# X-board® <GP8> Bootloader Description

Document Revision 0.6

PRELIMINARY

... always a Jump ahead!

# CONTENTS

# 1. USER INFORMATION

## 1.1 About This Manual

This document provides information about products from Kontron Embedded Computers GmbH and/or its subsidiaries. No warranty of suitability, purpose, or fitness is implied. While every attempt has been made to ensure that the information in this document is accurate, the information contained within is supplied “as-is” and is subject to change without notice.

For the  circuits, descriptions and tables indicated, Kontron assumes no responsibility as far as patents or other rights of third parties are concerned.

## 1.2 Copyright Notice

## 1.3 Trademarks

The following lists the trademarks of components used in this board.

- IBM, XT, AT, PS/2 and Personal System/2 are trademarks of International Business Machines Corp.
- Microsoft is a registered trademark of Microsoft Corp.
- Intel is a registered trademark of Intel Corp.
- All other products and trademarks mentioned in this manual are trademarks of their respective owners.

## 1.4 *License conditions*

The bootloader for the X-board<GP8> is distributed under the conditions of the eCos license, which is a modified version of the GPL (General Public License). All included third party sources are licensed under the license mentioned in the corresponding documentation. According to this, we offer to provide the source code covered by an open source license to the public. Please contact our support department if you wish to receive the source code.

**Note: We can't provide any support for firmware modifications. If you want to do it anyways, you will do it completely on your own risk. X-board<GP8>s with hardware failure or damaged flash contents can be sent to Kontron but the usual repair fees apply.**

### 1.4.1. LIMITED LIABILITY

**Note: Please read and take note of the following information**

# 2. INTRODUCTION

The bootloader that is provided with the Kontron X-board<GP8> is based on the RedBoot Bootloader which was originally released by RedHat. RedBoot is an acronym for "Red Hat Embedded Debug and Bootstrap". Shipped with the board is a specially adapted version for the X-board<GP8>. This documentation only covers the basic features and the Kontron specific enhancements of the bootloader. For a general information on RedBoot please refer to http://ecos.sourceware.org/redboot/.

Additionally there may be a second bootloader, called CEBoot, stored on the module that is able to start Microsoft Windows CE. This bootloader is included in the Windows CE BSP of the X-board<GP8> and can be removed, added and changed without touching the main bootloader. Please have a look at UPDATING THE WINCE BOOTLOADER and winceexec for details.

## 2.1 *Features*

- Linux boot support
- Microsoft Windows CE boot support
- Terminal via
  - Ethernet
  - RS232
- Ethernet configuration via BOOTP
- Data download via
  - HTTP
  - TFTP
  - Local disk (IDE)
  - X-/YMODEM
- Non volatile configuration
- Boot script
- Wake On LAN support

## 2.2 *Bootloader Revisions*

All bootloader builds have native Linux kernel support.

Bootloader versions starting with XBD7R103_B01 have also native EXT2 filesystem support for IDE devices.

Bootlogo support is available since bootloader revision XBD7R109_B01.

Support for both RS232 interfaces is available since revision XBD7R109_B04.

The default bootloader shell interface is only COM1 since revision XBD7R110_B02.

Contact your local Kontron support if your bootloader needs to be updated.

**Note: Bootloader revision numbers ending with a _Bxx string are test versions. Where xx is an increasing number. This string is removed when the bootloader revision is regarded as stable. Do not use _Bxx versions for production!**

# 3. SETTING UP A CONNECTION TO THE TARGET SYSTEM

## 3.1 *Serial connection*

To connect to GP8 during the boot process, the boot loader displays status information and a boot menu over the COM1 serial port. You can view this information and select menu items from a Windows host using the Microsoft HyperTerminal terminal emulation application. Use a null modem cable to connect the COM1 port of the X-board<GP8> to your development workstation. This is the bottom connector on the Kontron X-board Eval Backplane. During the boot process you can also enable COM2 to provide the boot shell. This is possible by sending the string “kontron” to the port directly after reset. Have a look at the darkboot configuration paramters for more information.

