» User Guide «

U-Boot

Bootloader

User Guide for the
AM4120

Doc. ID: 1046-1856, Rev. 1.0
3 November 2011

Revision History

Publication Title:	U-Boot Bootloader User Guide for the AM4120	
Doc. ID:	1046-1856	
Rev.	**Brief Description of Changes**	**Date of Issue**
1.0	Initial issue	3-Nov-2011

Imprint

Kontron Modular Computers GmbH may be contacted via the following:

MAILING ADDRESS

Kontron Modular Computers GmbH
Sudetenstraße 7
D - 87600 Kaufbeuren Germany

TELEPHONE AND E-MAIL

+49 (0) 800-SALESKONTRON
sales@kontron.com

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

Disclaimer

Table of Contents

This page has been intentionally left blank.

Copyrights and Licensing

U-Boot is Free Software. It is copyrighted by Wolfgang Denk and many others who contributed code (see the actual source code for details). You can redistribute U-Boot and/or modify it under the terms of version 2 of the GNU General Public License as published by the Free Software Foundation. Most of it can also be distributed, at your option, under any later version of the GNU General Public

License -- see individual files for exceptions.

NOTE! This license does *not* cover the so-called "standalone" applications that use U-Boot services by means of the jump table provided by U-Boot exactly for this purpose - this is merely considered normal use of U-Boot, and does *not* fall under the heading of "derived work".

The header files "include/image.h" and "include/asm-*/u-boot.h" define interfaces to U-Boot. Including these (unmodified) header files in another file is considered normal use of U-Boot, and does *not* fall under the heading of "derived work".

Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the U-Boot source code) is copyrighted by me and others who actually wrote it.

-- Wolfgang Denk

==========

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software -- to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Obtaining Source Code

The software included in this product contains copyrighted software that is licensed under the GPL. A copy of that license is included in this document beginning on page 5. You may obtain the complete Corresponding Source code from Kontron for a period of three years after our last shipment of this product. Please contact Kontron Support for further assistance in obtaining the source code.

1. Introduction to U-Boot

U-Boot is an open source boot loader software developed and maintained by DENX Software Engineering GmbH (http://www.denx.de). Kontron provides U-Boot with all its standard features as well as Kontron specific features for usage with Kontron's Freescale™ QorIQ P-series based AMC CPU modules.

This user guide provides specific information on Kontron's implementation of U-Boot and its usage. Please refer to the DENX website for up-to-date on-line documentation of all of U-Boot's standard features.

2. Standard U-Boot Commands

U-Boot is provided with a library of standard commands for which documentation is provided on the DENX website. Some of the below listed standard commands have sub-groups which can be displayed when help for the main group command is requested.

Where relevant, further information concerning the usage of standard commands is provided in this guide to assist users in performing specific functions.

Table 1: Standard U- Boot Commands

COMMAND	DESCRIPTION
?	Alias for 'help'
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
bootm	Boot application image from memory
bootp	Boot image via network using BOOTP/TFTP protocol
bootvx	Boot vxWorks from an ELF image
chpart	Change active partition
cmp	Memory compare
coninfo	Print console devices and information
cp	Memory copy
cpu	Multiprocessor CPU boot manipulation and release
crc32	Checksum calculation
dhcp	Boot image via network using DHCP/TFTP protocol
echo	Echo args to console

Table 1: Standard U- Boot Commands

COMMAND	DESCRIPTION
editenv	Edit environment variable
env	Environment handling commands
erase	Erase FLASH memory
exit	Exit script
ext2load	Load binary file from a Ext2 filesystem
ext2ls	List files in a directory (default /)
false	Do nothing, unsuccessfully
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
fsinfo	Print information about filesystems
fsload	Load binary file from a filesystem image
go	Start application at address 'addr'
help	Print command description/usage
i2c	I2C subsystem
iminfo	Print header information for application image
imls	List all images found in flash
imxtract	Extract a part of a multi-image
interrupts	Enable or disable interrupts
irqinfo	Print information about IRQs
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
ls	List files in a directory (default /)
md	Memory display

