile=uImage
loadaddr=1000000
loadsata=y
loadusb=n
pci_debug=n
sata_speed=low
dluboot=setenv ipaddr $ipaddr; setenv gatewayip $gatewayip; setenv serverip $serverip;
tftp 0x2000000 u-boot.bin
bootdisk=ext2load sata 0:1 0x1000000 vmlinuz-2.6.25.ppc.smp; ext2load sata 0:1 0x2000000
initrd-2.6.25.ppc.smp.img; bootm 0x1000000 0x2000000 -
ethaddr=00:00:de:50:c0:5c
eth1addr=00:00:de:50:c0:5d
eth2addr=00:00:de:50:c0:5e
eth3addr=00:00:de:50:c0:5f
stdin=serial
stdout=serial
stderr=serial
ethact=eTSEC1

Environment size: 1368/8188 bytes
```

4.2 U-Boot pre-defined Environment Variables

U-Boot from Kontron is provided with a set of pre-defined environment variables that are described in the table below:

Variable Name	Default Value	Description
loadsata	y	Enable/Disable SATA driver initialization at startup.
loadusb	n	Enable/Disable USB driver initialization at startup.
pci_debug	n	Enable/Disable PCI bus messages displayed at startup.
sata_speed	low	Enable SATA II (3 Gbps) support if set to 'high'
optim_vme_vsr_ctl	0x10	Set optimal VME performance for VSR_CTL register in ALMA2f PCI/VME bridge. See ALMA2f User Manual for more information.
optim_plx_prefetch	0x03	Set optimal PCI PREFETCH performance for PCI Express-to-PCI bridge. See ExpressLane™ PEX 8112 datasheet for more information.
optim_plx_max_read	0x04	Set optimal PCI MAX_READ performance for the PCI Express-to-PCI bridge. See ExpressLane™ PEX 8112 datasheet for more information.
optim_plx_cache	0x20	Set optimal PCI CACHELINE performance for PCI Express-to-PCI bridge. See ExpressLane™ PEX 8112 datasheet for more information.
dhcp_vendor_class_id	VM6250	Set DHCP Vendor Class Identifier
dhcp_client_id	-	Set DHCP Client Identifier

Each environment variables can be modified by using the **setenv** command.

The example below disables the SATA driver initialization at system startup:

```
VM6250 => setenv loadsata n
VM6250 => saveenv
VM6250 => reset
```

4.3 Set Ethernet Parameters

The VM6250 board has two Ethernet interfaces that the user needs to initialize before updating the firmware using network interface. Depending on the VM6250 board class (SA or RC), the Ethernet ports are accessible either on front panel or on rear P0 backplane via the Rear Transition Module (RTM).

The 2 front panel Ethernet ports are named eTSEC1 and eTSEC2 and the rear Ethernet ports are named eTSEC3 and eTSEC4.

In order to update firmware from network interface, set the following Ethernet parameters:

ipaddr	IP address of the VM6250
gatewayip	IP address of the gateway between the server and the VM6250
serverip	IP address of the server

> Example:

VM6250 IP address	192.93.167.109
Gateway IP address	192.193.167.46
Server IP address	192.93.167.102

```
VM6250 => setenv ipaddr 192.93.167.109
VM6250 => setenv gatewayip 192.93.167.46
VM6250 => setenv serverip 192.93.167.102
```

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

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

```
VM6250 => saveenv
Saving Environment to EEPROM...
VM6250 =>
```

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:

```
VM6250 => ping 192.93.167.102
Enet starting in 1000BT/FD
Speed: 1000, full duplex
Using eTSEC2 device
host 192.93.167.102 is alive
VM6250 =>
```

4.4 Create and Run User Parameters

U-Boot firmware on the VM6250 allows the user to create his own parameters using the U-Boot **setenv** command.

For example:

- to download an executable file "**example.bin**" from the network using the tftp protocol, load to address 0x20000000 (which is a safe SDRAM address enought to store a 128-MBytes file) and to execute the associated code
- to create a user parameter named "**flash_file**"

use the procedure described below:

The network parameters are supposed to be set as described in the sections 4.3 "Set Ethernet Parameters" page 10.

```
VM6250 =>  setenv flash_file 'tftp 0x2000000 example.bin;go 0x2000000'
VM6250 =>  saveenv
VM6250  =>
```

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

At this point the user parameter named "**flash_file**" is stored in the VM6250 non-volatile memory. To execute this parameter, type the following command:

```
VM6250 =>  run flash_file
VM6250 =>
```

Another example:

- to set Ethernet parameters and download an image file "**u-boot.bin**", create a user parameter named "**dluboot**" using the procedure described below:

```
VM6250 =>  setenv dluboot  'setenv ipaddr 192.93.167.109; setenv gatewayip 192.93.167.46;
setenv serverip 192.93.167.102; tftp 0x2000000 u-boot.bin'
VM6250 =>  saveenv
VM6250 =>
```

To execute this parameter, type the following command:

```
VM6250 =>  run dluboot
Enet starting in 1000BT/FD
Speed: 1000, full duplex
Using eTSEC2 device
TFTP from server 192.93.167.102; our IP address is 192.93.167.109
Filename 'u-boot.bin'.
Load address: 0x2000000
Loading: *_##########################
done
Bytes transferred = 389120 (5f000 hex)
VM6250 =>
```

Chapter 5 - Update U-Boot Firmware

The VM6250 U-Boot user firmware on Main System Flash can be either updated (User Mode) or restored from the RESCUE Flash (Rescue mode).

THE U-Boot ON THE RESCUE FLASH IS NEVER UPDATED !!!

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 previous chapter.

5.1 Update Main Flash from User Mode

> VM6250/SA Front Panel Overview

> The procedure to update U-Boot via Ethernet on MAIN Flash is:

1. Check first if DIP switch SW2_2 on VM6250 board is set to OFF (Boot on Main Flash) .
2. Connect a serial console via a RJ-14 cable to the front serial port of the VM6250 board.
3. Connect an Ethernet cable either on Ethernet Port 0 (eTSEC1) or Port 1 (eTSEC2) on front panel or on Ethernet Port 2 (eTSEC3) or Port 3(eTSEC4) on the RTM depending on the VM6250 board class (SA/RC).
4. Set the speed serial console to 115200 bauds.
5. Power-On the VM6250 board and wait for **`VM6250 =>`** prompt .

6. Set the correct Ethernet parameters by running the command **`run dluboot`**:

```
VM6250 => run dluboot
```


This command is an alias for the following actions:

