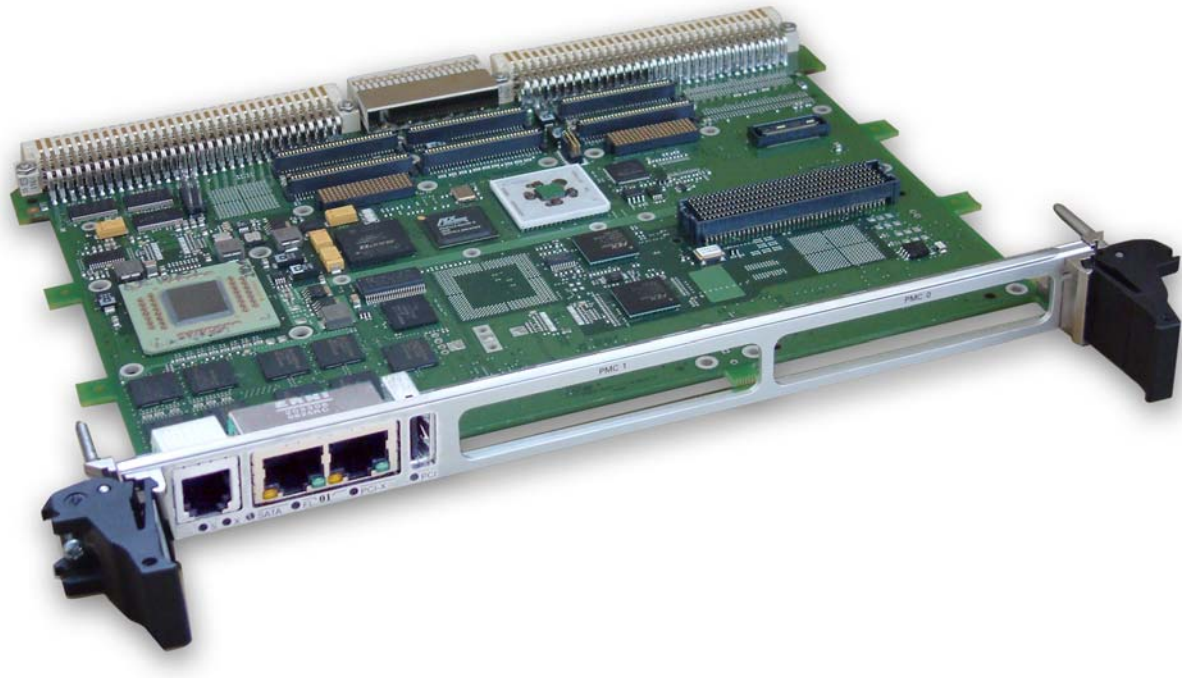
» VM6250 «



PBIT User's Guide

SD.DT.F35-2e - March 2012

Revision History

Publication Title:	VM6250 PBIT User's Guide	
Doc. ID:	SD.DT.F35-2e	
Rev.	Brief Description of Changes	Date of Issue
2e	New release V1.5 ID 12081 Add of section 2.11 “ByPass PBIT Execution by GPIO”	03-2012
1e	New release V1.4 ID 10077	03-2010
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 - PBIT Overview

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

This document describes the PowerOn Built In Test (PBIT) for Kontron VM6250 boards.

PBIT is a key product available under the VM6250 U-Boot 1.3.3 firmware environment.

PBIT is implemented as a binary executable file located in the System Flash and included in the U-Boot firmware image.

PBIT product includes among others the following services:

- a list of tests can be added or removed from a run list by a command according to the desired compromise between time to boot and coverage rate,
- the run list can be configured to be executed automatically or not after a power ON, a normal reset or a watchdog reset,
- PBIT can be run automatically at boot time or in an interactive mode under U-Boot firmware prompt,
- for a specific customized configuration, it is possible, to extend the PBIT, creating a new test and adding these new test to a new U-Boot image.
- tests configurations and results are stored in the EEPROM I2C and can be accessed and reconfigured under an Operating System such as Linux or VxWorks.

1.1 Related Documents

Kontron Documentation

Hardware

- VM6250 6U VME SBC User's Guide CA.DT.A65
- VM6250 Hardware Release Notes CA.DT.A66

Firmware

- VM6250 U-Boot User's Manual SD.DT.F36

DENX Software Engineering Documentation

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

1.2 PBIT Release Content

The PBIT release is distributed in one DVD-ROM referenced:

FW−VM6250 BIT V1.5 ID12081
PBIT Firmware for VM6250

This DVD-ROM includes:

- This User's Guide: SD.DT.F35.
- A U-Boot image containing the PBIT image to download into the VM6250 board System Flash.
- The PBIT Development Tool Kit release providing the necessary support for extending the PBIT test components.
- An ISO image of the Embedded Linux Development Kit (ELDK) version 4.2. It includes the GNU cross development tools, such as the compilers, binutils, gdb, etc., and a number of pre-built target tools and libraries necessary to provide some functionality on the target system.

1.3 PBIT Installation and Activation

1.3.1 PBIT Installation and Activation

The PBIT software comes pre−installed in the system Flash, along with the U-Boot firmware on the VM6250 boards.

PBIT is a key product that can be activated on any VM6250 board with a specific U-Boot command. Please contact Kontron support for more information.

To install a new U-Boot version including new PBIT version, please consult the VM6250 U-Boot User Reference Manual - SD.DT.F36.

When PBIT is activated, no test is configured to run automatically at system start-up.

1.3.2 DVD-ROM Installation

The PBIT DVD-ROM release must be installed in order to develop, compile and install a new PBIT test.

To add a new test, copy the DVD-ROM content on a Linux machine where the U-Boot compiler will run and install it in your home directory.

The DVD-ROM contains also the following files and directories:

- **ppc-2008-04-01_freescale.iso** eldk-4.2 ELDK=Embedded Linux Development Kit from the DENX project
- **pbit_VX3230-*YYDDD*.tgz** . . . . with *YYDDD* = current release ID
- **SD.DT.F35-1e.pdf** . . . . . current file

Refer to Chapter 3 “Customized PBIT Test Developing” page 25 for all details about Tool Kit installation and configuration.

1.4 PBIT Run List Mode Concept

PBIT is programmable to be run in three differents run modes

- ROM Run Mode
- RAM Run Mode
- COMMAND Run Mode

Each Run Mode is associated to a Run List:

- One run list usable when U-Boot is running from the ROM.
- One run list usable when U-Boot is relocated into the RAM and before the U-Boot prompt is displayed.
- One run list usable when a run command is entered.

These three Run Modes are fully described in the following sections.

1.4.1 ROM Run Mode

Several PBIT tests can be programmed to be run automatically when U-Boot is running from the ROM (for example, RAM test or U-Boot System Flash validity test).