Table 1: Standard U- Boot Commands

COMMAND	DESCRIPTION
md5sum	Create/check md5sum message digest
mii	MII utility commands
mm	Memory modify (auto-incrementing address)
mmc	MMC sub system
mmcinfo	Display MMC info
mtdparts	Define flash/nand partitions
mtest	Simple RAM read/write test
mw	Memory write (fill)
nand	NAND subsystem
nboot	Boot from NAND device
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
reginfo	Print register information
reset	Perform RESET of the CPU
run	Run commands in an environment variable
saveenv	Save environment variables to persistent storage
saves	Save S-Record file over serial line
setenv	Set environment variables
setexpr	Set environment variable as the result of eval expression
sf	SPI flash subsystem
showvar	Print local hushshell variables
sleep	Delay execution for some time
source	Run script from memory
test	Minimal test like /bin/sh
tftpboot	Boot image via network using TFTP protocol
true	Do nothing, successfully

Table 1: Standard U- Boot Commands

COMMAND	DESCRIPTION
ubi	ubi commands
ubifsload	Load file from an UBIFS filesystem
ubifsls	List files in a directory
ubifsmount	Mount UBIFS volume
ubifsumount	Unmount UBIFS volume
version	Print monitor, compiler and linker version

3. Kontron Specific Commands

3.1 Command List

Kontron's implementation of U-Boot includes certain enhancements to provide specific functions not incorporated in the U-Boot standard library. The following additional functions have been implemented in U-Boot:

- flsw (FLash SWitch)
- fru (Field Replaceable Unit)
- fwum (FirmWare Update Manager
- kboardinfo (Kontron BOARDINFO)
- kcs (Keyboard Controller Style)
- tlbdbg (Translation Look-aside Buffer DeBuG)
- vpd (Vital Product Data)

3.2 Command Syntax Reference

The following provides command syntax reference information, a short description, and, in some cases, usage examples.

Where an ellipsis (…) appears in the command syntax it means that the command is continued on the next line. Observe spaces before the ellipsis.

flsw (FLash SWitch)

FUNCTION:	Indicate or select currently active SPI NOR flash
SYNTAX:	**`flsw [s|r]`** where: flsw command: issuing the command without arguments will indicate the currently active SPI NOR flash (also returns 'true' or 'false' depending on currently active flash) s option: standard selects the standard SPI NOR flash as the active flash r option: recovery selects the recovery SPI NOR flash as the active flash
DESCRIPTION:	This command is used to determined the currently active SPI NOR flash or to select either the standard SPI NOR flash or the recovery SPI NOR flash as the currently active flash. In addition, this command returns 'true' if the standard SPI NOR flash is selected or 'false' if the recovery SPI NOR flash is selected. This is used in the update scripts to prevent the recovery flash from being updated. Besides this command, the currently active SPI NOR flash may also be selected either via a DIP Switch setting or IPMI OEM command. The output of this command always shows the current state.
USAGE:	Query flash status: COMMAND / RESPONSE: **`=> flsw`** **`standard boot flash active`** **`=>`**
	Select standard SPI NOR flash as currently active flash: COMMAND / RESPONSE: **`=> flsw s`** **`=>`**

fru (Field Replaceable Unit)

FUNCTION:

Provides read/write access to the board's FRU repository as well as displaying FRU data

SYNTAX:

```
fru info <FRU nr>|…
    read <FRU nr> <address> <size>|…
    write <FRU nr> <address> <size>
```

where:

fru	command:
info	option:
	displays FRU data for <FRU nr> specified
<FRU nr>	parameter: hexadecimal
	<0, 1, … n>
	identification number of FRU device for option specified
read	option:
	reads FRU data for <FRU nr> specified
<address>	parameter: hexadecimal
	<[x …]x>
	address where data is to be stored or read from
<size>	parameter: hexadecimal
	<[x …]x>
	length of data in bytes to be read or written
write	option:
	writes FRU data to <FRU nr> specified

DESCRIPTION:

This command can be used to display basic information about the FRU repository, read out the repository content to RAM, and, if required, to update the contents of the FRU repository.

WARNING!

Writing incorrect FRU data to the FRU repository can result in an inoperable board (E-Keying information incorrect). Users requiring modification to the board's FRU data are requested to contact Kontron for assistance before making any changes.

