20G052A00 E2 • 2017-03-22

G52A

QorIQ Enhanced Network CPU Board

3U CompactPCI Serial

User Manual

Contents

Figures

Tables

About this Document

This user manual is intended only for system developers and integrators, it is not intended for end users.

It describes the design, functions and connection of the product. The manual does not include detailed information on individual components (data sheets etc.).

G52A product page with up-to-date information and downloads:
www.men.de/products/g52a/

History

Issue	Comments	Date
E1	First issue	2016-07-13
E2	General update, minor errors corrected	2017-03-22

Conventions

Indicates important information or warnings concerning situations which could result in personal injury, or damage or destruction of the component.

Indicates important information concerning electrostatic discharge which could result in damage or destruction of the component.

Indicates important information or warnings concerning proper functionality of the product described in this document.

The globe icon indicates a hyperlink that links directly to the Internet. When no globe icon is present, the hyperlink links to specific information within this document.

Italics	Folder, file and function names are printed in *italics*.
Comment	Comments embedded into coding examples are shown in green text.
IRQ# /IRQ	Signal names followed by a hashtag "#" or preceded by a forward slash "/" indicate that this signal is either active low or that it becomes active at a falling edge.
In/Out	Signal directions in signal mnemonics tables generally refer to the corresponding board or component, "in" meaning "to the board or component", "out" meaning "from the board or component".
`0xFF`	Hexadecimal numbers are preceded by "`0x`".
`0b1111`	Binary numbers are preceded by "`0b`".

Product Safety

Electrostatic Discharge (ESD)

Computer boards and components contain electrostatic sensitive devices. Electrostatic discharge (ESD) can damage components. To protect the PCB and other components against damage from static electricity, you should follow some precautions whenever you work on your computer.

- Power down and unplug your computer system when working on the inside.
- Hold components by the edges and try not to touch the IC chips, leads, or circuitry.
- Use a grounded wrist strap before handling computer components.
- Place components on a grounded antistatic pad or on the bag that came with the component whenever the components are separated from the system.
- Only store the product in its original ESD-protected packaging. Retain the original packaging in case you need to return the product to MEN for repair.

Legal Information

Changes

MEN Mikro Elektronik GmbH ("MEN") reserves the right to make changes without further notice to any products herein.

Warranty, Guarantee, Liability

MEN makes no warranty, representation or guarantee of any kind regarding the suitability of its products for any particular purpose, nor does MEN assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages. TO THE EXTENT APPLICABLE, SPECIFICALLY EXCLUDED ARE ANY IMPLIED WARRANTIES ARISING BY OPERATION OF LAW, CUSTOM OR USAGE, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR USE. In no event shall MEN be liable for more than the contract price for the products in question. If buyer does not notify MEN in writing within the foregoing warranty period, MEN shall have no liability or obligation to buyer hereunder.

The publication is provided on the terms and understanding that:

1. MEN is not responsible for the results of any actions taken on the basis of information in the publication, nor for any error in or omission from the publication; and
2. MEN is not engaged in rendering technical or other advice or services.

MEN expressly disclaims all and any liability and responsibility to any person, whether a reader of the publication or not, in respect of anything, and of the consequences of anything, done or omitted to be done by any such person in reliance, whether wholly or partially, on the whole or any part of the contents of the publication.

Conditions for Use, Field of Application

The correct function of MEN products in mission-critical and life-critical applications is limited to the environmental specification given for each product in the technical user manual. The correct function of MEN products under extended environmental conditions is limited to the individual requirement specification and subsequent validation documents for each product for the applicable use case and has to be agreed upon in writing by MEN and the customer. Should the customer purchase or use MEN products for any unintended or unauthorized application, the customer shall indemnify and hold MEN and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim or personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that MEN was negligent regarding the design or manufacture of the part. In no case is MEN liable for the correct function of the technical installation where MEN products are a part of.

Qualified Personnel

The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Conformity

MEN products are no ready-made products for end users. They are tested according to the standards given in the Technical Data and thus enable you to achieve certification of the product according to the standards applicable in your field of application.

RoHS

Since July 1, 2006 all MEN standard products comply with RoHS legislation.

Since January 2005 the SMD and manual soldering processes at MEN have already been completely lead-free. Between June 2004 and June 30, 2006 MEN's selected component suppliers have changed delivery to RoHS-compliant parts. During this period any change and status was traceable through the MEN ERP system and the boards gradually became RoHS-compliant.

WEEE Application

The WEEE directive does not apply to fixed industrial plants and tools. The compliance is the responsibility of the company which puts the product on the market, as defined in the directive; components and sub-assemblies are not subject to product compliance.
In other words: Since MEN does not deliver ready-made products to end users, the WEEE directive is not applicable for MEN. Users are nevertheless recommended to properly recycle all electronic boards which have passed their life cycle.

Nevertheless, MEN is registered as a manufacturer in Germany. The registration number can be provided on request.

Germany

MEN Mikro Elektronik GmbH
Neuwieder Straße 3-7
90411 Nuremberg
Phone +49-911-99 33 5-0

info@men.de
www.men.de

France

MEN Mikro Elektronik SAS
18, rue René Cassin
ZA de la Châtelaine
74240 Gaillard
Phone +33-450-955-312

info@men-france.fr
www.men-france.fr

USA

MEN Micro Inc.
860 Penllyn Blue Bell Pike
Blue Bell, PA 19422
Phone 215-542-9575

sales@menmicro.com
www.menmicro.com

China

MEN Mikro Elektronik (Shanghai) Co., Ltd.
Room 808-809, Jiaxing Mansion, No. 877 Dongfang Road
200122 Shanghai
Phone +86-21-5058-0961

sales@men-china.cn
www.men-china.cn

1 Product Overview

1.1 Product Description

New Generation of CPU Boards

The G52A is a high-performance multicore CPU platform based on NXP (formerly Freescale) QorIQ T4x series. The G52A is a new branch of CPU boards for CompactPCI Serial specifically designed for high data bandwidth based on PCIe 3.0, PCIe 2.0 and Gigabit Ethernet via the backplane. The G52A provides high data bandwidth on the front panel via two 10 Gigabit Ethernet interfaces on M12 connectors or on RJ45 connectors (on request). The G52A paired with I/O cards can be ideally used for transferring data from and to storage media, the Internet via LTE, WiFi, copper or optical Ethernet. Its up to 12 processor cores make the board ideally suited for virtualization applications. Serial interfaces at the rear I/O connectors are one USB 2.0, two SATA interfaces, three PCI Express x4 links and one PCI Express x2 link and three Gigabit Ethernet interfaces.

Three-channel DDR3 DRAM

The memory configuration of the G52A includes a fast DDR3 DRAM with ECC which is soldered to the board to guarantee optimum shock and vibration resistance. A microSD card device offers space for user applications or can be used as a local boot medium.

Board Supervision

The G52A features thermal supervision of the processor and a watchdog for the operating system.

Perfect for Harsh Environments

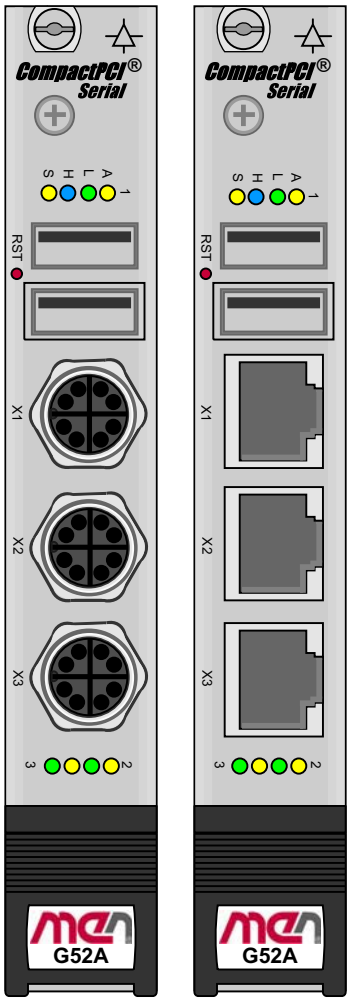
The G52A comes with a tailored heat sink within 4 HP height. All components are soldered for protection against shock and vibration according to applicable DIN, EN or IEC industry standards. The G52A is also ready for coating so that it can be used in humid and dusty environments and has a guaranteed minimum standard availability of 15 years. These features make the G52A perfectly suited for harsh environments.

Software

The G52A operates in Linux environments or as a development option in VxWorks or QNX environments.

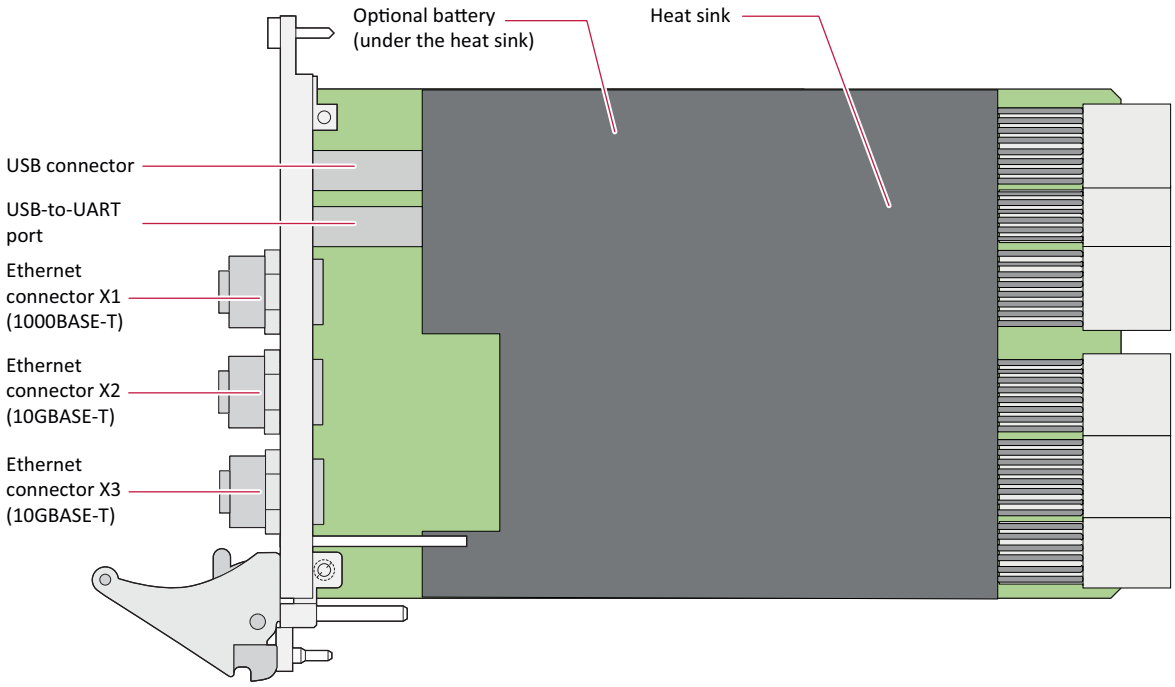
1.2 External Interfaces

Figure 1. Front interfaces (M12 connectors on the left, RJ45 connectors on the right)



