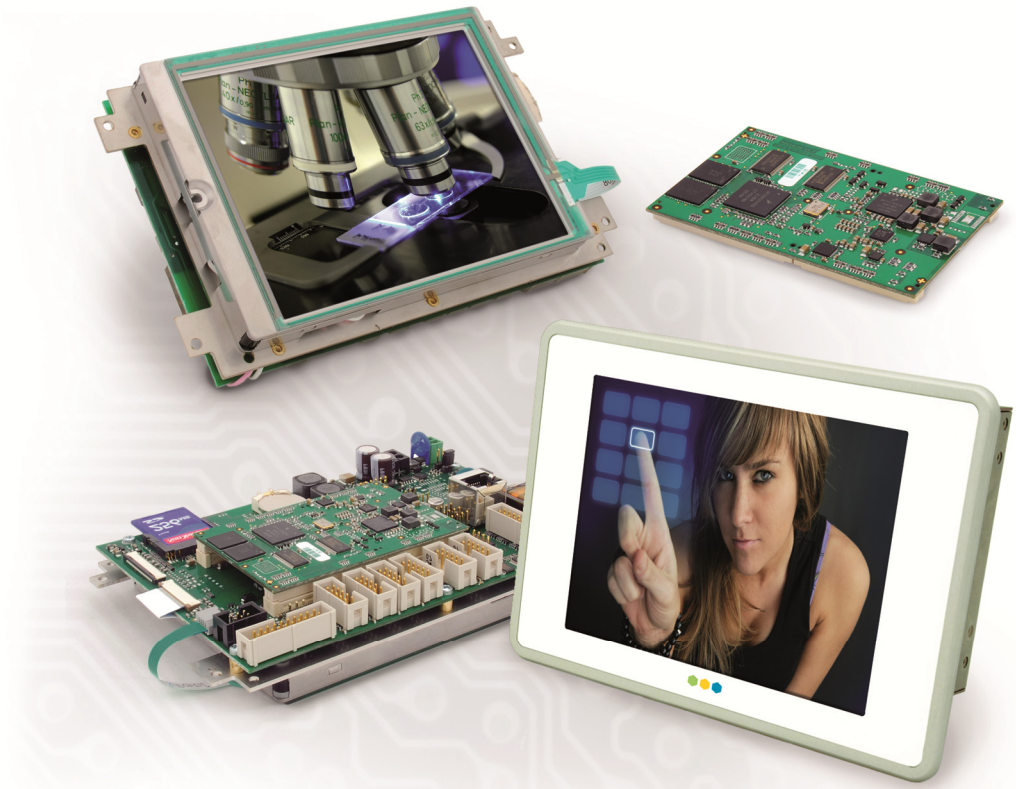
Garz & Fricke

Embedded Systems

Garz & Fricke

Embedded Computer Systems



RedBoot · User Manual

freescale™ Enabled

Windows® Embedded Silver Partner

Zuverlässige Qualität Made in Germany

Important hints

Thank you very much for purchasing a Garz & Fricke product. Our products are dedicated to professional use and therefore we suppose extended technical knowledge and practice in working with such products.

The information in this manual is subject to technical changes, particularly as a result of continuous product upgrades. Thus this manual only reflects the technical status of the products at the time of printing. Before design-in the device into your or your customer's product, please verify that this document and the therein described specification is the latest revision and matches to the PCB version. We highly recommend contacting our technical sales team prior to any activity of that kind. A good way getting the latest information is to check the release notes of each product and/or service. Please refer to the chapter [Online Support].

The attached documentation does not entail any guarantee on the part of Garz & Fricke GmbH with respect to technical processes described in the manual or any product characteristics set out in the manual. We do not accept any liability for any printing errors or other inaccuracies in the manual unless it can be proven that we are aware of such errors or inaccuracies or that we are unaware of these as a result of gross negligence and Garz & Fricke has failed to eliminate these errors or inaccuracies for this reason.
Garz & Fricke GmbH expressly informs that this manual only contains a general description of technical processes and instructions which may not be applicable in every individual case. In cases of doubt, please contact our technical sales team.

In no event, Garz & Fricke is liable for any direct, indirect, special, incidental or consequential damages arising out of use or resulting from non-compliancy of therein conditions and precautions, even if advised of the possibility of such damages.

Before using a device covered by this document, please carefully read the related hardware manual and the quick guide, which contain important instructions and hints for connectors and setup.

Embedded systems are complex and sensitive electronic products. Please act carefully and ensure that only qualified personnel will handle and use the device at the stage of development. In the event of damage to the device caused by failure to observe the hints in this manual and on the device (especially the safety instructions), Garz & Fricke shall not be required to honour the warranty even during the warranty period and shall be exempted from the statutory accident liability obligation. Attempting to repair or modify the product also voids all warranty claims

Before contacting the Garz & Fricke support team, please try to help yourself by the means of this manual or any other documentation provided by Garz & Fricke or the related websites.
If this does not help at all, please feel free to contact us or our partners as listed below.
Our technicians and engineers will be glad to support you. Please note that beyond the support hours included in the Starter Kit, various support packages are available. To keep the pure product cost at a reasonable level, we have to charge support and consulting services per effort.

Shipping address:
Garz & Fricke GmbH
Tempowerkring 2
21079 Hamburg
Germany

Support contact:
Phone +49 (0) 40 / 791 899 - 30
Fax +49 (0) 40 / 791 899 – 39
Email ▶ support@garz-fricke.com
URL ▶ www.garz-fricke.com

Content

1 Introduction

Garz & Fricke systems based on Freescale i.MX use an adapted version of Red Hat's RedBoot ROM Monitor as BIOS. RedBoot is a royalty-free open-source embedded system debug/bootstrap environment from Red Hat providing a complete bootstrap environment for all operating systems supported by Garz & Fricke i.MX systems. RedBoot is based on the hardware abstraction layer of the open-source operating system eCos.
This manual contains information on the usage of RedBoot on Garz & Fricke systems, modifications and extensions to RedBoot introduced by Garz & Fricke and the interface between RedBoot and operating systems. For further documentation on RedBoot and eCos please refer to the official eCos web-sites:

- http://ecos.sourceware.org
- http://www.ecoscentric.com

RedBoot, eCos and all modifications and extensions provided by Garz & Fricke are released under a modified version of the GNU General Public License v2 (GPLv2). For details about the license please refer to Annex B.

RedBoot as provided by Garz & Fricke is based on extensions by Freescale Semiconductor that currently have not been integrated into the standard eCos distribution. Garz & Fricke has made some larger incompatible changes to key components of RedBoot. Those modifications are currently not contributed back to the standard eCos distribution. The full source-code is available from Garz & Fricke, though.

2 Using RedBoot

Garz & Fricke systems are usually shipped in a configuration that will automatically load and start an operating system pre-installed in flash-memory. To get access to RedBoot in this 'default' configuration either the automatic boot-process must be cancelled by the user (described below) or the automatic boot-process must be disabled from the OS, if supported. If and how this can be done is documented in the manual of the respective operating system.

2.1 Boot-up sequence

The most common boot-up sequence is separated into two steps:

- At first the RedBoot BIOS is loaded from flash into RAM and executed to initialize the basic system functions.
- RedBoot then usually loads the operating system from flash into RAM and executes it (controlled by a user-configurable boot-script).

Some early embedded devices may be equipped with both, NOR and NAND flashes. RedBoot can only handle one kind of flash at a time. Therefore a command (factive) is provided to switch between different kinds of flash and restart RedBoot. On these systems the boot-up sequence may comprise three steps:

- RedBoot BIOS is loaded from flash (e.g. NOR-flash) into RAM and executed to initialize the basic system functions.
- Controlled by a user-configurable boot-script, RedBoot then may switch to a different kind of flash (e.g. NAND) and restart itself from RAM (optional)
- RedBoot then usually loads the operating system from flash into RAM and executes it (controlled by a user-configurable boot-script).

The user-configurable boot-script and all other configurable settings are always read from the currently active flash. Therefore on systems equipped with both NOR and NAND flash, different boot-scripts (and other settings) may be stored in NOR and NAND flash and will be used as soon as the respective flash-type is activated.

2.2 Connecting embedded devices

RedBoot currently supports connections via serial debug port using a terminal program or via Telnet-protocol over Ethernet. The available connection types may vary on the respective system and may also depend on the current configuration of the system.
Independent of the kind of connection used for the console (terminal program via RS-232 or Telnet via Ethernet), file uploads to the system can be done via RS-232 or Ethernet, too. So it is possible to control RedBoot via terminal program over RS-232, but transfer files via Ethernet, or vice versa.

A serial connection is usually used to configure settings, update the BIOS or display debugging messages, but can also be used to upload operating system images or other files. Due to the low transfer rate of a serial connection, it is recommended to use Ethernet transfers for larger files, though.

2.2.1 Connection via serial port (RS-232)

All RedBoot-based systems shipped by Garz & Fricke support connections via serial RS-232 port. On some few systems this may be a designated serial debug port, which is otherwise not available for applications. On most systems, though, the first regular serial port (RS232-1) doubles as serial debug port. The serial debug console on such systems can be turned on and off [▶ 5.2 Configuration settings] with a user-settable variable. By default, the serial debug console is enabled. If disabled, during the boot-sequence, no outputs will be sent to the port and no connection to RedBoot via the serial port is possible. Instead the respective serial port will be completely available to applications, like any other serial port of the system.

To establish a serial connection, connect a serial Null-Modem cable between the serial debug port of your system – and a COM port of your PC and start a terminal program (e.g. HyperTerminal, Tera Term, Minicom, Eltima Advanced Serial Terminal) on your PC for the respective COM port.

Set the communication parameters for the connection to the following values:

Serial connection settings	
Speed [Baud]	115200
Data bits	8
Stop bits	1
Parity	None
Flow control	None

If you now power-on your Garz & Fricke system you will see boot-up messages from RedBoot – assuming the serial debug console has not been disabled – similar to the following output:

```
Found NIC LAN9218 Rev. 0
Using device name: "GFMM668348"
Found PHY LAN92xx PHY Rev. 3
Ethernet eth0: MAC address 00:07:8e:0a:32:bc
IP: 172.20.102.159/255.255.0.0, Gateway: 172.20.0.13
Default server: 172.20.0.13

Clock input is 26 MHz
Booting from [NAND flash]
[0x04000000 bytes]: 4096 blocks of 32 pages of 512 bytes each.


RedBoot(tm) bootstrap and debug environment [ROMRAM]
Non-certified release, version G&F 1.5.7298 - built 16:26:25, Oct 25 2007

Platform: Garz&Fricke ADELAIDE (Freescale i.MX31 based) PASS 1.1 [x32 DDR]
Copyright (C) 2000, 2001, 2002, 2003, 2004 Red Hat, Inc.
Copyright (C) 2003, 2004, 2005, 2006 eCosCentric Limited

RAM: 0x00080000-0x08000000, [0x00094220-0x000f4000] available
FLASH: 0xe0000000 - 0xe4000000, 4093 blocks of 0x00004000 bytes each.
== Executing boot script in 0.000 seconds - enter ^C to abort
```

Note: If a boot script has been configured [▶ 5.2 Configuration settings], e.g. to automatically load and execute an operating system, this can be cancelled with the key-combination **Ctrl-C** (**Strg-C**) on your work station, not on the device itself.

2.2.2 Connection via Ethernet (TCP/IP)

An Ethernet connection may be used to control RedBoot using the Telnet protocol similar to a serial connection even if the serial debug console has been turned off. Ethernet connections are preferred for large file transfers such as operating system images.

To establish an Ethernet connection, the system must be connected to the LAN via Ethernet cable – either to a switch, a hub or directly to a PC (the last may require a cross-over cable).
The network configuration of the Garz & Fricke system may have to be re-configured to match your network (pls. see [▶ 5.2 Configuration settings]). This can be done in multiple ways:

- Using a serial connection [▶ 2.2.1 Connection via serial port (RS-232)]
- Using configuration files or dialogs of the operating system booting by default on the system. (Please refer to the respective OS documentation if and how the operating system does support this.)
- Connecting the system via LAN directly to a PC that has been configured to network settings compatible with the default network settings.

All Garz & Fricke systems are shipped with a pre-configured static IP configuration:

TCP/IP configuration	
IP address	192.168.1.1
Netmask	255.255.255.0
Gateway	192.168.1.100
DHCP/BOOTP	off

Once the Ethernet configuration has been setup, a Telnet connection to the system can be established by executing the command “telnet <IP-address>” on your PC.
File transfers via Ethernet use the TFTP-protocol and require a TFTP-server running on your PC (or any other PC available on the network) and are established via the RedBoot command “load”.

Note: To speed up to boot-process, RedBoot does not initialize the Ethernet port when a boot-script is activated. Therefore it is not possible to cancel a boot-script via Telnet connections. To enter RedBoot mode, please use RS-232 connection.

2.3 Transferring files

Files can be transmitted to RedBoot either via serial communication or via Ethernet. The respective connection has to be setup properly as described in the previous sections.

2.3.1 Uploading files via serial communication (RS-232)

To upload files, a terminal program supporting the XMODEM-protocol is required (most terminal programs provide such a function). After a serial connection has been established, enter the following command on the RedBoot command line:

```
load -r -v -b <address in RAM> -m xmodem <filename>
```

After entering this command RedBoot will wait for the start of the upload. Now send the desired file using your terminal program (for all usable address-ranges in RAM refer to [► 4 RedBoot Memory Layout]. The address 0x80100000 will be supported on all Garz & Fricke devices).

2.3.2 Uploading files via Ethernet (TCP/IP)

To upload a file via Ethernet, a TFTP-server must be available in your network. Usually such a program will be installed on the local PC. Depending on the TFTP server application, different configuration settings must be made. Please check the following:

- A TFTP root directory has been configured in your TFTP server, i.e. *C:\tftp-root*
- All files to be transmitted have been placed into this TFTP root directory
- The security settings of your TFTP server have been configured to allow at least sending/transmitting of files.
- The security settings of your TFTP server have been configured to allow connections from the IP address of your Garz & Fricke system
- Any firewalls between your TFTP server and the device allow TFTP transfers.

Once your TFTP-server and the network configuration of RedBoot have been setup properly, a file can be uploaded by the RedBoot command:

```
load -r -v -b <address in RAM> -h <IP of your TFTP-server> <filename>
```

While the file is being transmitted a rotating line below the RedBoot prompt will show activity.

Some TFTP-servers (or specific versions of a TFTP-server) do not fully comply with the TFTP protocol. If a TFTP-transfer fails to start or stops without reason although the TFTP-server and RedBoot network configuration are correct, please try a different TFTP server.
There are several free TFTP-servers available that work together with RedBoot, like “TFTPUtil”, “TFTPServerMT” or “SolarWinds TFTP-Server”. Please note, though, that there might be compatibility problems with some TFTP-servers and RedBoot for download images of 32 MB or larger.