To configure HyperTerminal (Windows):

- From the Windows Start menu, choose 'All Programs' and then choose 'Accessories'.
- Choose 'Communications' and then choose 'HyperTerminal'.
- In the Connection Description dialog box, in the Name box, type a name for the connection to your X-board<GP8>.
- From the Icon list, choose an icon to represent your connection and then choose OK.
- In the Connect To dialog box, in the Connect using box, choose the communications (COM) port on the development workstation through which you want to receive messages from the X-board<GP8>.
- The COM port that you choose must be the COM port on the development workstation to which you attached the null modem cable.
- Choose OK.
- In the COM<Port Number> Properties dialog box, modify the settings for your connection so that the settings are correct for your BSP (default: 38400, 8N1).

To configure for minicom (Linux):

- Install minicom if not already available
- Setup minicom (with -s option) and specify following connection parameter: ttyS0,38400,8,n,1.

```
$ minicom -s
```

  - Choose "Serial port setup" dialog
  - Select setting A and specify /dev/ttyS0,
  - Select setting E and specify GQ<Enter> (for 38400, 8N1)

- Select setting F to disable hardware flow control
- Leave the "Serial port setup" dialog
- Save as default using dialog "Save setup as dfl"
- Exit from minicom

- Execute minicom to start a terminal session

  $ minicom
- Switch on the target system. After short time the start up screen of the bootloader should appear.

The following table shows the correct settings:

| Bits per second | Data bits | Parity | Stop bits | Flow control |
|---|---|---|---|---|
| 38400 | 8 | None | 1 | None |

The output should look similar to this:

```
Looking for IP configuration from BOOTP server..
Ethernet eth0: MAC address 00:e0:4b:0d:2f:b4
IP: 192.168.1.107/255.255.255.0, Gateway: 192.168.1.1
Default server: 192.168.1.2, DNS server IP: 0.0.0.0
Ethernet console enabled on port 23 TCP

RedBoot(tm) bootstrap and debug environment [ROM]
Kontron X-board<GP8> RedBoot release
         version XBD7R105_B01
         built 16:30:51, Oct  5 2005

Platform: X-board<GP8> (XScale)
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.

RAM: 0x00000000-0x08000000, 0x0002fee8-0x07fd1000 available
FLASH: 0xf0000000 - 0xf2000000, 256 blocks of 0x00020000 bytes each.
RedBoot>
```

- Now you are able to execute commands in the bootloader shell.

  Example: RedBoot> help

**Note: You can also enable COM2 to access the bootloader and/or disable COM1 by changing the darkboot configuration.**

## 3.2 Ethernet connection

The bootloader is able to handle ethernet telnet connections if the ethernet interface of the board is configured correctly and the ethernet console is enabled. If this is the case you can simply establish a telnet connection to your board. The default telnet port is 23, but can be changed to other values with fconfig. A value of 0 disables the Ethernet console.

**Note: The default configuration of the bootloader is to listen to TCP port 23 for ethernet connections. For production use you most likely want to disable the Ethernet console, as it may be a security risk.**

# 4. FLASH USAGE

## 4.1 Overview

The X-board<GP8> is equipped with onboard memory that is used for the following purposes:

- Bootloader
- Bootloader configuration
- Space for customer specific applications

The RedBoot bootloader provides a flash management interface which allows the user to change the flash content without the need of a dedicated operating system.

## 4.2 Flash Image System (FIS)

FIS is a system that manages the available on-board flash of the X-board<GP8>. You should not access the flash without using it. Otherwise you will risk to damage important parts of the flash content which may make the board inoperable.

## 4.3 Default flash content

After execution of the "fis list" command you will get a list of the current flash content. In the default configuration of a 32 MB flash version the output will look like this:

```
RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0xF0000000  0x00100000  0x00040000  0x00100040
CEboot            0xF0040000  0x00100000  0x00040000  0x00100000
RedBoot config    0xF1FC0000  0xF1FC0000  0x00001000  0x00000000
FIS directory     0xF1FE0000  0xF1FE0000  0x00020000  0x00000000
```

**RedBoot**

The first 256 kB are reserved for the RedBoot bootloader. You should never try to erase or overwrite this region without prior consultation of our support department. If something goes wrong while updating the bootloader the board will become unbootable and you will have to send it to Kontron for repair.

### CEboot

The second 256 kB are usually used for the Windows CE Bootloader called CEboot. This bootloader is called by the “winceexec” command if a Windows CE image (NK.bin) should be started. This flash region can be removed if no CE bootloader is needed. You can update the CE Bootloader by simply overwriting the existing one.

### RedBoot configuration and FIS directory

The last 256 kB of the available flash memory are reserved for the RedBoot configuration and the FIS directory. You should never overwrite this area manually as you may render the board unbootable.