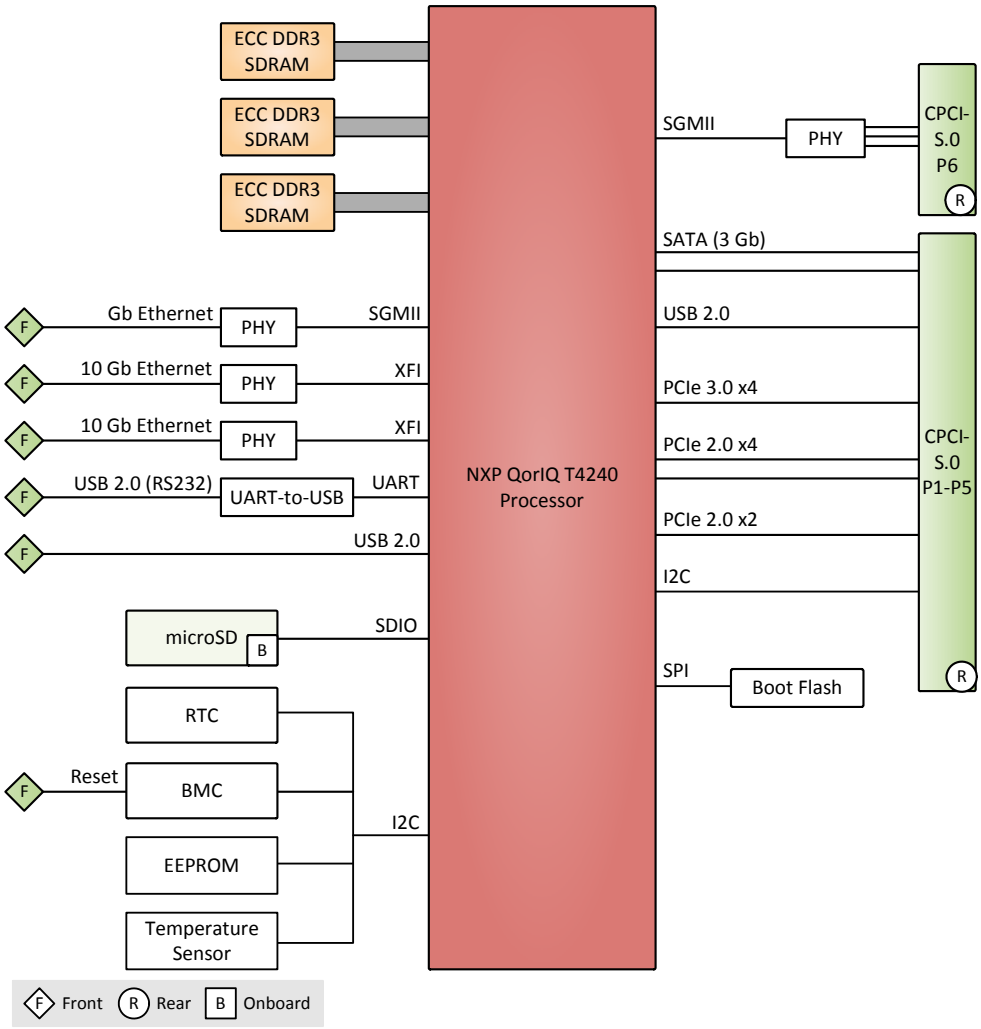
1.3 Board Layout

Figure 2. Board layout – top view



1.4 Block Diagram

Figure 3. Block diagram



1.5 Technical Data

CPU

- The following CPU types are available:
  - NXP QorIQ T4240, twelve cores, 1.5 to 1.8 GHz
  - NXP QorIQ T4160, eight cores, 1.5 to 1.8 GHz (on request)
  - NXP QorIQ T4080, four cores, 1.5 to 1.667 GHz (on request)

Memory

- System Memory
  - Soldered DDR3, ECC support
  - Up to 3 channels
  - 4 GB, 6 GB, 8 GB or 12 GB
- Boot Flash
  - 32 MB

Mass Storage

- The following mass storage device can be assembled:
  - microSD card (up to 32 GB)

Front Interfaces

- USB
  - One Series A connector, USB 2.0
  - One RS232 configuration port, USB 2.0
- Ethernet
  - Two 8-pin M12 connectors, X-coded, 10GBASE-T, or
  - Two RJ45 connectors, 10GBASE-T (option)
  - One 8-pin M12 connector, X-coded, 1000BASE-T, or
  - One RJ45 connector, 1000BASE-T (option)
  - One link and activity LED per Ethernet channel
- Front-panel LEDs for board status
- Hot-plug LED
- Reset button

Rear Interfaces

- SATA
  - Two channels, SATA Revision 2.x
- USB
  - One channel, USB 2.0
- Ethernet
  - Three channels, 1000BASE-T
- PCI Express
  - Two x4 links, PCIe 2.x
  - One x2 link, PCIe 2.x
  - One x4 link, PCIe 3.x

Supervision and Control

- Board controller
- Watchdog timer
- Temperature measurement
- Real-time clock with supercapacitor or battery backup

Backplane Standard

- Compliance with CompactPCI Serial PICMG CPCI-S.0 Specification
- System or peripheral slot

Electrical Specifications

- Supply voltages
  - +12V (9.5..15.5V)
- Power consumption
  - The following values are valid for product model 02G052A00, which uses a T4240 processor.
  - +12V: 4 A nominal, 8 A maximum

Mechanical Specifications

- Dimensions
  - 3U, 4 HP
- Weight: approx. 266 g (model 02G052A00)

Environmental Specifications

- Temperature range (operation)
  - EN 50155 class T1, T2, T3 or TX
  - Airflow 2.5 m/s
  - Depends on system configuration (CPU, hard disk, heat sink...)
- Temperature range (storage): -40°C to +85°C
- Cooling concept
  - Air-cooled, or
  - Conduction-cooled in MEN CCA frame
- Humidity: EN 60068-2-30, EN 50155
- Altitude: -300 m to +3000 m
- Shock: EN 50155 cat 1 class b
- Vibration: EN 50155 cat 1 class b

Safety

- Flammability
  - UL 94V-0
- Electrical Safety
  - EN 62368-1 (former EN 60950-1)

EMC Conformity

- EN 55022 class B, EN 50121-3-2 (radiated and conducted emission)
- EN 55024, EN 50121-3-2 (immunity)

BIOS

- U-Boot Universal Boot Loader

Software Support

- Linux

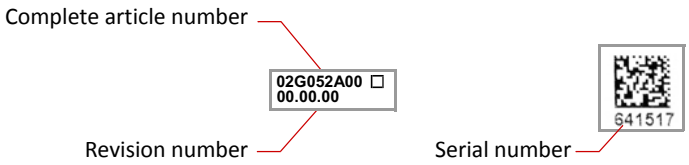
See the MEN website for more information on supported operating system versions and drivers.

1.6 Product Identification

MEN user documentation may describe several different models and/or design revisions of the G52A. You can find information on the article number, the design revision and the serial number on a label affixed to the board.

- **Article number:** Indicates the product family and model. This is also MEN's ordering number. To be complete it must have 9 characters.
- **Revision number:** Indicates the design revision of the product.
- **Serial number:** Unique identification assigned during production.

Figure 4. Product labels



2 Getting Started

2.1 Configuring the Hardware

You should check your hardware requirements before installing the board in a system, since most modifications are difficult, or even impossible, to do once the board is mounted in a rack.

The following chapters provide an overview on configuration possibilities.

2.1.1 microSD Card

The board is shipped without a microSD card. You should check your needs and install a suitable microSD card.

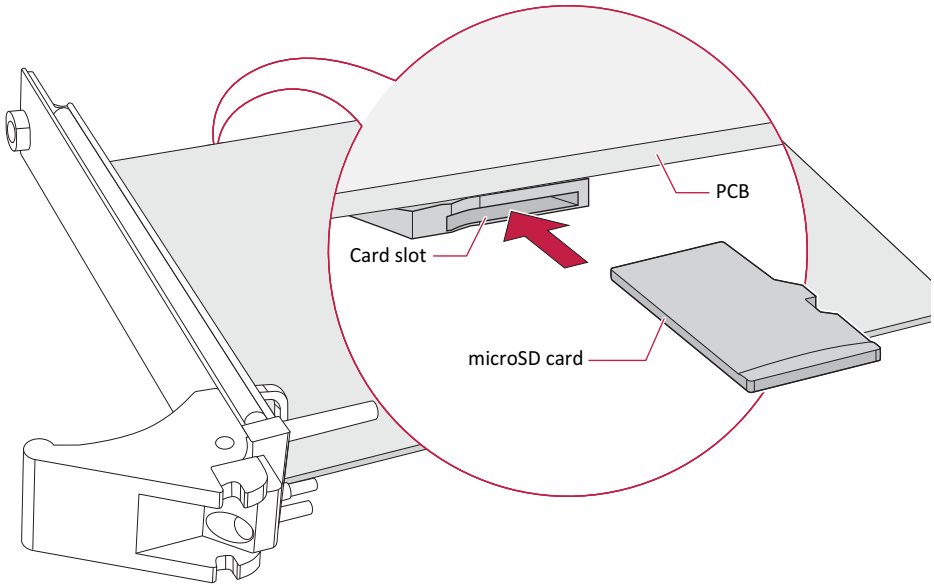
See the MEN website for ordering information:
www.men.de/products/g52a/#ord

See Chapter 3.7.1 microSD Card on page 28 for a detailed description.

2.1.1.1 Inserting and Extracting a microSD Card on the G52A

To install a microSD card, please stick to the following procedure.

- Power down your system and remove the G52A from the system.
- Put the board on a flat surface.
- Insert the microSD card into the slot with the contacts at the top.



- Make sure that it clicks into place properly.
- For extracting the card push it down and pull it out.

2.2 Connecting and Starting

You can use the following check list when installing the board in a system for the first time and with minimum configuration.

» Power down the system.

» Remove all boards from the CompactPCI Serial system.

» Insert the G52A into the system slot of your CompactPCI Serial system, making sure that the backplane connectors are properly aligned.

Note: The system slot of every CompactPCI Serial system is marked by a △ triangle with a plus sign behind it on the backplane and/or at the front panel. It also has red guide rails.

» Connect a terminal to the USB-to-UART interface at the front panel.

See the MEN website for a suitable service cable:
www.men.de/products/g52a/#ord

Note: An FTDI driver must be installed on the PC your terminal is running on.

See the FTDI chip website for all available drivers:
http://www.ftdichip.com/FTDrivers.htm

» Set your terminal to the following protocol:

- 115 200 baud data transmission rate
- 8 data bits
- 1 stop bit
- No parity

» Power up the system.

» U-Boot will load and then display a command line.

» Now you can make configurations in the U-Boot.

See Chapter 4 U-Boot Firmware on page 37.

2.3 Troubleshooting at Start-up

If you have any problems at start-up of the G52A, you can check if the front-panel status LED gives an error flash code.

See Chapter 3.3.3.1 Board Status LED on page 25.

You can also start the board with firmware default settings for troubleshooting.

See Chapter 4 U-Boot Firmware on page 37 for a detailed description.

2.4 Installing Operating System Software

By default, no operating system is installed on the G52A.

- Please refer to the respective manufacturer's documentation on how to implement the operating system.
- See the MEN website for all available software: www.men.de/products/g52a/#downl

See Chapter 4.2 Getting Started: Setting Up Your Operating System on page 37 for details on the first steps of how to get your operating system running with U-Boot.

2.5 Installing Driver Software

For a detailed description on how to install driver software, please refer to the respective documentation of the software package to be installed.

See the MEN website for all available software: www.men.de/products/g52a/#downl

2.6 Using the G52A under Linux

This chapter describes how to install and use Linux software together with the G52A. A detailed step-by-step description is given where needed.

2.6.1 Accessing Board Management Functions

There are two ways to access board management functions, e.g., the board management controller (BMC), under Linux:

- Using MEN software tools for Linux.
- Using standard Linux I2C tools.

More information on supported functions and hardware implementation

- See Chapter 3.3 Supervision and Management on page 24.
- See Chapter 5.3 BMC API (Application Programming Interface) on page 55 for a detailed description of the BMC API.
- See Chapter 5.2 I2C Devices on page 54.