The ROM Mode is used to debug a board that cannot start correctly, but is not recommended in custom configuration.

To display the tests that are programmed to run in ROM Mode, enter the following command:

```
VM6250 => diag rom
```

1.4.2 RAM Run Mode

RAM tests are all the tests available to run automatically once U-Boot is relocated in RAM and before the U-Boot User's prompt is displayed or the boot command (**bootcmd**) is launched.

All existing tests can be configured to be run in RAM Mode.

To display the tests that are programmed to run in RAM Mode, enter the following command:

```
VM6250 => diag ram
```

1.4.3 COMMAND Run Mode

COMMAND tests are the test that are programmed to be executed only at a user request. This is done by entering the command `diag run` or `diag run <test number | test name>`.

All existing tests can be configured to run in COMMAND Mode.

To display the tests that are programmed to run in COMMAND Mode and other tests that are available but not yet programmed to run in COMMAND Mode, enter:

```
VM6250 => diag command
```

or simply

```
VM6250 => diag
```

1.5 PBIT Tests List

1.5.1 Default Tests List

When PBIT are available on VM6250, run the command `diag` to display the complete tests list:

```
VM6250 => diag
```

The list of the available tests and their configuration parameter is displayed below. All the tests are configured to be in the COMMAND Run list by default when PBIT are activated.

```
mem_data (0) - Checks Memory/ECC data lines
mem_addr (1) - Checks Memory/ECC address lines
mem_pattern1 (2) - Checks Memory/ECC using pattern 0x00000000
mem_pattern2 (3) - Checks Memory/ECC using pattern 0xFFFFFFFF
mem_pattern3 (4) - Checks Memory/ECC using pattern 0x55555555
mem_pattern4 (5) - Checks Memory/ECC using pattern 0xAAAAAAAA
mem_bitflip (6) - Checks Memory/ECC using bit-flip pattern ((1 << (offset % 32))
mem_addrpat (7) - Checks Memory/ECC using address pattern (offset)
mem_addrpat2 (8) - Checks Memory/ECC using address pattern (~offset)
cpu0 (10) - Checks CPU0
cpu1 (11) - Checks CPU1
pcieswitch (12) - Checks the pciExpress switch
pciepci64switch (13) - Checks the pciExpress to PCI64 (PMC site A) switch
pciepci32switch (14) - Checks  PCIe/PCI32 switchs (PMCB/ALMA2e/USB)
serial (16) - Checks the serial line 1
battery (20) - Checks battery.
rtc (21) - Checks the RTC is running.
sysflash (22) - Checks the uboot and uboot rescue checksum in system flash
cpld (24) - Checks System CPLD Configuration
temp_sensors (31) - Checks if all temperature sensors are detected.
temperature (32) - Checks if temperatures are OK.
nvsram (40) - Checks NVsRAM device.
ether_loop0 (54) - Checks Ethernet 0 in Loopback mode
ether_loop1 (55) - Checks Ethernet 1 in Loopback mode
ether_loop2 (56) - Checks Ethernet 2 in Loopback mode
ether_loop3 (57) - Checks Ethernet 3 in Loopback mode
sata0_test (67) - Checks sata0 controller
sata1_test (68) - Check sata1 controller
eeprom_vpd (71) - Checks EEPROM VPD content is correct
watchdog (72) - Checks watchdog timer and frequency
vme (74) - Checks ALMA VME device access
pmcAxmc_check (76) - Checks PMCA 64bits/XMC path & slot
pmcBxmc_check (78) - Checks PMCB 32bits/XMC path & slot
usb0_controller (84) - Checks usb0 controller
usb1_controller (85) - Checks usb1 controller
```

To run the default PBIT COMMAND Run list, enter:

```
VM6250 => diag run
```

1.5.2 System Configuration Dependent Tests List

The following tests are considered as system configuration dependent and are not included in the default PBIT COMMAND Run list.

These tests are displayed at the end of the **diag** command after the message: **Other PBITs available but not yet configured**:

```
 Other PBITs available but not yet configured
ether_lnk0 (50) - Checks the link status is "UP" on Ethernet 0 interfaces.
ether_lnk1 (51) - Checks the link status is "UP" on Ethernet 1 interfaces.
ether_lnk2 (52) - Checks the link status is "UP" on Ethernet 2 interfaces.
ether_lnk3 (53) - Checks the link status is "UP" on Ethernet 3 interfaces.
 sata_0_detected (65) - Checks if a sata0 disk is present
 sata_1_detected (66) - Checks if a sata1 disk is present
 pmcAxmc_present (77) - Checks if device is present on PMCA 64bits/XMC slot
 pmcBxmc_present (79) - Checks if device is present on PMCA 32bits/XMC slot
 usb_0_dev_see (80) - Checks if a usb 0 device is present
 usb_1_dev_see (81) - Checks if a usb 1 device is present
 usb_2_dev_see (82) - Checks if a usb 2 device is present on P0
 usb_3_dev_see (83) - Checks if a usb 3 device is present on P0
 faultytest (98) - A dummy test that returns FAIL
 hangtest (99) - A dummy test that will hang
```

Tests 98 and 99 help to create some faulty test case and must not be configured at all in normal case.

1.6 PBIT Executive Time

The default PBIT run list lasts 1 minute 11 seconds for a 2 GB of RAM. This does not include the system dependent configuration tests.

Here is the detail of executive time for all tests. The tests that have a significant duration compared to other tests appear in bold red.