**Note: The content of the RedBoot config area is unique for every single module. Never directly copy the data from one module to another. Always use the fconfig command to change the content of this area.**

### Available space

The flash memory that is not covered by a FIS entry is available to the user and can be changed with the help of the **FIS** tools.

### Flash write protection

With hardware revision CE130 and later the flash is completely write protected at system start up. The boot loader automatically disables the write protection if needed. Usually the write protection gets enabled again after the write access. If you start an operating system which needs read write access to the onboard flash, you will have to unlock the flash before that. The easiest way to do so is to use the ‘fis unlock’ command with the appropriate parameters. You can use ‘fconfig’ to include this command in your auto-start script to automatically unlock the flash before the operating system starts.

# 5. REDBOOT COMMAND OVERVIEW

The RedBoot shell provides many commands that can help you testing the module and booting the desired operating system. You have also the possibility to use the RedBoot shell commands during manufacturing to initialize the module.

## 5.1 Getting help

**help** `help [<topic>]`

Print a list of all available commands if no argument is provided; otherwise print usage information to the given topic.

## 5.2 Hardware diagnostics

**diag** `diag`

The diag command will provide you with menu where you can start different hardware tests.

**ping**

```
ping [-v] [-n <count>] [-l <length>] [-t <timeout>]
            [-r <rate>] [-i <IP_addr>] -h <IP_addr>
```

This function is comparable to the PC command ping. It can be used to test the network connection between the onboard Ethernet adapter and an external host. You have to provide at least the IP address of the target with the –h option to use this function.

**Parameters:**

| | |
|---|---|
| -n | Number of packages to send |
| -l | Length of a single package |
| -t | Timeout value |
| -r <rate> | Delay between sending two packages in ms. 0 means as fast as possible. |
| -i <IP_addr> | Override local IP address; can be used for example if bootp has failed and no IP address has been assigned to the local network interface. |
| -h <IP_addr> | Target IP address |

## 5.3 Changing the RedBoot configuration

**alias** `alias {name} [value]`

RedBoot has the ability to manage aliases for example if some values are needed regularly. The alias can be stored in flash and is then available even after rebooting the hardware. On the console everything in the form %{name} will be replaced by [value].

After executing

```
alias loadkernel "load -r vmlinuz -b 0x100000"
```

you can simply type the command

```
%{loadkernel}
```

instead of

```
load -r vmlinuz -b 0x100000
```

**baudrate** `baudrate [-b <rate>]`

Use this command to query/set the baudrate of the system console.

**fconfig** `fconfig [-i] [-l] [-n] [-f] [-d] | [-d] nickname [value]`

This command should be used to change the RedBoot bootloader configuration. Please refer to the RedBoot Configuration chapter for details to the available configuration options.

You will have to reset the module before the changed configuration options take effect.

If no parameter is given you will be asked every possible option. If you don't want to change a specific option simply press return.

When providing an option nickname, you will only be asked for the new configuration of this option. If additionally a value is delivered then the option value will be automatically set without further question.

**Parameters:**

| | |
|---|---|
| -i | Initialize configuration, set everything to the factory default values |
| -l | List currently set values |
| -n | Show nicknames instead of full names |
| -f | Show full names (default) |
| -d | Dump terminal, no editing possible |
| -r | Only RAM, don't update flash |

**ip_address** `ip_address [-l <local_ip_address>] [-h <server_address>]`

This command enables you to change the Ethernet setup at runtime for the current session. The **–h** parameter specifies the TFTP server to be used for data download. With the **–l** option you can change the current IP address of the module. This command overrides any settings provided by an BOOTP/DHCP server.

If no option is given, the current configuration is printed.

**channel** **channel [-1|<channel number>]**

The channel command allows the user to switch between different channels. For example the serial consoles and the Ethernet console. Each console has a different channel number which can be seen for the current console when executing channel without parameters. If the **-1** parameter is provided RedBoot uses the first console that receives input.

## 5.4 *Data handling*

**cksum** `cksum -b <location> -l <length>`

The cksum function calculates a 32 bit Posix compatible checksum of the given memory area.

**disks** `disks`

This command lists all partitions recognized by RedBoot.

**load** `load [-r] [-v] [-d] [-h <host>] [-m <varies>] [-c <channel_number>] [-b <base_address>] <file_name>`

The load command can be used to copy data from a media like local disk or network to the local memory. Before loading data into the memory check that the data fits into the memory and that the memory base address is already not used. You can do this by looking at the output of the **version** command. The line starting with RAM gives you an indication about the available memory range.