2.6.1.1 MEN Tools

MEN provides a number of MDIS software tools for accessing G52A functions via the SMBus, which are included in the G52A Linux board support package (BSP):

- *xm01bc_ctrl* is the tool for accessing the BMC
- *smb2_eetemp* is the tool for accessing the temperature sensor

In the following you can find an exemplary description of how to access the BMC:

» Install the G52A Linux BSP.

See the G52A Linux BSP user manual for a detailed BSP description:
www.men.de/products/g52a/#doc

» Print a list of all possible parameters of the *xm01bc_ctrl* tool:

```
$ sudo xm01bc_ctrl <device name> <device name> is a place holder for the name of the device

Usage: xm01bc_ctrl [<opts>] <device> [<opts>]

Function: Control XM01BC PIC

Options:
  device      device name
  -v          show voltage values
  -s          do voltage supervision (requires option -v)

  -r=0xdead   perform SW warm reset (!! dangerous !!)
  -R=0xdead   perform SW cold reset (!! dangerous !!)

  -e          show error counters
  -c          clear error counters
  -n          get number of error counters

  -f          show firmware revision
  -F          show firmware revision extended

  -w          show last reset reason
  -x          show last error
  -y          show power failure flag
  -z          clear failure registers

  -o          show operating hours counter
  -p          show power cycle counter

  -l          get LED state
  -L=<state>  set LED state

  -u          get power resume mode
  -a          get EXT_PWR_OK resume mode
  -b          get RESET_IN mode

  -h          get hardware variant ID

  -q          exit QM-Mode (for production tests only)

(c) 2008 by MEN mikro elektronik GmbH
```

» For example, if you want to look up the voltage values, use the following command:

```
$ sudo xm01bc_ctrl -v xm01bc_1
```

2.6.1.2 Standard Linux I2C Tools

Carry out the following steps to access the BMC.

This procedure is also applicable to other SMBus devices.

» Find out the number of the SMBus the BMC is located on by listing the I2C devices:

```
$ sudo i2cdetect -l (small l not number 1)

[...]
i2c-x smbus     SMBus adapter                   SMBus adapter
[...]

For example:
i2c-9 smbus     SMBus adapter                   SMBus adapter
```

» Look up the SMBus address of the BMC in Chapter 4.1 SMBus Devices on page 42. In this example, the BMC address is 0x4D (7-bit notation).

» Display the devices of the SMBus (i2c-9) to look for the BMC address 0x4D:

```
$ sudo i2cdetect -y 9

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- 08 -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 1f
20: 20 21 22 -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- 32 -- -- -- -- 37 -- -- -- -- -- -- -- --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- 4d -- --
50: -- -- -- -- -- -- 56 57 -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

» In this example, the BMC has the address 0x4D, i.e. it is located on SMBus number 9.

» There might be several SMBuses in the list. If you do not find the BMC address on the first SMBus, list the devices on the other SMBuses. For SMBus 1 (i2c-1), for example, use this command:

```
$ sudo i2cdetect -y 1
```

» When you have found out the address of the BMC and the number of the SMBus you can dump the registers of the BMC using this command:

```
$ sudo i2cdump -y 9 0x4d
```

» Read out values using command *i2cget*:

```
$ sudo i2cget -y 9 0x4d
```

In the following you can find some examples showing how to use the watchdog:

- Setting the watchdog time to 10 seconds (0x64):

```
$ sudo i2cset -y 9 0x4d 0x14 0x64 w
```

- Enabling the watchdog (if it's not enabled inside the BIOS firmware):

```
$ sudo i2cset -y 9 0x4d 0x11 0x00
```

- Triggering the watchdog:.