PBIT "mem_data"	(cpu0,slow,simple)	141 ms
PBIT "mem_addr"	(cpu0,slow,simple)	273 ms
PBIT "mem_pattern1"	(cpu0,slow,simple)	**9.107 s**
PBIT "mem_pattern2"	(cpu0,slow,simple)	**9.107 s**
PBIT "mem_pattern3"	(cpu0,slow,simple)	**9.107 s**
PBIT "mem_pattern4"	(cpu0,slow,simple)	**9.107 s**
PBIT "mem_bitflip"	(cpu0,slow,simple)	**9.107 s**
PBIT "mem_addrpat"	(cpu0,slow,simple)	**9.107 s**
PBIT "mem_addrpat2"	(cpu0,slow,simple)	**9.107 s**
PBIT "cpu0"	(cpu0,fast,simple)	136 ms
PBIT "cpu1"	(cpu1,fast,simple)	136 ms
PBIT "pcieswitch"	(cpu0,fast,simple)	18ms
PBIT "pciepci64switch"	(cpu0,fast,simple)	300 ms
PBIT "pciepci32switch"	(cpu0,fast,simple)	31 ms
PBIT "serial"	(cpu0,fast,simple)	155 ms
PBIT "battery"	(cpu0,fast,simple)	33 ms
PBIT "rtc"	(cpu0,fast,simple)	940 ms
PBIT "sysflash"	(cpu0,fast,simple)	1.173 s
PBIT "cpld"	(cpu0,fast,simple)	122 ms
PBIT "temp_sensors"	(cpu0,fast,simple)	21 ms
PBIT "temperature"	(cpu0,fast,simple)	22 ms
PBIT "ether_lnk0"	(cpu0,fast,simple)	42 ms
PBIT "ether_lnk1"	(cpu0,fast,simple)	42 ms
PBIT "ether_lnk2"	(cpu0,fast,simple)	30 ms
PBIT "ether_lnk3"	(cpu0,fast,simple)	30 ms
PBIT "ether_loop0"	(cpu0,fast,simple)	**3.4 s**
PBIT "ether_loop1"	(cpu0,fast,simple)	**3.4 s**
PBIT "ether_loop2"	(cpu0,fast,simple)	**3.4 s**
PBIT "nvsram"	(fast,simple)	24 ms
PBIT "ether_loop3"	(cpu0,fast,simple)	**3.4 s**
PBIT "sata_0_detected"	(cpu0,fast,simple)	31 ms
PBIT "sata_1_detected"	(cpu0,fast,simple)	31 ms
PBIT "sata0_test"	(cpu0,fast,simple)	50 to 300 ms
PBIT "sata1_test"	(cpu0,fast,simple)	50 to 300 ms
PBIT "eeprom_vpd"	(cpu0,fast,simple)	69 ms
PBIT "watchdog"	(cpu0,fast,simple)	148 ms

PBIT "vme"	(cpu0,fast,simple)	27 ms
PBIT "pmcAxmc_check"	(cpu0,fast,simple)	19 ms
PBIT "pmcAxmc_present"	(cpu0,fast,simple)	19 ms
PBIT "pmcBxmc_check"	(cpu0,fast,simple)	33 ms
PBIT "pmcBxmc_present"	(cpu0,fast,simple)	28 ms
PBIT "usb_0_dev_see"	(cpu0,fast,simple)	**The first USB test require** 6.5 seconds (device init) then other tests require around 0.8 s
PBIT "usb_1_dev_see"	(cpu0,fast,simple)	~ 80 ms
PBIT "usb_2_dev_see"	(cpu0,fast,simple)	~ 80 ms
PBIT "usb_3_dev_see"	(cpu0,fast,simple)	~ 80 ms
PBIT "usb0_controller"	(cpu0,fast,simple)	37 ms
PBIT "usb1_controller"	(cpu0,fast,simple)	37 ms

To have the best coverage rate and to reduce the boot time delay, it is advised to remove unused ethernet interface tests and eventually RAM tests `mem_pattern1` to `mem_bitflip` because `mem_data` + `mem_addr` (testing all data lines and address lines) + `mem_addrpat` and `mem_addrpat2` (testing the full RAM using the ram address as data and then the Complement of RAM address as data) are considered as having a sufficient and quasi complete RAM testing coverage.

Chapter 2 - PBIT Command Line Interface

PBIT are configured and executed using the U-Boot command « `diag` ».

The following sections explain the various `diag` command parameters.

2.1 On-line Help

Under U-Boot prompt, enter `help diag` to display the full on-line help.

In command format

- `[ ]` means optional parameters
- `|` means a OR choice between several parameters
- `...` means an undetermined number of repeated previous parameters

```
VM6250 => help diag

diag

  ----- Usage -----

Print list of PBITs and infos about them :

  diag [rom|ram|command] [<PBITname>|<PBITnum> ...]

    rom         : Display PBIT(s) that are run automatically from ROM

                  (before relocation to RAM)

    ram         : Display PBIT(s) that are run automatically from ROM

                  (after relocation to RAM)

    command     : Display PBIT(s) that are run from command line

    (default)     (diag command)


    <PBITname>|<PBITnum> ...

              list of PBIT(s) to display. All if the list is empty.

              PBIT(s) are referenced using their name or their number.
```

```
Run PBIT(s) from command line :
  diag run [loop <count>] [<PBITname>|<PBITnum> ...]

    loop <count>
              run PBIT(s) <count> times instead of once

    <PBITname>|<PBITnum> ...
              list of PBIT(s) to run. All if the list is empty.
              PBIT(s) are referenced using their name or their number.
Print PBIT(s) status :
  diag stat [<PBITname>|<PBITnum> ...]
    <PBITname>|<PBITnum> ...
              list of PBIT(s) to display. All if the list is empty.
              PBIT(s) are referenced using their name or their number.

Clear PBIT(s) status :
  diag [rom|ram|command] clrstat|clrallstat [<PBITname>|<PBITnum> ...]

    rom        : Clear status for PBIT(s) that are run automatically from ROM
                 (before relocation to RAM)
    ram        : Clear status for PBIT(s) that are run automatically from ROM
                 (after relocation to RAM)
    command    : Clear status for PBIT(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

    <PBITname>|<PBITnum> ...
              list of PBIT(s) to clear. All if the list is empty.
              PBIT(s) are referenced using their name or their number.

Restore default PBIT configuration :
  diag default
```

```
Configure PBIT(s) :

  diag [rom|ram|command] cfg <cfgarg> ... <PBITname>|<PBITnum> ...

    rom         : Configure PBIT(s) that are run automatically from ROM
                  (before relocation to RAM)
    ram         : Configure PBIT(s) that are run automatically from ROM
                  (after relocation to RAM)
    command     : Configure PBIT(s) that are run from command line
    (default)     (diag command)

    cfg <cfgarg> : Configure one or several PBIT(s).
                   <cfgarg> is either :
                   - "delete" to delete PBIT(s) from the list of configured PBITs
                   - "default" to configure PBIT(s) with a default run mode
                   - a comma separated list of runflags defining a PBIT
                     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

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


    [<PBITname>|<PBITnum>] ...

                    list of PBIT(s) to configure.

                    PBIT(s) are referenced using their name or their number.

                    All missing tests if the list is empty (but the cfgarg is default).


Toggle PBIT running log information:

  diag silent


Display PBIT version :

  diag version

VM6250 =>
```

2.2 Display the List of Configured Tests

To display both the existing tests list and their configuration in the ROM, RAM or COMMAND Run lists, use the following command:

```
diag [rom|ram|command] [<PBITname>|<PBITnum> ...]
```

- `diag` or `diag command` prints the list of tests usable with the **run** parameter
- `diag ram` prints the list of tests texecuted automatically when U-Boot is starting and is executed from RAM but before U-Boot command prompt.
- `diag rom` prints the list of tests executed when U-Boot is running from ROM