USAGE:

Query FRU data for FRU 0:

COMMAND / RESPONSE:

```
=> fru info 0
FRU 0 size is 0x1000 bytes
```

fru (Field Replaceable Unit)

USAGE:	**Read FRU data for FRU 0:**
	COMMAND / RESPONSE:
	`=> fru read 0x0 0x1000000 0x1000`
	`Progress:####################################`
	`####################################`
	`####################################`
	`####################################`
	`##################################`
	`=>`

	Write FRU data to FRU 0:
	COMMAND / RESPONSE:
	`=> fru write 0x0 0x1000000 0x1000`
	`Progress:####################################`
	`####################################`
	`####################################`
	`####################################`
	`##################################`
	`=>`

fwum (FirmWare Update Manager)

FUNCTION: Provides functions for managing and updating the module's MMC firmware

SYNTAX:

```
fwum info|…
     status|…
     upgrade <address> <size>|…
     rollback
```

where:

fru	command:
info	option:
	displays information concerning 'fwum' services
status	option:
	displays information concerning the status of 'fwum' services
upgrade	option:
	updates MMC firmware using parameters as specified by <address> and <size>
<address>	parameter: hexadecimal
	<[x …]x>
	address in RAM where data is to be read from
<size>	parameter: hexadecimal
	<[x …]x>
	length of data in bytes to be read
rollback	option:
	executes manual rollback to previous firmware version

fwum (FirmWare Update Manager)

DESCRIPTION:	This command can be used to: 1. Info - show information about FWUM service present on IPMI firmware 2. Status - show actual status of firmware banks 3. Upgrade - download pointed firmware into MMC and initiate firmware upgrade procedure 4. Rollback - initiate manual firmware rollback to switch back to previously used firmware WARNING! Users requiring modification to the MMC's firmware are requested to contact Kontron for assistance before making any changes.

USAGE:	Query firmware service currently available:
	COMMAND / RESPONSE:
	`=> fwum info`
	
	`FWUM info`
	`=========`
	`Protocol Revision : 07h`
	`Controller Device Id : 00h`
	`Firmware Revision : 1.01`
	`Number Of Memory Bank : 2`
	`=>`

fwum (FirmWare Update Manager)

USAGE:

Query the status of the currently available firmware images:

COMMAND / RESPONSE:

```
=> fwum status

FWUM status
=========

Bank State 0                    : Last Known Good
Firmware Length                 : 143088 bytes
Firmware Revision               : 1.00 SDR 16

Bank State 1                    : Previous Good
Firmware Length                 : 146864 bytes
Firmware Revision               : 1.00 SDR 32
=>
```

Upgrade MMC firmware:

COMMAND / RESPONSE:

```
=> fwum upgrade 0x20000000 0x22ef0
Start uploading firmware into bank 0
Loading: ####################################
         ####################################
         ####################################
         ####################################
         ####################################
         ####################################
         ####################################
         ####################################
         ########################
Firmware update initiated
=>
```

Perform manual rollback of MMC firmware

COMMAND / RESPONSE:

```
=> fwum rollback
Firmware rollback initiated
=>
```

kboardinfo (Kontron BOARD INFO)

FUNCTION:	Displays a summary of board and configuration information.
SYNTAX:	**kboardinfo** where: kboardinfo command:
DESCRIPTION:	This command collects information from various board sources and provides a summary listing of this information:
USAGE:	Display board information COMMAND / RESPONSE: **=> kboardinfo** **Board id: 0xd040** **Hardware rev.: 0x0** **Logic rev.: 0x1** **Boot flash: Recovery Flash** **In system slot: na** **Geographic address: 5** **Material number: na** **Serial number: na** **U-Boot article name: SK-FIRM-UBOOT-D0401** **U-Boot material num: 1046-1471** **=>**

kcs (Keyboard Controller Style)

FUNCTION:

Provides capability for transmitting raw IPMI commands from the payload CPU to the MMC and displaying response from the MMC.

SYNTAX:

```
kcs raw [lun <lun>] <NetFn> <CMD>
… [Request Data Bytes]
```

where:

kcs	command:
raw	option:
	send raw data over KCS interface
lun	option:
	if present: set up desired lun number of message to send to MMC
	if absent: lun is assumed to be 0
<lun>	parameter: hexadecimal
	parameter range: <0, 1, 2, 3>
<NetFn>	parameter: hexadecimal
	<[x …]x>
<CMD>	parameter: hexadecimal
Request Data Bytes	parameter: hexadecimal: 1 ... n bytes (space as delimiter between bytes)
	command parameters

DESCRIPTION:

This command can be used to send IPMI commands in raw form to the MMC over the KCS interface and print response.