3 Common Operations

This chapter provides some samples of how to perform common operations, such as updating the OS or changing the boot logo.

3.1 RedBoot update

To update to a newer RedBoot release, establish a connection to the current RedBoot on your system first, either via serial port or Ethernet. Then execute the following sequence:

Via serial

```
factive nand
load -r -v -b 0x80100000 -m xmodem redboot.bin
exec
romupdate
```

Via Ethernet

```
factive nand
load -r -v -b 0x80100000 -h <TFTP-Server IP> redboot.bin
exec
romupdate
```

If you have a system that is booting from NOR-flash, execute the sequence twice: once with “factive nand” and once with “factive nor”.

Note: When updating from a RedBoot release prior to v1.4 to a release v1.4 or newer, the fis- and configuration data has to be re-initialized by the following sequence:

```
fis init
fconfig
```

On systems booting from NOR-flash this sequence also has to be executed twice: once after executing “factive nand” and once after “factive nor”. Please be aware that during an update from an earlier version to v1.4+ the Ethernet MAC-address will get reset to the default value and has to be manually set to the proper value (“0x00:0x07:0x8e:<Serial# of Mini-Module>”) as described in 5.2 Configuration settings].

3.2 Operating system update/installation

Although it is possible to update an OS completely via serial connection, it is recommended that you have a properly setup Ethernet configuration and a TFTP-server running on your PC for the actual file-transfer, due to the size of OS images. Nevertheless you need a properly working RS-232 connection to your work station to enter RedBoot mode for the required operations [▶ 2.2.1 Connection via serial port (RS-232)].

While the name of the fis partition used to store the OS image into is arbitrary, “OS” is used for simplicity. It also doesn’t matter where in (NAND-)flash the OS image is stored. Sample addresses and sizes are used for explanation, but these can be changed according to your needs, e.g. to install multiple OS images, larger/smaller OS images, or larger/smaller flash-disks. Therefore it might be feasible to name a FIS partition used for an OS image according to the OS stored inside, like “WinCE 5.0”, “WinCE 6.0”, or “Linux”.

Unless otherwise ordered, all systems shipped by Garz & Fricke will usually have an OS installed in a partition called “OS”. The common case will be to update that OS. To do so, please delete the existing OS first by executing:

```
fis delete OS
```

Note: After confirmation the OS is deleted. This condition is critical, because in case of a loss of power, the boot script will attempt to load the OS and hang up in RedBoot. In this case you have to do a hard reset, please refer to [▶ 5.4.3 Resetting configuration variables by hardware (RedBoot v1.9 and earlier)] and [▶5.4.4 Getting control over a miss-configured system (RedBoot v.10 and later)].

Please continue with the instructions of the next chapter.

3.2.1 Installing a Windows Embedded CE image

To install a Windows Embedded CE OS image, establish a serial connection with the RedBoot running on your system [▶ 2.2.1 Connection via serial port (RS-232)]. Ensure that you have a TFTP-server running on your PC and that the OS image you want to upload to your device is in your TFTP-root directory. If you already have an OS installed on your device, delete this first as shown above. Now execute the following commands:

```
load -r -v -b 0x80100000 -h <TFTP-serer IP> nk.bnx
fis create -b 0x80100000 -l 0x01f00000 -s 0x01f00000 -e 0x80100000 -r 0x80100000 OS
```

Note: Filename (“nk.bnx”), partition-name (“OS”), and length (“-l 0x01f00000 –s 0x01f00000”) may have to be changed for your OS image.

You also may add the “-f”-option to explicitly specify where in flash the OS image should be written to, e.g. “-f 0xe0100000”. If the “-f”-parameter is omitted (recommended usage), RedBoot will write the OS image to the first free area in flash providing enough space.
Windows-CE images compatible to RedBoot releases prior to v1.4 reserved one half of the NAND flash as flash disk, which could not be changed. Therefore RedBoot releases prior to V1.4 only had access to half of the NAND flash.

Starting with RedBoot v1.4 RedBoot has access to the whole NAND flash and Windows-CE images compatible with this release require that a fis-partition is created for the flash-disk. Therefore starting with RedBoot v1.4 it is possible to change location and size of the flash disk, as well. The name of the fis-partition must be the same name as the mount-point in Windows-CE (usually “NAND Flash”). To create a flash-disk partition in the same configuration as used by prior RedBoot releases, please execute the command:

```
fis create -l 0x02000000 -n “NAND Flash”
```

Note: The length of 0x02000000 bytes (32 MB) given here is just an example, which should work on all RedBoot-based systems from Garz & Fricke. Most systems will ship with a NAND flash, which allows for much larger flash-disk partitions.

Note: Some Windows-CE images, e.g. on all Garz & Fricke systems using i.MX35 SoCs, expect the name of the flash-disk partition to be “FlashDisk” instead of “NAND Flash”. Also pay attention to white-spaces and case-sensitivity of the partition name. If a Windows-CE image fails to find its flash-disk, it will issue a warning or error message on the debug-console specifying the name of the partition is was looking for, though.

As with the OS image itself, you may also use the “-f”-parameter to explicitly specify where in flash the partition will be placed.

(Windows-CE releases compatible with RedBoot v1.4+ do not make any assumptions about the location of the OS image or the flash-disk in NAND flash. Provided enough space, you may even install multiple, probably different Windows-CE images in flash or setup a multi-boot system with other operating systems in flash, as well).

Note: Windows-CE images built since May 2010 may additionally require a second flash-disk partition for storing the Windows-CE Registry. This partition must be called “Registry”, should have a length of at least 0x00600000 bytes (6 MB) and may be created similarly to the “NAND Flash” partition.

Garz & Fricke systems with i.MX35 SoCs usually store the Windows-CE Registry not in flash but in the onboard battery-backed SRAM. These systems don’t need the second “Registry” flash-disk partition. If an additional “Registry” flash-disk partition is created on these systems, this will not cause any problems, but as this partition won’t be used, it wastes space on the NAND flash that may be used for other purposes, e.g. a larger “FlashDisk”-partition.

3.2.2 How to start an OS directly

When developing OS images yourself, e.g. when using a Garz & Fricke Binary-BSP for Windows-CE, or when creating a customized Linux-image, it is inconvenient to update the OS image in flash each time you make some changes. In this case it is much easier to turn-off the boot-script configured to load the OS automatically from flash, and load and execute OS images directly. This can be accomplished similarly to an OS-Update (you should have a TFTP-server setup for this, as well):

```
load -r -v -b 0x80100000 -h <TFTP-server IP> nk.bnx
exec
```

This will execute the OS image directly from RAM without storing it in flash first. Note that in this way even OS images may be used that are (currently) too large to fit into flash, e.g. when additional debugging code is included.

3.2.3 How to automatically boot an OS

To automatically boot an OS (default configuration when shipping from Garz & Fricke) a boot script must be configured in RedBoot. To do so, please establish a connection the RedBoot on your system, first. If you have a system that starts booting from NOR-flash you have to ensure that you have configured a boot-script in NOR-flash that switches to NAND-mode (On systems booting directly from NAND flash this step is not necessary.):

```
RedBoot> factive nor
[...]
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> factive nand
>>
Boot script timeout (1000ms resolution): 2
[...]
```

When asked if the RedBoot non-volatile configuration should updated please confirm.
Now, in NAND flash, a boot script must be configured to load and start the OS image. It is assumed that you already have installed an OS image in a partition called “OS”.

```
RedBoot> factive nand
[...]
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis load OS
>> exec
>>
Boot script timeout (1000ms resolution): 2
[...]
```

Again, when asked if the RedBoot non-volatile configuration should be updated please confirm.
Now, when resetting or powering on your system, the OS will be loaded and started automatically.

3.3 How to set up a boot logo

Starting with v1.8 RedBoot, it is possible to display a boot-logo, which will be shown until the display driver in the OS takes over control (provided the OS supports the boot-logo, as well).

The following examples assume a JUPITER/ADELAIDE with 5.7” Kyocera VGA display. Other systems will require a different display configuration file but will work the same way.

We show sample commands with uploads via serial port using XMODEM protocol. TFTP via Ethernet, as shown in other sections, can be used just as well.

3.3.1 First time set up

If you want to set up a boot logo on a system for the first time, a couple of files are required:

- A RedBoot v1.8 or later that supports boot logos (please check the release notes of your product as per chapter [▶ 10 Online support])
- An OS that supports boot logos (please check release notes [▶ 10 Online support])
- An XML-configuration file containing the display-configuration for your system
- An XML-configuration file containing the boot-logo license (on RedBoot v1.9+ only required if custom logos not distributed by Garz & Fricke are to be displayed)
- A boot logo in PNG-format

All of these files are available from Garz & Fricke. We assume that you already have installed the RedBoot and OS releases on your device. Please refer to [▶ 3.1 RedBoot update] and/or [▶ 3.2 Operating system update/installation].

Three steps must be executed to install everything else - the order is not important, though:

- Upload and install the display configuration:

```
load -r -v -b 0x80100000 -m xmodem rbdisp-kyocera-vga-tcg057vg1ac.xml
xconfig import -b 0x80100000
```

- Upload and install the boot-logo license (may be skipped on RedBoot v1.9+ if boot logos from Garz & Fricke are to be displayed):

```
load -r -v -b 0x80100000 -m xmodem rb-logolicense.xml
xconfig import -b 0x80100000
```

- Upload and install the boot-logo itself:

```
load -r -v -b 0x80100000 -m xmodem <filename.png>
fis create -b 0x80100000 -l <filesize> logo.png
```

After a power-cycle the boot logo should be displayed. If no logo is shown, error messages on the serial debug console should tell if the display configuration is missing or invalid, the license is missing or invalid, or the boot-logo file is corrupted. Please also verify (with “fis list”) that the FIS partition containing the logo does have the name “logo.png”; other names for the logo are not supported.

3.3.2 Changing an existing boot logo

If you have a system that already displays some boot-logo, changing to a different logo works quite similar to updating an OS (we assume upload via serial-port, here; uploads via Ethernet work just as well):

- Delete the current boot-logo

```
fis delete logo.png
```

- Upload the new logo

```
load -r -v -b 0x80100000 -m xmodem <filename.png>
```

- Create new logo partition in flash

```
fis create -b 0x80100000 -l <filesize> logo.png
```

- Power-cycle the system and the new logo will be shown

3.3.3 Disabling the boot-logo

To turn-off the boot-logo, just delete the FIS partition of the logo:

```
fis delete logo.png
```

RedBoot will still initialize the display, but just show an empty (black) screen.

3.3.4 Installing a boot logo license

Starting with RedBoot v1.9 boot logo images distributed by Garz & Fricke do not require a boot logo license. Therefore systems may be setup and shipped with a boot logo but without a boot logo license that allows custom PNG images to be used as boot logo. To be able to display PNG files not distributed by Garz & Fricke on such a system a boot logo license must be obtained from Garz & Fricke and installed on the device:

- Upload and install the boot-logo license (necessary to display PNG-files not distributed by Garz & Fricke):

```
load -r -v -b 0x80100000 -m xmodem rb-logolicense.xml
xconfig import -b 0x80100000
```

4 RedBoot Memory Layout

RedBoot on Garz & Fricke systems runs with MMU and caches enabled. While different systems may have different physical memory layouts, RedBoot configures a virtual memory layout that looks similar on all platforms, i.e. all addresses used in sample-commands throughout this document will work on all RedBoot-based systems shipped by Garz & Fricke.

All RedBoot commands expecting address-parameters use virtual addresses (V-address in the following table).

Before executing an OS (or application) image, the MMU and caches are disabled.

4.1 Virtual Address Space on i.MX25-, i.MX31- and i.MX35-based Systems

The following table shows the mapping of virtual to physical address space on systems using Freescale i.MX25, i.MX31, or i.MX35 SoCs for RedBoot V1.12 (in previous versions some of these areas might be unmapped).

V-Address	P-Address	Size	Description	Attributes[1]
0x00000000	0x80000000	1 MB	SDRAM	C/B
0x00100000	n/a	255 MB	Unmapped	n/a
0x10000000	0x10000000	256 MB	Internal SRAM	UC/UB
0x20000000	n/a	256 MB	Unmapped	n/a
0x30000000	0x30000000	1 MB	L2CC	UC/UB
0x30100000	n/a	318 MB	Unmapped	n/a
0x43F00000	0x43F00000	961 MB	Internal Registers	UC/UB
0x80000000	0x80000000 / 0x90000000	512 MB	SDRAM (SDCS0/SDCS1) [2]	C/B
0xA0000000	0xA0000000	32 MB	CS0	UC/UB
0xA2000000	n/a	96 MB	Unmapped	n/a
0xA8000000	0xA8000000	32 MB	CS1	UC/UB
0xAA000000	n/a	96 MB	Unmapped	n/a
0xB0000000	0xB0000000	128 MB	CS2, CS3, CS4, CS5	UC/UB
0xB8000000	0xB8000000	128 MB	EIM control	UC/UB
0xB9000000	n/a	256 MB	Unmapped	n/a
0xD0000000	0x80000000 / 0x90000000	512 MB	SDRAM (SDCS0/SDCS1) [2]	UC/UB
0xE0000000	n/a	n/a	Pseudo-virtual addresses for NAND flash (only usable for RedBoot “fis”-commands).	
0xF0000000	0x00000000	1 MB	ROM	UC/UB
0xF0100000	n/a	255 MB	Unmapped	n/a

[1] Attributes: C: cacheable / UC: un-cacheable / B: bufferable / UB: un-bufferable

[2] All physically equipped SDRAM will be mapped in one contiguous area. SDRAM connected to SDCS0 will therefore be mapped 1:1, SDRAM connected to SDCS1 will not be mapped 1:1 but immediately following the SDRAM on SDCS0, instead.

4.2 Virtual Address Space on i.MX27-based Systems

The following table shows the mapping of virtual to physical address space on systems using Freescale i.MX27 SoCs for RedBoot V1.12 (in previous versions some of these areas might be unmapped).

V-Address	P-Address	Size	Description	Attributes[1]
0x00000000	0xA0000000	1 MB	SDRAM	C/B
0x00100000	n/a	255 MB	Unmapped	n/a
0x10000000	0x10000000	1 MB	Internal Registers	UC/UB
0x10100000	n/a	255 MB	Unmapped	n/a
0x20000000	0xA0000000 / 0xB0000000	512 MB	SDRAM (SDCS0/SDCS1) [2]	UC/UB
0x40000000	n/a	768 MB	Unmapped	n/a
0x70000000	0x80000000	1 MB	CSI/ATA Registers	UC/UB
0x70100000	n/a	255 MB	Unmapped	n/a
0x80000000	0xA0000000 / 0xB0000000	512 MB	SDRAM (SDCS0/SDCS1) [2]	C/B
0xA0000000	n/a	512 MB	Unmapped	n/a
0xC0000000	0xC0000000	192 MB	Ext. Chip-Selects	UC/UB
0xD8000000	0xD8000000	128 MB	Int. Periph./PCMCIA	UC/UB
0xE0000000	n/a	n/a	Pseudo-virtual addresses for NAND flash (only usable for RedBoot “fis”-commands).	
0xF0000000	0x00000000	255 MB	ROM	C/B
0xFFF00000	0xFFF00000	1 MB	VRAM	UC/UB

[1]Attributes: C: cacheable / UC: un-cacheable / B: bufferable / UB: un-bufferable

[2]All physically equipped SDRAM will be mapped in one contiguous area. SDRAM connected to SDCS0 will therefore be mapped 1:1, SDRAM connected to SDCS1 will not be mapped 1:1 but immediately following the SDRAM on SDCS0, instead.