```
VM6250 => setenv ipaddr 192.93.167.109
VM6250 => setenv gatewayip 192.93.167.46
VM6250 => setenv serverip 192.93.167.102
VM6250 => tftp 0x2000000 uboot.bin
```

Depending on your network settings, you may change the **`serverip/ipaddr/gatewayip`** environment variables to retrieve the **`uboot.bin image`** file from the TFTP server.

> Example:

```
VM6250 => run dluboot
Auto-neg error, defaulting to 10BT/HD
eTSEC1: No link.
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC2 device
TFTP from server 192.93.167.102; our IP address is 192.93.167.109
Filename 'uboot.bin'.
Load address: 0x2000000
Loading: *_Got error 4
T  ##########################
done
Bytes transferred = 372480 (5af00 hex)
```

7. Type **`uboot update`**:

> Example:

```
VM6250 => uboot update
Checking update image at 0x2000000 ... OK
Updating U-Boot on Flash #0 (MAIN U-Boot image)
Protect off 0xfff00000 0xfff6c7ff
....... done
Un-Protected 7 sectors
Erase 0xfff00000 0xfff6c7ff
....... done
Erased 7 sectors
Copy to Flash...  done
Verify Flash... OK
Protect on 0xfff00000 0xfff6c7ff
....... done
Protected 7 sectors
Update OK
```

The message **`OK`** appears after **`Verify Flash...`** if the update is completed successfully.

If the message **`FAILED`** appears after **`Verify Flash...`**, check environment parameters by using **`printenv`** command and repeat from step 5.

8. Type **reset** on VM6250 prompt or Power Off/On the rack or push Reset Button of the board and wait for prompt **VM6250 =>**

   The new version is displayed by running **ver** command.

   > Example :

```
VM6250 => ver

U-Boot 1.3.3 ID10047 for KONTRON VM6250 board
VM6250 =>
```

   Instead of typing **ver**, the single **uboot** command without parameter shows the current version on both Flash along with the current active Flash.

   > Example:

```
VM6250 => uboot
------------------------------------------------------------------------
-> DIP SW2 not set: boot from Flash #0
------------------------------------------------------------------------
-> Flash #0 and Flash #1 are NOT Write-Protected.
------------------------------------------------------------------------
Flash #0 (MAIN U-Boot image) : Current active flash
-> U-Boot Version = U-Boot 1.3.3 ID10047 for KONTRON VM6250 board
------------------------------------------------------------------------
Flash #1 (RESCUE U-Boot image)
-> U-Boot Version = U-Boot 1.3.3 ID09345 for KONTRON VM6250 board
------------------------------------------------------------------------
VM6250 =>
```

By default, the **uboot update** command does not change any environment variables set previously by the user. To restore the default environment variables, add the **-e** option after the **uboot update** command.

```
VM6250 => uboot update -e
Erasing EEPROM to reload default environment variables
Checking update image at 0x2000000 ... OK
Updating U-Boot on Flash #0 (MAIN U-Boot image)
Protect off 0xfff00000 0xfff6b6ff
....... done
Un-Protected 7 sectors
Erase 0xfff00000 0xfff6b6ff
....... done
Erased 7 sectors
Copy to Flash...  done
Verify Flash... OK
Protect on 0xfff00000 0xfff6b6ff
....... done
Protected 7 sectors
Update OK
VM6250 => reset
```

After reseting the board, a message indicating that there is a bad CRC is displayed:

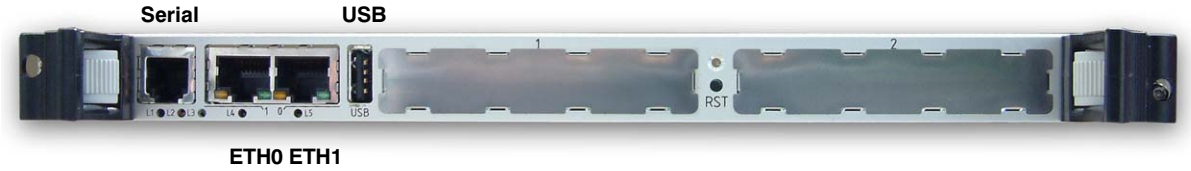
***** Warning - bad CRC, using default environment**

This is a normal message to warn the user that the default environment variables containing in the current U-Boot image have been loaded in the EEPROM and are ready to use.

The user needs to save the current environment variables by using the **saveenv** command in order to not have this message displayed at next reset. If the user wants to set its own environment variables, he can use the **setenv** command and then the **saveenv** command.

5.2 Restore U-Boot version

> VM6250/SA Front Panel Overview



Sometimes, the Main Flash might be corrupted and it is not possible to have the U-Boot VM6250 prompt.

In order to repair the corrupted Flash, the user can switch on the Rescue Flash by setting the DIP switch SW2_2 to ON.

The procedure to restore a MAIN corrupted Flash is shown below:

1. Check first if DIP_SW2 on VM6250 board is set to ON (Boot on Rescue Flash).
2. Connect a serial console via a RJ-14 cable to the front serial port of the VM6250 board.
3. Connect an Ethernet cable either on Ethernet Port 0 or Port 1 on front of the board.
4. Set the speed serial console to 115200 bauds.
5. Power-On the VM6250 board and wait for **`VM6250-RESCUE =>`** prompt.
6. Type **`uboot`** to display the current configuration

   > Example:

```
VM6250-RESCUE => uboot
-------------------------------------------------------------------------
-> DIP SW2 set: boot from Flash #1  !!!!!! RESCUE IMAGE !!!!!!
-------------------------------------------------------------------------
-> Flash #0 and Flash #1 are NOT Write-Protected.
-------------------------------------------------------------------------
Flash #1 (RESCUE U-Boot image) : Current active flash
-> U-Boot Version = U-Boot 1.3.3 ID09345 for KONTRON VM6250 board
-------------------------------------------------------------------------
Flash #0 (MAIN U-Boot image)
-> U-Boot Version = U-Boot 1.3.3 ID10047 for KONTRON VM6250 board
-------------------------------------------------------------------------
VM6250-RESCUE =>
```

7. Type **uboot restore**:

   > Example:

```
VM6250-RESCUE => uboot restore
Checking U-Boot image on Flash #1 (RESCUE) ...
OK
Loading U-Boot image from Flash #1 (RESCUE) ...
Updating U-Boot Flash #0 (MAIN)
Protect off 0xfef00000 0xfef6b5ff
....... done
Un-Protected 7 sectors
Erase 0xfef00000 0xfef6b5ff
....... done
Erased 7 sectors
Copy to Flash...  done
Verify Flash... OK
Protect on 0xfef00000 0xfef6b5ff
....... done
Protected 7 sectors
Update OK
```

   If the message **OK** appears after **Verify Flash...** , go to next step.

   If the message **FAILED** appears after **Verify Flash...** , repeat step 5.

8. Type **uboot** to check if the copy is done from Rescue flash to Main flash

   > Example:

```
VM6250-RESCUE => uboot
-------------------------------------------------------------------
-> DIP SW2 set: boot from Flash #1  !!!!!! RESCUE IMAGE !!!!!!
-------------------------------------------------------------------
-> Flash #0 and Flash #1 are NOT Write-Protected.
-------------------------------------------------------------------
Flash #1 (RESCUE U-Boot image) : Current active flash
-> U-Boot Version = U-Boot 1.3.3 ID09345 for KONTRON VM6250 board
-------------------------------------------------------------------
Flash #0 (MAIN U-Boot image)
-> U-Boot Version = U-Boot 1.3.3 ID09345 for KONTRON VM6250 board
-------------------------------------------------------------------
VM6250-RESCUE =>
```

Chapter 6 - U-Boot Command Line Interface

The U-Boot implementation delivered with VM6250 contains the complete boot code infrastructure. Our Quality Insurance process has thoroughly qualified proper operation of the U-Boot commands required to operate VM6250. 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 VM6250 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.

6.1 Commands Summary

6.1.1 Proprietary Qualified Commands

List of U-Boot proprietary qualified commands available only for U-Boot on VM6250 boards.

Command Name	Description	Command Type	See section
diag	Perform board diagnostics. **Available ONLY if ordered.**	Proprietary	6.2.1 page 20
uboot update	Update U-Boot image on MAIN flash	Proprietary	6.2.2 page 20
uboot restore	Restore U-Boot image from RESCUE flash to MAIN flash	Proprietary	6.2.2 page 20
uboot	Display U-Boot image on MAIN flash and RESCUE flash	Proprietary	6.2.2 page 20
vpdutil pld	Display current version of PLD on the board	Proprietary	6.2.3 page 21
vpdutil show	Display Vital Product Data (VPD) information of the board	Proprietary	6.2.3 page 21

6.1.2 Standard Qualified Commands

List of U-Boot standard qualified commands available on VM6250 boards.

Command Name	Description	Command Type	See section
?	alias for 'help'	Miscellaneous	6.11.10 page 47
bdinfo	print Board Info structure	Information	6.3.1 page 22
bootd	boot default, i.e., run 'bootcmd'	Environment Var.	6.8.5 page 36
bootelf	boot from an ELF image in memory	Execution Control	6.6.2 page 31
bootline	boot parameters	Execution Control	6.9.1 page 37
bootm	boot application image from memory	Execution Control	6.6.3 page 31
bootp	boot image via network using bootp/TFTP protocol	Network	6.7.1 page 33
bootvx	boot VxWorks from an ELF image	Execution Control	6.9.2 page 37
cmp	memory compare	Memory	6.4.3 page 26
coninfo	print console devices and information	Information	6.3.2 page 22
cp	memory copy	Memory Flash Memory	6.4.4 page 27
crc32	checksum calculation	Memory	6.4.2 page 26
date	get/set/reset date & time	Miscellaneous	6.11.1 page 45
dcache	enable or disable data cache	Special	6.10.1 page 40
dtt	Digital Thermometer and Thermostat	Miscellaneous	6.11.2 page 45
echo	echo args to console	Miscellaneous	6.11.3 page 45
erase	erase FLASH memory	Flash Memory	6.5.2 page 29
ext2load	load binary file from a filesystem image	Filesystem Sup.	6.9.4 page 38
ext2ls	list files in a directory (default/)	Filesystem Sup.	6.9.3 page 37
flinfo	print FLASH memory information	Information Flash Memory	6.3.3 page 23
go	start application at addr 'addr'	Execution Control	6.6.4 page 32
help	print online help	Information	6.3.6 page 24
i2c	I2C Sub-system	Special	6.10.2 page 40
icache	enable or disable instruction cache	Special	6.10.3 page 40
icrc32	checksum calculation	Special	6.10.4 page 40
iloop	infinite loop on address range	Special	6.10.5 page 40
imd	I2C memory modify (auto-incrementing)	Special	6.10.6 page 40
iminfo	print header information for application image	Information	6.3.4 page 24
imm	I2C memory modify (auto-incrementing)	Special	6.10.7 page 41
imw	I2C memory write (fill)	Special	6.10.8 page 41
inm	I2C memory modify (constant address)	Special	6.10.9 page 41
iprobe	probe to discover valid I2C chip addresses	Special	6.10.10 page 41
loop	infinite loop on address range	Memory	6.4.9 page 28
md	memory display	Memory	6.4.5 page 27
mii	MII utility command	Network	6.7.9 page 34
mm	memory modify (auto-incrementing)	Memory	6.4.6 page 27
mtest	Simple RAM Test	Memory	6.4.10 page 28
mw	memory write (fill)	Memory	6.4.7 page 27
nfs	boot image via network using NFS protocol	Network	6.7.5 page 34

Command Name	Description	Command Type	See section
nm	memory modify (constant address)	Memory	6.4.8 page 27
pci	list and access PCI configuration space	Special	6.10.11 page 41
ping	send ICMP ECHO_REQUEST to network host	Network	6.7.6 page 34
printenv	print environment variables	Environment Var.	6.8.1 page 35
protect	enable or disable FLASH write protection	Flash Memory	6.5.3 page 30
rarpboot	boot image via network using RARP/TFTP protocol	Network	6.7.7 page 34
reginfo	print registers information	Information	6.3.7 page 25
reset	perform reset of the CPU	Miscellaneaous	6.11.6 page 45
run	run commands in an environment variable	Environment Var.	6.8.4 page 36
sata	SATA sub system	Special	6.10.12 page 42
saveenv	save environment variables to persistent storeage	Environment Var.	6.8.2 page 35
setenv	set environment variables	Environment Var.	6.8.3 page 35
sleep	delay execution for some time	Miscellaneous	6.11.7 page 46
tftpboot	boot image via network using TFTP protocol	Network	6.7.8 page 34
usb	USB sub system	Special	6.10.13 page 43
usbboot	Boot from USB device	Special	6.10.14 page 44
version	print monitor version	Miscellaneous	6.11.9 page 46

6.1.3 Standard Delivered Commands

List of U-Boot standard delivered commands available on VM6250 boards.

Command Name	Description	Command Type	See section
autoscr	run script from memory	Execution Control	6.6.1 page 31
base	print or set address offset	Memory	6.4.1 page 26
exit	exit script	Execution Control	6.11.4 page 45
fatinfo	print information about filesystem	Filesystem Sup.	6.9.5 page 38
fatload	load binary file from a dos filesystem	Filesystem Sup.	6.9.6 page 38
fatls	list files in a directory (default/)	Filesystem Sup.	6.9.7 page 38
fdt	flattened device tree utility commands	Filesystem Sup.	6.9.8 page 39
imls	list all images found in flash	Information	6.3.5 page 24
imxtract	extract a part of a multi-image	Filesystem Sup.	6.9.10 page 39
itest	return true/false on integer compare	Miscellaneous	6.11.5 page 45
loadb	load binary file over serial line (kermit mode)	Network	6.7.2 page 33
loads	load s-record file over serial line	Network	6.7.3 page 34
loady	load binary file over serial line (ymodem mode)	Network	6.7.4 page 34
test	minimal test like /bin/sh	Miscellaneous.	6.11.8 page 46

6.2 Proprietary Commands

6.2.1 diag - perform board diagnostics

This command is fully described in document SD.DT.F35 - ”VM6250 PBIT User's Guide”.

6.2.2 uboot update/restore commands - manage U-Boot image

Refer to Chapter 5 for details.

At any time, type **`help uboot`** to retrieve all available commands as described below:

```
VM6250 => help uboot
uboot - display Flash configuration on Flash #0 (MAIN) and Flash #1 (RESCUE)
uboot update [<image address>] [-e] - update U-Boot on Flash #0 (MAIN)
uboot restore - restore U-Boot on Flash #0 (MAIN) from Flash #1 (RESCUE)

VM6250 =>
```

6.2.3 vpdutil - manage VPD (Vital Product Data)

```
VM6250 => help vpdutil
vpdutil read offset length              - read from VPD EEPROM
vpdutil write offset value              - write to VPD EEPROM
vpdutil update                          - update vpd data of the board
vpdutil show                            - show vpd data of the board
vpdutil pld                             - show revision of PLD
vpdutil default                         - fill VPD EEPROM with default values

VM6250 =>
```

Customer does not need to use those commands except for debug information purpose, for example to provide at Kontron support the revision of the PLD or any useful information of the board.

Example:

```
VM6250 => vpdutil pld
-> PLD revision: 07
VM6250 => vpdutil show

************* VPD *************

-> Board Data 0

Board Id           : 00470000
Hardware Index     : 0013
Software Index     : 0003
Main Memory Size   : 40000000
Flash Memory Size  : 00400000
Nvsram Memory Size : 00000000
-> Board Data 1

Serial Number      : 0000000000002
CPU Id             : 0047
Frequency 0        : 1f78a400
Frequency 1        : 4ead9aa8
Num of MAC         : 04
MAC 0              : 0000de403634
MAC 1              : 0000de403635
MAC 2              : 0000de403636
MAC 3              : 0000de403637
CRC Board Data 1   : 006466

-> Production Data

Order Code         : VM6250-2SA44-01110
EC Level           : 12345
Variant            : 0230046004020000

VM6250 =>
```

6.3 Information Commands

6.3.1 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.

```
VM6250 => help bdinfo
bdinfo - No help available.
VM6250 => bdinfo
memstart      = 0x00000000
memsize       = 0x80000000
flashstart    = 0xFEC00000
flashsize     = 0x00800000
flashoffset   = 0x00000000
sramstart     = 0x00000000
sramsize      = 0x00000000
bootflags     = 0x00000001
intfreq       =   1056 MHz
busfreq       =    528 MHz
ethaddr       = 00:00:DE:40:36:34
eth1addr      = 00:E0:0C:00:01:FD
eth2addr      = 00:E0:0C:00:02:FD
eth3addr      = 00:E0:0C:00:03:FD
IP addr       = 192.168.1.100
baudrate      = 115200 bps
VM6250 =>
```

6.3.2 coninfo - print console devices and information

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

```
VM6250 => coninfo
List of available devices:
serial   80000003 SIO stdin stdout stderrhelp coninfo
VM6250 =>
```

6.3.3 flinfo - print FLASH memory information

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

```
VM6250 => help flinfo
flinfo       - print information for all FLASH memory banks
flinfo N     - print information for FLASH memory bank #N

VM6250 => flinfo

Bank # 1: CFI conformant FLASH (16 x 16)  Size: 4 MB in 64 Sectors
  AMD Standard command set, Manufacturer ID: 0x01, Device ID: 0x227E
  Erase timeout: 16384 ms, write timeout: 2 ms
  Buffer write timeout: 5 ms, buffer size: 32 bytes


Sector Start Addresses:
  FFC00000 E      FFC10000 E      FFC20000 E      FFC30000 E      FFC40000 E
  FFC50000 E      FFC60000 E      FFC70000 E      FFC80000 E      FFC90000 E
  FFCA0000 E      FFCB0000 E      FFCC0000 E      FFCD0000 E      FFCE0000 E
  FFCF0000 E      FFD00000 E      FFD10000 E      FFD20000 E      FFD30000 E
  FFD40000 E      FFD50000 E      FFD60000 E      FFD70000 E      FFD80000 E
  FFD90000 E      FFDA0000 E      FFDB0000 E      FFDC0000 E      FFDD0000 E
  FFDE0000 E      FFDF0000 E      FFE00000 E      FFE10000 E      FFE20000 E
  FFE30000 E      FFE40000 E      FFE50000 E      FFE60000 E      FFE70000 E
  FFE80000 E      FFE90000 E      FFEA0000 E      FFEB0000 E      FFEC0000 E
  FFED0000 E      FFEE0000 E      FFEF0000 E      FFF00000   RO   FFF10000   RO
  FFF20000   RO   FFF30000   RO   FFF40000   RO   FFF50000   RO   FFF60000   RO
  FFF70000        FFF80000        FFF90000        FFFA0000        FFFB0000
  FFFC0000        FFFD0000        FFFE0000        FFFF0000

Bank # 2: CFI conformant FLASH (16 x 16)  Size: 4 MB in 64 Sectors
  AMD Standard command set, Manufacturer ID: 0x01, Device ID: 0x227E
  Erase timeout: 16384 ms, write timeout: 2 ms
  Buffer write timeout: 5 ms, buffer size: 32 bytes