WARNING!

As "ipmi raw" functions provide access to the majority of MMC functionality, care must be exercised when invoking raw commands. Improper use may cause the board to become inoperable (e.g. damage to FRU data).

kcs (Keyboard Controller Style)

USAGE:

Send IPMI "Get Device ID" command (lun 0, NetFn 6, cmd 1, no data)

COMMAND / RESPONSE:

```
=> kcs raw lun 0x00 0x06 0x01
KCS transaction successfully completed,
rsp_size: 18 (dec)
 1c 01 00 10 80 01 00 51 b9 98 3a 00 00 d0 10…
 05 00 00
=>
```

Response bytes:
- first byte presents return NetFn combined with lun
- second presents command number
- third presents completion code
- further bytes are response data

In this example, the first byte (0x1c) is decoded as lun 0 (two least significant bits) and NetFn 7 (six most significant bits)

Send IPMI "Get FRU Inventory Area" command to get information about FRU 0 repository (lun 0, NetFn 16, cmd 10, data byte 0x00)

COMMAND / RESPONSE:

```
=> kcs raw 0x0a 0x10 0x00
KCS transaction successfully completed,…
rsp_size: 6 (dec)
2c 10 00 00 10 00
=>
```

tlbdbg (Translation Look-aside Buffer DeBuG)

FUNCTION:	Displays current configuration of TLB0 and TLB1
SYNTAX:	**`tlbdbg`** where: tlbdbg command:
DESCRIPTION:	This command provides information on the translation look-aside buffers TLB0 ad TLB1 for debugging purposes during U-Boot development or for debugging OS startup issues
USAGE:	Display TLB0/TLB1 information COMMAND / RESPONSE:

```
=> tlbdbg
TLBx Configuration Register : 04110200 101bc010


TLB0: [check 512 entries]
IDX   PID EPN       SIZE   V TS    RPN       U0-U3 WIMGE UUUSSS
-------------------------------------------------------------

TLB1: [check 16 entries]
IDX   PID EPN       SIZE   V TS    RPN       U0-U3 WIMGE UUUSSS
-------------------------------------------------------------
  0d: 00  00000000  1GB    V 0d  -> 0_00000000 0000 ----- ---RWX
  1d: 00  40000000  1GB    V 0d  -> 0_40000000 0000 ----- ---RWX
  2d: 00  ffe00000  1MB    V 0d  -> f_ffe00000 0000 -I-G- ---RWX
  3d: 00  80000000  1GB    V 0d  -> e_80000000 0000 -I-G- ---RWX
  4d: 00  ffc00000  256kB  V 0d  -> e_ffc00000 0000 -I-G- ---RWX
  5d: 00  f0000000  64MB   V 0d  -> f_f0000000 0000 -I-G- ---RWX
  6d: 00  f8000000  16MB   V 0d  -> f_f8000000 0000 -I-G- ---RWX
  7d: 00  ff000000  4kB    V 0d  -> f_ff000000 0000 -I-G- ---RWX
  8d: 00  c0000000  256MB  V 0d  -> d_c0000000 0000 -I-G- ---RWX
  9d: 00  d0000000  256MB  V 0d  -> d_d0000000 0000 -I-G- ---RWX
 10d: 00  fffff000  4kB    V 0d  -> 0_1ffff000 0000 -I-G- ---RWX
=>
```

vpd (Vital Product Data)

FUNCTION:	Provides display and importing functions for vital product data entities
SYNTAX:	**vpd print [<name>] | …** **import <name> | all_params** where: vpd — command: print — option: displays VPD information (source: System EEPROM) (if <name> is not used, all VPD entities are displayed) <name> — parameter: text string <[x …]x> name of VPD entity addressed by option import — option: imports VPD information to the U-Boot Environment (source: System EEPROM; target: RAM) all_params — parameter: text constant all_params selects all VPD entities for importing to the U-Boot Environment
DESCRIPTION:	Vital Product Data are information stored in the System EEPROM which are required for proper operation of the board. With this command the VPD entities can be displayed or imported to the U-Boot Environment in RAM. Among the VPD entities are, for example, the board serial number and the board's Ethernet MAC addresses. If the option: 'import' is invoked, existing VPD entities in the Environment in RAM are overwritten. If a 'saveenv' is then invoked, the previously stored values in the currently active SPI NOR flash Environment area are overwritten.
USAGE:	Display all VPD entities COMMAND / RESPONSE: **=> vpd print** ***<response: displays all VPD entities>***

vpd (Vital Product Data)