Normally the command expects a file of the SREC format as source, if you want to load binary data you have to use the **–r** switch. It's also possible to load gzipped data and transparently decompress it into RAM. This is especially useful if you are loading from a slow media or if storage space is an issue.

**Warning:** **Loading images into the wrong memory location can permanently damage the module!**

**Parameters:**

| | |
|---|---|
| -r | Load raw data instead of SREC format files |
| -v | Verbose output |
| -d | Decompress data that is compressed with the gzip algorithm during download |
| -h <host> | Host address of the source server for tftp or http download. Overrides the server setting in the RedBoot configuration. |
| -m <type> | Protocoll to use for download.<br><br>disk: local disk<br><br>http: via ethernet from server<br><br>tftp: via ethernet from server<br><br>xmodem, ymodem: via serial port |
| -c <channel> | Channel number which should be used for x/ymodem download |
| -b | Memory base address where data will be copied to. |

### Loading data from an attached IDE device

You can have a look at all supported IDE partitions attached to the module by executing the disk command. The output will contain the internal device name and the partition type of the module. Currently supported partition types are FAT and EXT2.

The first IDE disk found on the bus is called hda, the second one hdb. The number at the end of the device name indicates the partition count on the device.

To load a binary file called “MyData” from the first partition of the IDE master to memory address 0x100000 you can execute the following:

```
RedBoot> disks

hda1     FAT16

RedBoot> load -m disk hda1:MyData -b 0x100000 –r

Raw file loaded 0x00100000-0x0023feff, assumed entry at 0x00100000
```

The output indicates where the file is located in the local memory. The assumed entry information is only usable for executable files like a Linux kernel. The value tells you where the bootloader will jump into the image if you execute the **go** or **exec** command.

### Loading data from a network server

The bootloader supports the HTTP and the TFTP protocol for ethernet data download. You can choose one of those protocols with the –m parameter. If you want to use an other server then the configured default server you can change the server with the –h switch. You can provide a complete source path as filename. For http downloads you always have to preface the path with a “/” character.

```
RedBoot> load -m http –h 192.168.1.1 –b 0x100000 /testfile.srec

Raw file loaded 0x00100000-0x0023feff, assumed entry at 0x00100000

RedBoot> load -m tftp –b 0x200000 –r testfile.bin

Raw file loaded 0x00200000-0x0033feff, assumed entry at 0x00100000
```

**Loading data from the serial interface**

You can use the serial port to download data if you choose either the xmodem or the ymodem protocol with the –m switch. You can change the used channel with the –c parameter if you don't want to use the current one. You don't need to provide a filename if you use the serial interface.

In order to use this feature you have to use a terminal that is able to handle the x-/ymodem protocol, for example **minicom**.

```
RedBoot> load -m xmodem –b 0x300000 –r

Raw file loaded 0x00100000-0x00102129, assumed entry at 0x00100000

xyzModem - CRC mode, 68(SOH)/0(STX)/0(CAN) packets, 8 retries
```

## 5.5 *Flash handling with FIS*

| | |
|---|---|
| fis create | **`fis create -b <mem_base> -l <image_length> [-s <data_length>] [-f <flash_addr>] [-e <entry_point>] [-r <ram_addr>] [-n] <name>`** |

This command can be used to store memory regions into the on-board flash of the X-board<GP8> and make them available under a name.

With the **–b** parameter you provide the base address of the memory written to flash. The **–l** parameter defines the size of the memory region used inside the flash memory. This size is adapted automatically to the block size of the memory.

If the length of the data you want to write to the flash is smaller then the space you have reserved for the image with the **–l** parameter you can use **–s** to define the actuall data size written to flash.

If no flash address is provided with the –f parameter the image will automatically resides at the first position inside the flash that has enough free space.

With the **–e** parameter it is possible to define or override an entry point for images that will be executed later on.

The **–r** parameter defines where the image will be located inside the memory if it is loaded with the **fis load** command.

If the name provided already exists inside the FIS directory this command will automatically update the content of the flash image.