Example:

```
VM6250 => diag

PBITs configured to run from command line :

 mem_data (0) - Checks Memory/ECC data lines

   capabilities : cpu0/cpu1,fast,simple

   run mode 1   : cpu0,fast,simple

 mem_addr (1) - Checks Memory/ECC address lines

   capabilities : cpu0/cpu1,fast,simple

   run mode 1   : cpu0,fast,simple

 mem_pattern1 (2) - Checks Memory/ECC using pattern 0x00000000

   capabilities : cpu0/cpu1,slow/fast,simple

   run mode 1   : cpu0,slow,simple

 mem_pattern2 (3) - Checks Memory/ECC using pattern 0xFFFFFFFF

   capabilities : cpu0/cpu1,slow/fast,simple

   run mode 1   : cpu0,slow,simple

 mem_pattern3 (4) - Checks Memory/ECC using pattern 0x55555555

   capabilities : cpu0/cpu1,slow/fast,simple

   run mode 1   : cpu0,slow,simple

 mem_pattern4 (5) - Checks Memory/ECC using pattern 0xAAAAAAAA

   capabilities : cpu0/cpu1,slow/fast,simple

   run mode 1   : cpu0,slow,simple

...
```

```
...
 Other PBITs available but not yet configured :
 ether_lnk0 (50) - Checks the link status is "UP" on Ethernet 0 interfaces.
   capabilities : cpu0/cpu1,fast,simple
 ether_lnk1 (51) - Checks the link status is "UP" on Ethernet 1 interfaces.
   capabilities : cpu0/cpu1,fast,simple
 ether_lnk2 (52) - Checks the link status is "UP" on Ethernet 2 interfaces.
   capabilities : cpu0/cpu1,fast,simple
 ether_lnk3 (53) - Checks the link status is "UP" on Ethernet 3 interfaces.
   capabilities : cpu0/cpu1,fast,simple
 sata_0_detected (65) - Checks if a sata0 disk is present
   capabilities : cpu0/cpu1,fast,simple
 sata_1_detected (66) - Checks if a sata1 disk is present
   capabilities : cpu0/cpu1,fast,simple
 pmcAxmc_present (77) - Checks if device is present on PMCA 64bits/XMC slot
   capabilities : cpu0/cpu1,fast,simple/complex
 pmcBxmc_present (79) - Checks if device is present on PMCB 32bits/XMC slot
   capabilities : cpu0/cpu1,fast,simple/complex
 usb_0_dev_see (80) - Checks if a usb 0 device is present
   capabilities : cpu0/cpu1,fast,simple
 usb_1_dev_see (81) - Checks if a usb 1 device is present
   capabilities : cpu0/cpu1,fast,simple
 usb_2_dev_see (82) - Checks if a usb 2 device is present on P0
   capabilities : cpu0/cpu1,fast,simple
 usb_3_dev_see (83) - Checks if a usb 3 device is present on P0
   capabilities : cpu0/cpu1,fast,simple
 faultytest (98) - A dummy test that returns FAIL
   capabilities : cpu0/cpu1,fast,simple
 hangtest (99) - A dummy test that will hang
   capabilities : cpu0/cpu1,fast,simple


Use ´help diag´ to get more info.
```

2.3 Execute PBIT from the Command Line

To run the PBIT list that is programmed to be launched from the command line, enter:

```
VM6250 => diag run

PBIT "mem_data" (cpu0,fast,simple)  PASSED
PBIT "mem_addr" (cpu0,fast,simple)  PASSED
PBIT "mem_pattern1" (cpu0,slow,simple)  PASSED
PBIT "mem_pattern2" (cpu0,slow,simple)  PASSED
PBIT "mem_pattern3" (cpu0,slow,simple)  PASSED
PBIT "mem_pattern4" (cpu0,slow,simple)  PASSED
PBIT "mem_bitflip" (cpu0,slow,simple)  PASSED
PBIT "mem_addrpat" (cpu0,slow,simple)  PASSED
PBIT "mem_addrpat2" (cpu0,slow,simple)  PASSED
PBIT "cpu0" (cpu0,fast,simple)  PASSED
PBIT "cpu1" (cpu1,fast,simple)  PASSED
PBIT "pcieswitch" (cpu0,fast,simple)  PASSED
PBIT "pciepci64switch" (cpu0,fast,simple)  PASSED
PBIT "pciepci32switch" (cpu0,fast,simple)  PASSED
PBIT "serial" (cpu0,fast,simple)  PASSED
PBIT "battery" (cpu0,fast,simple)   NOT EQUIPPED PASSED
PBIT "rtc" (cpu0,fast,simple)  PASSED
PBIT "sysflash" (cpu0,fast,simple)  PASSED
PBIT "cpld" (cpu0,fast,simple)  PASSED
PBIT "temp_sensors" (cpu0,fast,simple)  PASSED
PBIT "temperature" (cpu0,fast,simple)  PASSED
PBIT "nvsram" (fast,simple) PASSED
PBIT "ether_loop0" (cpu0,fast,simple)  PASSED
PBIT "ether_loop1" (cpu0,fast,simple)  PASSED
PBIT "ether_loop2" (cpu0,fast,simple)  PASSED
PBIT "ether_loop3" (cpu0,fast,simple)  PASSED
PBIT "sata0_test" (cpu0,fast,simple)  PASSED
PBIT "sata1_test" (cpu0,fast,simple)  PASSED
PBIT "eeprom_vpd" (cpu0,fast,simple)  PASSED
PBIT "watchdog" (cpu0,fast,simple)  PASSED
PBIT "vme" (cpu0,fast,simple)  PASSED
PBIT "pmcAxmc_check" (cpu0,fast,simple)  PASSED
PBIT "pmcBxmc_check" (cpu0,fast,simple)  PASSED
PBIT "usb0_controller" (cpu0,fast,simple)  PASSED
PBIT "usb1_controller" (cpu0,fast,simple)  PASSED

VM6250 =>
```

To run a single test or a limited serie of tests, enter:

```
VM6250 => diag run <PBIT number | PBIT name ...>
```

For example:

```
VM6250 => diag run usb0_controller
PBIT "usb0_controller" (cpu0,fast,simple)  PASSED
```

is equivalent to

```
VM6250 => diag run 84
PBIT "usb0_controller" (cpu0,fast,simple)  PASSED
```

2.4 Execute PBIT in Loop Mode

To run all PBIT in loop mode, enter:

```
VM6250 => diag run loop <count>
```

where **count** is the number of loop to do.

Example:

```
VM6250 => diag run loop 10
```

To run a single test in loop mode, enter:

```
VM6250 => diag run loop <count> <PBIT number | PBIT name ...>
```

where **count** is the number of loop to do.