Display eth1addr entity

COMMAND / RESPONSE:

```
=> vpd print eth1addr
eth1addr=00:80:82:47:12:02
```

Import eth1addr entity to Environment

COMMAND / RESPONSE:

```
=> vpd import eth1addr
import eth1addr = 00:80:82:47:12:02 to …
Environment
```

Import all VPD entities to Environment

COMMAND / RESPONSE:

```
=> vpd import all_params

<response: displays all imported VPD
entities; format for each imported VPD entity
as follows:>

import <name> = <value> to Environment
.
.
.
import <name> = <value> to Environment
```

4. U-Boot Access and Startup

Communication with U-Boot is achieved via a serial console configured for 115200 baud, 8N1.

Initially, U-Boot executes the commands defined in the environment variable ‘preboot’. Then, if not otherwise interrupted, U-Boot pauses for the time defined in the environment variable ‘bootdelay’ and then executes the statements stored in the environment variable ‘bootcmd’. To gain access to the U-Boot command prompt type in any single character during the boot delay time.

If required, the boot delay function can be so configured that even when the boot delay is set to ‘0’ to have characters, which are sent over the serial interface prior to the boot wait time, be recognized to allow operator intervention in the boot process.

5. Environment

The Environment is stored in the same flash as U-Boot, usually in the last sector. This provides the possibility to update U-Boot without changing the Environment. The environment can be modified by the user with the typical commands of the ‘env’ command group: ‘setenv’, ‘editenv’, ‘printenv’ and ‘saveenv’.

Furthermore, if a larger number of boards require updating the environment can be updated by a script, loaded from the SD card, onboard NAND flash, or a network.

A typical user modification would be to set the variable ‘bootcmd’ so that the user’s OS will boot automatically.

6. Working with U-Boot

6.1 General Operation

Most operations are carried out using the main memory as an intermediate step. It is not possible for example to boot a kernel image directly from a tftp server. Instead, the kernel image is first loaded to memory and then booted from there with another command.

The same is true when writing new contents to the SPI NOR flashes.

This concept is very flexible since it separates the commands which handle the loading of data from the commands that carry out actions like booting.

6.2 Using the Network

U-Boot provides support for multiple Ethernet interfaces for transferring files from a file server. This is accomplished using the environment variables: “ethact” and “ethrotate”.

“ethact” is used to select the required interface.

In the case of the AM4120 this is as follows:

- ethact eTSEC1 - interface is the front panel connector J2 (B)
- ethact eTSEC2 - interface is either AMC port 1 or the front panel connector J3 (A) (depends of the hardware configuration; settable per DIP switch)
- ethact eTSEC3 - interface is AMC port 0

“ethrotate” can be used to force the selection of the next available interface if, for example, there is no link available for the selected interface.

If set to “yes” (default is “no”), U-Boot provides a status message that the next interface is selected and updates the “ethact” variable accordingly. U-Boot then tries to download the file again. This is repeated until either the file is downloaded or all interfaces have been exhausted.

In the event the link is active for the selected interface and ethrotate is yes, U-Boot tries to download the file. If it cannot download the file, it tries the next available interface. If the file is not available on the server, U-Boot stops trying and issues an error message.

In addition, to be able to transfer files from a tftp server to a module, the module’s IP address (environment variable ‘ipaddr’) and the IP address of the server must be set (environment variable ‘serverip’). Alternatively, it is possible to use the ‘dhcp’ or ‘bootp’ commands.

They can be set using the ‘setenv’ command. Please note, that these settings are lost after a reset. To retain the environment permanently, use the command ‘saveenv’, which saves the complete environment to flash.

To transfer a file from a tftp server to memory, the ‘tftpboot’ command is used, for example:

```
tftpboot 100000 filename
```

6.3 Using SD Cards

SD Cards are supported (read only) with the ‘ext2’ or ‘fat’ file system.