**Note: You can't change the size of an image after it has been created.**

**Parameters:**

| | |
|---|---|
| -b <mem_base> | Memory base address of the image |
| -l <image_length> | Image length inside the flash memory |
| -s <data_length> | Length of the actual data copied to the flash |
| -f <flash_addr> | Location where the image should be created inside the flash. |
| -e <entry_point> | Entry point when executing the image |
| -r <ram_addr> | Memory address where the image should be located if it is loaded into the RAM. |
| -n | Don't touch the flash image, only update the FIS directory entry. |
| <name> | Name of the FIS directory entry |

**fis delete** `fis delete <name>`

This command deletes an image from the FIS directory and erases the content of the flash.

**Note: It is not possible to delete all images, some are protected and will issue a warning. If you want to delete the CEboot image you first will have to erase the flash region with the fis erase command, before you can delete it with fis delete.**

**fis erase** `fis erase -f <flash_addr> -l <length>`

This command will erase a flash area which starts at the memory address that is given with the **–f** parameter and has the size provided with **–l**. The size provided with **–l** will be rounded up to the block size of the flash.

**Warning: There is no crosschecking if there is an image stared at the provided address. You can make the board unbootable if you erase reserved flash areas like the configuration or bootloader images.**

**fis free** `fis free`

This command outputs the flash regions which are currently not used and available for new images.

**fis init** `fis init [-f]`

This command initializes the FIS directory. It can be used to delete all user created images. When the **–f** switch is provided it will also erase the complete flash memory excepting the reserved areas.

**fis list** `fis list [-c] [-d]`

This command provides a list with all images stored inside the FIS directory. The command also provides the Flash address, size, memory address and entry point of an image. The memory address and entry point entries are used when loading an image to the RAM and executing it.

With the **–c** switch the command displays the checksum of the images instead of the memory address field.

With the **–d** switch the command displays the amount of valid data that is actually stored inside the image.

**fis load** `fis load [-d] [-b <memory_load_address>] [-c] name`

**fis lock** `fis lock [-f <flash_addr> -l <length>] [name]`

This command locks a flash region or a FIS entry. This prevents the image to be overwritten without first unlocking it.

**Note: The fis create command automatically unlocks the flash before writing to it.**

**fis unlock** `fis unlock [-f <flash_addr> -l <length>] [name]`

This command unlocks a flash region or a FIS entry. This allows the flash region to be overwritten and erased by everyone.

**fis write** `fis write -f <flash_addr> -b <mem_base> -l <image_length>`

This command writes data from the memory address given with the –b parameter to the flash address provided with –f. The amount of data provided with –l is rounded up to the actual block size of the flash.

**Warning: fis write does not check if the flash region that is to be overwritten belongs to an image. The board will become unbootable if you overwrite reserved flash areas like the configuration or bootloader images.**

## 5.6 *Boot logo support*

**display** **display [-l] [-p -c [-s <logo>] [-r <JILI id>] [-e <1|0>]] [-w] [-u] [-t]**

The display command helps when configuring the boot logo for the X-board<GP8>.

To enable boot logo support you first have to use fconfig and enable the boot logo for the desired interface(s) and choose a display mode. The available JILI modes are listed in the following table:

| JILI ID | Mode |
|---|---|
| 0x2 | 640x480, 60 Hz |
| 0x3 | 800x600, 60 Hz |
| 0x5 | 1024x768, 60 Hz |
| 0x6 | 1280x1024, 60 Hz |
| 0x80000006 | 320x240, 60 Hz for panel Sharp LQ057Q3DC03 |
| 0x80000007 | 480x234, 52 Hz for panel Au A070FW03 |

If the table doesn't contain a display timing that works for the desired device, then a custom display mode can be configured with the display command.
The display command is also able to search the flash for a valid boot logo and permanently safe its position to flash. Before you can select a custom boot logo you first have to create it with the xbd7_logotool that is available for Microsoft Windows and Linux. After creating the logo binary you can put it into the onboard flash with the FIS commands or by adding it to an uncompressed image (for example WinCE NK.bin) which is stored inside the flash. If you store the image inside a file system you have to keep in mind that the logo binary mustn't be fragmented in order to work!

**Note: If you store the image inside a file system you have to keep in mind that the logo binary mustn't be fragmented in order to work!**

**Parameters:**

| | |
|---|---|
| -l | Search the flash for available boot logos. |
| -p | Edit settings for the panel (use alone to edit custom display timings). |
| -c | Edit settings for the CRT (use alone to edit custom display timings). |
| -s <logo> | Select logo with number <logo> to be display on a specific screen . |
| -r <JILI id> | Reset custom display timings to predefined JILI mode timings. |
| -e <0\|1> | Enable/disable display |
| -w | Store current settings to flash |
| -u | Update screens |
| -t | Show test-screen |

## 5.7 *Image execution*

**go** `go [-w <timeout>] [entry]`

The go command sets the instruction pointer to the entry point that was assumed last when loading a image with either fis load or load**.** You can change the entry point manually by providing it to the command.