```
$ sudo i2cset -y 9 0x4d 0x13 0x00
```

See Chapter 5.3 BMC API (Application Programming Interface) on page 55 for a detailed description of the BMC API.

3 Functional Description

The BSP for the operating system may not support all the functions of the G52A. See the MEN website for more information on hardware support on the respective BSP product page.

3.1 Power Supply

The G52A is supplied via the backplane.

3.2 Processor Core

The G52A is equipped with a twelve-core NXP QorIQ T4240 processor, which includes twelve 64-bit Power Architecture e6500 cores.

See the NXP website for more information on the QorIQ T4xxx processor family.

The following CPU types are available:

- NXP QorIQ T4240, twelve cores, 1.5 to 1.8 GHz
- NXP QorIQ T4160, eight cores, 1.5 to 1.8 GHz (on request)
- NXP QorIQ T4080, four cores, 1.5 to 1.667 GHz (on request)

3.2.1 Thermal Considerations

The power dissipation of G52A heavily depends on its processor and I/O configuration and on the workload.

Power dissipation of a G52A equipped with a high-end 45 W processor is up to 72 W (45 W for the high-end processor and 27 W for the rest of the board).

The G52A provides a very high computing power on a small space. For this reason, it is vital to provide sufficient airflow (2.5 m/s).

A suitable heat sink is provided to meet thermal requirements.

If you use any other heat sink than that supplied by MEN, or no heat sink at all, warranty on functionality and reliability of the G52A may cease. Please contact MEN if you have any questions or problems regarding thermal behavior.

3.3 Supervision and Management

The G52A provides an intelligent board management controller (BMC) with the following main features:

- System watchdog
- Operating hours counter
- Power cycle counter
- Voltage supervision
- Error state logging

3.3.1 Watchdog

The watchdog device monitors the CPU board on operating system level. If enabled, the watchdog must be triggered by application software. If the trigger is overdue, the watchdog initiates a board reset and in this way can put the system back into operation when the software hangs.

The watchdog unit can be enabled or disabled, as required and the watchdog timeout can be set in 100-ms steps from 100 ms up to 1:49:10 (hh:mm:ss) - 65536 steps.

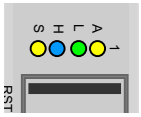
3.3.2 Temperature Measurement

The G52A uses a temperature device to measure the local CPU board temperature.

The G52A automatically shuts down when reaching the temperature 105°C.

3.3.3 Status LEDs

Table 1. General status LEDs

Appearance	Position	Label	Color	Function
	Front panel	*S*	Yellow	Board status LED
	Front panel	*H*	Blue	Hot-Swap LED
	Bottom side of PCB		Yellow	User LED
	Bottom side of PCB		Yellow	User LED

3.3.3.1 Board Status LED

The yellow status LED shows G52A status messages. Th e LED is controlled by a GPIO pin of the board controller. It is switched on when the G52A firmware starts, switched off when the G52A is switched off.

During normal operation the LED can be switched on and off using software.

In case of an error, the LED displays the following error messages by repeatedly flashing *n* times and then pausing for one second:

Table 2. Error codes signaled by board management controller via LED flashes

Number of Flashes	Error
1	+3.3 V failure
2	Input voltage failure
3	External power supply failure
4	CPU too hot
5	G52A firmware timeout
>5	Internal error

3.3.3.2 Hot-Swap LED

The hot-swap LED signals the hot-swap status.

Table 3. Hot-swap LED

Hot-Swap LED	Description
Off	The hot-swap switch is closed.
Starts flashing	The hot-swap switch is opened while the system is running (start of hot-swap sequence).
Lights continuously	The system has been shut down (end of hot-swap sequence).
	The hot-swap switch is open during power-up. The board controller delays the power-up sequence until the hot-swap switch is closed.
Stops flashing	The hot-swap switch has been closed while the hot-swap sequence is in progress. The board controller no longer waits for system shutdown.

If the hot-swap switch is closed after system shutdown, the board controller initiates Power Resume.

3.3.4 Software Support

Supervision and management functions are accessible by software.

- See Chapter 2.6 Using the G52A under Linux on page 19 for details on using MEN driver software.
- See the MEN website for all available software: www.men.de/products/g52a/#downl
- See Chapter 5.3 BMC API (Application Programming Interface) on page 55 for a detailed documentation of the BMC API.

3.4 Reset

The G52A is equipped with a reset button which is recessed within the front panel and requires a tool, e.g. paper clip to be pressed, preventing the button from being inadvertently activated.

3.5 Real-Time Clock (RTC)

The G52A includes a real-time clock connected to the processor as a system RTC. The real-time clock device is connected to the CPU via SMBus.

For data retention during power off the RTC is backed up by a supercapacitor. The supercapacitor gives an autonomy of up to 96 hours when fully charged.

For retention of time/date data after a power off of more than 96 hours the RTC can optionally be backed by a battery.

See the MEN website for ordering information.

Due to the RTC's reduced current consumption, the data retention time supported by the supercapacitor or battery can be increased considerably compared to the RTC integrated in the CPU.

Please note that the real-time clock integrated in the processor is **not used**. Configuring the date and time through the means provided by the operating system **does not set** the system RTC.

You can set the system date and time through the U-Boot firmware.

If you use dedicated MEN driver software supporting the system RTC, you can use the functions provided there to set the system date and time also via software.

See the MEN website for downloads and documentation.

3.6 Memory

3.6.1 DRAM System Memory

The DRAM system memory of G52A is scalable.

3.6.2 Boot Flash

The boot Flash memory contains the U-Boot firmware. It can also contain an operating system image and application software.

See Chapter 4.7.1 Boot Flash Memory Map on page 48.

3.7 Mass Storage

3.7.1 microSD Card

The G52A provides an onboard microSD card slot on the bottom side of the PCB. The slot supports the Secure Digital 2.0 specification (microSDHC) with a storage capacity of up to 32 GB.

The slot is ready-to-use.

See the MEN website for ordering information.

3.7.2 Serial ATA (SATA)

The G52A supports RAID operation.

3.7.2.1 Connection

- See the CompactPCI Serial standard PICMG CPCI-S.0 for the exact position of the SATA ports on the rear I/O connectors.
- See Chapter 3.11 CompactPCI Serial on page 34 for the position of the SATA interfaces in a CompactPCI Serial system.

3.7.2.2 Using the G52A in a CompactPCI Serial Peripheral Slot

If the G52A is used in a peripheral slot, the SATA interfaces on the CompactPCI Serial connectors are switched off.

3.8 USB

The G52A supports:

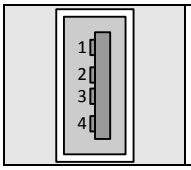
- EHCI implementation

3.8.1 Front Connection

Connector type:

- 4-pin USB Series A receptacle according to Universal Serial Bus Specification Revision 2.0
- Mating connector:
  4-pin USB Series A plug according to Universal Serial Bus Specification Revision 2.0

Table 4. Pin assignment - USB 2.0



1	+5V
2	USB_D-
3	USB_D+
4	GND

Table 5. Signal mnemonics - USB 1.0/2.0

Signal	Direction	Function
+5V	out	+5 V power supply
GND	-	Digital ground
USB_D+, USB_D-	in/out	USB lines, differential pair

3.8.2 Rear Connection

- Refer to the CompactPCI Serial standard PICMG CPCI-S.0 for the exact position of the USB ports on the rear I/O connectors.
- See Chapter 3.11 CompactPCI Serial on page 34 for the exact position of the USB ports in a CompactPCI Serial system.

3.8.2.1 Using the G52A in a Peripheral Slot

If the G52A is used in a peripheral slot, the USB interface on the CompactPCI Serial connectors is switched off.

3.8.3 USB-to-UART Interface

The USB signals of this interface are converted to UART signals using a bridge chip, e.g., when you connect a PC or other remote station using a USB port. The remote computer does not need a UART port.

In this way it can be used as a COM interface for terminal connection even without additional driver software.

See the MEN website for a service cable for USB:
www.men.de/products/g52a/#ord

3.9 Ethernet

See Figure 2, Board layout – top view on page 12 for the exact position of the Ethernet ports.

3.9.1 Front Connection

3.9.1.1 M12

Table 6. Connector types – Ethernet M12

Connector	Type
On G52A	8-pin M12 receptacle X-coded, e.g., Phoenix Contact SACC-CI-M12FSX-8CON-L90
Mating	8-pin M12 plug X-coded

Table 7. Pin assignment – Ethernet (8-pin M12)

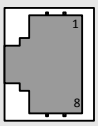
		10GBASE-T/ 1000BASE-T	10/100BASE-T
	1	BI_DA+	TX+
	2	BI_DA-	TX-
	3	BI_DB+	RX+
	4	BI_DB-	RX-
	5	BI_DD+	
	6	BI_DD-	
	7	BI_DC-	
	8	BI_DC+	

3.9.1.2 RJ45

Table 8. Connector types – Ethernet (RJ45)

Connector	Type
On G52A	Modular 8/8-pin receptacle according to FCC68
Mating	Modular 8/8-pin plug according to FCC68

Table 9. Pin assignment – Ethernet (RJ45)

		1000BASE-T/ 10GBASE-T	10/100BASE-T
	1	BI_DA+	TX+
	2	BI_DA-	TX-
	3	BI_DB+	RX+
	4	BI_DC+	-
	5	BI_DC-	-
	6	BI_DB-	RX-
	7	BI_DD+	-
	8	BI_DD-	-

3.9.2 Rear Connection

- Refer to the CompactPCI Serial standard PICMG CPCI-S.0 for the exact position of the Ethernet ports on the rear I/O connectors.
- See Chapter 3.11 CompactPCI Serial on page 34 for the exact position of the Ethernet ports in a CompactPCI Serial system.

3.9.3 Signal Mnemonics

Table 10. Signal mnemonics – Ethernet

Signal	Direction	Function
BI_Dx+/-	in/out	Differential pairs of data lines for 1000BASE-T or 10GBASE-T
RX+/-	in	Differential pair of receive data lines for 10/100BASE-T
TX+/-	out	Differential pair of transmit data lines for 10/100BASE-T

3.9.4 Ethernet MAC Addresses

The unique MAC address is set at the factory and should not be changed. Any attempt to change this address may create node or bus contention and thereby render the board inoperable.

The naming of the interfaces may differ depending on the operating system. The MAC addresses on G52A are:

Table 11. Ethernet MAC addresses

Interface	Position	Base Address
X1	Upper front	`0x 74 8F 4D 11 00 00`
X2	Centre front	`0x 74 8F 4D 11 80 00`
X3	Lower front	`0x 74 8F 4D 11 A0 00`
1_ETH	CompactPCI Serial connector P6	`0x 74 8F 4D 11 20 00`
2_ETH	CompactPCI Serial connector P6	`0x 74 8F 4D 11 40 00`
3_ETH	CompactPCI Serial connector P6	`0x 74 8F 4D 11 60 00`

"74 8F 4D" is the MEN vendor code. The last six digits describe the range from which the addresses for the board are taken. The serial number is added by the last three digits in the range:

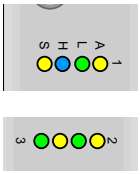
Serial number 42 (X1): `0x0000 + 0x002A = 0x002A.`

See Chapter 1.6 Product Identification on page 16.

3.9.5 Ethernet Status LEDs

The G52A provides a total of six Ethernet status LEDs at the front panel, two for each Ethernet channel. They signal the link and activity status.

Table 12. Ethernet status LEDs at front panel

Appearance	Label	Color	Function
	*[1:3]/A*	Yellow	Activity LED ▪ On: Tx/Rx activity ▪ Off: No activity ▪ Blinking: Tx/Rx activity
	[1:3]/L	Green	Linkup LED ▪ On: Link up ▪ Off: No link ▪ Blinking: n/a

3.10 PCI Express

- Refer to the CompactPCI Serial standard PICMG CPCI-S.0 for the exact position of the PCI Express ports on the rear I/O connectors.
- See Chapter 3.11 CompactPCI Serial on page 34 for the exact position of the PCI Express ports in a CompactPCI Serial system.

3.11 CompactPCI Serial

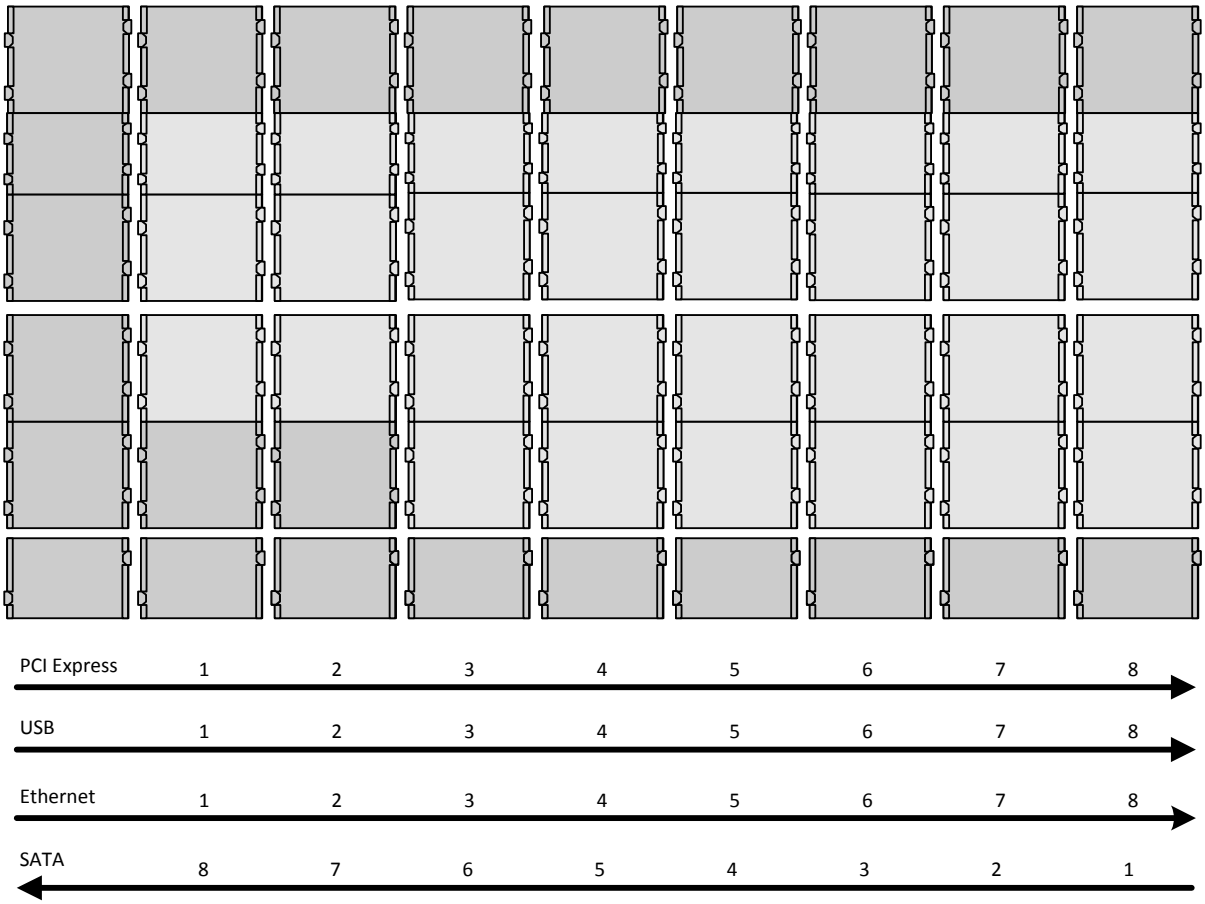
Refer to the CompactPCI Serial standard PICMG CPCI-S.0 for detailed information regarding the rear I/O connectors.

- CompactPCI Serial Specification PICMG CPCI-S.0 Revision 2.0: 2015; PCI Industrial Computers Manufacturers Group (PICMG) www.picmg.org
- Introduction to CompactPCI Serial on Wikipedia: en.wikipedia.org/wiki/CompactPCI_Serial

3.11.1 Backplane Filling Order

The CompactPCI Serial standard supports a maximum of 2 PCI Express x8 links (fat pipe), 6 PCI Express x4, 8 SATA, 8 USB and 8 Ethernet interfaces.

Table 13. CompactPCI Serial backplane filling order

System Slot 1	Fat Pipe Periph. Slot 2	Fat Pipe Periph. Slot 3	Periph. Slot 4	Periph. Slot 5	Periph. Slot 6	Periph. Slot 7	Periph. Slot 8	Periph. Slot 9
								
Implementation on the G52A								
	PCI Express 3.x x4	PCI Express 2.x x4	PCI Express 2.x x2	PCI Express 2.x x4				
	USB 2.0							
	1GBASE-T Ethernet	1GBASE-T Ethernet	1GBASE-T Ethernet					
							SATA	SATA

3.11.2 Using the G52A as a Peripheral Board

It is possible to use more than one G52A board within a CPCI-S.0 system to build a redundant system or a cluster with more processing power. The communication between the boards is done via Ethernet in this case and the other high-speed interfaces cannot be used. The G52A cannot be booted via SATA in such a configuration.

In peripheral mode the following interfaces available on the CPCI-S.0 connector are disabled by the board firmware automatically:

- USB
- SATA
- All control signals which are only available for system boards

The board firmware detects if the board is inserted in a peripheral slot by monitoring the *SYS_EN#* pin on CompactPCI Serial connector P1.

4 U-Boot Firmware

4.1 General

U-Boot is the G52A firmware that is invoked when the system is powered on.

The basic tasks of U-Boot are:

- Initialize the CPU and its peripherals
- PCI configuration
- Provide debug/diagnostic features on the U-Boot command line
- Boot the operating system

- See the MEN website for the current U-Boot (patch file and complete binaries, i.e. prebuilt main U-Boot image): www.men.de/products/g52a/#downl
- U-Boot is open source software. The source code that the G52A U-Boot is based on is available on request. Please contact MEN.

The following description only includes product-specific features. See the DENX U-Boot and Linux Guide (DULG) available under www.denx.de/wiki/DULG/WebHome for a general description and in-depth details on U-Boot. (See Chapter 2.3 Availability for a PDF version.)

4.2 Getting Started: Setting Up Your Operating System

When U-Boot starts up for the first time, it does not know yet which operating system (OS) to load and normally stops the boot procedure by its prompt. If you don't see the U-Boot prompt, reset the G52A again and press any key during start-up.

You need to make the necessary settings first and then load a boot image, e.g., via network. The exact steps and settings depend on the operating system.

- See the documentation of the respective MEN board support package (BSP) for G52A for OS-specific details and examples: www.men.de/products/g52a/#doc
- See Chapter 4.3.4.1 Boot Methods on page 42 for supported boot methods.

4.3 Interacting with U-Boot

U-Boot uses a shell similar to the Linux Hush shell with a command history and autocompletion support.

4.3.1 Setting Up a Console Connection

To interact with U-Boot, you can use the USB-to-UART interface as a serial console port.

You can select the active console by means of environment variables *stdin*, *stdout* and *stderr*.

U-Boot command *coninfo* lists all active consoles.

The default setting of the COM ports is 115 200 baud, 8 data bits, no parity, and one stop bit. You can set the baud rate through environment variable *baudrate*.

See Table 17, U-Boot – Environment variables – Console on page 50 for a list of all console variables.

4.3.2 Entering the U-Boot Command Line

During normal boot, you can abort the booting process by pressing any key during start-up.

You can use U-Boot environment variable *bootdelay* to configure the autoboot behavior.

If an autoboot time is set, U-Boot waits for this amount of seconds (measured from its beginning) before it starts the operating system, to give the user a chance to abort booting and enter the command line.

See Table 15, U-Boot – Environment variables – OS boot on page 48.

4.3.3 User Interface Basics

4.3.3.1 Help and Navigation

Use the *help* command to get a list of available commands.

Arrow keys "up" ↑and "down" ↓ let you navigate in the command line history.

The <*TAB*> key autocompletes commands and variables.

You can press <*CTRL*> <*c*> to abort.

4.3.3.2 Configuring Your System

Use environment variables to configure your system. They can be viewed using the *printenv* command. To set or add variables, you can use commands *editenv* and *setenv*. To save the changed parameters use *saveenv*.

Command *defenv* resets the environment variables to default values.

To display an environment variable, you can do the following:

```
=> printenv baudrate
baudrate=115200
```

To print all variables:

```
=> printenv
baudrate=115200
bootdelay=-1
...
```

Shell variable expansion:

```
=> echo "Server IP: ${serverip}"
Server IP: 192.1.1.22
```

To edit, modify the variable in the edit line and press <Enter>:

```
=> editenv ipaddr
edit: 192.1.1.023
```

To edit a variable directly:

```
=> setenv ipaddr 192.1.1.123
```

To delete a variable completely:

```
=> setenv ipaddr
```

Set all variables to default:

```
=> defenv
```

You need to save any changes made, otherwise they will be lost after the next reset:

```
=> saveenv
Saving Environment to Flash...
```

See Chapter 4.7.2 Environment Variables on page 48 for a list of the G52A environment variables.

4.3.3.3 Working with Scripts and Applications

You can use scripts or stand-alone applications for more complex tasks. Scripts can be stored in environment variables and executed by the *run* command.

You can enter a sequence of commands using different separators:

- ; separated = all commands are executed
- && separated = next command is executed only if no error occurred
- || separated = next command is executed only if an error occurred

Simple Scripts using the Command Line

You can create a script using the command line, and you can store it in an environment variable:

» Create script (i.e. store list of commands in a variable), for example:

```
=> setenv menu_script 'echo "1=Boot Linux"; echo "2=Boot Alternative
OS"; echo "3=Memory Test"; askenv _number; if test ${_number} = 1;
then run script_linux; elif test ${_number} = 2; then run
script_altos; else mtest; fi'
```

See www.denx.de/wiki/view/DULG/CommandLineParsing.

» Save the script in an environment variable (optional):

```
=> saveenv
```

» Execute the script:

```
=> run menu_script
```

Scripts using Source Files

For more complex scripts, you can write a text file on your host computer, convert it, load it into the G52A Flash and run it on the G52A from the source file. The following shows how an example script is created on a Linux host computer:

» Write the script as a TXT file. In the example, we have written a file called *brd_info.txt*:

```
# example U-Boot script (show board info)
#
# convert:
# mkimage -A ppc -O linux -T script -C none -a 0 -e 0 -n "board info
script" -d ./brd_info.txt ./brd_info.scr

echo
echo Version:
echo -------- \\c      # \\c = no new line
version
echo
echo Board:
echo ------ \\c
eeprod
echo
clocks
echo
echo Network:
echo --------
echo Interface: ${ethact}
echo Target: ${ipaddr} (${ethaddr})
echo Server: ${serverip}
echo
echo bdinfo:
echo --------
bdinfo
echo
```

» Convert the TXT script to *.scr* format, e.g., under Linux:

```
$ mkimage -A ppc -O linux -T script -C none -a 0 -e 0 -n "board info
script" -d /examples/brd_info.txt /tftpboot/brd_info.scr
```

» Download the script via network using the U-Boot command line:

```
=> tftpboot 192.1.1.22:/tftpboot/brd_info.scr
```

» Execute the script:

```
=> source ${loadaddr}
```

4.3.4 Booting an Operating System

The G52A U-Boot supports the standard commands for booting the supported operating systems.

You can completely configure how U-Boot boots the operating system through environment variables. Variables *bootargs* and *bootcmd* include the arguments to be set and commands to be executed at boot-up to start the operating system.

- See the documentation of the respective MEN board support package (BSP) for G52A for OS-specific details and examples: www.men.de/products/g52a/#doc
- See Chapter 4.2 Getting Started: Setting Up Your Operating System on page 37.

Please remember to save the settings you have made in the environment variables using *saveenv*.

4.3.4.1 Boot Methods

The following boot methods are supported and recommended for G52A.

OS Boot via Network

U-Boot command *tftpboot* allows loading of the operating system via Ethernet using the TFTP protocol. The following interfaces are supported:

- X1 front interface and all three rear interfaces

OS Boot via Mass Storage Devices

U-Boot allows loading of the operating system via the following mass storage devices:

- SATA
- microSD
- USB

The following U-Boot commands can be used for booting:

- *ext2load*
- *fatload*

The command to be used depends on the file system of the device.

OS Boot via Boot Flash

U-Boot allows loading of an operating system binary stored in the user space of the onboard boot Flash.

4.4 U-Boot Images

4.4.1 Images

U-Boot has one image for normal operation called "Image 1". There is a dedicated Image 1 for CompactPCI Serial system slots and peripheral slots.

See below, Chapter 4.4.1.1 Boot Flash Memory Devices for CompactPCI Serial and Peripheral Slots on page 43.

U-Boot also has a fallback image. However, this is only intended for factory usage and is not automatically executed. If Image 1 fails, you can make use of the G52A's Flash memory set-up to repair the image.

See below, Repairing Image 1 in Case of a Boot Failure.

4.4.1.1 Boot Flash Memory Devices for CompactPCI Serial and Peripheral Slots

G52A has two separate but identically mapped Flash memory devices. One device contains the U-Boot images for booting the G52A in a CompactPCI Serial system slot, and the other is used when the board is located in a CompactPCI Serial peripheral slot.

By default, the G52A automatically uses the Flash memory device matching the slot type that the board is inserted in, i.e. system or peripheral slot.

One memory device is always selected, and any accesses to U-Boot settings or updates will be active on the selected memory device. U-Boot command *men_sw_spi* allows to switch between the two memory devices.

Repairing Image 1 in Case of a Boot Failure

If the G52A does not start up U-Boot Image 1, you should do the following to repair Image 1:

- Power down your system.
- Remove the G52A from the CompactPCI Serial slot it was inserted in.
- Insert the G52A into a different slot:
  - If the former slot was the system slot, select a peripheral slot.
  - If the former slot was a peripheral slot, select the system slot.
- Power up the system.

The G52A now boots from the respective other memory device.

- Enter the U-Boot command line.
- Use command *men_sw_spi* to change memory access back to the corrupted memory device:

```
=> men_sw_spi 0     to switch to system slot boot Flash device
or
=> men_sw_spi 1     to switch to peripheral slot boot Flash device
```

- Update your corrupted Image 1 by a working image file:

See Chapter 4.4.2 Updating U-Boot Image 1 on page 44.

4.4.2 Updating U-Boot Image 1

Updating the boot Flash may damage the G52A!
Read the following instructions carefully.
Please be aware that you do boot Flash updates at your own risk.

You can update U-Boot Image 1 via network, mass storage or serial console. Do the following to update U-Boot:

» Download the current U-Boot Image 1 update from the MEN website.

www.men.de/products/g52a/#downl

» Unzip the downloaded file, e.g., *14g052a00.zip*, into a temporary directory on your host system or on a mass-storage device.

» If you use a mass-storage device, change U-Boot environment variable *img_name* to the binary file name, e.g.:

```
For the system slot image:
=> setenv img_name u-boot_std_sys_14g052a00.bin
For the peripheral slot image:
=> setenv img_name u-boot_std_per_14g052a00.bin
```

» If you use a host system, change U-Boot environment variable *img_path_name* to the path and binary file name, e.g.:

```
For the system slot image:
=> setenv img_path_name /tftpboot/temp/u-boot_std_sys_14g052a00.bin
For the peripheral slot image:
=> setenv img_path_name /tftpboot/temp/u-boot_std_per_14g052a00.bin
```

» If you use a serial console or network connection, connect your host computer to the G52A.

» If you use a USB or other mass storage device, connect the device to the G52A.

» Power on the G52A and enter the U-Boot command line.

» Run the update script that applies in your case:

```
Via network:
=> run update_tftp

Via USB:
=> run update_usb

Via serial console:
=> run update_uart
```

» If you need an update script for a mass-storage device other than USB, take the *update_usb* script as a template and create a new, specific environment variable. Make sure to leave the Flash address as specified.

See Table 18, U-Boot – Environment variables – Other on page 50.

» When the update procedure has completed, reset the G52A.

4.5 Updating User Data in the Boot Flash

You can write binary files to the user space in the boot Flash via network, a mass storage device, or a serial console connection. U-Boot provides commands specific for each medium to load a binary update file, and the following general commands to program the boot Flash:

- *sf*

Updating the boot Flash may damage the G52A!
Read the following instructions carefully.
Please be aware that you do boot Flash updates at your own risk.

See Chapter 4.7.1 Boot Flash Memory Map on page 48 for the exact addresses and size of the user space.

4.5.1 Update via Network

You can use U-Boot command *tftpboot* to download the binary update file from a TFTP server in the network.

4.5.2 Update via Mass Storage Devices

You can also make a Flash update from a mass storage device, e.g., a USB Flash drive or plugged microSD card. The following U-Boot commands are supported to load the binary update file:

- *ext2load*
- *fatload*

4.5.3 Update via the Serial Console

U-Boot provides the *loady* tool to download a binary update file.

The terminal emulation program must be configured to start the upload via the "Ymodem" and send the required file.

Set the terminal emulation program to the following protocol:

- 115 200 baud
- 8 data bits
- 1 stop bit
- No parity
- No handshaking

4.5.4 Performing an Update

To perform an update, e.g., of your operating system image inside the Flash, use the following procedure.

See Chapter 4.4.2 Updating U-Boot Image 1 on page 44 for instructions on how to update the U-Boot code itself.

» Make sure the size of your binary file is within the boundaries of the user address space and will be written to the correct address.

See Chapter 4.7.1 Boot Flash Memory Map on page 48.

» Download the update file to the G52A:

```
Via network, e.g.:
=> tftpboot ${loadaddr} ${serverip}:/path/file.bin

Via mass storage, e.g., USB:
=> usb start; fatload usb 0:1 ${loadaddr} /file.bin
                              0:1 = device 0 partition 1

Via serial console, e.g., with Y protocol:
=> loady
```

» Erase the part of the Flash that you want to update:

```
=> sf probe; sf erase <write address> +${filesize}
                        <write address> is the update address, e.g., 0xffffffff
```

» Write the file to Flash:

```
=> sf write ${loadaddr} <write address> ${filesize}
```

4.6 Diagnostic Tests

4.6.1 Power-On Self Tests

The G52A U-Boot includes a number of power-on self tests (POST). If a test fails, U-Boot stops booting and enters the command line.

You can use the *failbootcmd* environment variable to execute special commands in case of an error. The settings of *failbootcmd* will be executed also if the fallback U-Boot image is forced to load.

The following tests are executed at power-on:

- **Main memory test:** Tests the address and data bus of the main memory.[1]
- **MAC address test:** Verifies if Ethernet MAC addresses are available in EEPROM.

The **main memory test** may take up to 20 seconds. Therefore, the test runs only once when the G52A is powered up for the very first time during production. If the test was executed successfully, environment variable *memtest_state* is set to *done*.

To force execution of a new main memory test, change environment variable *memtest_state* to *run*, save the environment variables using *saveenv* and restart the G52A.

[1] Only the first 2 GB of memory are tested.

4.7 U-Boot Implementation on G52A

4.7.1 Boot Flash Memory Map

Note: The G52A has two boot Flash memory devices. The memory map below is identical for both devices.

See Chapter 4.4.1.1 Boot Flash Memory Devices for CompactPCI Serial and Peripheral Slots on page 43.

Table 14. U-Boot – Boot Flash memory map

Address Range	Size	Description
0x 0000 0000 – 0x 0003 FFFF	256 KB	RCW, PBL, SPL U-Boot
0x 0004 0000 – 0x 000F FFFF	768 KB	U-Boot binary, Image 1
0x 0010 0000 – 0x 0010 1FFF	8 KB	U-Boot environment, Image 1
0x 0010 2000 – 0x 0010 FFFF	56 KB	Reserved
0x 0011 0000 – 0x 0012 FFFF	128 KB	FMAN (FrameManager) firmware for networking functions
0x 0013 0000 – 0x 0018 FFFF	768 KB	U-Boot binary, fallback image; only for factory usage
0x 001F 0000 – 0x 001F 1FFF	8 KB	U-Boot environment, fallback image; only for factory usage
0x 001F 2000 – 0x 001F FFFF	56 KB	Reserved
0x 0020 0000 – 0x 0027 FFFF	512 KB	User Area 1
0x 0028 0000 – 0x 004F FFFF	2.5 MB	User Area 2
0x 0050 0000 – 0x 01FF FFFF	27 MB	User Area 3

4.7.2 Environment Variables

U-Boot uses environment variables stored in Flash to configure the target. The available variables are specific for the G52A.

See Chapter 4.3.3.2 Configuring Your System on page 39 for editing commands and examples.

Table 15. U-Boot – Environment variables – OS boot

Variable	Description	Default	Access
bootargs	Boot arguments when booting an OS image	The boot command will set *bootargs*.	r/w
bootcmd	Command string that is automatically executed after reset	echo "no boot command defined"	r/w
bootdelay	Delay before the default image is automatically booted, in seconds. Set to -1 to disable autoboot	-1	r/w
bootfile	Name of the image to load through command *tftpboot*	Empty	r/w

Variable	Description	Default	Access
failbootcmd	Code string to be executed in case of a boot failure, e.g., during power-on self tests.	*echo "POSTFAIL detected, no OS boot"*	r/w
linux_path[1]	Linux boot path and file used by default *bootcmd*	*/tftpboot/G052A/os_linux*	r/w
linux_setargs[1]	Used to set *bootargs* Default: *setenv bootargs root=/dev/ram rw console=ttyS0,${baudrate}*		r/w
linux_mmc[1]	Example script for booting Linux via microSD	See U-Boot, *print linux_mmc*	r/w
linux_tftp[1]	Example script for booting Linux via network	See U-Boot, *print linux_tftp*	r/w
linux_usb[1]	Example script for booting Linux via USB	See U-Boot, *print linux_usb*	r/w
os[1]	Operating system to boot (e.g., *linux*, *vxworks*)	*--- do not boot ---*	r/w

Table 16. U-Boot – Environment variables – Network

Variable	Description	Default	Access
ethact	Controls which network interface is currently active; automatically set to current network interface Possible values: ▪ *dTSEC4* (X1) ▪ *dTSEC1* (rear, 1_ETH) ▪ *dTSEC2* (rear, 2_ETH) ▪ *dTSEC3* (rear, 3_ETH) ▪ not defined (empty)	Empty	r/w
ethaddr *eth1addr* *eth2addr* *eth3addr* *eth4addr* *eth5addr*	MAC address of Ethernet interface (see Chapter 3.9.4 Ethernet MAC Addresses on page 32). You can pass a MAC address to the OS using the *cpenv* command.	*74:8f:4d:11:x0:00*[1]	r
ethprime	Primary Ethernet controller	*dTSEC4* (X1)	r/w
gatewayip	IP address of the gateway (router) to use	*192.1.1.22*	r/w
hostname	Target host name	Empty	r/w
ipaddr	IP address; needed for *tftpboot* command	*192.1.1.254*	r/w
loadaddr	Default load address for U-Boot commands	*0x01000000*	r/w
netmask	Subnet mask	*255.255.255.0*	r/w
serverip	TFTP server IP address; needed for *tftpboot* command	*192.1.1.22*	r/w
user	TFTP user name	Empty	r/w

[1] *x* stands for the base address of the respective channel.

Table 17. U-Boot – Environment variables – Console

Variable	Description	Default	Access
baudrate	Baud rate for serial console Possible values: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	*115200*	r/w
stderr	Standard error console Possible values: *serial*	*serial*	r/w
stdin	Standard input console Possible values: *serial*	*serial*	r/w
stdout	Standard output console Possible values: *serial*	*serial*	r/w

Table 18. U-Boot – Environment variables – Other

Variable	Description	Default	Access
cust_env	For factory use only	0	r
dbg_env	For factory use only	0	r
hwconfig	Describes the used hardware configuration	*hwconfig=fsl_ddr:ctlr_intlv=3way_4KB,bank_intlv=null;usb1:dr_mode=host,phy_type=utmi*	r
img_path_name	Image location and name of U-Boot Image 1 update file. Used in command *run update_tftp*.	*/tftpboot/G052A/u_boot/u-boot_std_sys_14G052A00.bin*	r/w
img_path_name_fman	Name of T4240 *fman* firmware update file. Used in command *run update_usb_fman*.	*fsl_fman_ucode_t4240_r2.0_106_4_15.bin*	r/w
img_name	Name of U-Boot update file. Used in command *run update_usb*.	*u-boot_std_sys_14G052A00.bin*	r/w
memorytest_state	State of start-up main memory test ▪ *run*: force memory test at next start-up ▪ *done*: memory test successfully executed; test will not run at next start-up	*run*	r/w
mp_holdoff	Setting for multi-processor usage	*n*	r/w
pre_delay	Delay time before release of external reset (n*100 ms; e.g., `0x05` = 500 ms)	`0x05`	r/w
post_delay	Delay time after release of external reset (n*100 ms; e.g., `0x0F` = 1500 ms)	`0x0F`	r/w
update_tftp	Updates U-Boot Image 1 via network. See Chapter 4.4.2 Updating U-Boot Image 1 (page 44).	See U-Boot, *print update_tftp*	r/w
update_uart	Updates U-Boot Image 1 via serial console. See Chapter 4.4.2 Updating U-Boot Image 1 (page 44).	See U-Boot, *print update_uart*	r/w

Variable	Description	Default	Access
update_usb	Updates U-Boot Image 1 via USB. See Chapter 4.4.2 Updating U-Boot Image 1 (page 44).	See U-Boot, *print update_usb*	r/w
update_usb_fman	Updates the T4240 *fman* firmware image via USB. **This function is intended for updates with files provided directly from the processor manufacturer.** It works similarly to the other update functions, e.g., *update_usb*. See Chapter 4.4.2 Updating U-Boot Image 1 (page 44).	See U-Boot, *print update_usb_fman*	r/w

4.7.3 U-Boot Commands

You can access the full U-Boot command list using the *help* command (or the *?* alias). More detailed information is displayed if you enter *help* <*command*>. U-Boot supports auto completion using the <*TAB*> key.

The following table provides only the G52A-specific commands.

Table 19. U-Boot – G52A-specific commands

Command	Description
bootu	Boot special application image from memory (customer defined, not for general usage)
cpenv	Copy environment string to address (default is `0x5000`), may be needed to forward some additional parameters to an OS without flat device tree support
defenv	Set environment variables to default values (use *saveenv* to store the changes)
eeprod	Display MEN production data
macerase	Reserved, for internal factory usage only
macread	Read all MAC addresses from EEPROM and display them
macwrite	Reserved, for internal factory usage only
men_info	Print register information (for debugging)
men_sw_spi	Switch boot Flash: select system or peripheral serial boot Flash See Chapter 4.4.1.1 Boot Flash Memory Devices for CompactPCI Serial and Peripheral Slots on page 43.
uptime	Get time since time base reset, in milliseconds

4.7.4 Hardware Interfaces Not Supported by U-Boot

The standard G52A U-Boot does **not** support the following hardware interfaces:

- 10GBASE-T front interfaces (X2 and X3)

5 Hardware/Software Interface

This chapter is intended for software developers or board integrators who need deeper knowledge of the implementation details of the G52A interfaces and its internal connections.

5.1 PCI Express Root Port Interrupt Mapping

Table 20. PCI Express Root Port Interrupt Mapping for Downstream Devices

PCI Express Controller	CompactPCI Slot	INTx	IRQn
PCI Express 1	CPCI slot 2 (x4)	INTA#	-
		INTB#	IRQ1
		INTC#	IRQ2
		INTD#	IRQ3
PCI Express 2	CPCI slot 1 (x4)	INTA#	-
		INTB#	IRQ5
		INTC#	IRQ6
		INTD#	IRQ7
PCI Express 3	CPCI slot 4 (x4)	INTA#	-
		INTB#	IRQ9
		INTC#	IRQ10
		INTD#	IRQ11
PCI Express 4	CPCI slot 3 (x2)	INTA#	-
		INTB#	IRQ0
		INTC#	IRQ4
		INTD#	IRQ8

5.2 I2C Devices

Table 21. I2C devices

Bus	8-Bit Address	7-Bit Address	Function	MDIS Device Name
I2C1	0xA8	0x54	EEPROM	*smb2_1ppc*
	0x68	0x34	EEPROM: write protect	
I2C3	0xA2	0x51	Customer EEPROM	
I2C1	0x90	0x48	Temperature sensor for CPU diode with integrated temperature sensor for board temperature	*smb2_1ppc*
	0x64	0x32	System RTC (see also Chapter 3.5 Real-Time Clock (RTC) on page 27)	
	0x9A	0x4D	Board Management Controller	
	0x80	0x40	VID Power Supply	
	0xD0	0x68	Clock Generator	

Note on 8-Bit/7-Bit Addressing

- **8-bit addressing** is compliant to the Windows nomenclature. The last bit, which is used as the read/write bit, is added to the address (0 = write, 1 = read).
  If you use MDIS driver software, use 8-bit addresses, with any OS.
- **7-bit addressing** is used, e.g., under Linux. A '0' is added at the beginning of the address so that all consecutive address bits are moved one bit to the right.
  If you use standard I2C commands under Linux, use 7-bit addresses.

5.3 BMC API (Application Programming Interface)

The G52A uses a generic command interface for communication between the CPU (host) and the BMC. Application software uses command packets to communicate with the BMC.

The application software controls the BMC via I2C/SMBus. The device address is `0x4D` (in 7-bit, non-shifted notation) or `0x9A`/`0x9B` (in 8-bit, shifted notation, write/read).

5.3.1 BMC Command Packets

5.3.1.1 Command Packet Protocol

From a logical point of view, the command protocol has the following characteristics:

- Commands are always initiated by the host. The BMC never sends packets without the host requesting it to do so.
- Packets are either
  - unidirectional from host to BMC, without an answer from the BMC
  - bidirectional, with an answer from the BMC
- Each command has a unique identifier, consisting of the command opcode and a packet type:

Table 22. BMC API – Packet types

Packet Type	Description	Request Data Host > BMC	Response Data BMC > Host	Error Signaling