4.3 RAM-Usage

Within the SDRAM area, RedBoot reserves the first MByte of memory. The display driver (introduced with RedBoot v1.8) reserves video memory from the end of RAM, if a valid display configuration is present in RedBoot’s configuration data. The size of this video memory can be configured in the display settings. If no display settings are included in RedBoot’s configuration, the display driver won’t start and won’t reserve any memory.
The entire space between the first MByte and the (optional) video-memory at the end of RAM is not implicitly used by RedBoot and can be used without restrictions for downloading files, inspection, modification, etc. The organization of the first MByte of RAM is shown below:

V-Address	Size	Description
0x80000000	64 KB	Reserved
0x80010000	16 KB	Linux-compatible ATAG parameter table
0x80014000	176 KB	reserved
0x80040000	256 KB	RedBoot RAM image
0x80080000	512 KB	RedBoot Heap

Note: In the table above, virtual RAM-addresses are shown. For the corresponding physical RAM-addresses please refer to the mapping table for the respective platform.

Once an OS (or other application) has been started and, if desired, parsed the ATAG parameter table, the first MByte of RAM may be overwritten. The optional video-memory should be used by an OS or application for the same purpose.

5 RedBoot configuration

RedBoot keeps two sets of configuration data for different purposes:

- FIS configuration data is used and manipulated by the “fis”-commands and provides partitioning information for flash-memory.
- Configuration settings shown and manipulated by the “fconfig”-command keep different kinds of settings in variable-like entries. These allow the configuration, e.g. of boot-scripts, Ethernet settings, etc.

While the standard eCos/RedBoot distributions save this data in binary format into flash or serial EEPROM, all RedBoot v1.4 or later releases from Garz & Fricke store all configuration data in flash in XML-format. This change was made to allow easier sharing of information between RedBoot, different operating systems, and different kinds of PC-based remote configuration and maintenance tools.

On Garz & Fricke systems FIS configuration data and other RedBoot configuration settings are stored together in a single configuration area and besides the current configuration area the last version of the configuration area is kept in a separate location (to be more specific: When writing configuration data to flash the oldest version in flash will be overwritten, thereby effectively switching between the two areas reserved for configuration data).

5.1 Flash partitioning

The RedBoot FIS configuration provides partitioning information for the flash chips. Commands are provided for initializing, showing, creating, writing, reading, and deleting partitions. This manual only describes commonly used commands as well as differences between RedBoot releases by Garz & Fricke and the standard eCos/RedBoot distributions. For a full list of all commands and options, please refer to the official eCos/RedBoot documentation.

5.1.1 Handling of NOR and NAND flash

The standard eCos/RedBoot distributions only support one kind of flash at any time. As some Garz & Fricke systems, e.g. the ADELAIDE Mini-Modules, may be equipped with both NOR- and NAND flashes at the same time, RedBoot releases for these systems support the “factive”-command introduced by Freescale to switch between NOR- and NAND flash:

```
factive nor|nand
```

Switching means, that RedBoot will be restarted (from RAM) and afterwards use the specified flash-type for all flash accesses. Even the configuration information will be re-read from the specified flash and therefore must be stored and maintained in both flash-types separately. The currently active mode is shown in the RedBoot boot-messages and can be determined by the flash-addresses used and shown by RedBoot:

- 0xA0000000-area flash addresses are used to access the NOR-flash
- Flash addresses starting at address 0xE0000000 are used to access the NAND flash. For systems equipped with more than 256 MB of NAND flash consecutive address regions are used, wrapping around to 0x00000000 if necessary, i.e. a system equipped with 1 GB of NAND flash will use the contiguous address-area 0xE0000000—0x1FFFFFFF to addresses the NAND flash.

In each mode only the currently active flash can be accessed, i.e. in NOR-flash mode any accesses to NAND flash are unsupported and vice versa.

Please note that addresses used to access NAND flash are only ‘pseudo-virtual’ addresses accepted only by commands meant to access the NAND flash (esp. “fis”, and “nand”-commands). Other commands that directly access memory, e.g. “mfill”, “iopeek”, iopoke”, “cksum”, “load”, etc. **cannot** access the NAND flash.

5.1.2 Bad-block handling in NAND flash

NAND flash chips may have so-called “bad” or “invalid” blocks, i.e. sections of the flash area that are not working reliably under all operating conditions and therefore should not be used for data-storage. This is an inherent property of NAND flash technology and even brand-new NAND flash chips may have bad blocks. The number of bad-blocks may even grow over the lifetime of a flash-chip (usually the flash-chip manufacturer guarantees that not more than 2% a flash-chip will become unusable over the lifetime of the chip). Therefore software directly accessing NAND flashes, like RedBoot and most OS, must be able to handle bad-blocks and agree on a consistent way how to mark flash-blocks as bad.
Garz & Fricke RedBoot releases prior to v1.4 scanned the whole NAND flash for bad-blocks once during boot-time, skipped these by concatenating all good blocks in a single contiguous area and otherwise provided no bad-block handling. This resulted in multiple problems when the number of bad-blocks grew during the lifetime of a NAND flash chip:

- Writes to the flash failing due to a new bad-block required that the user manually marked the respective block as bad, rebooted the device and retried the write operation.
- As all good blocks were collected in a single contiguous area skipping over bad-blocks, all flash addresses referring to data after a new bad-block became invalid.
- RedBoot configuration data could get lost when hitting on new bad-blocks in the configuration area.

To fix all these problems starting with v1.4 a semi-automatic non-transparent bad-block handling scheme has been implemented:

- The whole NAND flash is always mapped into the RedBoot address space, including bad-blocks, so that no references to flash-addresses will ever become invalid due to a silent un-mapping of new bad-blocks.
- All read and load commands accessing NAND flash have been changed to skip over bad-blocks encountered in the specified flash area, concatenating all data read from good blocks to a single, contiguous area in RAM.
- All erase and write commands accessing NAND flash now automatically mark blocks as bad when an erase/write operation fails (according to recommendations from the NAND flash manufacturer).
- All erase and write commands accessing NAND flash have been modified to ask the user if the specified flash-area should be enlarged to compensate for bad-blocks in this area. If the user agrees, the area (in flash) will be enlarged accordingly (maybe multiple times) so that the whole source-image in RAM will fit. If the user refuses only as much of the source-data from RAM will be written to flash, as fits in good NAND flash blocks.
- RedBoot image and configuration areas have been configured large enough to allow for some bad blocks in the respective areas.
- The boot-process and read/write accesses to the RedBoot image and configuration areas have been changed to automatically skip bad-blocks on all operations without any user intervention. (These areas will never be grown due to bad-blocks and therefore always stay in the pre-configured areas.)

Any operating system or application accessing NAND flash must handle flash accesses in a way compatible with the bad-block scheme used by RedBoot.

For NAND flash chips with 528 byte sectors, the first 512 bytes (data-area) of each sector are used for actual data-storage and the last 16 bytes (spare-area) of each sector are used to store meta-data about each sector/block:

- Byte 5 “BLOCK_INVALID” (6th byte in the spare area) determines if a block may be used for data-storage or not: If this byte of any of the first two sectors of a block is not set to 0xff, the respective block is considered invalid and will not be used to store any data. (This follows the convention used by NAND flash manufacturers).
- Bytes 6-10 “ECC” are used to maintain error-correction-codes for each sector via the NAND flash controller of the system.

For NAND flash chips with 2112 byte sectors (2048 bytes data-area + 64 bytes spare-area), RedBoot on Freescale i.MX-based systems breaks with the convention used by the NAND flash manufacturers of implementing a single contiguous data-area followed by a single contiguous spare-area in each sector due to the way data is transmitted by the NAND flash controllers of these systems. Instead RedBoot interprets these 2112 byte sectors as a concatenation of 4 528 byte sectors (i.e. 512d + 16s + 512d + 16s + 512d + 16s + 512d + 16s). NAND flash manufacturers use the first byte of the (contiguous) spare-area, i.e. byte 2048 of the 2112 byte sector, to indicate a bad-block. To remain compatible with this convention RedBoot swaps byte 2048 (1st byte in spare-area “BLOCK_INVALID” using NAND-manufacturer convention) with byte 517 (6th byte in first spare-area “BLOCK_INVALID” in RedBoot convention).

This swapping takes place as the very first step after loading each sector and as the very last step before writing each sector.

Operating systems must implement the same swapping of BLOCK_INVALID bytes on NAND flashes with 2112 bytes sectors, otherwise the OS will consider blocks written by RedBoot as invalid and vice versa.

5.1.3 Flash layout

The locations and sizes for the RedBoot binary image and all configuration areas in flash are hard-coded into RedBoot and cannot be changed. The “fis init”-command will always reserve these areas and it is not possible to delete them.
Up to RedBoot v1.4 the size and location of configuration areas changed several times due to the changes to bad-block management and storage format of the configuration data. Starting with v1.4 these areas are expected to remain fixed in size, location, and configuration, but operating systems and applications should not rely on this. Instead they should use the areas passed to them by RedBoot (see chapter [▶ 8 Information passed to OS]).
This is a sample (default) partitioning in NAND flash on a system with a NAND flash with 528-byte sectors:

```
RedBoot> fis list
... Read from 0x00099238-0x000a9238 at 0xe0050000:
....
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0xE0000000  0xE0000000  0x00040000  0x00000000
Redundant FIS     0xE0040000  0xE0040000  0x00010000  0x00000000 backup
FIS directory     0xE0050000  0xE0050000  0x00010000  0x00000000 current
```

The RedBoot binary image is always stored at the very beginning of the flash. In NAND flash this area will always be followed immediately by two FIS areas: one to keep the current configuration and one to keep the last configuration as backup. All RedBoot configuration settings are also stored in these two FIS areas; therefore also always keeping a backup of all configurable variables.

All other areas in flash not occupied by these areas may be freely used for any user data (operating system images, eCos applications, flash-disks, etc.).

The two FIS areas are used alternatively, always overwriting the oldest version. New commands to explicitly switch between both areas and to overwrite the currently selected one are provided (see below). The area currently selected is marked as “current” in the output of the “fis list” command. The area marked as “backup” will be overwritten when saving the configuration (implicitly or explicitly) turning the “current” area into the new “backup” area.

Please note that on systems with NAND flashes with 2112-byte sectors the sizes and locations of these default areas will be different.

Starting with RedBoot v1.6, an additional area is reserved for the Logbook-feature. This area is not fixed, though. It requires a space of two valid blocks in NAND flash and is not created by the “fis init”-command. Instead when RedBoot starts-up and the Logbook-feature is enabled, RedBoot will look for a FIS -partition with the name “Logbook”. If none is found, RedBoot will automatically create one itself, which will be used from then on for the Logbook, provided that there is still sufficient free-space in flash.

5.1.4 FIS commands

This chapter gives a brief summary of the supported FIS commands. For a detailed description, please refer to the official eCos/RedBoot documentation.

5.1.4.1 fis init

```
fis init [-f] [-y]
```

-f erases the whole flash
-y does not ask for confirmation of the init

The “fis init”-command (re-)initializes the FIS partitioning to a hard-coded default [▶ 5.1.3 Flash layout]. All other partitioning data will get lost.

5.1.4.2 fis list

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

-c shows the checksums stored for partitions
-d shows the actual data-length occupied inside partitions

The “fis list”-command shows the current FIS partitioning data. The partitions will be shown sorted by the physical starting address in flash, i.e. in NAND flashes >512 MB, the starting addresses may wrap-around.

5.1.4.3 fis create

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

The “fis create” command is used to create a new partition in flash. The following options are supported:

-b <mem_base>	source address (usually in RAM) to read the image from that is to be written to the flash-partition
-l <image_length>	size of the area in flash that shall be reserved for the partition
-s <data_length>	size of the source image (usually in RAM) to be written to the partition. This must be smaller or equal to the size given with “-l”
-f <flash_addr>	address in flash specifying the start of the created partition
-e <entry_point>	entry point (usually in RAM) that shall be used later on for the “exec”-command after this image has been loaded via “fis load”.
-r <ram_addr>	address (usually in RAM) that this image shall be loaded to later on when loading it via “fis load”
-n:	the specified area in flash will only be reserved, but not be erased or written to (“-b” and “-s” parameters will be ignored)
<name>	name for the partition

Usually a newly created partition will be immediately filled with data, e.g. an operating system image. Just reserving space for a partition without filling it with data may be used, e.g. for reserving an area to be used as flash-disk by an operating system, i.e. an area that is not dedicated to be used by RedBoot.

RedBoot reserves some partition names for internal use – namely: “RedBoot”, “Redundant FIS”, and “FIS directory” – these will be created automatically by the “fis init”-command and should not be used otherwise. Sizes and locations of these areas are hard-coded in RedBoot. Operating systems should not rely on any partitions being located in a fixed area, though. RedBoot will pass the addresses and sizes of the configuration areas to the operating system (or eCos application) on start-up. Operating systems should use this information to read the configuration data from flash and use the partitioning information contained therein.

The “-f”-parameter to specify the starting address of a partition in flash is optional and may be omitted when creating new partitions, leaving the allocation of free space in flash up to RedBoot.

Starting with RedBoot v1.6 the partition name “Logbook” is reserved as well. The position of this area is not fixed, though. It will be created automatically when RedBoot is started with the logbook-feature being enabled, but no “Logbook”-partition is found.

Starting with RedBoot v1.8 the partition names “logo.png” and “logo.test” are reserved as well. The “logo.png”-partition, if present, is expected to contain a PNG-file to be shown as boot-logo until the OS graphics driver takes over control of the display. The “logo.test”-partition, if present, causes RedBoot to show a hard-coded test-image on screen. The contents of this partition are ignored.

Except for the reserved partition names, RedBoot does not make any assumptions about partition names or their contents. Some operating systems might require that partitions with certain names are present, though.