Example:

```
VM6250 =>diag run loop 10 54
```

is equivalent to

```
VM6250 => diag run loop 10 ether_loop0
```

2.5 Consult the PBIT Results

The U-BOOT firmware displays the PBIT results by running the **`diag stat`** command:

Here is an example with a faulty test:

```
VM6250 => diag stat
Status of PBITs configured to run automatically from ROM :
 None

Status of PBITs configured to run automatically from RAM :
 None

Status of PBITs configured to run from command line :
 PASSED : mem_data (cpu0,fast,simple)
 PASSED : mem_addr (cpu0,fast,simple)
 PASSED : mem_pattern1 (cpu0,slow,simple)
 PASSED : mem_pattern2 (cpu0,slow,simple)
 PASSED : mem_pattern3 (cpu0,slow,simple)
 PASSED : mem_pattern4 (cpu0,slow,simple)
 PASSED : mem_bitflip (cpu0,slow,simple)
 PASSED : mem_addrpat (cpu0,slow,simple)
 PASSED : mem_addrpat2 (cpu0,slow,simple)
 PASSED : cpu0 (cpu0,fast,simple)
 PASSED : cpu1 (cpu1,fast,simple)
 PASSED : pcieswitch (cpu0,fast,simple)
 PASSED : pciepci64switch (cpu0,fast,simple)
 PASSED : pciepci32switch (cpu0,fast,simple)
 PASSED : serial (cpu0,fast,simple)
 PASSED : battery (cpu0,fast,simple)
 PASSED : rtc (cpu0,fast,simple)
 PASSED : sysflash (cpu0,fast,simple)
 FAILED : cpld (cpu0,fast,simple)
 PASSED : temp_sensors (cpu0,fast,simple)
 PASSED : temperature (cpu0,fast,simple)
 PASSED : nvsram (cpu0,fast,simple)
 PASSED : ether_loop0 (cpu0,fast,simple)
 PASSED : ether_loop1 (cpu0,fast,simple)
 PASSED : ether_loop2 (cpu0,fast,simple)
 PASSED : ether_loop3 (cpu0,fast,simple)
 PASSED : sata0_test (cpu0,fast,simple)
 PASSED : sata1_test (cpu0,fast,simple)
 PASSED : eeprom_vpd (cpu0,fast,simple)
 PASSED : watchdog (cpu0,fast,simple)
 PASSED : vme (cpu0,fast,simple)
 PASSED : pmcAxmc_check (cpu0,fast,simple)
 PASSED : pmcBxmc_check (cpu0,fast,simple)
 PASSED : usb0_controller (cpu0,fast,simple)
 PASSED : usb1_controller (cpu0,fast,simple)

RUN    : 35
PASSED : 34
FAILED : 1
NOT_RUN: 0

VM6250 =>
```

2.6 Clear the PBIT Results

- To clear all PBIT statistics for PBIT executed from COMMAND Mode:

```
VM6250 => diag clrallstat
```

- To clear all PBIT statistics for PBIT executed in RAM Mode:

```
VM6250 => diag ram clrallstat
```

- When a PBIT FAILED, a statistic command is recorded. To keep this information, a reduce clear statistics command can be run:

```
VM6250 => diag clrstat
```

2.7 Configure PBIT Test List

For each one of the three possible run modes (ROM/RAM/COMMAND), the PBIT list to execute can be modified. Each test can be added, removed, and programmed with specific attributes.

The possible specific attributes are defined with the following test flags:

- **CPU Flags**

  cpu0 : run on CPU0

  cpu1 : run on CPU1

  These flags can be mixed to be executed on both CPUs (if possible)

- **SPEED flags (can NOT be mixed)**

  slow : run in slow mode (full testing)

  fast : run in fast mode (fast testing)

  None test implements a difference between fast and slow mode in this PBIT version

- **CONFIG flags (can NOT be mixed)**

  simple : run in simple mode (no external hardware)

  complex : run in complex mode (needs external hardware)

  Complex mode requires a supplementary device and are reserved for Factory test. Do not use this mode.

- **HALT flags (can NOT be mixed)**

  haltonfail : halt immediately (hang) if test fails

  promptonfail : halt at U-Boot prompt (no OS) if test fails

  This flag offers the possibility to halt all tests execution when an error is detected. The promptonfail flags will cause the U-Boot to use the environment variable **failbootcmd** if defined instead of **bootcmd.**

- **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

  This flag offers the possibility to select after what source of reset the PBIT will be automatically executed. It is only significant for test ROM or RAM test list.

2.8 Add/Remove Test in PBIT List to Execute

2.8.1 Adding a Test to the Current Run List

```
VM6250 => diag [rom|ram|command] cfg <cfgarg> ... <PBITname>|<PBITnum> ...
```

`cfgarg` allows to choice the test run mode. Use keyword default to set a default mode (typically CPU0, FAST, SIMPLE, ALLRESETS)

Examples:

To add test 80 (usb_0_dev_see) in default mode in the test list executed by COMMAND, enter:

```
VM6250 => diag command cfg default 80
```

To add this test in list executed automatically when U-Boot run in RAM, enter:

```
VM6250 => diag ram cfg default 80
```

To add this test in RAM list with flags `poweronreset` (and other default flag), enter:

```
VM6250 => diag ram cfg poweronreset 80
```

To add this test in RAM list with flags `poweronreset + promptonfail` , enter:

```
VM6250 => diag ram cfg poweronreset,promptonfail 80
```

Execute `diag ram` or `diag ram 80` to check the configuration:

```
VM6250 => diag ram 80
 usb_0_dev_see (80) - Checks if a usb 0 device is present
   capabilities : cpu0/cpu1,fast,simple,allresets
   run mode 1   : cpu0,fast,simple,promptonfail,poweronreset
```

2.8.2 Removing a Test from the Current Run List

- From the COMMAND list

  Example for removing test number 80

  ```
  VM6250 => diag command cfg delete 80
  ```

- For automatically executed RAM list

  Example for removing test number 80

  ```
  VM6250 => diag ram cfg delete 80
  ```

2.8.3 Adding all Existing Tests in Current Run List

It's possible to add all existing tests in the current run list with a given flag running a single command. The command will include all tests that are not yet configured for the specified mode (ROM/RAM/COMMAND).