  Sector Start Addresses:
  FEC00000 E      FEC10000 E      FEC20000 E      FEC30000 E      FEC40000 E
  FEC50000 E      FEC60000 E      FEC70000 E      FEC80000 E      FEC90000 E
  FECA0000 E      FECB0000 E      FECC0000 E      FECD0000 E      FECE0000 E
  FECF0000 E      FED00000 E      FED10000 E      FED20000 E      FED30000 E
  FED40000 E      FED50000 E      FED60000 E      FED70000 E      FED80000 E
  FED90000 E      FEDA0000 E      FEDB0000 E      FEDC0000 E      FEDD0000 E
  FEDE0000 E      FEDF0000 E      FEE00000 E      FEE10000 E      FEE20000 E
  FEE30000 E      FEE40000 E      FEE50000 E      FEE60000 E      FEE70000 E
  FEE80000 E      FEE90000 E      FEEA0000 E      FEEB0000 E      FEEC0000 E
  FEED0000 E      FEEE0000 E      FEEF0000 E      FEF00000        FEF10000
  FEF20000        FEF30000        FEF40000        FEF50000        FEF60000
  FEF70000        FEF80000        FEF90000        FEFA0000        FEFB0000
  FEFC0000        FEFD0000        FEFE0000        FEFF0000
VM6250 =>
```

6.3.4 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.

```
VM6250 => 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)
VM6250 =>
```

6.3.5 imls - list all images found in flash

```
VM6250 => help imls
imls - Prints information about all images found at sector boundaries in flash.
VM6250 =>
```

6.3.6 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:

```
VM6250 => help 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.
```

6.3.7 reginfo - print register information

```
VM6250 => help reginfo
reginfo --pci              - display PCI-Express Registers
reginfo --mem              - display SDRAM CTRL1 Registers
reginfo --eth              - display eTSEC Registers
reginfo --pld              - display cPLD Registers
reginfo --all              - display all Registers

VM6250 => reginfo

Local Access Window Configuration
             LAWBAR0 : 0x00000000, LAWAR0 : 0x00000000
             LAWBAR1 : 0x00080000, LAWAR1 : 0x8000001c
             LAWBAR2 : 0x00000000, LAWAR2 : 0x80f0001e
             LAWBAR3 : 0x000a0000, LAWAR3 : 0x8010001c
             LAWBAR4 : 0x000e8000, LAWAR4 : 0x80400017
             LAWBAR5 : 0x000e0000, LAWAR5 : 0x80000019
             LAWBAR6 : 0x000e4000, LAWAR6 : 0x80100019
             LAWBAR7 : 0x000fe000, LAWAR7 : 0x80400018
             LAWBAR8 : 0x000ec000, LAWAR8 : 0x80400017
             LAWBAR9 : 0x000c0000, LAWAR9 : 0x80c0001c
Local Bus Controller Registers
             BR0  0xFF001001  OR0  0xFF006FF7
             BR1  0xFE001001  OR1  0xFF006FF7
             BR2  0xF4000801  OR2  0xFF006FF7
             BR3  0xEC000801  OR3  0xFF006FF7
             BR4  0xE8000801  OR4  0xFF006FF7
             BR5  0x00000000  OR5  0x00000000
             BR6  0x00000000  OR6  0x00000000
             BR7  0x00000000  OR7  0x00000000
=======================================
Hardware Implementation Registers
             HID0          0x8482D100
             HID1          0x80008C80
=======================================
Machine State Register
             MSR           0x00009030
=======================================
Cache/Memory Subsystem Register
             L2CR          0xB0000000
             ICTRL         0x00000000
             LDSTCR        0x00000000
             MSSCR0        0x00008000
             MSSSR0        0x00000000
VM6250 =>
```

6.4 Memory Commands

6.4.1 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:

```
VM6250 => help base
base          - print address offset for memory commands
base off      - set address offset for memory commands to 'off'
VM6250 =>
```

6.4.2 crc32 - checksum calculation

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

```
VM6250 => crc 100004 3FC
CRC32 for 00100004 ... 001003ff ==> 8a231c50
VM6250 =>
```

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

```
VM6250 => crc 100004 3FC 100000
CRC32 for 00100004 ... 001003ff ==> 8a231c50
VM6250 => md 100000 4
00100000: 8a231c50 deadbeef deadbeef deadbeef    .#.P............
VM6250 =>
```

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

6.4.3 cmp - memory compare

With the **cmp** command you can test if 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.

```
VM6250 =>help cmp
cmp [.b, .w, .l] addr1 addr2 count - compare memory
VM6250 =>
```

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.

6.4.4 cp - memory copy

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

```
VM6250 =>help cp
cp [.b, .w, .l] source target count - copy memory
VM6250 =>
```

6.4.5 md - memory display

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

```
VM6250 =>help md
md [.b, .w, .l] address [# of objects] - memory display
VM6250 =>
```

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.

6.4.6 mm - memory modify (auto-incrementing)

```
VM6250 =>help mm
mm [.b, .w, .l] address - memory modify, auto increment address
VM6250 =>
```

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 .).

6.4.7 mw - memory write (fill)

```
VM6250 =>help mw
mw [.b, .w, .l] address value [count] - write memory
VM6250  =>
```

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.

6.4.8 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:

```
 VM6250 =>help nm
nm [.b, .w, .l] address - memory modify, read and keep address
VM6250  =>
```

6.4.9 loop - infinite loop on address range

```
VM6250 =>help loop
loop [.b, .w, .l] address number_of_objects - loop on a set of addresses
VM6250  =>
```

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!

6.4.10 mtest - simple RAM test

```
VM6250 => help mtest
mtest [start [end [pattern]]] - simple RAM read/write test
VM6250 =>
```

Defaut start address = 0x00200000 and end address = 0x00400000 with pattern:

0x00000001 /* single bit */
0x00000003 /* two adjacent bits */
0x00000007 /* three adjacent bits */
0x0000000F /* four adjacent bits */
0x00000005 /* two non-adjacent bits */
0x00000015 /* three non-adjacent bits */
0x00000055 /* four non-adjacent bits */
0xaaaaaaaa /* alternating 1/0 */

6.5 Flash Memory Commands

6.5.1 cp - memory command

```
VM6250 =>help cp
cp [.b, .w, .l] source target count - copy memory
VM6250 =>
```

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.

6.5.2 erase - erase FLASH memory

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.

```
VM6250 =>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
VM6250 =>
```

6.5.3 protect - enable or disable FLASH write protect

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.