PT_SB	Send command only	None	No response	-
PT_RBD	Send command, get one data byte from BMC	None	1 byte	Response byte = `0xFF`
PT_WBD	Send command, send one data byte to BMC	1 byte	No response	-
PT_RWD	Send command, get two data bytes from BMC	None	2 bytes	Response byte = `0xFFFF`
PT_WWD	Send command, send two data bytes to BMC	2 bytes	No response	-

The packet types are directly mapped to the corresponding SMBus “bus protocols” as defined in the System Management Bus Specification.

Table 23. BMC API – Packet types mapping on SMBus

Packet Type	SMBus Protocol
PT_SB	Send byte
PT_RBD	Read byte
PT_WBD	Write byte
PT_RWD	Read word
PT_WWD	Write word

5.3.1.2 Watchdog Control Commands

Table 24. BMC API – Watchdog commands

Command	Packet Type	Opcode	Functional Description
WDOG_ON	*PT_SB*	0x11	Enable watchdog
WDOG_OFF	*PT_WBD*	0x12	Disable watchdog
WDOG_TRIG	*PT_SB*	0x13	Trigger watchdog
WDOG_TIME_SET	*PT_WWD*	0x14	Set watchdog timeout value
WDOG_TIME_GET	*PT_RWD*	0x14	Get watchdog timeout value
WDOG_STATE_GET	*PT_RBD*	0x17	Get watchdog state
WDOG_ARM	*PT_SB*	0x18	Arm watchdog and BIOS timeouts
ARM_STATE	*PT_RBD*	0x19	Get watchdog arming state

Command WDOG_ON

Opcode: 0x11	**Packet Type:** ***PT_SB***

Command WDOG_OFF

Opcode: 0x12								**Packet Type:** ***PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	0x69							

Command WDOG_TRIG

Opcode: 0x13	**Packet Type:** ***PT_SB***

Commands WDOG_TIME_SET and WDOG_TIME_GET

Command WDOG_TIME_SET

Opcode: 0x14								**Packet Type: *PT_WWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	*WD_TOUT* (LSB)							
Data 1	*WD_TOUT* (MSB)							

Command WDOG_TIME_GET

Opcode: 0x14								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	*WD_TOUT* (LSB)							
Data 1	*WD_TOUT* (MSB)							

Bit Field	**Description**
WD_TOUT	Trigger timeout, in steps of 100 ms ▪ 0x0001: 100 ms ▪ 0x0002: 200 ms ▪ ... ▪ 0xFFFF: Error

Command WDOG_STATE_GET

Opcode: 0x17								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*WD_STATE*							

Bit Field	**Description**
WD_STATE	Watchdog state ▪ 0x00: Off ▪ 0x01: On ▪ 0xFF: Error

Command WDOG_ARM

Opcode: 0x18	Packet Type: *PT_SB*

Command WDOG_ARM_STATE

Opcode: 0x19								Packet Type: *PT_RBD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*ARM_STATE*							

Bit Field	Description
ARM_STATE	Watchdog arming state ▪ 0x00: Not armed ▪ 0x01: Armed ▪ 0xFF: Error

5.3.1.3 Power Resume Mode Commands

These commands allow configuring the behavior of the G52A in case the power is reapplied after a power failure and input voltages return to their allowed limits.

This setting is only used when the G52A is in Master mode. When the G52A is in Slave mode, the BMC always starts the power-up sequence.

See Chapter 5.3.1.12 Board Controller Mode on page 70.

The setting is persistent, i.e. it is stored in non-volatile memory.

The default resume mode after factory programming is "On".

Table 25. BMC API – Power resume mode commands

Command	Packet Type	Opcode	Functional Description
RESUME_MODE_SET	*PT_WBD*	0x20	Set power resume mode
RESUME_MODE_GET	*PT_RBD*	0x20	Get power resume mode

Table 26. BMC API – Power resume modes

Resume Mode	System State at Power Loss	Resume Action
On	On	Start power-up sequence
	Off	Start power-up sequence
Off	On	Stay in S4/S5 state
	Off	Stay in S4/S5 state
Former	On	Start power-up sequence
	Off	Stay in S4/S5 state

S0 to S5 are the power states as defined in the ACPI specification, or an equivalent state

Please refer to the ACPI Specification for more details on the power states S0 to S5:
Advanced Configuration and Power Interface Specification Version 6.1
January, 2016
Unified EFI Forum
uefi.org/specifications

Commands RESUME_MODE_SET and RESUME_MODE_GET

Command RESUME_MODE_SET

Opcode: 0x20								**Packet Type: *PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*RES_MODE*							

Command RESUME_MODE_GET

Opcode: 0x20								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*RES_MODE*							

Bit Field	Description
RES_MODE	Resume mode ▪ 0x00: Off ▪ 0x01: On ▪ 0x02: Former

5.3.1.4 External Power Supply Failure Mode

These commands allow configuring the behavior of the G52A upon assertion of an external power supply fail signal.

Modes:

- Ignore: Assertion of external power failure signal is completely ignored.
- Treat as error: Assertion of external power failure is treated as an error; i.e. event is counted as an error and G52A is reset.

The setting is persistent, i.e. it is stored in non-volatile memory.

The default external power supply fail signal mode after factory programming is "Ignore".

Table 27. BMC API – External power supply failure mode commands

Command	Packet Type	Opcode	Functional Description
EXT_PWR_FAIL_MODE_SET	*PT_WBD*	`0x21`	Set external power supply failure mode
EXT_PWR_FAIL_MODE_GET	*PT_RBD*	`0x21`	Get external power supply failure mode

Commands EXT_PWR_FAIL_MODE_SET and EXT_PWR_FAIL_MODE_GET

Command EXT_PWR_FAIL_MODE_SET

Opcode: `0x21`								**Packet Type:** ***PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*EXT_PWR_FAIL_MODE*							

Command EXT_PWR_FAIL_MODE_GET

Opcode: `0x21`								**Packet Type:** ***PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*EXT_PWR_FAIL_MODE*							

Bit Field	Description
EXT_PWR_FAIL_MODE	External power supply failure mode ▪ `0x00`: Ignore ▪ `0x01`: Treat as error ▪ `0xFF`: Error

5.3.1.5 Reset Signal Blocking

These commands allow blocking of G52A reset inputs. The setting is persistent, i.e. it is stored in non-volatile memory.

In a system with master and slave CPU boards, normally the slave boards will get a reset whenever the master board resets. With "Reset Signal Blocking" configuration it is possible to decide at runtime for the slave boards whether they should get a reset whenever the master board resets or whether the slave board should operate independently. Additionally with this functionality it is possible to disable external reset for the master board where needed.

The default mode after factory programming is "Reset enabled".

Table 28. BMC API – Reset signal blocking commands

Command	Packet Type	Opcode	Functional Description
RESET_IN_MODE_SET	*PT_WBD*	0x22	Set reset input mode
RESET_IN_MODE_GET	*PT_RBD*	0x22	Get reset input mode

Commands RESET_IN_MODE_SET and RESET_IN_MODE_GET

Command RESET_IN_MODE_SET

Opcode: 0x22								**Packet Type: *PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*RESET_IN_MODE*							

Command RESET_IN_MODE_GET

Opcode: 0x22								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*RESET_IN_MODE*							

Bit Field	Description
RESET_IN_MODE	Reset input mode ▪ 0x00: Reset enabled ▪ 0x01: Reset masked ▪ 0xFF: Error

5.3.1.6 External Power Supply Control

In Master mode, the BMC uses the EXT_PS_ON signal to switch the external power supply on and off. In Slave mode, the BMC does not control the EXT_PS_ON signal.

Table 29. BMC API – External power supply control commands

Command	Packet Type	Opcode	Functional Description
EXT_PS_ON_MODE_SET	*PT_WBD*	0x23	Set *EXT_PS_ON* mode
EXT_PS_ON_MODE_GET	*PT_RBD*	0x23	Get *EXT_PS_ON* mode

Commands EXT_PS_ON_MODE_SET and EXT_PS_ON_MODE_GET

Command EXT_PS_ON_MODE_SET

Opcode: 0x23								**Packet Type:** ***PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*EXT_PS_ON_MODE*							

Command EXT_PS_ON_MODE_GET

Opcode: 0x23								**Packet Type:** ***PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*EXT_PS_ON_MODE*							

EXT_PS_ON_MODE	External power supply on/off mode ▪ 0x00: Invalid ▪ 0x01: Always ▪ 0x02: Switched ▪ 0xFF: Error

5.3.1.7 Software Reset

These commands allow performing CPU resets under application software control.

Different types of resets are available:

- *SW_RESET* issues a “warm reset”.
- *SW_COLD_RESET* issues a “cold reset”.
- *SW_RTC_RESET* issues a “cold reset”, together with an RTC reset.

Resets will be performed by writing the data word `0xDEAD`, see below.

Table 30. BMC API – Software reset commands

Command	Packet Type	Opcode	Functional Description
SW_RESET	*PT_WWD*	`0x31`	Initiate software reset (warm reset)
SW_COLD_RESET	*PT_WWD*	`0x32`	Initiate cold reset
SW_RTC_RESET	*PT_WWD*	`0x35`	Initiate cold reset combined with RTC reset

Command SW_RESET

Opcode: `0x31`								**Packet Type:** ***PT_WWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	`0xAD`							
Data 1	`0xDE`							

Command SW_COLD_RESET

Opcode: `0x32`								**Packet Type:** ***PT_WWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	`0xAD`							
Data 1	`0xDE`							

Command SW_RTC_RESET

Opcode: `0x35`								**Packet Type:** ***PT_WWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	`0xAD`							
Data 1	`0xDE`							

5.3.1.8 Power Button

These commands allow initiating power button events.

Table 31. BMC API – Power button commands

Command	Packet Type	Opcode	Functional Description
PWRBTN	*PT_WBD*	0x33	Perform pressing of power button
PWRBTN_OVRD	*PT_WBD*	0x34	Perform power button override, i.e. assert the power button for more than 4 seconds to initiate system shutdown

Command PWRBTN

Opcode: 0x33								**Packet Type:** ***PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	0x69							

Command PWRBTN_OVRD

Opcode: 0x34								**Packet Type:** ***PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	0x69							

5.3.1.9 Voltage Supervision

The voltage supervision commands the customer application to monitor different voltages on the G52A.

Table 32. BMC API – Voltage supervision commands

Command	Packet Type	Opcode	Functional Description
VOLT_LOW(0)	*PT_RWD*	0x40	Get lower limit of +3.3 V (in mV)
VOLT_LOW(1)	*PT_RWD*	0x41	Get lower limit of +5 V (in mV)
VOLT_LOW(2)	*PT_RWD*	0x42	Get lower limit of +12 V (in mV)
VOLT_LOW(3)	*PT_RWD*	0x43	Get lower limit of +5 V standby (in mV)
VOLT_LOW(4)	*PT_RWD*	0x44	Get lower limit of battery voltage (in mV)
VOLT_HIGH(0)	*PT_RWD*	0x50	Get upper limit of +3.3 V (in mV)
VOLT_HIGH(1)	*PT_RWD*	0x51	Get upper limit of +5 V (in mV)
VOLT_HIGH(2)	*PT_RWD*	0x52	Get upper limit of +12 V (in mV)
VOLT_HIGH(3)	*PT_RWD*	0x53	Get upper limit of +5 V standby (in mV)
VOLT_HIGH(4)	*PT_RWD*	0x54	Get upper limit of battery voltage (in mV)
VOLT_ACT(0)	*PT_RWD*	0x60	Get actual value of +3.3 V (in mV)
VOLT_ACT(1)	*PT_RWD*	0x61	Get actual value of +5 V (in mV)
VOLT_ACT(2)	*PT_RWD*	0x62	Get actual value of +12 V (in mV)
VOLT_ACT(3)	*PT_RWD*	0x63	Get actual value of +5 V standby (in mV)
VOLT_ACT(4)	*PT_RWD*	0x64	Get actual value of battery voltage (in mV)
NUM_VOLTS	*PT_RBD*	0x8E	Get number of supervised voltages

Command VOLT_LOW(x)

Opcode: 0x40 + x								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Lower limit of voltage x (LSB)							
Data 1	Lower limit of voltage x (MSB)							

Command VOLT_HIGH(x)

Opcode: 0x50 + x								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Upper limit of voltage x (LSB)							
Data 1	Upper limit of voltage x (MSB)							

Command VOLT_ACT(x)

Opcode: 0x60 + x								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Actual value of voltage x (LSB)							