In both cases, the card must be rescanned first.

```
mmc rescan 0
```

After that, the contents can be verified with:

```
ext2ls mmc 0
```

in case of the ext2 file system, or

```
fatls mmc 0
```

in case of the fat file system.

To load a file into memory the commands ‘ext2load’ or ‘fatload’ can be used, for example:

```
ext2load mmc 0 100000 kernel.bin
```

which loads the file ‘kernel.bin’ from the SD card to memory address 0x100000.

6.4 Using the Onboard NAND Flash

The onboard NAND Flash is supported with the ‘ubi’ filesystem. The access is read only so the filesystem and its contents must be prepared with Linux first.

As a prerequisite, the environment variables ‘mtdids’ and ‘mtdparts’ must be set correctly.

‘mtdids’ identifies the NAND chip to use while mtdparts defines the partitions.

Example:

```
setenv mtdids nand0=chip1
setenv mtdparts mtdparts=chip1:-(all)
```

This defines the first NAND chip (nand0) to be used with the name ‘chip1’. The chip contains one partition ‘all’ which occupies the whole chip.

The next command sets the partition ‘all’ to be used with the ‘ubi’ layer:

```
ubi part all
```

Now, an ‘ubi’ volume can be mounted; in this example volume ‘boot’:

```
ubifsmount boot
```

After the volume is mounted, its contents can be listed:

```
ubifsls
```

or a file loaded, in this case ‘kernel.bin’ to address 0x100000:

```
ubifsload 100000 kernel.bin
```

6.5 Using the Onboard Parallel NOR Flash

The onboard parallel NOR flash is not used together with a file system, it is used raw. It does not contain any U-Boot components and is completely free for the user usage. It's primary function is to store VxWorks® bootroms and images.

To program an image to the NOR flash, it must first be loaded to memory from an arbitrary source. It can then be programmed with the ‘cp’ command. The flash must be erased before the NOR flash is programmed. To achieve this, use the ‘erase’ command.

Example: Programming a test file ‘test.img’ from an SD card using the ‘ext2’ file system:

```
erase f0000000 f000ffff
mmc rescan 0
ext2load mmc 0 100000 test.img
cp.b 100000 f0000000 ${filesize}
```

This example assumes, that the size of ‘test.img’ is less than 64 kB. The environment variable ‘filesize’ is set automatically when a file is loaded to memory and can be used for convenience here.

6.6 Booting an OS

6.6.1 Booting Linux

To boot Linux, at least a kernel image and a FDT (Flattened Device Tree) must be loaded to memory. Optionally, an ‘initrd’ can be loaded.

Furthermore, a command line must be prepared in the environment variable ‘bootargs’.

The boot itself is initiated with the ‘bootm’ command.

To simplify the setup of the board, three predefined scripts are already programmed in the default environment:

- ‘nfsboot’ to boot from a tftp server and mount the root over NFS
- ‘nandboot’ to boot from the NAND flash and also mount it as root
- ‘sdboot’ to boot from a SD Card and also mount it as root

For a one-time-only bootup, this can be accomplished with the ‘run’ command, for example:

```
run nfsboot
```

To make this permanent and have the board execute it automatically, it must be stored in the ‘bootcmd’ environment variable and the environment must be saved to flash.

Example:

```
setenv bootcmd 'run nandboot'
saveenv
```

6.6.2 Booting VxWorks

To boot a Wind River VxWorks image, a U-Boot Image file of the corresponding (ROM-able) VxWorks binary image and an FDT (Flattened Device Tree) must be loaded to memory.

By default U-Boot operates on ‘uImage’ files (U-Boot Image) which contain a special header and in the data portion the operating system binary image. The special header defines various properties of the U-Boot image (e.g. load address and entry point for the binary image in the data portion). Both the header and the data portion of the U-Boot image are secured and checked against corruption by a CRC32 checksum at U-Boot load time.

All VxWorks (ROM-able) binary images will be converted to an U-Boot image at build time of the suiting Wind River Workbench projects based on the dedicated Kontron Modular Computers VxWorks BSP (Board Support Package). This conversion will be carried out by a Kontron Modular Computers JAVA based version of the U-Boot tool ‘mkImage’ which is invoked by BSP provided Wind River Workbench project settings automatically.