```
VM6250 =>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 writable
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
VM6250 =>
```

6.6 Execution Control Commands

6.6.1 autoscr - run script from memory

```
VM6250 =>help autoscr
autoscr [addr] - run script starting at addr - A valid autoscr header must be present
VM6250 =>
```

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.

6.6.2 bootelf - boot from an ELF image in memory

```
VM6250 =>help bootelf
bootelf [address] - load address of the ELF image.
VM6250 =>
```

6.6.3 bootm - boot application image from memory

```
VM6250 =>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 argument, a bd_info struct will be passed instead
VM6250 =>
```

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).

6.6.4 go - start application at address “addr”

```
VM6250 =>help go
go addr [arg ...] - start application at address 'addr'passing 'arg' as arguments
VM6250 =>
```

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.

6.7 Network Commands

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

```
VM6250 =>help bootp
bootp [loadAddress] [bootfilename]
VM6250 =>
```

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:

```
VM6250 =>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)
VM6250 =>
```

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

```
VM6250 =>help loadb
loadb [ off ] [ baud ]
     - load binary file over serial line with offset 'off' and baudrate 'baud'
VM6250 =>
```

6.7.3 loads - load S-Record file over serial line

```
VM6250 =>help loads
loads [ off ] [ baud ]
     - load S-Record file over serial line with offset 'off' and baudrate 'baud'
VM6250 =>
```

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

```
VM6250 =>help loady
loady [ off ] [ baud ]
     - load binary file over serial line with offset 'off' and baudrate 'baud'
VM6250 =>
```

6.7.5 nfs - boot image via network using NFS protocol

```
VM6250 =>help nfs
nfs [loadAddress] [host ip addr:bootfilename]
VM6250 =>
```

6.7.6 ping - send ICMP ECHO_REQUEST to network host

```
VM6250 =>help ping
ping pingAddress
VM6250 =>
```

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

```
VM6250 =>help rarp
rarpboot [loadAddress] [bootfilename]
VM6250 =>
```

6.7.8 tftpboot - boot image via network using TFTP protocol

```
VM6250 =>help tftpboot
tftpboot [loadAddress] [[hostIPaddr:]bootfilename]
VM6250 =>
```

6.7.9 mii - MII utility command

```
VM6250 => help mii
mii device                              - list available devices
mii device <devname>                    - set current device
mii info   <addr>                       - display MII PHY info
mii read   <addr> <reg>                 - read  MII PHY <addr> register <reg>
mii write  <addr> <reg> <data>          - write MII PHY <addr> register <reg>
mii dump   <addr> <reg>                 - pretty-print <addr> <reg> (0-5 only)

addr and/or reg may be ranges, e.g. 2-7.
```

6.8 Environment Variables Commands

6.8.1 printenv - print environment variables

```
VM6250 =>help printenv
printenv                  - print values of all environment variables
printenv name ...         - print value of environment variable 'name'
VM6250 =>
```

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.

6.8.2 saveenv - save environment variables to persistent storage

```
VM6250 =>help saveenv
saveenv - No help available.
VM6250 =>
```

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.

6.8.3 setenv - set environement variables

```
VM6250 =>help setenv
setenv name value ...     - set environment variable 'name' to 'value ...'
setenv name               - delete environment variable 'name'
VM6250 =>
```

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!

6.8.4 run - run commands in an environment variable

```
VM6250 =>help run
run var [...]             - run the commands in the environment variable(s) 'var'
VM6250 =>
```

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.

6.8.5 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.

6.9 Filesystem Support

6.9.1 bootline - Boot Parameters

The VM6250 U-Boot firmware includes commands to manage bootrom parameters exactly like under VxWorks bootrom prompt. This is done by the VM6250 U-Boot command "bootline"

For example, to display bootrom paramaters:

```
VM6250 => bootline print

boot device                : motetsec
unit number                : 0
processor number           : 0
host name                  : sunblade
file name                  : /home/Vx-6.6SMP/vxworks-
6.6/target/config/VM6250/vxWorks
inet on ethernet (e)       : 192.54.144.163:0xffffff00
host inet (h)              : 192.54.144.248
user (u)                   : vxworks
ftp password (pw)          : target
flags (f)                  : 0x8
target name (tn)           : vxworks1
```

To setup bootrom parameters:

```
VM6250 => bootline set

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

boot device                : motetsec0
processor number           : 0
host name                  : sunblade
file name                  : /home/Vx-6.6SMP/vxworks-
6.6/target/config/vm6250/vxWorks
inet on ethernet (e)       : 192.54.144.163:0xffffff00
inet on backplane (b)      :
host inet (h)              : 192.54.144.248
gateway inet (g)           :
user (u)                   : vxworks
ftp password (pw) (blank = use rsh): target
flags (f)                  : 0x8
target name (tn)           : vxworks1
startup script (s)         :
other (o)                  :
```

6.9.2 bootvx - boot VxWorks from an ELF image

```
VM6250 => help bootvx
bootvx  [address] - load address of vxWorks ELF image.
```

6.9.3 ext2ls - list files in a directory (default/)

```
VM6250 => help ext2ls
ext2ls <interface> <dev[:part]> [directory]
     - list files from 'dev' on 'interface' in a 'directory'
```

6.9.4 ext2load - load binary file from a filesystem image

```
VM6250 => help ext2load
ext2load <interface> <dev[:part]> [addr] [filename] [bytes]
    - load binary file 'filename' from 'dev' on 'interface'
      to address 'addr' from ext2 filesystem
```

For example, in order to boot a Linux image from a SATA HDD, you have to execute following commands:

```
VM6250 => ext2load sata 0:1 0x1000000 /uImage
VM6250 => ext2load sata 0:1 0x2000000 /initrd.gz.uboot
VM6250 => bootm 0x1000000 0x2000000 -
```

where: sata = device SATA,

0:1=port number(0=port SATA0, 1=port SATA1):partition number

0x1000000=target address in SDRAM where the image file will be

/uImage=name of the binary file on the SATA filesystem under / directory

6.9.5 fatinfo - print information about filesystem

```
VM6250 => help fatinfo
fatinfo <interface> <dev[:part]>
     - print information about filesystem from 'dev' on 'interface'
```

6.9.6 fatload - load binary file from a dos filesystem

```
VM6250 => help fatload
fatload <interface> <dev[:part]>  <addr> <filename> [bytes]
     - load binary file 'filename' from 'dev' on 'interface'
       to address 'addr' from dos filesystem
```

6.9.7 fatls - list file in a directory (default/)