With the **–w** parameter you can provide the time in seconds the the execution of the command should be delayed.

**Note: The board will lock if you execute memory regions which do not contain valid executable data.**

**exec**

```
exec [-w <timeout>] [-b <load addr> [-l <length>]]
[-r <ramdisk addr> [-s <ramdisk length>]]
[-c "kernel command line"] [<entry_point>]
```

This command executes an image with the MMU turned of. This can be used to start Linux kernels. It is also possible to pass a kernel command line and an INITRD ramdisk to the kernel. If no entry point is provided the command will use the entry point assumed by the last load command.

With the **–w** parameter you can provide the time in seconds for that the execution of the command should be delayed.

| **Parameters:** | | |
|---|---|---|
| | -w <timeout> | Delay before the execution will be started in seconds. It is possible to stop the execution by sending “CTRL-c” to the terminal. |
| | -b <load addr> | Start address of the image inside the memory |
| | -l <length> | Length of the image |
| | -r <ramdisk addr> | Start address of the INITRD ramdisk |
| | -s <ramdisk length> | Size of the INITRD ramdisk |
| | -c “kernel command line” | Linux kernel command line string |
| | <entry point> | Entry point |

```
RedBoot> fis load linux

RedBoot> exec -c "root=/dev/mtdblock2 rootfstype=jffs2"

Using base address 0x00100000 and length 0x00150000

The boot tags are located at 0xA0000100

Booting the kernel...

Uncompressing Linux........................ done, booting the kernel.
```

**Note: The board will lock if you execute memory regions which do not contain valid executable data**

**winceexec** `winceexec [-v] [-n] [-t] [-s] [-o] [-F] [-S] [[-L <device>] [-p <path>]]`

This command starts the Windows CE bootloader which then executes a CE image. This command will fail if no CEBoot is stored at the predefined position inside the flash.

Please look at the X-board<GP8> Windows CE BSP Manual for additional information regarding WinCE and CEBoot.

| **Parameters:** | | |
|---|---|---|
| | -v | Show CEBoot version |
| | -n | No bootloader menu delay |
| | -t | Long bootloader menu delay |
| | -s | Execute boot menu |
| | -o | Show debug messages |
| | -F | Write image into onboard flash after download |
| | -L <device> | Load image from <device><br>Valid choices are:<br>flash: onboard flash<br>cf: CF card, configured as primary master<br>eth: Ethernet download from Microsoft Windows CE Platform Builder |
| | -S | Swap WinCE COM ports to get the bootloader messages from COM2 instead of COM1 |

```
RedBoot> winceexec -L cf

Booting WinCE...
```

**Note: The board will lock if you execute this command without providing a valid CEboot.**

## 5.8 *Special functions*

**jida** `jida`

The jida command retrieves module specific information, like serial number, manufacturing date and so on.

**poweroff** **poweroff**

This function initiates a soft power off.

**reset** `reset`

This function resets the system.

**version** `version`

This command provides RedBoot version information as well as information about the memory configuration.

# 6. REDBOOT CONFIGURATION

The RedBoot runtime behaviour can be adjusted via the fconfig command. The following table contains all available configuration options:

| **Nickname** | **Full Name** | **Default** | **Description** |
|---|---|---|---|
| boot_script | Run Script at boot | true | Aktivate this to start a boot script at each startup. The execution can be interrupted when sending "CTRL-c" to the terminal. |
| boot_script_data | Enter script | | Everything entered here will be executed at startup.<br>Only available when `boot_script` is set to true. |
| boot_script_timeout | Boot script timeout | 0 | Delay in seconds before the boot-script gets executed. If set to zero you will have to send the break signal directly after powerup to prevent the bootloader from executing the script.<br><br>Only available when `boot_script` is set to true. |
| bootp | Use BOOTP for network configuration | true | Try to get the Ethernet configuration via BOOTP. |
| bootp_my_gatway | Gateway IP address | 0.0.0.0 | Address of the Ethernet gateway for contacting servers that are not in the local network.<br>Only available when `bootp` is set to false. |
| bootp_my_ip | Local IP address | 0.0.0.0 | IP address of the onboard Ethernet interface.<br>Only available when `bootp` is set to false. |
| bootp_my_ip_mask | Local IP address mask | 0.0.0.0 | Netmask of the local network.<br>Only available when `bootp` is set to false. |
| bootp_my_server_ip | Default server IP address | 0.0.0.0 | IP address of the local TFTP server.<br>Only available when `bootp` is set to false. |
| console_baud_rate | Console baud rate | 38400 | Baudrate of the serial console. |
| darkboot | Enable darkboot | true | If this feature is set to true the serial console will be disabled |