On successful build of the VxWorks binary (ROM-able) image an additional U-Boot image file, containing the VxWorks (ROM-able) binary image will be generated in the project default build folder with the following naming conventions:

U-BOOT IMAGE NAME	VXWORKS IMAGE NAME
uImage.bootrom.bin	bootrom.bin
uImage.bootrom_uncmp.bin	bootrom_uncmp.bin
uImage.vxWorks_rom.bin	vxWorks_rom.bin
uImage.vxWorks_romCompress.bin	vxWorks_romCompress.bin

Please note, that the resulting U-Boot image contains all needed information for a proper U-Boot load process and start of the contained VxWorks binary (ROM-able) image. Therefore it is strongly recommended to utilize the corresponding U-Boot image listed above when using U-Boot for booting VxWorks.

The VxWorks U-Boot image file and FDT are typically stored in and loaded from onboard parallel NOR flash.

The boot itself is initiated with the ‘bootm’ command. To perform autobooting of a VxWorks image requires that appropriate U-Boot environment variables or script(s) be defined for the boot operation to be performed. For more detailled information with examples of boot command sequences, refer to the Kontron VxWorks BSP online documentation.

For more information on how to configure and build VxWorks images and how to utilize them e.g. for a subsequent VxWorks boot process, please refer to the appropriate Wind River documentation.

6.7 Getting Help

U-Boot was configured with support for longhelp. That means, that for every command, online help is available while working with the system. To access the online help, enter '?' or 'help' at the console prompt. This will show an overview over all available commands. To get specific help, enter '? <command/command group' or 'help <command/command group'.

For example to get help on the 'saves' command enter '? saves'.

```
=> ? saves
saves - save S-Record file over serial line
Usage:
saves [ off ] [size] [ baud ]
    - save S-Record file over serial line with offset 'off', size
'size' and
      baudrate 'baud'
=>
```

To get help on the mmc command group enter ‘? mmc’.

```
=> ? mmc
mmc - MMC sub system
Usage:
mmc read <device num> addr blk# cnt
mmc write <device num> addr blk# cnt
mmc rescan <device num>
mmc part <device num> - lists available partition on mmc
mmc list - lists available devices
=>
```

6.8 Update

The environment contains two scripts which allow an update of various components, e.g. U-Boot, bootrom for VxWorks, data in E2PROMs, etc.

The script ‘update’ checks for a U-Boot script ‘update’ in the directory ‘update’ in the first partition of the SD card with ‘ext2’ or ‘fat’ filesystem. If unsuccessful, the check continues with the first NAND chip, volume ‘boot’, and again U-Boot searches in the subdirectory ‘update’ for the script ‘update’. If the script ‘update’ is found, it is loaded to memory and executed.

So, to actually execute an update, e.g. a SD card should be prepared with a directory 'update' on the first partition. Kontron provides an update e.g. for U-Boot as a compressed archive (zip, tar.bz2, tar.gz) which must be unpacked in the directory ‘update’.

After the SD card is inserted, U-Boot should be stopped at the console after power up. To manually start the update, enter the following command:

```
run update
```

In the case of a U-Boot update, only the standard SPI NOR flash is updated.

The script ‘netupdate’ tries to load a U-Boot script ‘update/update’ from the server. If found, it is loaded to memory and executed as in the case of the SD card.

As the script ‘netupdate’ requires access to a server, the environment variable ‘serverip’ must be set correctly. Alternatively, it is possible to use the ‘dhcp’ or ‘bootp’ commands.

An automatic run of the update script at every startup takes place if the update script is started in the preboot environment variable:

```
setenv preboot 'run update'
saveenv
```

6.9 Recovery Mechanism

The are two SPI NOR flashes available with each device holding a copy of U-Boot. In case the contents of the standard SPI NOR flash have been corrupted (e.g. as a result of a power failure during an update), the IPMI subsystem detects the problem, switches the flashes and restarts the CPU. The board starts from the recovery SPI NOR flash. In this state, the standard SPI NOR flash can be programmed again with the ‘update’ or ‘netupdate’ scripts described in the previous chapter “6.8 Update”.

The update scripts provided ensure that prior to the update the standard SPI NOR flash is selected and the U-Boot update image is available and correct.

The contents of the recovery SPI NOR flash should never be updated in order to avoid a completely inoperable system with no accessing capability.