```
VM6250 => help fatls
fatls <interface> <dev[:part]> [directory]
     - list files from 'dev' on 'interface' in a 'directory'
```

6.9.8 fdt - flattened device tree utility commands

```
VM6250 => help fdt
fdt addr   <addr> [<length>]
     - Set the fdt location to <addr> (if <addr>= '-' then the local device tree is used)
fdt boardsetup                          - Do board-specific set up
fdt move   <fdt> <newaddr> <length>     - Copy the fdt to <addr> and make it active
fdt print  <path> [<prop>]              - Recursive print starting at <path>
fdt list   <path> [<prop>]              - Print one level starting at <path>
fdt set    <path> <prop> [<val>]        - Set <property> [to <val>]
fdt mknode <path> <node>                - Create a new node after <path>
fdt rm     <path> [<prop>]              - Delete the node or <property>
fdt header                              - Display header info
fdt bootcpu <id>                        - Set boot cpuid
fdt memory <addr> <size>                - Add/Update memory node
fdt rsvmem print                        - Show current mem reserve
fdt rsvmem add <addr> <size>            - Add a mem reserve
fdt rsvmem delete <index>               - Delete a mem reserve
fdt chosen                              - Add/update the /chosen branch in the tree
NOTE: If the path or property you are setting/printing has a '#' character
       or spaces, you MUST escape it with a \ character or quote it with ".

VM6250 =>
```

6.9.9 fsinfo - print information about filesystems

```
VM6250 =>help fsinfo
fsinfo - print information about filesystems
VM6250 =>
```

6.9.10 imxtract - extract a part of a multi-image

```
VM6250 => help imxtract
imxtract addr part [dest]
    - extract <part> from legacy image at <addr> and copy to <dest>
```

6.10 Special Commands

6.10.1 dcache - enable or disable data cache

```
VM6250 =>help dcache
dcache [on, off]           - enable or disable data (writethrough) cache
VM6250 =>
```

6.10.2 i2c - I2C sub-system

```
VM6250 => help i2c
i2c dev [dev]       - show or set current I2C bus
i2c speed [speed] - show or set I2C bus speed
i2c md chip address[.0, .1, .2] [# of objects] - read from I2C device
i2c mm chip address[.0, .1, .2] - write to I2C device (auto-incrementing)
i2c mw chip address[.0, .1, .2] value [count]  - write to I2C device (fill)
i2c nm chip address[.0, .1, .2] - write to I2C device (constant address)
i2c crc32 chip address[.0, .1, .2] count - compute CRC32 checksum
i2c probe - show devices on the I2C bus
i2c loop chip address[.0, .1, .2] [# of objects] - looping read of device
```

Those commands are new I2C commands for manage I2C sub-system:

```
i2c md => imd
i2c mm => imm
i2c probe => iprobe
i2c nm => inm
i2c mw => imw
i2c crc32 => icrc32
```

6.10.3 icache - enable or disable instruction cache

```
VM6250 =>help icache
icache [on, off]           - enable or disable instruction cache
VM6250 =>
```

6.10.4 icrc32 - checksum calculation

```
VM6250 =>help icrc32
icrc32 chip address [.0, .1, .2] count - compute CRC32 checksum
VM6250 =>
```

6.10.5 iloop - infinite loop on range address

```
VM6250 =>help iloop
iloop chip address[.0, .1, .2] [# of objects] - loop, reading a set of addresses
VM6250 =>
```

6.10.6 imd - I2C memory display

```
VM6250 =>help imd
imd chip address[.0, .1, .2] [# of objects] - i2c memory display
VM6250 =>
```

6.10.7 imm - I2C memory modify (auto-incrementing)

```
VM6250 =>help imm
imm chip address[.0, .1, .2] - memory modify, auto increment address
VM6250 =>
```

6.10.8 imw - I2C memory write (fill)

```
 VM6250 =>help imw
imw chip address[.0, .1, .2] value [count] - memory write (fill)
VM6250 =>
```

6.10.9 inm - I2C memory modify (constant address)

```
VM6250 =>help inm
inm chip address[.0, .1, .2] - memory modify, read and keep address
VM6250 =>
```

6.10.10 iprobe - probe to discover valid I2C chip addresses

```
VM6250 =>help iprobe
iprobe -discover valid I2C chip addresses
VM6250 =>
```

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 (6.10.4 to 6.10.9)

Example of **iprobe** command output:

```
VM6250 =>iprobe
Valid chip addresses: 48
VM6250 =>
```

6.10.11 pci - list and access PCI configuration space

```
VM6250 =>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
```

6.10.12 sata - SATA sub-system

```
VM6250 => help sata
sata reset | start         - reset and start (rescan) SATA controller
sata info                  - show available SATA devices
sata device [dev]          - show or set current device
sata part [dev]            - print partition table
sata read addr blk# cnt
sata write addr blk# cnt
```

Example:

```
VM6250 => sata info

SATA device 1: Model: WDC WD2500YD-01NVB1 Firm: 10.02E01 Ser#:      WD-WCANK6722813
            Type: Hard Disk
            Supports 48-bit addressing
            Capacity: 239372.4 MB = 233.7 GB (490234752 x 512)
            Speed: 1.5Gbit/s

VM6250 => sata part 1

Partition Map for SATA device 1  --   Partition Type: DOS

Partition     Start Sector     Num Sectors     Type
    1              63          401562    83
    2          401625       489821850    8e
VM6250 => sata device

SATA device 0: Model:  Firm:  Ser#:
            Type: # 1F #
            Capacity: not available
            Speed: 1.5Gbit/s
VM6250 => sata device 1

SATA device 1: Model: WDC WD2500YD-01NVB1 Firm: 10.02E01 Ser#:      WD-WCANK6722813
            Type: Hard Disk
            Supports 48-bit addressing
            Capacity: 239372.4 MB = 233.7 GB (490234752 x 512)
            Speed: 1.5Gbit/s
... is now current device
VM6250 => sata device

SATA device 1: Model: WDC WD2500YD-01NVB1 Firm: 10.02E01 Ser#:      WD-WCANK6722813
            Type: Hard Disk
            Supports 48-bit addressing
            Capacity: 239372.4 MB = 233.7 GB (490234752 x 512)
            Speed: 1.5Gbit/s

VM6250 => sata part


Partition Map for SATA device 1  --   Partition Type: DOS

Partition     Start Sector     Num Sectors     Type
    1              63          401562    83
    2          401625       489821850    8e
```

6.10.13 usb - USB sub-system