To add all existing tests in RAM mode with a default run mode, enter:

```
VM6250 => diag ram cfg default

Configuration of all PBITs in default mode only
List of test to configure
--> mem_data (0) - Checks Memory/ECC data lines
--> mem_addr (1) - Checks Memory/ECC address lines
--> mem_pattern1 (2) - Checks Memory/ECC using pattern 0x00000000
--> mem_pattern2 (3) - Checks Memory/ECC using pattern 0xFFFFFFFF
--> mem_pattern3 (4) - Checks Memory/ECC using pattern 0x55555555
--> mem_pattern4 (5) - Checks Memory/ECC using pattern 0xAAAAAAAA
--> mem_bitflip (6) - Checks Memory/ECC using bit-flip pattern ((1 << (offset % 32))
--> mem_addrpat (7) - Checks Memory/ECC using address pattern (offset)
--> mem_addrpat2 (8) - Checks Memory/ECC using address pattern (~offset)
--> cpu0 (10) - Checks CPU0
--> cpu1 (11) - Checks CPU1
--> pcieswitch (12) - Checks the pciExpress switch
--> pciepci64switch (13) - Checks the pciExpress to PCI64 (PMC site A) switch
--> pciepci32switch (14) - Checks  PCIe/PCI32 switchs (PMCB/ALMA2e/USB)
--> serial (16) - Checks the serial line 1
--> battery (20) - Checks battery.
--> rtc (21) - Checks the RTC is running.
--> sysflash (22) - Checks the uboot and uboot rescue checksum in system flash
--> cpld (24) - Checks System CPLD Configuration
--> temp_sensors (31) - Checks if all temperature sensors are detected.
--> temperature (32) - Checks if temperatures are OK.
--> nvsram (40) - Checks NVsRAM device.
--> ether_lnk0 (50) - Checks the link status is "UP" on Ethernet 0 interfaces.
--> ether_lnk1 (51) - Checks the link status is "UP" on Ethernet 1 interfaces.
--> ether_lnk2 (52) - Checks the link status is "UP" on Ethernet 2 interfaces.
--> ether_lnk3 (53) - Checks the link status is "UP" on Ethernet 3 interfaces.
--> ether_loop0 (54) - Checks Ethernet 0 in Loopback mode
--> ether_loop1 (55) - Checks Ethernet 1 in Loopback mode
--> ether_loop2 (56) - Checks Ethernet 2 in Loopback mode
--> ether_loop3 (57) - Checks Ethernet 3 in Loopback mode
--> sata_0_detected (65) - Checks if a sata0 disk is present
--> sata_1_detected (66) - Checks if a sata1 disk is present

--> sata0_test (67) - Checks sata0 controller
--> sata1_test (68) - Check sata1 controller
--> eeprom_vpd (71) - Checks EEPROM VPD content is correct
--> watchdog (72) - Checks watchdog timer and frequency
--> vme (74) - Checks ALMA VME device access
--> pmcAxmc_check (76) - Checks PMCA 64bits/XMC path & slot
--> pmcAxmc_present (77) - Checks if device is present on PMCA 64bits/XMC slot
--> pmcBxmc_check (78) - Checks PMCB 32bits/XMC path & slot
--> pmcBxmc_present (79) - Checks if device is present on PMCA 32bits/XMC slot
--> usb_0_dev_see (80) - Checks if a usb 0 device is present
--> usb_1_dev_see (81) - Checks if a usb 1 device is present
--> usb_2_dev_see (82) - Checks if a usb 2 device is present on P0
--> usb_3_dev_see (83) - Checks if a usb 3 device is present on P0
--> usb0_controller (84) - Checks usb0 controller
--> usb1_controller (85) - Checks usb1 controller
VM6250 =>
```

To add all test in one command in ROM mode with **poweronreset** flags, enter:

```
VM6250 => diag rom cfg poweronreset

Configuration of all PBITs in default mode only

List of test to configure
--> mem_data (0) - Checks Memory/ECC data lines
--> mem_addr (1) - Checks Memory/ECC address lines
--> sysflash (22) - Checks the uboot and uboot rescue checksum in system flash

VM6250 => diag rom

PBITs configured to run automatically from ROM :
 mem_data (0) - Checks Memory/ECC data lines
   capabilities : cpu0,fast,simple,allresets
   run mode 1   : cpu0,fast,simple,poweronreset
 mem_addr (1) - Checks Memory/ECC address lines
   capabilities : cpu0,fast,simple,allresets
   run mode 1   : cpu0,fast,simple,poweronreset
 sysflash (22) - Checks the uboot and uboot rescue checksum in system flash
   capabilities : cpu0,fast,simple,allresets
   run mode 1   : cpu0,fast,simple,poweronreset

Other PBITs available but not yet configured :

 None

Use ´help diag´ to get more info.

VM6250 =>
```

The command will not include test number greater than 95 because these tests are considered as being tool test case for PBIT validation.

2.8.4 Restoring Default Run List

For COMMAND list, enter:

```
VM6250 => diag command default
```

or

```
VM6250 => diag default
```

To restore automatically executed RAM test list, enter:

```
VM6250 => diag ram default
```

Since none test is configured by default in RAM mode, this command remove all tests from RAM mode test list.

2.9 Run Pbit in Silencious Mode

To avoid PBIT displaying any test message execution and error report, use the toggle command:

```
VM6250 =>  diag pbit silent
PBIT set in silent mode
VM6250 =>
```

To remove the silent mode, re-execute the same command:

```
VM6250 => diag pbit silent
PBIT silent mode removed
VM6250 =>
```

2.10 Display PBIT version

Following PBIT command displays the PBIT version:

```
VM6250 => diag version
PBIT VERSION 1.2
VM6250 =>
```

2.11 ByPass PBIT Execution by GPIO

PBIT execution bypass can be controlled by GPIO. When PBIT are configured to run they can be bypassed according to a GPIO level programming.

To bypass PBIT execution through a GPIO, define the 2 following UBOOT environment variables `'activeGPIOpbit'` and `'levelGPIOpbit'`.

- `activeGPIOpbit` Activate the GPIO bypass functionality and Set the GPIO line (1 to 3).
- `levelGPIOpbit` Set the activation level 0 or 1. PBIT will be executed only when the GPIO logical level corresponds to the value of `levelGPIOpbit`

Example:

To set the GPIO 2 on a Low level :

```
VM6250 => setenv  activeGPIOpbit 2
VM6250 => setenv  levelGPIOpbit  0
```

Then PBIT will be bypassed when GPIO 2 logical level is 1 (high) and will be executed when GPIO 2 logical level is 0 (low)

To disable this functionality, remove the environment value:

```
VM6250 => setenv  activeGPIOpbit
VM6250 => setenv  levelGPIOpbit
```

Chapter 3 - Customized PBIT Test Developing

The PBIT package allows to build and add in the U-Boot image any customized PBIT test. For example, a test for a specific XMC/PMC or any other device.

Thereby, U-Boot libraries and Makefile are delivered in the PBIT Tool Kit. To be available, the new customized test will just need to be linked with the U-Boot libraries.

The user needs:

- to install the ELDK GNU compiler & tools on a Linux machine, (ELDK = Embedded Linux Development Kit from the DENX project) ,
- install the PBIT Tool Kit,
- add a **newtest.c** file,
- modify the PBIT Makefile
- run a **make** command to build the new U-Boot image including this new test.