5.1.4.4 fis load

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

The “fis load” command loads the contents of a partition into RAM

-d	treats the partition contents as gzip-compressed data and unpacks it during loading
-b <memory_load_address>	overrides the memory load address specified for the partition during creation.
-c	shows the checksum calculated (and stored) for the partition
<name>	name of the partition to be loaded

5.1.4.5 fis delete

```
fis delete [-e] [-y] <name>
```

The “fis delete” command deletes existing partitions

-e	keeps the partition information in the FIS-table and just erases the flash area occupied by the partition. If not specified, the whole partition including the entry in the FIS-table is deleted.
-y	does not ask for confirmation of delete.
<name>	the name of the partition to be deleted

5.1.4.6 Commands for direct flash manipulation

```
fis erase –f <flash_addr> -l <length>
fis read –f <flash_addr> -b <mem_base> -l <length
fis write –f <flash_addr> -b <mem_base> -l length
```

These instructions access the flash directly independent of any partitioning information. Therefore these instructions should be used with extreme care.

-f <flash_addr>	start-address in flash
-l <length>	length of the flash-area to be read/written/erased
-b <mem_base>	start address (in RAM) to read/write data from/to

5.1.4.7 fis switch

```
fis switch
```

This command is used to switch between the two sets of configuration data (current and backup) kept in flash. On reset, the latest set of configuration data is always used as ‘current’. Using the “fis switch” command, it is possible to switch to the ‘backup’-copy instead. No data in flash is changed by this command. Therefore this command alone does not change, which configuration area RedBoot chooses as current during boot-up. It is safe to use this command multiple times.
Together with the fis save command, this command may be used to restore the backup configuration data:

```
fis switch
fis save
```

Executing this sequence of commands will first switch to the backup configuration data and then save it as new current configuration settings.

5.1.4.8 fis save

```
fis save
```

The “fis save” command explicitly overwrites the ‘backup’ configuration-data with the ‘current’ one, i.e. if used alone (without “fis switch”) a duplicate of the current configuration will be created. If used immediately after a “fis switch”-command a duplicate of the ‘backup’-data is created effectively reverting the configuration data permanently to the backup-copy.

5.2 Configuration settings

RedBoot provides a couple of configuration settings that may be set by users. The “fconfig” command is used to display and change these:

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

If used without options all configuration settings are shown and queried to be changed.

-i re-initializes all configuration data to default values
-l list the current settings
-n shows the actual names of the configuration variables, instead of the descriptive text (can be combined with other options)
-f shows the descriptive text for each configuration variable. Can be combined with “-n” to show both, the name and the descriptive text
-d switches to “dumb-terminal-mode” turning off editing options

The following section lists all supported settings. As the configurable settings may depend on the version of RedBoot and the specific system it is running on, it may be possible that not all settings are available on all systems/versions.

5.2.1 Boot scripts

A boot script may be configured that will be executed automatically when RedBoot starts-up, e.g. to automatically load and execute an operating system image, or to automatically switch between NOR- and NAND flash mode for systems that boot from a NOR-flash but keep OS images in NAND flash (in this case two different boot scripts would be configured: one in NOR-flash to switch automatically to NAND-mode, and a different boot script in NAND flash to automatically load and execute an OS image).
Three variables are involved in the boot script:

```
boot_script (Run script at boot)
boot_script_data (Boot script)
boot_script_timeout (Boot script timeout (1000ms resolution))
```

The boot script itself is stored as a (multi-line) string in the “boot_script_data” variable. It is enabled or disabled by the Boolean variable “boot_script”. The timeout to wait before starting the boot script is specified by the “boot_script_timeout”-variable.

In RedBoot releases supporting a logbook (see below) with the logbook-feature enabled the boot script will only be executed if the logbook of the previous boot-process has been closed successfully. With this feature it is possible to safely configure a boot_script_timeout of 0 seconds.

5.2.2 Logbook

The logbook-feature has been introduced in RedBoot v1.6 to provide reliable access to the system in case of a failing OS boot-process and to reduce the boot-time of the system by allowing to safely configure a boot-script timeout of 0 seconds. A single variable is used to configure the logbook:

```
enable_logbook (Enable Logbook)
```

If the logbook is enabled, boot-script execution will only be allowed if the logbook of the previous boot-process indicates that the OS or application booted-up properly. [► 5.2.2 Logbook]

Note: As enabling of the Logbook will result in a system refusing to execute the boot-script and therefore waiting in RedBoot for maintenance if the boot-process is interrupted, e.g. by a power-cycle during boot-up, the Logbook feature is **disabled** by default in RedBoot v1.8 or later.

5.2.3 Ethernet configuration

The MAC-address and device name are configured with three variables:

```
eth0_esa (Set eth0 network hardware address [MAC])
eth0_esa_data (eth0 network hardware address [MAC])
eth0_name (Device name)
```

The MAC-address is stored in the “eth0_esa_data”-variable as a string of six hexadecimal numbers separated by ‘:’-characters. The Boolean variable “eth0_esa” decides if the MAC-address stored in “eth0_esa_data” is to be used (if enabled) or if a factory default is to be used, if disabled. The default MAC-address is “0x00:0x07:0x8E:0xAD:0x00:0x01”. (“eth0_esa” has been removed and is considered to always be enabled starting with RedBoot v1.6).

In regular operation no system should be configured to use the default address, because MAC-address collisions with other devices also configured to the default address will occur.

MAC-addresses for Garz & Fricke devices are constructed as follows:

- The first three hex-numbers are set to the Garz & Fricke OUI registered with IEEE: “0x00:0x07:0x8E”
- The remaining three hex-numbers should be set to the serial number of the Mini-Module (in hex).

For example a Mini-Module with the serial number 668348 should be configured to the MAC-address “0x00:0x07:0x8E:0x0A:0x32:0xBC”.

Depending on the RedBoot version the MAC-address might be reset back to the default MAC-address when resetting configuration data to defaults, e.g. with the “fis init” or “fconfig –i” command.

Note: Updates between different RedBoot versions prior and up to v1.4 will also reset the MAC-address to default. For correct operation the MAC-address must be configured to the regular MAC-address format described above! For updates from v1.4 to newer versions this should no longer be necessary; the MAC-address should be preserved.

The device name is user-configurable via the variable “eth0_name”. The usual device name used for Garz & Fricke systems is of the form “GFMM<Serial# of Mini-Module>”. This name will also be generated automatically from the MAC-address if no device name is specified.

To speed up the boot-process, starting with RedBoot v1.6 the Ethernet hardware will no longer be initialized by RedBoot when a boot script is enabled and being executed. The Ethernet hardware will only be initialized automatically by RedBoot, when

- The boot script is disabled
- The boot script is enabled, but will not be executed because the logbook of the last boot-process was not finished successfully.
- The boot script is enabled, but boot script execution is cancelled by the user via Ctrl-C.

Therefore starting with RedBoot v1.6 an OS can no longer rely on the Ethernet hardware already being initialized to a certain state. Also before using commands in boot-scripts that require network access, RedBoot must be forced to initialize the Ethernet hardware. RedBoot may be explicitly forced to initialize the Ethernet hardware via the new “net_init”-command.

5.2.4 IPv4 configuration

Five configuration variables are provided to set the IPv4 configuration (IPv6 is currently not supported)

```
bootp (Use BOOTP for network configuration)
bootp_my_gateway_ip (Gateway IP address)
bootp_my_ip (Local IP address)
bootp_my_ip_mask (Local IP address mask)
bootp_server_ip (bootp_server_ip)
gfdiscover_server (Enable G&F Discovery Server)
```

The Boolean variable “bootp” allows the device to receive its IP-address automatically from a DHCP or BOOTP server. If this is disabled (default) a static IP-address configuration provided by the other variables is used. The string variable “bootp_my_ip” specifies the (static) IP-address to be used if BOOTP/DHCP is disabled; the default address is “192.168.1.1”. The string variable “bootp_my_ip_mask” specifies the corresponding netmask – default is “255.255.255.0”. The string variable “bootp_my_gateway_ip” specifies the address of the gateway server. All IP-packets addressing nodes outside the local network will be sent to this IP for forwarding. Default is “192.168.1.100”. The string variable “bootp_server_ip” specifies the address of the server to be contacted for TFTP downloads (if not overridden on the command-line). A default is not configured.

The variable “gfdiscover_server” enables/disables the Garz & Fricke Discovery Server (enabled by default). [▶7 Discovery Server]

5.2.5 Serial console configuration

The serial console for RedBoot can be configured by two variables:

```
console_baud_rate (Console baud rate)
enable_serialdiag (Enable serial Diag-Port)
```

The baud-rate to be used for the serial console can be configured with the integer variable “console_baud_rate”. The default baud-rate is 115200 baud. Other parameters of the serial communication cannot be configured and are fixed to: 8 data bits, no parity, no hand-shaking.

The Boolean variable “enable_serialdiag”, if disabled, turns-off the serial console. No output will be sent to the serial debug port and no serial connections will be possible (default is ‘true’, i.e. serial console is enabled). If the serial console is disabled only connections via Telnet will be supported by RedBoot.

This variable should also be interpreted accordingly by all operating systems: If the serial console is disabled the operating system should make the serial debug port available as a regular serial port available for customer applications just like all other serial ports of the system. If this variable is enabled, the operating system may provide a debug console on this port itself and/or use this port to output debugging and logging messages. In this case this serial port may not be directly available to applications, but debug output messages by the application may - depending on the OS and its configuration - be directed automatically to this port.

5.2.6 Operating system specific settings

RedBoot provides some configuration variables that are not interpreted by RedBoot but used by operating systems. These are included in RedBoot so that even miss-configured operating systems can be recovered via RedBoot.

```
clear_flashdisk (Clear OS-flashdisk during boot)
clear_registry (Clear OS-registry during boot)
```

The Boolean variable “clear_flashdisk” requests the OS to format all supported flash disks during boot, ignoring and discarding all data possibly stored on them.

The Boolean variable “clear_registry” requests the OS to start with hard-coded default configuration settings. A Windows-CE operating system, for example, should erase all user-made changes to the Registry and instead start with a factory default Registry hard-coded into the OS image.

5.2.7 Miscellaneous settings

RedBoot provides a couple of standard settings that are currently not officially supported in the Garz & Fricke releases.

```
brd_specs (Board specifics)
gdb_port (GDB connection port)
info_console_force (Force console for special debug messages)
net_debug (Network debug at boot time)
```

Usage of these settings is currently discouraged; these settings should be left at their default values. Effects of changing these settings are undefined.

These settings may be removed without notice in future RedBoot releases.

5.3 Boot Logo

Starting with v1.8, RedBoot provides support for displaying a boot-logo (on i.MX27-based systems starting with RedBoot v1.9). Besides needing RedBoot v1.8+ some additional requirements must be fulfilled:

- A display configuration file for the display of the system must be imported into the RedBoot XML-configuration. Due to the many different supported displays, RedBoot does not include any default display settings.
- A FIS partition with the name “logo.png” must exist containing the image to be shown as boot logo. As of v1.8 any valid image file in PNG-format that fits on the screen should work.
- A license for showing the boot-logo must be imported into the RedBoot XML-configuration. This license may be obtained from Garz & Fricke. Starting with RedBoot v1.9 boot logo images provided by Garz & Fricke can be shown without a separate license in the XML-configuration.
- The OS or application to be booted by RedBoot should provide support for a boot-logo. If the OS does not explicitly support a boot-logo it should still work, but may cause a corrupted screen during the boot phase of the OS.

RedBoot and OS images with boot-logo support, corresponding BSPs to build a custom OS, display configuration files for all supported display-types and sample logo files will be provided by Garz & Fricke in the public download area. [▶ 10 Online support]

The license file required to show a customized boot-logo must be obtained separately from Garz & Fricke. Please contact us for further information.

RedBoot v1.8 provides support for logo files conforming to “PNG specification, version 1.2”:
All “critical” chunks of the PNG specification are fully supported including any bit depths, color types, compression methods, filter methods, and interlace methods.

The following “ancillary” chunks are supported:

- tRNS: for logos with transparency
- bKGD: to specify a background color
- All other chunks are ignored.

There is no need for the bit depth or color type of the image to match the configuration of the display, e.g. you can show gray-scale, 4 bpp palletized, or 32 bpp true-color images on a display that is configured for a 16 BPP format.

Images with a resolution smaller than the display resolution will be centered on the screen. Images with a resolution larger than the display resolution are not supported.

Display configuration, boot-logo license file and the boot-logo image itself are uploaded to the device in the same way as a RedBoot or OS image [see ▶ 2.3 Transferring files]. Display configuration and boot-logo license are XML-configuration files that must be imported into RedBoot’s XML-configuration via the “xconfig import”-command. The boot-logo itself must be written into a FIS partition with the name “logo.png” using the “fis create” command similar to writing an OS image to the flash.

To verify that the display and display configuration are working properly a hard-coded test-logo can be shown even without a proper logo file or a boot logo license, by creating an empty partition with the name

“logo.test” and a size of at least one byte (similar to creating a “NAND Flash”-partition for Windows-CE). Any changes made to the logo or display configuration require a reset or power-cycle of the system to take effect.

For a more detailed “How-To” description please refer to [▶ 3.3 How to set up a boot logo].

Note for RedBoot v1.9 or later: Starting with v1.9 boot logo images distributed by Garz & Fricke can be shown without obtaining a separate boot logo license. On older RedBoot releases with boot-logo support, these images can also be used but still require a separate boot logo license.

5.3.1 Compatibility Notes

As proper boot-logo support requires cooperation of RedBoot and OS, some remarks on compatibility are in order:

- RedBoot v1.8+ (with logo support) should boot an OS without boot-logo support without any problems at all when no display configuration is installed in RedBoot.
- RedBoot v1.8+ (with logo support) should boot an OS without boot-logo support but may show a corrupted display when a valid display configuration is installed in RedBoot (regardless of whether a logo is actually shown or not).
- RedBoot v1.8+ (with logo support) may fail to boot an OS with boot-logo support if no display configuration is installed in RedBoot, i.e. an OS with boot-logo support may rely on RedBoot providing the display configuration.
- An OS with boot-logo support may support being booted by RedBoot prior to v1.8 (without logo support) but is not required to. Such an OS, necessarily, must provide its own display configuration. Started by a RedBoot release that supports a boot-logo, such an OS will use the display configuration from RedBoot and ignore its own display configuration settings.

5.4 Resetting configuration to save defaults

Several ways are provided to reset the configuration to save defaults. It is possible to

- reset just the configuration variables, keeping the FIS partitioning information
- reset just the FIS partitioning information, keeping the configuration variables
- reset both.

Please note: Save defaults will usually not be identical to the system state as shipped by Garz & Fricke: Resetting the FIS-information will erase all operating system images and configured flash disks, resetting the configuration variables will disable and discard any boot scripts configured. RedBoot versions prior to v1.4 will also reset the Ethernet MAC-address to the hard-coded default address and must be reconfigured manually (as described above) to a correct MAC-address to prevent address collisions on the network.