```
VM6250 => help usb
usb reset                  - reset (rescan) USB controller
usb stop [f]               - stop USB [f]=force stop
usb tree                   - show USB device tree
usb info [dev]             - show available USB devices
usb storage                - show details of USB storage devices
usb dev [dev]              - show or set current USB storage device
usb part [dev]             - print partition table of one or all USB storage devices
usb read addr blk# cnt
             - read ‘cnt’ blocks starting at block ‘blk#’to memory address ‘addr’
```

Examples:

```
VM6250 => usb start
(Re)start USB...
USB:   scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
VM6250 => usb tree

Device Tree:
  1  Hub (480MBit/s, 0mA)
  |  Philips EHCI
  |
  | +-2  Mass Storage (480MBit/s, 100mA)
       SMART eUSB 20090921048BAED4

  3  Hub (480MBit/s, 0mA)
     Philips EHCI

VM6250 => usb part
print_part of 0

Partition Map for USB device 0  --   Partition Type: DOS

Partition     Start Sector     Num Sectors     Type
    1           63   31457216                  c

print_part of 1
## Unknown partition table

print_part of 2
## Unknown partition table

print_part of 3
## Unknown partition table

print_part of 4
## Unknown partition table
```

```
VM6250 => usb info
1: Hub,  USB Revision 2.0
 - Philips EHCI
 - Class: Hub
 - PacketSize: 64  Configurations: 1
 - Vendor: 0x1131  Product 0x1562 Version 1.0
   Configuration: 1
   - Interfaces: 1 Self Powered 0mA
     Interface: 0
     - Alternate Setting 0, Endpoints: 1
     - Class Hub
     - Endpoint 1 In Interrupt MaxPacket 8 Interval 255ms

2: Mass Storage,  USB Revision 2.0
 - SMART eUSB 20090921048BAED4
 - Class: (from Interface) Mass Storage
 - PacketSize: 64  Configurations: 1
 - Vendor: 0x0e39  Product 0x2b00 Version 135.79
   Configuration: 1
   - Interfaces: 1 Bus Powered 100mA
     Interface: 0
     - Alternate Setting 0, Endpoints: 2
     - Class Mass Storage, Transp. SCSI, Bulk only
     - Endpoint 1 In Bulk MaxPacket 512
     - Endpoint 2 Out Bulk MaxPacket 512

3: Hub,  USB Revision 2.0
 - Philips EHCI
 - Class: Hub
 - PacketSize: 64  Configurations: 1
 - Vendor: 0x1131  Product 0x1562 Version 1.0
   Configuration: 1
   - Interfaces: 1 Self Powered 0mA
     Interface: 0
     - Alternate Setting 0, Endpoints: 1
     - Class Hub
     - Endpoint 1 In Interrupt MaxPacket 8 Interval 255ms

VM6250 => usb stor
  Device 0: Vendor: SMART    Rev: 874F Prod: eUSB
            Type: Hard Disk
            Capacity: 15360.0 MB = 15.0 GB (31457280 x 512)
```

6.10.14 usbboot - boot from USB device

```
VM6250 => help usbboot
usbboot loadAddr dev:part
VM6250 =>
```

6.11 Miscellaneous Commands

6.11.1 date - get/set/reset date & time

```
VM6250 =>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
VM6250 =>
```

6.11.2 dtt - Digital Thermometer and Thermostat

```
VM6250 =>help dtt
Read temperature from digital thermometer and thermostat.
VM6250 =>
```

Example

```
VM6250 => dtt

DTT: CPU sensors
DTT1: 34 C
DTT2: 46 C (55)

DTT: BOARD sensors
DTT1: 30 C
DTT2: 26 C
DTT3: 32 C
VM6250 =>
```

6.11.3 echo - echo args to console

```
VM6250 =>help echo
echo [args..]              - echo args to console; \c suppresses newline
VM6250 =>
```

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

6.11.4 exit script

```
VM6250 => help exit
exit     - exit functionality
```

6.11.5 itest - return true/false on integer compare

```
VM6250 =>help itest
itest [.b, .w, .l, .s] [*]value1 <op> [*]value2
VM6250 =>
```

6.11.6 reset - perform reset of the CPU

The **reset** command reboots the system.

6.11.7 sleep - delay execution for some time

```
VM6250 =>help sleep
sleep N       - delay execution for N seconds (N is _decimal_ !!!)
VM6250 =>
```

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

6.11.8 test - minimal test like /bin/sh

```
VM6250 => help test
test [args..]              - test functionality
VM6250 =>
```

6.11.9 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**)

6.11.10 ? - alias for 'help'

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

```
VM6250 => ?
?       - alias for 'help'
autoscr - run script from memory
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootelf - Boot from an ELF image in memory
bootline - print/set/default vxworks bootrom parameters
bootm   - boot application image from memory
bootp   - boot image via network using BootP/TFTP protocol
bootvx  - Boot vxWorks from an ELF image
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
date    - get/set/reset date & time
dcache  - enable or disable data cache
diag    - perform board diagnostics
dtt     - Digital Thermometer and Thermostat
echo    - echo args to console
erase   - erase FLASH memory
exit    - exit script
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
fdt     - flattened device tree utility commands
flinfo  - print FLASH memory information
go      - start application at address 'addr'
help    - print online help
i2c     - I2C sub-system
icache  - enable or disable instruction cache
icrc32  - checksum calculation
iloop   - infinite loop on address range
imd     - i2c memory display
iminfo  - print header information for application image
imls    - list all images found in flash
imm     - i2c memory modify (auto-incrementing)
imw     - memory write (fill)
imxtract- extract a part of a multi-image
inm     - memory modify (constant address)
iprobe  - probe to discover valid I2C chip addresses
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mii     - MII utility commands
mm      - memory modify (auto-incrementing)
mtest   - simple RAM test
mw      - memory write (fill)
nfs     - boot image via network using NFS protocol
nm      - memory modify (constant address)
```

```
pci     - list and access PCI Configuration Space
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
protect - enable or disable FLASH write protection
rarpboot- boot image via network using RARP/TFTP protocol
reginfo - print register information
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
sata    - SATA sub system
saveenv - save environment variables to persistent storage
setenv  - set environment variables
sleep   - delay execution for some time
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
uboot   - manage U-Boot updates
usb     - USB sub-system
usbboot - boot from USB device
version - print monitor version
vpdutil - VPD EEPROM utility
VM6250 =>
```

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.