| | | | |
|---|---|---|---|
| | | | at the next startup. This feature can be disabled by sending the escape key into the serial terminal directly after power up or with fconfig via the ethernet console. You also have to configure the COM ports for which this feature should be active. |
| `darkboot_com1` | Enable darkboot for COM1 | `false` | Enable darkboot for COM1. Only available when darkboot is set to true. |
| `darkboot_com2` | Enable darkboot for COM2 | `true` | Enable darkboot for COM2. Only available when darkboot is set to true. |
| `darkboot_key` | Darkboot escape key. | `kontron` | Access key to temporarily disable darkboot. The key can have a maximum length of 15 characters. **If the key is empty darkboot can not be disabled!** Only available when darkboot is set to true. |
| `dns_ip` | DNS server IP address | `0.0.0.0` | IP address of a DNS server for resolving host names. |
| `gdb_port` | GDB/Telnet connection port | `23` | Port for Ethernet connections. To use this feature the Ethernet port has to be configured correctly. Use 0 to disable this feature. |
| `gfx_crt_enable` | Enable CRT bootlogo | `false` | Enable boot logo on the CRT interface |
| `gfx_crt_jili` | CRT JILI id | `0x2` | Kontron JILI id of the predefined display timing (default is VGA timing) |
| `gfx_crt_jili_custom` | CRT use custom values | `false` | Use custom timing instead of JILI modes. Timing values can be set with the display command. |
| `gfx_crt_logo_addr` | CRT logo address | `0x0` | Address of the logo file inside the flash. 0x0 uses the default logo. |
| `gfx_panel_enable` | Enable panel bootlogo | `false` | Enable boot logo on the panel interface |
| `gfx_panel_jili` | Panel JILI id | `0x2` | Kontron JILI id of the predefined display timing (default is VGA timing) |
| `gfx_panel_jili_custom` | Panel use custom values | `false` | Use custom timing instead of JILI modes. Timing values can be set with the display command. |
| `gfx_panel_logo_addr` | Panel logo address | `0x0` | Address of the logo file inside the flash. 0x0 uses the default logo. |
| `ide_spinup_wait` | IDE drive max. spinup wait | `2` | Max. delay when waiting for an IDE device to get ready. |
| `wol_enable` | Enable Wake On LAN | `false` | Enables Wake On LAN with the next reboot. |

| | |
|---|---|
| | Wakes the board if a magic packet is received. |

**Note: Be careful when changing the console baud rate. If you use values your terminal can't handle you won't be able to get a serial console anymore. You should always enable the Telnet connection port as fallback when changing the baud rate the first time.**

**Note: When using the darkboot feature you can lock yourself out permanently if you forget or mistype the escape key! You should always enable the Telnet connection port as fallback when enabling darkboot the first time. If you don't provide an escape key you won't be able to use the COM ports for which the darkboot feature is enabled anymore until the feature is disabled via another communication channel (Telnet, or other COM port)!**

# 7. UPDATING THE WINCE BOOTLOADER

## 7.1 *Installing/Updating CEboot*

In this step by step instructions we assume that you are using the Kontron Eval backplane and have a CF card ready. If this is not the case you have to adapt the steps 1, 4, 5 and 6 to your needs.