5.4.1 Resetting FIS partitioning information

The FIS partitioning information can be reset with the RedBoot command

```
fis init
```

All partitioning information except for the reserved partitions (RedBoot image and configuration partitions) will be deleted by this command.

5.4.2 Resetting configuration variables with RedBoot commands

All configuration variables can be reset with the RedBoot command

```
fconfig -i
```

This will also remove any configured boot script, display settings or boot-logo license. Starting with RedBoot v1.4 this command will preserve the configured MAC-address.

5.4.3 Resetting configuration variables by hardware (RedBoot v1.9 and earlier)

If the system condition even rejects connections to RedBoot (1.4 or later), it is possible to reset configuration variables by rebooting the system with a closed CLEAR_ALL jumper that is available on most systems by Garz & Fricke. Please consult the respective hardware user manual for your system on where to find this jumper. Note that this option is no longer supported starting with RedBoot v1.10).

5.4.4 Getting control over a miss-configured system (RedBoot v.10 and later)

If the system has been configured to a state where it cannot be controlled anymore, e.g. the serial debug port has been turned-off and an OS image is started automatically, but crashes during boot, starting with RedBoot v1.10 it is possible to get control over the system again without permanently resetting any configuration data made so far. Starting with RedBoot v1.10 booting a system with a closed CLEAR_ALL jumper/button will no longer make any changes to the stored configuration data, but instead temporarily

- enable the serial debug port,
- disable the boot-script,
- initialize the Ethernet-port with the saved configuration data.

In this way it should be possible to make any desired configuration changes to the system to make it boot properly again, without requiring a complete re-configuration of the system.

Starting with RedBoot v1.10 the current name “CLEAR_ALL”-jumper or –button in product manuals therefore is somewhat misleading, as no permanent changes are made to the system, and might be changed to BOOT_MODE in future product manuals.

5.5 Logbook feature

In RedBoot v1.6 and later the Logbook feature has been implemented for logging the progress and success of the current and last boot-process. Logbook is enabled by default in v1.6 and disabled by default in RedBoot v1.8 and later. It provides reliable access to the system in case the OS or application boot-process fails; provided the OS or application supports the RedBoot Logbook, as well.

Note: This feature requires a boot-process without interruption, e.g. by a power-cycle. Each time the boot-process is interrupted before the OS/application signals a properly and completely booting, RedBoot will require manual intervention to re-enable automatic booting of the OS/application via the “logbook close” command.

On a system with an OS/application supporting the logbook feature, the logbook works completely automatically and does not require intervention or maintenance by the user. RedBoot provides a full set of maintenance commands [►explanation in 6.1.8 logbook].

The logbook is stored in flash occupying two blocks in the flash memory using the reserved fis-partition name “Logbook”. This partition will be created automatically if necessary. Each block occupied by the logbook is used for a single “entry” logging the progress of one boot-process in so-called “steps”, each occupying a single sector in flash. Each step comprises a single 32-byte, zero-terminated ASCII-string, with the first four characters identifying the component that logged the step, and the remainder giving information on what happened.

The remaining 480 bytes of a sector used for a logbook-step are reserved for future use. The additional 1536 bytes available in NAND flashes with 2112-byte sector will never be used.

RedBoot defines the following steps with the identifier set to “BOOT”:

- A logbook-entry will always start with a line formatted “BOOT: started #<number>”. The value <number> is incremented continuously at each boot-process, thereby also ordering the entries in the logbook.
- Each time something is loaded in RedBoot via “load” or “fis load” commands, a line formatted “BOOT: <source>:<name>” is written to the current logbook entry.
- Each time an “exec”, “go”, or “run” command is executed in RedBoot, a line formatted “BOOT: <cmnd> <address>” is written to the current logbook entry
- To declare a logbook entry as successfully closed, an OS or application must write a line formatted as “BOOT: finished” to the current logbook entry. On the next reset, RedBoot will only execute the boot-script if the previous logbook entry has been closed by such a line.

Each time the system is booted, RedBoot erases the oldest logbook entry (a single flash-block) and start writing progress steps in consecutive sectors of this block. An OS or application may add further lines to log certain critical milestones passed in the boot-process or to log errors in case of critical failures.

For the logbook feature to work properly the OS or application started by RedBoot must write a “BOOT: finished” step to the end of the most recent logbook entry if (and only if) the OS/application was started completely and successfully.

Each logbook entry can hold up to a maximum of 32 steps, with at least 3 steps being written by RedBoot and another step reserved for the “BOOT : finished”-step that must be written by the OS/application.
Except for the “BOOT: started #<number>” and “BOOT: finished” steps, the contents or presence/absence of no other step must be interpreted or be relied on. All other progress-steps are only intended for informational purposes for users or maintenance personal.

5.6 Configuration data format

Unlike the standard eCos/RedBoot distributions, which use a proprietary binary format, the Garz & Fricke RedBoot releases save all configuration data (FIS-table and configurable variables) as ASCII XML strings in flash. This change was made to allow for easier sharing the configuration data between RedBoot, different operating systems and remote configuration tools running on PCs.

As described in chapter [▶ 5.1.3 Flash layout], all Garz & Fricke devices supporting RedBoot currently store the configurable variables together with the FIS-table in one area. This area is kept redundantly, i.e. besides the current configuration data the last configuration data is stored as back-up. Each time configuration data is saved the old backup data will be overwritten and the roles of the two areas will be exchanged: the old backup data will be overwritten with the new data and turn into the new ‘current’ area; the old current area will be turned into the new ‘backup’ area and not be modified.

While this way of storage is current practice on all Garz & Fricke systems using RedBoot, this is not mandatory: RedBoot may also be configured to save the FIS-table and configurable variables in different areas, and/or not to keep a backup copy. All operating systems accessing the configuration data should be aware of this. The chapter [▶ 8 Information passed to OS] documents how operating systems can tell how and where configuration data is stored in flash.

For remote-configuration applications running on PCs that might want to access the configuration data directly, so far no mechanism has been defined how to find out how and where configuration data is stored. This will be specified in a future release.

5.6.1 Storage in flash

The configuration data is stored as a single 0-terminated 7-Bit ASCII-string immediately followed by a 32-Bit CRC checksum covering the whole ASCII-string including the terminating 0. RedBoot uses the free CRC32 implementation by Gary S. Brown included in the eCos source-tree. While operating systems and applications may use different implementations it is strongly recommended that the same implementation is used to guarantee compatibility.
The checksum will be verified by RedBoot on boot-up. The contents of configuration area(s) with invalid checksums will not be read or interpreted (but will be overwritten with valid configuration data, if configuration data is saved).

5.6.2 XML data format

The XML data is encoded in ASCII format, as specified by the XML header. Other encodings are not supported.

Currently no schema or DTD is provided describing the syntax of the XML data. This will be specified in a future release, but even then it should not be expected that RedBoot will perform conformance checks and/or reject illegal data. Therefore care must be taken when making changes to the configuration data directly.
While storage of additional data not specified here is theoretically possible, this will not be officially supported and is strongly discouraged.

RedBoot uses the free ezXML-v0.8.6 XML parsing C Library by Aaron Voisine, which is part of the standard eCos/RedBoot distribution, for parsing and modification of the XML data. While it is possible to

use different libraries it is recommended that operating systems and other applications accessing the XML configuration data should use the same library for guaranteed compatibility. (The library is available separately from http://ezxml.sourceforge.net/)

5.6.2.1 XML header and root tag

All XML-configuration data always starts with a fixed XML header and root tag:

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>

<configurationFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</configurationFile>
```

It must contain exactly one <configurationFile>-tag. All configuration data will be contained in this tag.

5.6.2.2 <flash>-tag

The configuration data will mostly contain one <flash>-tag describing the flash partitioning (FIS-table) of the flash-area the configuration data is saved in. This tag supports one numerical attribute:

```
<flash generation=”{generation count}”>
</flash>
```

The “generation”-attribute is used by RedBoot on boot-up to determine, which configuration data contains the most recent, i.e. current, configuration data and which area contains the backup. The area with the higher generation count is defined as current.

Therefore when configuration data is modified the “generation”-attribute must be incremented for RedBoot to correctly find the current area.

5.6.2.2.1 <partition>-tag

The <flash>-tag contains multiple <partition>-tags describing each FIS partition:

```
<partition
    name=”{name of partition}”
    flash_base=”{flash base address}”
    flash_length=”{flash length}”
    ram_base=”{RAM base address}”
    ram_entry=”{RAM entry point}”
    data_length=”{length of actual data}”
    data_cksum=”{CRC32 checksum over the data}” >
</partition>
```

All attributes are of numerical type and are mandatory. The values must be specified in the number-syntax used by the C programming language.

name:	name of the partition
flash_base:	start (in flash) of the partition; corresponds to the “-f”-parameter of the “fis create”-command
flash_length:	size of the partition in flash; corresponds to the “-l”-parameter of the “fis create”-command
ram_base:	load-address (in RAM) for the contents of the partition; corresponds to the “-r”-parameter of the “fis create”-command. When “fis load” is called for this partition, the data will be loaded to this address in RAM.
ram_entry:	entry point (in RAM) for the contents of the partition; corresponds to the “-e”-parameter of the “fis create”-command. When “exec” is called after loading this partition with “fis load”, this will become the default execution address.
data_length:	length of the actual data contained in the partition; corresponds to the “-s”-parameter of the “fis create”-command
data_cksum:	CRC32 checksum over the partition; will be calculated and saved automatically by the “fis create”-command.

Note: Certain partitions and partition names are reserved, must always be present and must never be modified. Failures may result in undefined behavior and flash corruption. This may even render a system completely unusable!

Note: The following partitions are reserved:

```
<partition name=”RedBoot” ... />
<partition name=”FIS directory” ... />
<partition name=”Redundant FIS” ... />
```

The following partitions are optional but have a special meaning for RedBoot and may require contents in a specific format when present:

```
<partition name=”Logbook” ... />
<partition name=”logo.png” ... />
<partition name=”logo.test” ... />
```

When using the “fis create”/”fis delete”-commands RedBoot ensures that all partitions are valid and do not overlap. When manipulating the XML-configuration data directly, this must be ensured by the operating system or application doing so.

Note: Specification of overlapping partitions will result in corruption of the overlapping areas. This may even render a system completely unusable if overlaps with reserved partitions are specified!

5.6.2.3 <variables>-tag

All user-configurable RedBoot variables are stored as sub-tags in a single <variables>-tag in the XML configuration data. No attributes are supported.

```
<variables>
</variables>
```

5.6.2.3.1 <setting>-tag

The single <variables>-tag contains multiple <setting>-tags, each describing a single user-configurable variable.

```
<setting
    key="{name of variable}"
    title="{description of variable}"
    type="{data-type of value contents}"
    enable_key="{name of ‘parent’ variable}"
    enable_sense="{enabling sense of ‘parent’ variable}"
....value="{value of variable}"
    regkey="{registry-key associated with variable}" >
</setting>
```

All attributes except for “regkey” are mandatory.

key: type:string, name of the variable; shown with the “-n”-parameter of “fconfig”.

title: type: string, description for the variable; shown when “fconfig” is used without parameters or with the “-f”-parameter.

type: type: string, data-type of the “value”-attribute (see table below).

enable_key: type: string, name of a (Boolean) variable that must be set to a certain value for this variable to be effective.

enable_sense: type: Boolean, the (Boolean) value the variable given in the “enable_key”-attribute must be set to for this variable to be effective.

value: type: specified by “type”-attribute, contains the actual data stored in the variable

regkey: type: string, this optional attribute is used by Windows-CE operating systems and specifies a registry setting this variable is associated with, so that XML configuration data can be synchronized automatically with the Windows-CE registry.

The variable data, stored in the “value”-attribute, is stored in different types depending on the specific variable. The data-type is defined by the “type”-attribute. As the XML configuration data is encoded in

ASCII representation, all data-types are encoded in ASCII format, too. The following data types are defined:

Type	Description	Example
“bool”	A Boolean data-type	“true” “false
“int”	A numerical, integer, non-negative value using number syntax compatible with the C-programming-language	“42” “0xe0040000”
“hex”	Identical to “int”, but forces fconfig to display the parameter in hex-format	
“string”	A single-line string value	“GFMM668348”
“script”	A multi-line string value	“fis load OS exec”
“esa”	Ethernet MAC-address, stored as colon-delimited list of 6 hex-numbers	“0x00:0x07:0x8e:0x0a:0x32:0xbc”
“ip”	An IPv4 address	“192.168.1.1” “255.255.255.0”

5.6.2.3.2 <display>-tag

Starting with RedBoot v1.8 the single <variables>-tag may contain at most one <display>-tag specifying the display configuration used for the system. The configuration settings are grouped in several mandatory sub-tags. The <display>-tag itself contains several attributes:

```
<display
            name="{descriptive name of the display"
            type="{type of controller}"
            xres="{horizontal resolution}"
            yres="{vertical resolution}"
            refresh="{refresh rate}"
            vidmem="{size of video memory}"
            regKey="{WinCE registry key for display settings}">
     <mode ... />
     <format ... />
     <rotation ... />
     <backlight ... />
     <hsync ... />
     <vsync ... />
     <clock ... />
     <data ... />
     <power_sequence ... />
</display>
```

All attributes and sub-tags are mandatory:

name: type: string, assigns a brief descriptive name to the display settings (may be shown by RedBoot or OS for information purposes, but not interpreted.)

type: type: string, the type of the display controller in the system to be used for the display. Values may be “SDC” for synchronous displays or “ADC” for asynchronous displays. RedBoot currently only supports “SDC”.

xres: type: int, horizontal resolution of the display in pixel

yres: type: int, vertical resolution of the display in pixel

refresh: type: int, vertical refresh rate of the display in Herz

vidmem: type: int, size of video memory to be reserved in bytes (should be at least 5*xres*yres*bytes_per_pixel)

regKey: type: string, Windows-CE registry path, where display settings should be written to.

5.6.2.3.2.1 <mode>-tag

The <mode>-tag specifies some general information about the display interface.

```
<mode
     id="{Interface to use}"
     custom_panel="{Custom or hard-coded settings}"
     type="{Hard-coded display configuration}"
/>
```

All attributes are mandatory:

id: type: string, the interface to the display. Possible values are: “PAL”, “NTSC” or “Device”. As of RedBoot v1.8 only “Device” is supported.

custom_panel: type: int, specifies whether custom configuration from the display configuration file should be used (value 1) or hard-coded internal settings (value 0). As of RedBoot v1.8 only “1” is supported.

type: type: int, is an index into a list of hard-coded display configurations. This parameter is ignored when “custom_panel”=”1”. Should be set to “3” for compatibility.

5.6.2.3.2.2 <format>-tag

The <format>-tag specifies information about the pixel-format in memory and on the display interface.