Refer to the following sections for a detailed description of:

- the ELDK 4.2 Gnu tools installation,
- the PBIT Tool Kit installation,
- the process to add a new test and create a new U-Boot image.

3.1 Supported Hosts Systems

The development environment is a GNU cross environment built for **target=powerpc-linux** on **host=i686-pc-linuxgnu**, that can be installed for example in directory **/opt/eldk-4.2** .

DENX indicates the following hosts systems can support the ELDK U-Boot compiler (see also http://www.denx.de/wiki/view/DULG/ELDKSupportedHostSystems)

The ELDK can be installed onto and operate with the following operating systems:

- Fedora Core 4, 5, 6 Fedora 7, 8, 9, 10
- Red Hat Linux 7.3, 8.0, 9
- SuSE Linux 8.x, 9.0, 9.1, 9.2, 9.3
- OpenSUSE 10.2, 10.3 (32 Bit); OpenSUSE 11.0 (32 and 64 Bit)
- Debian 3.0 (Woody), 3.1 (Sarge) and 4.0 (Etch)
- Ubuntu 4.10, 5.04, 6.10, 8.04
- FreeBSD 5.0

Users also reported successful installation and use of the ELDK on the following host systems:

- Suse Linux 7.2, 7.3
- Mandrake 8.2
- Slackware 8.1beta2
- Gentoo Linux 2006.1

3.2 Installing the ELDK U-Boot Compiler

To install the ELDK compiler you can follow what is indicated on DENX web site and download the compiler from this site or use the one provided in this release.

If you use the ELDK compiler provided in this release, then follow the process decribed below:

- Copy the iso image to `/tmp` directory, for example:

```
$ cp /mnt/cdrom/ppc-2008-04-01_freescale.iso      /tmp/ppc-2008-04-01_freescale.iso
$
```

- Log as root and mount the ISO image of ELDK product `ppc-2008-04-01_freescale.iso` on the directory of your choice, for example `/media/cdrom`:

```
$  mount /tmp/ppc-2008-04-01_freescale.iso /media/cdrom -t iso9660 -o loop
$
```

- Check the mounted directory:

```
$ ls /media/cdrom
common          etc       ppc_6xx    ppc_85xxDP    RPMS    version
ELDK_FIXOWNER   icons     ppc_74xx   ppc_8xx       sys
ELDK_MAKEDEV    install   ppc_85xx   README.html   tools
$
```

- Chose a new directory where the ELDK is to be installed, for example: **/opt**
- Under root logging, install the ELDK tool with the following command:

```
$ whoami
root
$ ./install -d /opt/eldk-4.2

 Do you really want to install into /opt/eldk-4.2 directory[y/n]?: y
 Installing cross RPMs
Done
Installing ppc_6xx target RPMs
 Preparing...                ########################################### [100%]
.....
$
```

Using this install command, the ELDK tool is installed for all target architectures, if you wish to install it only for the a specific target architecture then indicate it as a second argument. For example for VM6250 you can limit the installation with `./install -d /opt/eldk-4.2  ppc_74xx`

After the initial installation is complete, to start working with the ELDK set and export the CROSS_COMPILE environment variable. Optionally, you may wish to add the bin and usr/bin directories of your ELDK installation to the value of your PATH environment variable. For instance, a sample ELDK installation and usage scenario looks as follows:

- After the installation utility completes, export the CROSS_COMPILE variable:

```
$ export CROSS_COMPILE=ppc_74xx-
$
```

The trailing '-' character in the CROSS_COMPILE variable value is optional and has no effect on the cross tools behavior. However, it is required when building Linux kernel and U-Boot images.

The value of the CROSS_COMPILE variable must correspond to the target CPU family you want the cross tools to work for. It is necessarily ppc_74xx- for VM6250 board. **Do not forget the trailing "-" at the variable end.**

- Add the directories **/opt/eldk/usr/bin** and **/opt/eldk/bin** to the PATH variable:

```
$   export PATH=$PATH:/opt/eldk-4.2/usr/bin:/opt/eldk-4.2/bin
$
```

- Display the ELDK version to check:

```
$ ppc_74xx-as --version
GNU assembler 2.17.50.0.12 20070128
Copyright 2005 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License.  This program has absolutely no warranty.
This assembler was configured for a target of `powerpc-linux´.
$
```

3.3 Recompiling a New U-Boot Image

3.3.1 Installing the U-Boot PBIT Tool Kit

Once the ELDK compiler is available, you need to extract the PBIT tool kit directory and run a **make** command.

To extract the PBIT toolkit directory from the DVD-ROM, run:

```
$ mount /dev/dvdrom /mnt/dvdrom
$
```

- Copy the full distribution to install into that specified directory for instance **/home/user**:

```
$ cp  /mnt/dvdrom/pbit_vm6250-"ID".tgz /home/user
$
```

- Extract the PBIT Tool Kit directory:

```
$ cd /home/user
$ tar xvzf  pbit_vm6250-"ID".tgz
$
```