1. Copy the bootloader file BOOT.nb0 on a CF card with either FAT or EXT2 file system.
2. Insert the CF card into the CF card slot of the Eval Backplane
3. Set up a connection to the X-board<GP8>
4. Execute the "disk" command

   Now you should get a list of all usable IDE partitions. For a CF card plugged into the CF card slot and having one partition it should show you a entry with the name hda1. There will be more entries with increased number if the CF card has more then one partition. If you have connected a slave IDE disk that one will show up as hdbX
5. Load the WinCE bootloader into the target RAM by executing the following command:

   ```
   load -m disk hda1:/BOOT.nb0 -r –v -b %{FREEMEMLO}
   ```

   If necessary, hda1 should be adapted to the desired source partition according to the information retrieved  in point 4. If your bootloader has the **.gz** extension you have to add the **–d** option to the command line.
6. Write the WinCE bootloader into the flash be executing the following command:

   ```
   fis create -b %{FREEMEMLO} -l 0x40000 -f 0xf0040000 CEBoot
   ```

   If this command fails you have to check if there is already a FIS image in the memory region reserved for CEboot. If this is the case you will have to delete that image with the fis delete command first.
7. Now the new Windows CE bootloader is permanently stored on the board and can be executed by invoking the "winceexec" command.

**Note: If you previously erased the CEBoot and now have reinstalled it, it won't be automatically protected by the bootloader. To enable the CEBoot protection again you have to reinitialize the flash with** fis init**, which will erase all user data from flash.**

## 7.2 *Erasing CEBoot*

If you don't use Windows CE you may want to free the flash memory that is used by CEboot. As the CEboot flash image is protected by the bootloader you can't simply use the fis delete command to remove the image.

The first of all you should check if CEboot is really stored on the right memory location by invoking the fis list command and checking if the FLASH address of CEBoot is 0xF0040000. If this is not the case it should be possible to delete CEBoot by simply executing "`fis delete CEBoot`" and no further steps have to be taken. Otherwise you will have to reinitialize FIS, which causes all user data inside the onboard flash to be lost!

In order to remove the CEBoot from the FIS directory you have to erase CEBoot manually with the fis erase command. This will not remove the FIS entry of CEBoot. To get rid of the entry you have to execute the “fis init” command.

**Warning: All user data inside the onboard flash will be erased when deleting CEBoot!**

```
Name                FLASH addr   Mem addr     Length       Entry point

RedBoot             0xF0000000   0x00100000   0x00040000   0x00100040

CEBoot              0xF0040000   0xF0040000   0x00040000   0x00100000

RedBoot config      0xF1FC0000   0xF1FC0000   0x00001000   0x00000000

FIS directory       0xF1FE0000   0xF1FE0000   0x00020000   0x00000000

RedBoot> fis unlock CEBoot

... Unlock from 0xf0040000-0xf0080000: ..

RedBoot> fis erase -f 0xf0040000 -l 0x40000

... Erase from 0xf0040000-0xf0080000: ..

RedBoot> fis init -f

About to initialize [format] FLASH image system - continue (y/n)? y

*** Initialize FLASH Image System

... Erase from 0xf0040000-0xf3fc0000: ...............................

... Erase from 0xf1fe0000-0xf1fe0000:

... Unlock from 0xf1fe0000-0xf4000000: .

... Erase from 0xf1fe0000-0xf4000000: .

... Program from 0x07fdf000-0x07fff000 at 0xf1fe0000: .

... Lock from 0xf1fe0000-0xf4000000: .

RedBoot> fis list

Name                FLASH addr   Mem addr     Length       Entry point

RedBoot             0xF0000000   0xF0000000   0x00040000   0x00000000

RedBoot config      0xF3FC0000   0xF3FC0000   0x00001000   0x00000000

FIS directory       0xF3FE0000   0xF3FE0000   0x00020000   0x00000000
```

# 8. RELATED LINKS AND DOCUMENTATION

- eCos License, http://ecos.sourceware.org/ecos/docs-2.0/user-guide/ecos-licensing.html
- General Public License, http://www.gnu.org/copyleft/gpl.html
- RedBoot bootloader, http://sources.redhat.com/redboot/

# 9. DOCUMENT REVISION HISTORY

| Revision | Date | Edited by | Changes |
|---|---|---|---|
| 0.1 | 14.11.05 | BMI | 1st preliminary version |
| 0.2 | 13.02.06 | BMI | Added WOL and CEBoot description |
| 0.3 | 02.06.06 | BMI | Added bootlogo description |
| 0.4 | 25.07.06 | BMI | Updated winceexec parameter list<br>Updated darkboot configuration parameters |
| 0.5 | 12.10.06 | BMI | Removed wake on link status change<br>Described flash write protection<br>Default RS232 shell interface has changed |
| 0.6 | 01.12.06 | CMO | Added disclaimer and correct X-board trademark |