```
<format
     depth="{pixel depth for main graphics plane}"
     video_depth="{pixel depth for video plane}"
     format="{pixel configuration on display interface}"
     EGPEFormat="{pixel format for Windows-CE}"
/>
```

All attributes are mandatory:

depth: type: int, pixel depth in bits for the main graphics plane. Possible value are “1”, “2”, “4”, “8”, “16”, “24”, or “32”. As of RedBoot v1.8 only “16”, “24”, and “32” are supported.

video_depth: type: int, pixel depth in bits for the video plane. Not used by RedBoot.

format: type: string, pixel configuration on the display interface. Possible values are “RGB565”, “RGB666”, “RGB24”, or “RGB888”. As of RedBoot v1.8 only “RGB565”, “RGB666”, and “RGB24” are supported.

EGPEFormat: type: int, Windows-CE EGPE pixel format (not used by RedBoot). Possible values:
0: 1 Bpp
1: 2 Bpp
2: 4 Bpp
3: 8 Bpp
4: 16 Bpp
5: 25 Bpp
6: 32 Bpp
7: 16 YCrCb
8: Device compatible
9: undefined

5.6.2.3.2.3 <rotation>-tag

The <rotation>-tag specifies display rotation:

```
<rotation
     value="{Angle of rotation}"
     regKey="{WinCE registry key for rotation settings}"
/>
```

All attributes are mandatory:

value: type: int, angle of rotation of the screen (in degree). Possible values are 0, 90, 180, or 270.

regKey: type: string, Windows-CE registry path, where rotation settings should be written to.

5.6.2.3.2.4 <backlight>-tag

The <backlight>-tag configures settings for backlight control.

```
<backlight
     level_ac="{AC-Power backlight level}"
     level_battery="{Battery-Power backlight level}"
     lut="{Look-Up table for backlight levels}"
     regKey="{WinCE registry key for backlight settings}"
/>
```

All attributes except for “lut” are mandatory:

level_ac: type: int, current backlight level when running on AC power as index into a 256-entry look-up table.

level_battery: type: int, current backlight level when running on battery power as index into a 256-entry look-up table.

lut: type: int[256], optionally specifies a 256-entry look-up table for backlight levels to compensate for non-linear backlight intensities. If omitted a hard-coded default table is used.

regKey: type: string, Windows-CE registry path, where backlight settings should be written to.

5.6.2.3.2.5 <hsync>-tag and <vsync>-tag

The <hsync>-tag configures the horizontal synchronization signal for the display, the <vsync>-tag the vertical synchronization signal.

```
<hsync
     width="{Width of sync signal}"
     start_width="{Width of sync start}"
     end_width="{Width of sync end}"
     polarity="{Polarity of sync signal}"
/>
<vsync
     width="{Width of sync signal}"
     start_width="{Width of sync start}"
     end_width="{Width of sync end}"
     polarity="{Polarity of sync signal}"
/>
```

All attributes are mandatory:

width: tpye:int, width of the active level of the sync signal in pixel clock cycles (hsync) or line clock cycles (vsync).

start_width: tpye:int, width from start of active sync to start of valid pixel data in pixel clock cycles (hsync) or line clock cycles (vsync).

end_width: tpye:int, width from end of valid pixel data to start of sync signal in pixel clock cycles (hsync) or line clock cycles (vsync).

polarity: tpye:int, polarity of the sync signal:
0: straight-through
1: inverted

5.6.2.3.2.6 <clock>-tag

The <clock>-tag configures the display clock signal.

```
<clock
     idle_enable="{Clock enabled durng vsync}"
     select_enable="{Clock enabled without data output}"
     polarity="{Polarity of clock signal}"
/>
```

All attributes are mandatory:

idle_enable: type: int, specifies whether the display clock is enabled (0) while vertical synchronization signal is active or disabled (1).

select_enable: type: int, specifies whether the display clock is disabled (1) when no data is output or always output (0).
polarity: tpye:int, polarity of the clock signal:
0: straight-through
1: inverted

5.6.2.3.2.7 <data>-tag

The <data>-tag configures the data signals.

```
<data
     mask_enable="0"
     polarity="0"
     oe_polarity="1"
/>
```

All attributes are mandatory:

mask_enable: type: int, when set to 1, forces all data output signals to 0. Should usually be set to 0.
polarity: tpye:int, polarity of the data signals:
0: straight-through
1: inverted
oe_polarity: tpye:int, polarity of the output enable signal:
0: straight-through
1: inverted

5.6.2.3.2.8 <power_sequence>-tag

The <power_sequence>-tag specifies the timing for the display power-up and power-down sequences (not to be confused with power-management settings in case of user-inactivity).

```
<power_sequence
      PowerOnToSignalOn="{val}"
      PowerOnToBacklightOn="{val}"
      BacklightOffBeforePowerOff="{val}"
      SignalOffBeforePowerOff="{val}"
      PowerOffToPowerOn="{val}"
/>
```

All attributes are mandatory:

PowerOnToSignalOn: tpye:int, time in milliseconds to wait between enabling the display power and enabling the display signals.
PowerOnToBacklightOn: tpye:int, time in milliseconds to wait between enabling the display power and turning on the backlight.
BacklightOffBeforePowerOff: tpye:int, time in milliseconds the backlight must be turned-off before the display power is disabled.
SignalOffBeforePowerOff: tpye:int, time in milliseconds the display signals must be turned-off before the display power is disabled.
PowerOffToPowerOn: tpye:int, time in milliseconds the display must remain turned-off before re-enabling power.

5.6.2.3.3 <logo-license>-tag

Starting with RedBoot v1.8 the single <variables>-tag may contain a <logo-license>-tag. The boot logo supported starting with v1.8 will only be shown if this tag is present and its contents valid.

```
<logo-license
     license="{Binary logo license key}"
/>
```

5.6.3 Sample XML-configuration

This section shows the (current) XML-configuration data of a sample JUPITER/ADELAIDE system running RedBoot v1.4. For samples of display configuration settings (RedBoot v1.8 or later) please refer to the display configuration files available for download from Garz & Fricke.

Note: This is **not** the default configuration Garz & Fricke are systems shipped with).

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>

<configurationFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<flash generation="24">
    <partition name="RedBoot"
        flash_base="0xe0000000" flash_length="0x00040000"
        ram_base="0xe0000000" ram_entry="0x00000000"
        data_length="0x00000000" data_cksum="0x00000000"></partition>
    <partition name="FIS directory"
        flash_base="0xe0050000" flash_length="0x00010000"
        ram_base="0xe0050000" ram_entry="0x00000000"
        data_length="0x00000000" data_cksum="0x00000000"></partition>
    <partition name="Redundant FIS"
        flash_base="0xe0040000" flash_length="0x00010000"
        ram_base="0xe0040000" ram_entry="0x00000000"
        data_length="0x00000000" data_cksum="0x00000000"></partition>
    <partition name="NAND Flash"
        flash_base="0xe2000000" flash_length="0x02000000"
        ram_base="0xe2000000" ram_entry="0xffffffff"
        data_length="0x02000000" data_cksum="0x00000000"></partition>
</flash>

<variables>
    <setting key="boot_script" title="Run script at boot"
        type="bool" enable_key="" enable_sense="true" value="false"
        regkey="HKLM\Software\RedBoot\BootscriptEnabled"></setting>
    <setting key="boot_script_data" title="Boot script"
        type="script" enable_key="boot_script" enable_sense="true" value=""
        regkey="HKLM\Software\RedBoot\Bootscript"></setting>
    <setting key="boot_script_timeout"
        title="Boot script timeout (1000ms resolution)"
        type="int" enable_key="boot_script" enable_sense="true" value="0"
        regkey="HKLM\Software\RedBoot\BootscriptTimeout"></setting>
    <setting key="bootp" title="Use BOOTP for network configuration"
        type="bool" enable_key="" enable_sense="true" value="false"
        regkey="HKLM\Comm\SMSC91181\Parms\TcpIp\EnableDHCP"></setting>
    <setting key="bootp_my_gateway_ip" title="Gateway IP address"
        type="ip" enable_key="bootp" enable_sense="false" value="172.20.0.13"
        regkey="HKLM\Comm\SMSC91181\Parms\TcpIp\DefaultGateway"></setting>
    <setting key="bootp_my_ip" title="Local IP address"
        type="ip" enable_key="bootp" enable_sense="false" value="172.20.102.159"
        regkey="HKLM\Comm\SMSC91181\Parms\TcpIp\IpAddress"></setting>
    <setting key="bootp_my_ip_mask" title="Local IP address mask"
        type="ip" enable_key="bootp" enable_sense="false" value="255.255.0.0"
        regkey="HKLM\Comm\SMSC91181\Parms\TcpIp\Subnetmask"></setting>
    <setting key="bootp_server_ip" title="Default server IP address"
        type="ip" enable_key="" enable_sense="true"
        value="172.20.0.13"></setting>
    <setting key="brd_specs" title="Board specifics"
        type="int" enable_key="" enable_sense="true" value="0"></setting>
    <setting key="clear_flashdisk" title="Clear OS-flashdisk during boot"
        type="bool" enable_key="" enable_sense="true" value="false"
        regkey="HKLM\Software\RedBoot\ClearFlashDisk"></setting>
    <setting key="clear_registry" title="Clear OS-registry during boot"
        type="bool" enable_key="" enable_sense="true" value="false"
        regkey="HKLM\Software\RedBoot\ClearRegistry"></setting>
    <setting key="console_baud_rate" title="Console baud rate"
        type="int" enable_key="" enable_sense="true" value="115200"
        regkey="HKLM\Software\RedBoot\ConsoleBaudrate"></setting>
```

```
    <setting key="enable_serialdiag" title="Enable serial Diag-Port"
        type="bool" enable_key="" enable_sense="true" value="true"
        regkey="HKLM\Software\RedBoot\EnableSerialDiag"></setting>
    <setting key="eth0_esa" title="Set eth0 network hardware address [MAC]"
        type="bool" enable_key="" enable_sense="true" value="true"></setting>
    <setting key="eth0_esa_data" title="eth0 network hardware address [MAC]"
        type="esa" enable_key="eth0_esa" enable_sense="true"
        value="0x00:0x07:0x8E:0x0A:0x32:0xBC"
        regkey="HKLM\Comm\SMSC91181\Parms\MACAddr"></setting>
    <setting key="eth0_name" title="Device name"
        type="string" enable_key="" enable_sense="true" value="GFMM668348"
        regkey="HKLM\Ident\Name"></setting>
    <setting key="gdb_port" title="GDB connection port"
        type="int" enable_key="" enable_sense="true" value="23"></setting>
    <setting key="info_console_force"
        title="Force console for special debug messages"
        type="bool" enable_key="" enable_sense="true" value="false"></setting>
    <setting key="info_console_number"
        title="Console number for special debug messages"
        type="int" enable_key="info_console_force" enable_sense="true"
        value="0"></setting>
    <setting key="net_debug" title="Network debug at boot time"
        type="bool" enable_key="" enable_sense="true" value="false"></setting>
</variables>

</configurationFile>
```

6 Additional RedBoot commands

This chapter documents commands that have been introduced by Freescale and Garz & Fricke and that are supported on Garz & Fricke platforms, as well as some other useful standard RedBoot commands that may not be available in all RedBoot releases.

6.1 New or changed commands

Compared to the standard eCos/RedBoot distribution, Freescale and Garz & Fricke have added several additional commands to RedBoot.

6.1.1 factive

```
factive nor|nand
```

The “factive”-command is supported on all Garz & Fricke systems that may be equipped with both NOR- and NAND flash and switches RedBoot between both modes. On platforms equipped with only one kind of flash the behavior of this command is undefined or the command may be missing.

6.1.2 romupdate

```
romupdate
```

The “romupdate”-command is provided to make RedBoot-updates easier and saver: Instead of manually overwriting the RedBoot image in flash with an untested new RedBoot image using the “fis create” command, “romupdate” will overwrite the RedBoot image in flash with the currently running image without requiring any parameters.
Thereby eliminating possible errors when specifying flash-addresses and demonstrating that the new version is in fact working before replacing the version in flash. [▶ 3.1 RedBoot].

6.1.3 fis delete

```
fis delete [-e] name
```

While the “fis delete” command is not new – it deletes the named partition, the “-e”-option has been added by Garz & Fricke. If specified, it will only erase the contents of the partition, but keep the partition entry itself.

6.1.4 fis read

```
fis read -f <flash_addr> -b <mem_base> -l <image_length>
```

While the standard RedBoot provides “fis write” and “fis erase” commands to directly manipulate flash areas independent of configured FIS-partitions, no corresponding function to read directly from flash is provided. The new “fis read” command implements just this: It reads the specified area from flash into RAM.

6.1.5 mallinfo

```
mallinfo
```

The “mallinfo”-command shows some information on the memory usage in the RedBoot RAM-heap. This may be useful for developers making extensions to the RedBoot source-code that allocate dynamic memory.

6.1.6 testram

```
testram -b <start_address> -l <length> [-i]
```

The “testram”-command performs a simple memory test in the specified area (only tests in RAM-like memory are supported) to check that memory is working properly.

If the “-i”-option is specified (RedBoot v1.6 or later) a more intensive RAM test will be executed in addition to the default test.

6.1.7 nand

So far, standard RedBoot distributions do not provide special functions to address the specifics of NAND flash chips. Freescale introduces the “nand”-command set to provide such functions. These are also supported on all Garz & Fricke systems equipped with NAND flash.

6.1.7.1 nand erase

```
nand erase -f <raw address> -l <length> [-o][-z]
```

The “nand erase”-command erases NAND flash areas similar to the “fis erase” command, but provides additional command-line options to address NAND-specific features:

-o erases not just the specified data-area but also all bad-block marks for the specified area of the NAND flash thus declaring bad-blocks as usable ‘good’ blocks again. This option should only be used for testing purposes: flash blocks marked as bad have been marked so because they have shown to be unreliable for data storage. Manufacturers of NAND flash chips allow a certain percentage of bad-blocks. Most new NAND flash chips already have some blocks marked as bad, that didn’t pass elaborate tests by the manufacturer and during the life-time of a NAND flash chip a certain number of additional blocks may turn bad. (For details on this subject please refer to the datasheet of the specific NAND flash chip.)

-z erases the given address range and marks it as bad. This should only be used for testing purposes.

6.1.7.2 nand info

```
nand info
```

The “nand info” command prints useful information on the NAND flash, like type and manufacturer, size, bad blocks, etc. This is a sample output on a JUPITER/ADELAIDE system:

```
RedBoot> nand info
Type: ST NAND512W3A
Total size:     0x4000000 bytes
Total blocks:   0x1000 (4096)
Block size:     0x4000 (16384)
Page size:      0x200 (512)
Bad blocks:
  block 20 at 0x50000
  block 2784 at 0x2b80000
  block 3520 at 0x3700000

  Total number of bad blocks: 3
```

6.1.7.3 nand load

```
nand load -f <flash_address> -b <mem_address> -l <length>
```

This command loads data directly from flash into RAM. The new “fis read” function provides the same function and is more consistent with other “fis”-commands, so it is recommended to use “fis read”.