A new directory pbit_vm6250-”ID” is created

3.3.2 Creating a New U-Boot Image

To compile a new U-Boot Image for testing, run following commands:

```
$ cd pbit_vm6250-"ID"
$ make

echo "#define UBOOT_ID        \""`date +´%y%j´`"\"">> include/version_autogenerated.h
for dir in  ; do make -C $dir _depend ; done

Generating include/autoconf.mk
make -C post/
make[1]: Entering directory `/home/user/U-Boot-1_3_3/pbit_distrib_09188/post´
ppc_74xx-gcc      -g            -Os                  -D__KERNEL__       -DTEXT_BASE=0xfff00000
-I/home/user/U-Boot-1_3_3/pbit_distrib_09188/include   -fno-builtin   -ffreestanding   -nostdinc
-isystem /opt/eldk-4.2/usr/bin/../lib/gcc/powerpc-linux/4.2.2/include -pipe  -DCONFIG_MPC86xx=1
-DCONFIG_MPC8641=1    -maltivec    -mabi=altivec    -msoft-float    -Wall    -Wstrict-prototypes
-I/home/user/U-Boot-1_3_3/pbit_distrib_09188 -c -o tests.o tests.c

ppc_74xx-gcc      -g            -Os                  -D__KERNEL__       -DTEXT_BASE=0xfff00000
-I/home/user/U-Boot-1_3_3/pbit_distrib_09188/include   -fno-builtin   -ffreestanding   -nostdinc
-isystem /opt/eldk-4.2/usr/bin/../lib/gcc/powerpc-linux/4.2.2/include -pipe  -DCONFIG_MPC86xx=1
-DCONFIG_MPC8641=1    -maltivec    -mabi=altivec    -msoft-float    -Wall    -Wstrict-prototypes
-I/home/user/U-Boot-1_3_3/pbit_distrib_09188 -c -o newtest.o newtest.c
ppc_74xx-ar crv libpost.a  tests.o newtest.o
a - tests.o
a - newtest.o
make[1]: Leaving directory `/home/user/U-Boot-1_3_3/pbit_distrib_09188/post´
make -C /home/user/U-Boot-1_3_3/pbit_distrib_09188/board/kontron/nice/ U-Boot.lds
make[1]: Entering directory `/home/user/U-Boot-1_3_3/pbit_distrib_09188/board/kontron/nice´
```

```
make[1]: Nothing to be done for `U-Boot.lds´.
make[1]: Leaving directory `/home/user/U-Boot-1_3_3/pbit_distrib_09188/board/kontron/nice´
ppc_74xx-ar xv lib/uboot_libs.obj; \
                UBOOT_LIB=`ls *.a`;\
                ppc_74xx-ar xv lib/mpc86xx.obj; \
                UNDEF_SYM=`ppc_74xx-objdump -x libnice.a libgeneric.a libkontron.a libmpc86xx.a
libppc.a libcramfs.a libfat.a libfdos.a libjffs2.a libreiserfs.a libext2fs.a libnet2.a libdisk.a
libatibiosemu.a libblock.a libdma.a libhwmon.a libi2c.a libinput.a libmisc.a libmtd.a libnand.a
libnand_legacy.a  libonenand.a  libnet.a  libsk98lin.a  libpci.a  libpcmcia.a  libspi.a  librtc.a
libserial.a libusb.a libvideo.a libcommon.a libfdt.a libapi.a libpost.a post/libpost.a| \
                sed  -n -e ´s/.*\(__u_boot_cmd_.*\)/-u\1/p´|sort|uniq`;\
                     cd /home/user/U-Boot-1_3_3/pbit_distrib_09188 && ppc_74xx-ld -Bstatic -T
/home/user/U-Boot-1_3_3/pbit_distrib_09188/board/kontron/nice/U-Boot.lds     -Ttext  0xfff00000
$UNDEF_SYM  start.o \
                       --start-group  libgeneric.a libkontron.a libmpc86xx.a libppc.a libcramfs.a
libfat.a  libfdos.a  libjffs2.a  libreiserfs.a  libext2fs.a  libnet2.a  libdisk.a  libatibiosemu.a
libblock.a libdma.a libhwmon.a libi2c.a libinput.a libmisc.a libmtd.a libnand.a libnand_legacy.a
libonenand.a libnet.a libsk98lin.a libpci.a libpcmcia.a libspi.a librtc.a libserial.a libusb.a
libvideo.a  libcommon.a  libfdt.a  libapi.a  libpost.a  libnice.a  post/libpost.a   --end-group  -L
/opt/eldk-4.2/usr/bin/../lib/gcc/powerpc-linux/4.2.2/m7400  -lgcc \
                        -Map U-Boot.map -o U-Boot
x - libgeneric.a
x - libkontron.a
x - libmpc86xx.a
x - libppc.a

x - libcramfs.a
x - libfat.a
x - libfdos.a
...
...
...
x - atibios.o
chmod +x ./tools/addinfo
./tools/addinfo U-Boot 1.3.3 nice

U-Boot size : 430336
U-Boot sum  : 58885
Patching Version String to file U-Boot at offset 262148 (decimal)
Patching Version "1.3.3" to file U-Boot at offset 262224 (decimal)
Patching ID "09189" to file U-Boot at offset 262240 (decimal)
Patching Size "430336" to file U-Boot at offset 262256 (decimal)
Patching Sum "58885" to file U-Boot at offset 262260 (decimal)
rm -f *.[oa]
ppc_74xx-objcopy --gap-fill=0xff -O srec U-Boot U-Boot.srec
ppc_74xx-objcopy --gap-fill=0xff -O binary U-Boot U-Boot.bin
$
```

The new created U-Boot image is **U-Boot.bin**. Refer to U-Boot User's Manual to update U-Boot on the VM6250 board.

If the generated U-Boot image is older than the version on your VM6250 board then contact Kontron support representative to receive a library update corresponding to the version of U-Boot you have. U-Boot version can be check under U-Boot prompt with command version.

3.4 Adding a New Test

Let's consider the PBIT Tool Kit has been installed in the directory **${PBITDISTRIB}**

To add a new test named **newtest** with associated number 33:

- Create the test file **${PBITDISTRIB}/post/newtest.c file** (the file already exist, you can edit/update it)
- Add **COBJS +=newTest.o** in the **${PBITDISTRIB}/post/Makefile** (this is already done for current example)
- Add the test entry point in **${PBITDISTRIB}/post/test.c** file in the **post_list[]** and the **post_default_run_list[]** arrays following example given with newtest. This is already done for current example in the file **post/test.c** but not compiled because the define **CFG_POST_NEWTEST** is not set.
- Remove the conditional compile flags **CONFIG_POST & CFG_POST_NEWTEST** in **tests.c** or add **CFG_POST_NEWTEST** at the end of file **${PBITDISTRIB}/include/configs/NICE.h**
- Build **newtest.o** and **post/test.o** simply running the commands:

```
$ cd ${PBITDISTRIB}
$ make
```

The new **U-Boot.bin** image can be download and flashed on U-Boot, example:

```
VM6250 => setenv ipaddr 192.93.161.25
VM6250 => setenv serverip 192.93.161.184
VM6250 => tftp 0x2000000 U-Boot.bin
Enet starting in 1000BT/FD
Speed: 1000, full duplex
Using eTSEC1 device
TFTP from server 192.93.161.184; our IP address is 192.93.161.25
Filename 'u-boot.bin'.
Load address: 0x2000000
Loading: ############################
done
Bytes transferred = 430336 (69100 hex)

VM6250 =>uboot update
Checking update image at 0x2000000 ... OK
Updating U-Boot on Flash #0 (MAIN U-Boot image)
Protect off 0xfff00000 0xfff690ff
....... done
Un-Protected 7 sectors
Erase 0xfff00000 0xfff690ff
....... done
Erased 7 sectors
Copy to Flash...  done
Verify Flash... OK
VM6250 => reset
```

At the new U-Boot prompt, check the new test number 33 is available running the **diag** command or **diag 33** command

```
VM6250 => diag 33
PBIT "33" is not configured to run from command line
VM6250 =>diag cfg default 33
VM6250 =>diag 33
 newtest (33) - Checks if new tests are OK.
   capabilities : cpu0/cpu1,fast,simple
   run mode 1   : cpu0,fast,simple
VM6250 => diag run 33
```

Without modification, the newtest test example will fail because it expects a specific PMC.

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.