Note: Support for this command has been removed in RedBoot v1.6.

6.1.7.4 nand show

```
nand show –f <flash_address> [-s]
```

The “nand show” command displays the contents of a single NAND flash page on the console. If the “-s”-option is specified, only the spare-area of the given page-address is shown.

6.1.8 logbook

RedBoot v1.6 introduced the logbook feature and provides a set of commands to display and modify the logbook from the command-line.

6.1.8.1 logbook show

```
logbook show [-c] [-l] [-a]
```

This command shows the contents of logbook entries on the console. The “-l”-option displays the logbook-entry of the previous boot process (default).

-c displays the logbook-entry currently being written.

-a displays all logbook-entries in order. On systems supporting only two logbook entries the “-a”-option is equivalent to specifying both “-c” and “-l”-options.

6.1.8.2 logbook note

```
logbook note –i <id> -n <note>
```

This command writes a new progress step to the currently open logbook-entry (if any). This command is mostly intended for maintenance and testing purposes but may also be useful when executing longer boot scripts to log progress of boot scripts.

6.1.8.3 logbook open

```
logbook open
```

This command opens a new logbook entry, overwriting the oldest existing logbook-entry. This command is only intended for maintenance and testing purposes.

6.1.8.4 logbook close

```
logbook close
```

This command closes the current logbook entry by writing a “BOOT: finished” step, signalling that the boot process has finished completely. This command is mostly intended for maintenance and testing purposes but may be useful if a boot script should be executed that does not start an application or OS, which then will be responsible for finishing the current logbook entry.

6.1.9 net_init

```
net_init
```

(Introduced with RedBoot v1.6) Explicitly forces RedBoot to initialize the Ethernet hardware. This should be executed if commands executed in a boot script require network access.

6.1.10 xconfig

In RedBoot v1.8 and later, several commands have been added to manipulate the XML-configuration data directly. Care should be taken when using these commands as miss-configured or corrupted XML-configuration data might render a system unusable.
Most of these commands allow a selection of one or more nodes of the XML configuration data via a “path”-option. This “path”-option expects a very small subset of the XPath 1.0 syntax for selection:

- only abbreviated syntax is supported
- only absolute location paths
- only the child-axis is supported for selection
- “*” node test is supported, but not “//”
- only attribute predicates of string-type with “=” comparison are supported

The description of the “xconfig list” command below shows some examples.
All xconfig-commands by default only work on the <variables>-subtree of the XML-configuration. The “-f”-option supported by all xconfig commands switches to the <flash>-subtree, instead, but this is strongly discouraged. The “fis”-commands should be used instead to manipulate partitions.

All commands changing the XML configuration, i.e. all but “config list”, support the “-y”-option. If specified, any changes are written to flash without asking the user for confirmation.

6.1.10.1 xconfig list

```
xconfig list [-v] [-f] [-p <path>]
```

This command shows the current contents of the XML configuration. If no path is specified, the whole tree is shown. If the “-v” option is not specified, only the general structure of the selected (sub-)tree is shown, if the “-v” option is specified, the whole (sub-)tree including all attributes is displayed in a format that is again suitable as input to the “xconfig import” command.

Samples:

A. List a complete tree similar to the one shown in section “Sample XML configuration”:

Command:

```
xconfig list -v
```

Result:

Similar to section „Sample XML configuration“

B. Show configuration of the horizontal sync signal of the display configuration:

Command:

```
xconfig list -v -p /configurationFile/variables/display/hsync
```

Result (depends on display type of the platform)**:**

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>
<configurationFile
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <variables>
        <display
            name="Kyocera TCG057VG1AC"
            type="SDC"
            xres="640"
            yres="480"
            refresh="50"
            vidmem="0x00300000"
            regKey="HKLM\Drivers\Display\DDGUF">
            <hsync
```

```
                width="63"
                start_width="144"
                end_width="16"
                polarity="0">
            </hsync>
        </display>
    </variables>
</configurationFile>
```

C. Show setting for the serial diagnostic port:

Command:

```
xconfig list -v -p /configurationFile/variables/setting[@key='enable_serialdiag']
```

Result:

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>
<configurationFile>
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <variables>
        <setting
            key="enable_serialdiag"
            title="Enable serial Diag-Port"
            type="bool"
            enable_key=""
            enable_sense="true"
            value="true"
            regkey="HKLM\Software\RedBoot\EnableSerialDiag">
        </setting>
    </variables>
</configurationFile>
```

6.1.10.2 xconfig import

```
xconfig import [-v] [-f] [-y] -b <addr>
```

This command imports an external XML configuration file into RedBoot's current XML configuration. The file must have been loaded to the address specified with the "-b"-option before. If the "-v"-option is specified, the command will output additional information about the changes made to the existing XML-configuration.

Sample (import boot-logo license):

```
load -r -v -b 0x80100000 -m xmodem rb-logolicense.xml
xconfig import -b 0x80100000
```

6.1.10.3 xconfig addnode

```
xconfig addnode [-f] [-y] -p <path> -n <name>
```

This command adds a new, empty node with the name specified via the "-n"-option as a child to each of the nodes selected with <path>. As any node requires certain attributes to be functional, usage of this command is not recommended, because all mandatory attributes must be added to the node(s) manually, as well.

6.1.10.4 xconfig delnode

```
xconfig delnode [-f] [-y] -p <path>
```

This command deletes all nodes selected with the specified <path> and all child nodes.

Sample (Delete display configuration):

```
xconfig delnode -p /configurationFile/variables/display
```

6.1.10.5 xconfig addattribute

```
xconfig addattribute [-f] [-y] -p <path> -n <attr-name> -v <value>
```

This command adds a new or changes an existing attribute with the name <attr-name> of all nodes selected with the specified <path> to the value specified with the “-v”-option.

Sample (rotate display by 180°):

```
xconfig addattribute -p /configurationFile/variables/display/rotation -n value -v
180
```

6.1.10.6 xconfig delattribute

```
xconfig delattribute [-f] [-y] -p <path> -n <attr-name>
```

This command deletes existing attributes with the name <attr-name> from all nodes selected with the specified <path>.

Sample (remove the backlight look-up table, causing the system to revert to an internal hard-coded table):

```
xconfig delattribute -p /configurationFile/variables/display/backlight -n lut
```

7 Discovery Server

Starting with RedBoot v1.6 the Garz & Fricke Discovery Server is included in the BIOS and enabled by default, which is required for the device to be accessible from the Garz & Fricke “DeviceControl”-application. DeviceControl is a separate PC application allowing remote discovery of and access to Garz & Fricke systems for maintenance purposes like configuration or firmware updates.

For a device to be findable and accessible via DeviceControl, the BIOS or operating system running on the device must provide the Garz & Fricke Discovery Server, which is a very simple UDP/IP server application running on the device, providing some information to remote systems.

The Discovery Server implements a simple, proprietary ASCII-based protocol transmitted via single UDP/IP packets on UDP-port 7311. The protocol is only used to find devices on the network and provide remote stations with some initial information about the device. Remote access to devices for configuration or update is not provided by this server, but only by other standard protocols like Telnet, FTP, HTTP, etc, that a user can also use with standard applications like Telnet- or FTP-programs or Web-Browsers.

The Discovery Server listens to UDP/IP-packets on port 7311 and accepts packets sent to three different kinds of addresses:

- The currently configured IP-address of the device (e.g. 192.168.1.1)
- The IPv4 network broadcast address (255.255.255.255)
- The directed broadcast address of the network (e.g. 192.168.1.255)

The Discovery Server accepts two different packets:

- A packet containing the string “GF DISCOVER QUERY” requests the device to send information about itself. The packet contains no further data.
- A packet containing the string “GF DISCOVER OK” signals to the device that the remote station has successfully received the information-packet sent by the device. The packet contains no further data.

The Discovery Server will send a single type of packet when asked to do so by a remote station:

- A packet containing the string “GF DISCOVER REPLY” informs a remote station about the device. This packet contains a set of ASCII-strings with information about the device.

Discovery data sent by RedBoot (when an operating system is running the contents of the fields documented here may be different and additional fields may be provided):

“Name: <name>”	Device-name configured via “fconfig”
“Description: <description>”	An informational description of the system
“System: RedBoot Vx.x.xxxx”	RedBoot version running on the device.
“Platform: <platform>”	Name of the platform of the system, e.g. “ADELAIDE”
“Version: x.x.xxxx”	for Redboot the same version as in “System” is sent
“Vendor: Garz & Fricke GmbH”	
“SerialModule: <serial#>”	The serial number of the Mini-Module. Currently this will be determined by the Ethernet MAC-address.
“Serial*: <serial#>”	Serial numbers of other components in the system. (Currently none sent).
“Services: <services>”	A comma-separated list of network services the system supports for remote access. Currently defined: TEL : Telnet FTP : FTP (not TFTP) HTP : HTTP

As of RedBoot v1.6, RedBoot only supports “TEL”. Future RedBoot releases and operating systems may also provide access via FTP, HTTP, or other standard protocols.

8 Information passed to OS

RedBoot provides different commands for executing an OS or application:

go: jumps to the given entry point with caches turned-off but MMU turned on
run: jumps to the given entry point with caches and MMU turned off
exec: jumps to the given entry point with caches and MMU turned off and passing parameters in a way compatible with Linux-ARM kernels

While all three commands are available in the RedBoot provided by Garz & Fricke, only “exec” is supported for full information passing to the OS. Therefore this is the recommended way to start an OS or application and should be supported by all operating systems.

8.1 Information passed in registers

When starting an OS (or application) via the “exec” command RedBoot passes information in registers, RAM and flash to the OS. The parameter passing is based on the conventions used by Linux-ARM kernels, i.e. some information is passed in CPU-registers and additional information is passed in a so called ATAGs-list in RAM. The address of this list is given in a register:

Register	Description
R0	Board type (currently unused; set to 0)
R1	Machine type (see below)
R2	Address of ATAGs list (physical; usually 0x80010000)

Machine type	Platform
447	Garz & Fricke system based on Freescale i.MX31 SoC
1965	Garz & Fricke system based on Freescale i.MX27 SoC
2213	Garz & Fricke system based on Freescale i.MX35 SoC
2671	Garz & Fricke system based on Freescale i.MX25 SoC

8.2 Information passed in RAM

RedBoot passes a variety of information via ATAG structures stored in RAM. The address of this list of structures is passed in a register (see above). The types of structures passed and their contents may be dependent on the parameters given to the “exec”-command and may be different for different RedBoot releases or systems.

8.2.1 ATAG_CORE(0x54410001)

```
struct tag_core {
    UINT32 flags;        /* bit 0 = read-only */
    UINT32 pagesize;
    UINT32 rootdev;
};
```

This is a mandatory structure. As of RedBoot v1.4 this structure is passed, but unused and set to 0.

8.2.2 ATAG_INITRD2 (0x54410006)

```
struct tag_initrd {
    UINT32 start;
    UINT32 size;
};
```

This structure is only passed when the ramdisk-parameters to the “exec” command are given. It specifies the physical start-address and size of the INITRD-ramdisk in RAM. (The ramdisk contents are not loaded automatically by RedBoot, though).

8.2.3 ATAG_MEM(0x54410002)

```
struct tag_mem32 {
    UINT32  size;
    UINT32  start;
};
```

This structure is always passed and specifies physical start address and size of SDRAM.

8.2.4 ATAG_REVISION(0x54410007)

```
struct tag_revision {
    UINT32  rev;
};
```

Starting with RedBoot V1.5 this structure is always passed and contains the RedBoot version. The version information is stored as follows:

31 … 24	23 … 16	15 … 0
Major	Minor	Build number

8.2.5 ATAG_SERIAL(0x54410006)

```
struct tag_serialnr {
    UINT32 low;
    UINT32 high;
};
```

This structure is always passed. As on Garz & Fricke systems the serial number is associated with the Ethernet MAC address this structure simply passes the MAC address as serial number. (For a description of the relationship between the MAC address and board serial number please see the description of the RedBoot configuration options).

8.2.6 ATAG_CMDLINE(0x54410009)

```
struct tag_cmdline {
    char cmdline[1];
};
```

This structure is always passed. The command-line string comprises certain parameters that are generated automatically by RedBoot and user-specified parameters given with the “-c”-option to “exec”. Automatically generated parameters are always passed first.

The following parameters will be generated automatically:

8.2.6.1 ip=<local-IP>:<server-IP>:<gateway>:<netmask>:<name>:<if>:<bootp>

This parameter passes the current network configuration used by RedBoot to the OS.

<local-IP> is the IPv4 address used by the device
<server-IP> is the IPv4 address for the default server
<gateway> is the IPv4 address for the default gateway server
<netmask> is the IPv4 netmask used by the device
<name> is the device name used by the device
<if> is the name of the network interface (usually “eth0”)
<bootp> specifies if BOOTP/DHCP is to be used to obtain the IP-configuration (set to “bootp”) or whether a static IP configuration is to be used (set to “off”).

8.2.6.2 rbfis=<cur-addr>,<backup-addr>,<length>[,<flash_length>]

This parameter specifies the location of the RedBoot FIS-configuration data in flash, i.e. the configuration data shown and manipulated with the fis-commands. Pseudo-virtual addresses are passed here, as used by the RedBoot fis-commands, i.e. addresses in the 0xA0000000-area denote configuration data in NOR-Flash. Other addresses denote addresses in NAND flash and are interpreted with a base-address of 0xE0000000., i.e. to get the physical address in NAND flash, 0xE0000000 must be subtracted from the given address.

<cur-addr> specifies the flash-address of the most recent FIS-configuration data.
<backup-addr> specifies the flash-address of the backup FIS-configuration data. (If RedBoot is not configured to keep redundant FIS data this address will be set to 0xFFFFFFFF).
<length> specifies the maximum length of data storing FIS-information, i.e. the FIS-data will require at most <length>-bytes for the actual data and CRC.
<flash_length> (introduced with RedBoot v1.6) specifies the area occupied in flash for the FIS area. If given, it will always be >= <length>. If not given, <length> also specifies the size of the area occupied in flash. The backup FIS-data, if present, occupies an area of the same size.

For a description of the contents and usage policy please refer to the chapter on RedBoot configuration data. By default all RedBoot versions released by Garz & Fricke use redundant FIS data, i.e. <backup-addr> will be a valid flash address and be different from <cur-addr>.

8.2.6.3 rbcfg=<cur-addr>,<length>[,<flash_length>]

This parameter specifies the location and size of RedBoot configuration data, i.e. configuration data shown and manipulated with the fconfig-command. Pseudo-virtual addresses are passed here, as used by the RedBoot fis-commands, i.e. addresses in the 0xA0000000-area denote configuration data in NOR-Flash. Other addresses denote addresses in NAND flash and are interpreted with a base-address of 0xE0000000., i.e. to get the physical address in NAND flash, 0xE0000000 must be subtracted from the given address.

<cur-addr> specifies the flash-address of the current configuration data
<length> specifies the maximum length of data storing configuration-information, i.e. the configuration-data will require at most <length>-bytes for the actual data and CRC.
<flash_length> (introduced with RedBoot v1.6) specifies the area occupied in flash for the configuration area. If given, it will always be >= <length>. If not given, <length> also specifies the size of the area occupied in flash.

If the configuration data is stored together with the FIS-data, <cur-addr> of the rbcfg-parameter will be identical to <cur-addr> of the rbfis-parameter. This is the default for all RedBoot versions released by Garz & Fricke.

8.2.6.4 vidmem=<bIsOn>,<vmem_start>,<vmem_length>

RedBoot versions with support for a boot logo (v1.8 or later) will always generate this parameter (even if no display configuration is installed in RedBoot).

<bIsOn> can be 0 or 1 and specifies if the display is on and showing something on the screen (1) or off (0). When the display is on, an OS should take care not to turn-off the display or display-controller or overwrite the video memory.

<vmem_start> physical start address of video memory reserved by RedBoot in RAM. Only valid if <vmem_length> != 0.

<vmem_length> size of video memory reserved by RedBoot. If no display configuration is installed or start-up of the video driver failed for some reason, <vmem_length> will be 0 to tell the OS that neither memory has been reserved for video, nor has the display or display-controller been initialized.

The amount of video memory reserved by RedBoot is specified in the display configuration files and should be sufficient for an OS (if not, the display configuration should be changed accordingly). RedBoot will only allocate one block of memory right at the start of this area as a framebuffer memory to be displayed on screen. The OS (and especially the display driver in the OS) should pay attention to this so that no garbage is displayed during the boot-phase of the OS.

8.2.6.5 -q

This is an optional parameter that is passed when the serial debug console of RedBoot has been disabled, telling the OS not to output debugging messages on any serial ports. This information is passed redundantly in the form of this command-line parameter and inside the XML-configuration data, so that the OS is aware of the availability of the serial debug port even before being able to access the XML configuration data.

8.2.6.6 bld=<build_date>:<svn-revision>

This parameter has been introduced with RedBoot v1.6 to provide additional version information about RedBoot to the OS and augments ATAG_REVISION:

<build_date> specifies the date-of-compilation of the RedBoot release.

<svn-revision> specifies the SubVersion revision number the RedBoot release was built from.

9 Compiling from Source

Garz & Fricke provides the full eCos source-tree including RedBoot and all required modifications and extensions to rebuild the binary RedBoot images shipped by Garz & Fricke. To rebuild the sources a properly setup build-environment is required:

- Developers wishing to recompile RedBoot on a Microsoft Windows host require a recent version of the Cygwin UNIX emulation systems. For detailed instructions on how to install Cygwin please refer to http://ecos.sourceware.org/cygwin.html .
- For RedBoot v1.9 and earlier an "**arm-elf**", for RedBoot v1.10 and later an "**arm-eabi**" GNU cross toolchain is required to compile the source-code. Suitable pre-built toolchains running under Linux and Cygwin are available for download from the eCos mirror sites (http://ecos.sourceware.org/mirror.html). The exact location of installation doesn't matter as long as the tools are available via the environment PATH variable from the bash or Cygwin shell.
- The eCos host configurations tools must be installed. It is suggested that these tools be installed into a folder accessible via the environment PATH variable, as well. (http://www.ecoscentric.com/devzone/configtool.shtml)
- The eCos source-tree provided by Garz & Fricke, which is provided as a Tar.bz2 packages must be installed. It includes the full eCos source-tree, so that the installation folder on the hard-disk can be directly specified as root of the eCos repository in the eCos host configuration tools.

To generate the RedBoot image we suggest using the graphical eCos host configuration tool "configtool". When starting the tool for the first time, the "root of the eCos repository tree" must be specified. Please select the folder you installed the source-code into. Garz & Fricke, for example, is using the Cygwin environment under Windows-XP, with the eCos source-tree installed into the folder "U:\ecos", so this would be the folder to specifiy as "root of the eCos repository tree".

To configure the build-tree open the menu "Build-->Templates...", select "Garz&Fricke i.MX31 ADELAIDE board" as hardware and select "redboot" as package in the dialog-box that opens (other selections are currently not supported by Garz & Fricke) and click on "OK". A "Resolve conflicts" dialog will open that should just be accepted by clicking on "Continue". Afterwards the configuration file describing the setup of RedBoot for ADELAIDE must be imported by opening "File-->Import...". The configuration file is located in the eCos source-tree in the folder "packages\hal\arm\mx31\guf-adl\current\misc" and called "redboot_ROMRAM.ecm". Another "Resolve conflicts" folder will open, which, again, should just be confirmed by clicking on "Continue".

Now everything should be setup to generate the image: clicking on "Build-->Library" will ask for a name and location to save the results to (this should be a folder outside the eCos tree) and start the compilation process. The final binary that can be installed on the ADELAIDE board will be the file "XXX_install\bin\redboot.bin" placed in the build-folder. This is a plain binary file without any special headers and may be downloaded, started and installed using an already installed RedBoot on an ADELAIDE system or it may be directly written to the start of the NOR and/or NAND flash using a JTAG-tool.

10 Online support

Support for your Garz & Fricke embedded device is available on the Garz & Fricke website. You may find the latest documentation and updates for your system on the web:

Product	Link to download area
ADELAIDE	▶ http://www.garz-fricke.com/adelaide-download
AUCKLAND	▶ http://www.garz-fricke.com/auckland-download
JUPITER	▶ http://www.garz-fricke.com/jupiter-download
CALLISTO	▶ http://www.garz-fricke.com/callisto-download
NESO	▶ http://www.garz-fricke.com/neso-download
CUPID	▶ http://www.garz-fricke.com/cupid-download
NESO LT	▶ http://www.garz-fricke.com/nesolt-download

Of course, help is also available via email to ▶ support@garz-fricke.com

Annex A: Trademarks and service marks

There are a number of proprietary logos, service marks, trademarks, slogans and product designations ("Marks") used in this document. By making the Marks available in this document, Garz & Fricke GmbH is not granting you a license to use them in any fashion.

The following Marks are the property of Garz & Fricke GmbH. This list is not comprehensive; the absence of a Mark from the list does not constitute a waiver of intellectual property rights established by Garz & Fricke GmbH in a Mark.

AUCKLAND, ADELAIDE, CALLISTO, CUPID, GANYMED, Flash'nGo, JUPITER, NESO, NESO LT, related XY *Starter Kits* and subversions (XY *"core*", "*open frame*", "*boxed*") are registered trademarks or products of Garz & Fricke GmbH, Hamburg.

Other product or service names may be the property of third parties. Marks owned by third parties include those listed below. This list is not comprehensive; the absence of a Mark from the list does not constitute a waiver of intellectual property rights established by the owner of a Mark.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off.

ARM is the registered trademark of ARM Limited. *ARMWXYZ* is the trademark of ARM Limited.

Dolby Digital, Dolby Surround®, Pro Logic® and the double-D symbol are registered trademarks of Dolby Laboratories; Dolby Digital is manufactured under license from Dolby Laboratories.

Java™ and all other Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

eCosCentric and *eCos* are registered trademarks of eCosCentric Ltd.

Microsoft, Windows, Windows Embedded CE, Windows NT, Visual Studio, Visual C++, Visual C#, MFC and *Visual C++* are registered trademarks, trademarks or products of Microsoft Corporation in the United States and/or other countries.

Sharp is a registered trademark of Sharp Electronics Europe GmbH.

RedBoot is a registered trademark of Red Hat Inc.

Linux is a registered trademark of Linus Torvalds.

Their use is subject to national and international laws and agreements. Every use of these names in this documentation occurs subject to the legal regulations. While trademark symbols may be omitted for the purpose of simplification, they are implied when the names of the trademarks are used in the remainder of this document and should be interpreted as present.

Annex B: License

The eCos and RedBoot source-code provided by Garz & Fricke is released, just as the regular eCos/RedBoot distribution, under a modified version of the GNU General Public License v2 (GPL). The following sections are verbatim citations of the eCos and GPL licenses.

B.1 eCos License

As of May 2002, eCos is released under a modified version of the well known GNU General Public License (GPL) (http://www.gnu.org/copyleft/gpl.html), now making it an official GPL-compatible Free Software License (http://www.gnu.org/philosophy/license-list.html). An exception clause has been added to the eCos license which limits the circumstances in which the license applies to other code when used in conjunction with eCos. The exception clause is as follows:

As a special exception, if other files instantiate templates or use macros or inline functions from this file, or you compile this file and link it with other works to produce a work based on this file, this file does not by itself cause the resulting work to be covered by the GNU General Public License. However the source code for this file must still be made available in accordance with section (3) of the GNU General Public License. This exception does not invalidate any other reasons why a work based on this file might be covered by the GNU General Public License.

The goal of the license is to serve the eCos user community as a whole. It allows all eCos users to develop products without paying anybody anything, no matter how many developers are working on the product or how many units will be shipped. The license also guarantees that the eCos source code will always be freely available. This applies not only to the core eCos code itself but also to any changes that anybody makes to the core. In particular, it should prevent any company or individual contributing code to the system and then later claiming that all eCos users are now guilty of copyright or patent infringements and have to pay royalties. It should also prevent any company from making some small improvements, calling the result a completely new system, and releasing this under a new and less generous license.

The license does not require users to release the source code of any applications that are developed with eCos. However, if anybody makes any changes to code covered by the eCos license, or writes new files derived in any way from eCos code, then we believe that the entire user community should have the opportunity to benefit from this. The license stipulates that these changes must be made available in source code form to all recipients of binaries based on the modified code, either by including the sources along with the binaries you deliver (or with any device containing such binaries) or with a written offer to supply the source code to the general public for three years. It is perhaps most practical for eCos developers to make the source code available online and inform those who are receiving binaries containing eCos code, and probably also the eCos maintainers, about the location of the code. See the full text of the GPL (http://www.gnu.org/copyleft/gpl.html) for the most authoritative definition of the obligations.

Although it is not strictly necessary to contribute the modified code back to the eCos open source project, we are always pleased to receive code contributions and hope that developers will also be keen to give back in return for what they received from the eCos project completely free of charge. The eCos maintainers are responsible for deciding whether such contributions should be applied to the public repository. In addition, a copyright assignment (http://sources.redhat.com/ecos/assign.html) is required for any significant changes to the core eCos packages.

The result is a royalty-free system with minimal obligations on the part of application developers. This has resulted in the rapid uptake of eCos. At the same time, eCos is fully open source with all the benefits that implies in terms of quality and innovation. We believe that this is a winning combination.

B.2 GNU General Public License v2

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

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 Lesser 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 non-commercial 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

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

Annex C: Document History

Release/Date	Title	Description
r0.1, 31.10.2007	Initial document release	
r0.2, 25.02.2008	Updated for RedBoot V1.6	Changed sections: 3 RedBoot Memory Layout 4.1.1 Handling of NOR and NAND flash 4.1.2 Bad-block handling in NAND flash 4.1.3 Flash layout 4.1.4.2 fis list 4.1.4.3 fis create 4.2.1 Boot scripts 4.2.3 Ethernet configuration 4.2.7 Miscellaneous settings 5.1.6 testram 5.1.7.3 nand load 6.2.1 Installing a Windows-CE OS image 8.2.6.2 rbfis 8.2.6.3 rbcfg New sections: 4.2.2 Logbook 4.4 Logbook feature 5.1.8 logbook 5.1.9 net_init 7 Discovery Server 8.2.6.5 bld 12 Annex B: Document History
r0.3, 05.03.2009	Updated for RedBoot V1.8	Changed sections: 2.3.2 Uploading files via Ethernet (added note on firewalls) 3 RedBoot Memory Layout (added additional mappings introduced since v1.6) (added information on video memory) 4.1.2 Bad-block handling in NAND flash (added note on required OS-support for 2k-sector flashes) 4.1.4.1 fis init (added “-y”-parameter) 4.1.4.5 fis delete (added “-y”-parameter) 4.2.2 Logbook (added note on logbook being disabled by default in v1.8+) 4.4 Resetting configuration (changed wording from “factory defaults” to “save defaults”) 4.5 Logbook feature (added note on logbook being disabled by default in v1.8+) 4.6.2.2.1 <partition>-tag (added note on additional partitions with special meaning) New sections: 4.3 Boot Logo 4.6.2.3.2 <display>-tag 4.6.2.3.3 <logo-license>-tag 5.1.10 xconfig 6.4 How To set up a Boot Logo 8.2.6.4 vidmem=<bIsOn>,<vmem_start>,<vmem_length> Minor corrections throughout the document.
r0.4, 01.04.2009	Update for RedBoot V1.9	Changed sections: 3 RedBoot Memory Layout (added memory-mapping for NESO-platform) 4.1.1 Handling of NOR and NAND flash (added remark on optional NOR flash and factive-command)

		4.1.2 Bad-block handling in NAND flash (added information on nature of bad-blocks) 4.3 Boot Logo (added information on license-free Garz & Fricke boot logo images for RedBoot V1.9+) 6.1 How To update RedBoot (fixed Ethernet command sequence) 6.4.1 (Boot Logo) first time set up (added information on license-free Garz & Fricke boot logo images for RedBoot V1.9+) New sections: 6.4.4 Installing a boot logo license Small changes throughout the document to make it clearer that this documentation is valid for all RedBoot-based Garz & Fricke systems. Removed product name from document title
r0.5, 07.05.2009	Redesign of the documentation	
r0.6, 18.06.2010	Update for RedBoot V1.12	Changed sequence of chapters Changed sections (old chapters numbers): Important hints 2.2.2 Connection via Ethernet (added note that cancellation of boot-scripts is impossible via Telnet) 3 RedBoot Memory Layout (Updated for RedBoot v1.12 and added i.MX25 and i.MX35 platforms) 4.6.1 Storage in flash (of XML data) (fixed information on storage of CRC-checksum on configuration data) 6.2.1 Installing a Windows-CE OS image (changed example for creating flash-disk partition and added some notes. Added note on “Registry” partition.) 8.1 Information passed in registers (added machine types for i.MX25 and i.MX35 systems) 9 Compiling from Source (updated toolchain requirements for RedBoot v.10 and later) New sections: 4.4.4 Getting control over a miss-configured system Removed most of the product names from the document to make it clearer that this documentation is valid for all RedBoot-based Garz & Fricke systems. Annex A: revised trademark information
V.0.6.1 13.7.2010	Minor changes	Changed: Revision to Version Changed: 2.2.1 f. & 3.2 ff. added information how to enter RedBoot mode

Art-Nr. 902-0127