Data 1	Actual value of voltage x (MSB)							

Command NUM_VOLTS

Opcode: 0x8E								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	Number of supervised voltages							

5.3.1.10 Error Counters

The error counter commands allow querying and clearing error counters.

The BMC provides error counters for each type of error that can occur. Using this information, the application software can determine how often certain errors have occurred, but it is not possible to determine the chronological order of the errors.

You can determine the actual number of error counters using *NUM_ERR_CNTRS*, up to a theoretical maximum of 255 error counters.

All counters are set to zero during factory programming or using command *ERR_CNT_CLR*.

Table 33. BMC API – Error counters

Counter	Error Condition / Error Clearing
1	External BMC watchdog timeout (application software timeout)
2	Internal BMC watchdog timeout
3	Internal brown-out (BMC undervoltage)
4	External power failure
5	BIOS life sign timeout
6	Processor too hot
7	Shutdown while too hot
8	Internal power failure
9	Handshake timeout
10	Platform reset timeout
11	Error cleared using system reset
12	Error cleared using power cycling
13	Error cleared using power cycling with resume reset
14	Error cleared using power cycling with RTC reset
15	Error could not be corrected

Table 34. BMC API – Error counter commands

Command	Packet Type	Opcode	Functional Description
ERRCNT_01	*PT_RBD*	0x70	Get error counter 1
ERRCNT_xx		0x70 + x	Get error counter xx
ERRCNT_15		0x7E	Get error counter 15
ERR_CNT_CLR	*PT_WBD*	0x7F	Clear error counters
NUM_ERR_CNTRS	*PT_RBD*	0x8D	Get number of error counters

Command ERRCNT_xx (1 to 15)

Opcode: 0x70 + x							**Packet Type: *PT_RBD***	
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	Value of error counter number xx							

Command ERRCNT_xx (16 to 32)

Opcode: 0xB0 + x								**Packet Type:** *PT_RBD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	Value of error counter number xx							

Command ERR_CNT_CLR

This command clears all error counters.

Opcode: 0x7F								**Packet Type:** *PT_WBD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	0x69							

Command NUM_ERR_CNTRS

Opcode: 0x8D								**Packet Type:** *PT_RBD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	Number of error counters							

5.3.1.11 Firmware Revision

The firmware revision commands allow querying the separate parts of the BMC firmware revision.

Table 35. BMC API – Firmware version commands

Command	Packet Type	Opcode	Functional Description
GETREV_WORD0	*PT_RWD*	0x80	Get firmware revision major part
GETREV_WORD1	*PT_RWD*	0x81	Get firmware revision minor part
GETREV_WORD2	*PT_RWD*	0x82	Get firmware revision maintenance part
GETREV_WORD3	*PT_RWD*	0x83	Get firmware revision build part
GETREV_WORD4	*PT_RWD*	0x84	Get firmware revision verification marker

Command GETREV_WORD0

Opcode: 0x80								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Firmware Revision Major Part (LSB)							
Data 1	Firmware Revision Major Part (MSB)							

Command GETREV_WORD1

Opcode: 0x81								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Firmware Revision Minor Part (LSB)							
Data 1	Firmware Revision Minor Part (MSB)							

Command GETREV_WORD2

Opcode: 0x82								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Firmware Revision Maintenance Part (LSB)							
Data 1	Firmware Revision Maintenance Part (MSB)							

Command GETREV_WORD3

Opcode: 0x83								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Firmware Revision Build Part (LSB)							
Data 1	Firmware Revision Build Part (MSB)							

Command GETREV_WORD4

Opcode: 0x84								Packet Type: *PT_RWD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	Firmware Revision Verification Marker (LSB)							
Data 1	Firmware Revision Verification Marker (MSB)							

5.3.1.12 Board Controller Mode

This command allows determining if the CPU is operated as a master or slave.

Table 36. BMC API – Board controller mode command

Command	Packet Type	Opcode	Functional Description
BOARD_MODE	*PT_RBD*	0x8B	Get board controller mode (Master/ Slave)

Command BOARD_MODE

Opcode: 0x8B								Packet Type: *PT_RBD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*BOARD_CTRL_MODE*							

Bit Field	Description
BOARD_CTRL_MODE	Board controller mode ▪ 0x00: Invalid ▪ 0x01: Master ▪ 0x02: Slave ▪ 0xFF: Error

5.3.1.13 Geographical Address

Table 37. BMC API – Backplane slot geographical address command

Command	Packet Type	Opcode	Functional Description
CPCI_SLOT_ADDRESS	*PT_RBD*	0x8C	Get CompactPCI peripheral slot address

Command SLOT_ADDRESS

Opcode: 0x8C								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*SLOT_ADDRESS*							

Bit Field	Description
SLOT_ADDRESS	CompactPCI backplane slot geographical board address ▪ 0x00 – 0x07: Information from GA[2:0] backplane pins ▪ 0xFF: Error

5.3.1.14 Hardware Board Type

This command allows the BMC to query the board type, i.e. a unique ID that MEN assigns to each hardware board the generic BMC is implemented on. The board type is programmed into the BMC during production. The setting is persistent, i.e. is stored in a non-volatile memory.

Command HW_BOARD_GET

Opcode: 0x8F								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	*BOARD* (LSB)							
Data 1	*BOARD* (MSB)							

Bit Field	Description
BOARD	Unique MEN board ID

5.3.1.15 Last Error

This command allows querying the last error.

Table 38. BMC API – Last error command

Command	Packet Type	Opcode	Functional Description
ERR_LAST	*PT_RBD*	0x90	Get last error

Command ERR_LAST

Opcode: 0x90								Packet Type: *PT_RBD*
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*LAST_ERR_CODE*							

Bit Field	Description
LAST_ERR_CODE	Last error ▪ 0x00: Initial value; no error was registered by the BMC since the Last Error Register was cleared ▪ 0x01: +3.3 V voltage failure ▪ 0x02: Input voltage failure ▪ 0x03: External power supply failure ▪ 0x04: CPU too hot ▪ 0x05: BIOS life sign timeout ▪ 0x06: System reset timeout ▪ 0x07: Platform reset failure ▪ 0x08: Chipset handshake failure ▪ 0x09: System power OK failure ▪ 0xFF: Error

5.3.1.16 Power Failure Flags

This command allows querying the power failure flags of the G52A.

Table 39. BMC API – Power failure flags command

Command	Packet Type	Opcode	Functional Description
ERR_PWR_FLAGS	*PT_RBD*	0x91	Get power failure flags

Command ERR_PWR_FLAGS

Whenever a power failure occurs, the respective flag is set to 1 until the Power Failure Flag Register is cleared.

Opcode: 0x91								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*BATT*	-	*EXT*	*SYS_ PWROK*	*12V*	*5V_ STDBY*	*5V*	*33V*

Bit Field	Description
Initial Value	0x00: No power failure was registered by the BMC since the Power Failure Flag Register was cleared.
BATT	Battery failure
EXT	External power supply failure
SYS_PWROK	System power OK failure
12V	+12 V input voltage failure
5V_STDBY	+5 V standby voltage failure
5V	+5 V input voltage failure
33V	+3.3 V voltage failure

5.3.1.17 Reset Reason

This command allows querying the reason of the last reset. The BMC maintains a Reset Reason Register that stores the reason for the last reset issued by the BMC.

Table 40. BMC API – Reset reason command

Command	Packet Type	Opcode	Functional Description
ERR_RST_RSN	*PT_RBD*	0x92	Get reason of last reset

Command ERR_RST_RSN

Opcode: 0x92								**Packet Type: *PT_RBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	*RST_REASON*							

Bit Field	Description
RST_REASON	Reason of last reset ▪ 0x00: Initial value; no reset was issued by the BMC since the Reset Reason Register was cleared ▪ 0x01: Regular reset ▪ 0x02: External BMC watchdog timeout (application software timeout) ▪ 0x03: Internal BMC watchdog timeout ▪ 0x04: Internal brown-out reset (BMC undervoltage) ▪ 0x05: External reset ▪ 0x06: Platform reset ▪ 0x07: Software warm reset ▪ 0x08: Software cold reset ▪ 0x09: Software cold reset with RTC reset ▪ 0x0A: Power failure ▪ 0x0B: Chipset handshaking timeout ▪ 0x0C: PLT_RST timeout ▪ 0x0D: BIOS life sign timeout

5.3.1.18 Clear Error Registers

This command allows clearing the Reset Reason Register, Last Error Register and Power Failure Flag Register, collectively called 'error registers'.

Table 41. BMC API – Clear error registers command

Command	Packet Type	Opcode	Functional Description
ERR_REG_CLR	*PT_WBD*	0x9F	Clear error registers

Command ERR_REG_CLR

Opcode: 0x9F								**Packet Type: *PT_WBD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	0x69							

5.3.1.19 Power Cycle Counter

The power cycle counter counts the number of power cycles of the external power supply, i.e. the number of times the system changes from S5 into S0 state.
S0 to S5 are the power states as defined in the ACPI specification, or an equivalent state

Please refer to the ACPI Specification for more details on the power states S0 to S5:
Advanced Configuration and Power Interface Specification Version 6.1
January, 2016
Unified EFI Forum
uefi.org/specifications

The counter is set to zero during factory programming.

Table 42. BMC API – Power cycle counter command

Command	Packet Type	Opcode	Functional Description
PWRCYCL_CNT	*PT_RWD*	0x93	Get power cycle counter

Command PWRCYCL_CNT

Opcode: 0x93								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	*PWR_CYCLES* (LSB)							
Data 1	*PWR_CYCLES* (MSB)							

Bit Field	Description
PWR_CYCLES	Number of power cycles on the external power supply

5.3.1.20 Operating Hours Counter

This command allows querying the operating hours counter. The operating hours counter counts the number of hours and minutes the board has been (at least partly) powered on, i.e. when the system is in S3 or S0 state.
S0 to S5 are the power states as defined in the ACPI specification, or an equivalent state

Please refer to the ACPI Specification for more details on the power states S0 to S5:
Advanced Configuration and Power Interface Specification Version 6.1
January, 2016
Unified EFI Forum
uefi.org/specifications

The counter is set to zero during factory programming.

Table 43. BMC API – Operating hours counter command

Command	Packet Type	Opcode	Functional Description
OP_HRS_CNT	*PT_RWD*	0x94	Get Operating Hours Counter

Command OP_HRS_CNT

Opcode: 0x94								**Packet Type: *PT_RWD***
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data 0	*OP_TIME* (LSB)							
Data 1	*OP_TIME* (MSB)							

Bit Field	Description
OP_TIME	Number of hours the board has been powered on

5.3.1.21 Status LED Control

This command allows controlling status LEDs, depending on implementation on the product.

Table 44. BMC API – Status LED control command

Command	Packet Type	Opcode	Functional Description
LED_CTRL_SET	*PT_WBD*	0xA0	Set LED state
LED_CTRL_GET	*PT_RBD*	0xA0	Get LED state

Command LED_CTRL_SET

Opcode: 0xA0							**Packet Type: *PT_WBD***	
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	-				*USR2*	*USR1*	*HTSWP*	*STA*

Command LED_CTRL_GET

Opcode: 0xA0							**Packet Type: *PT_RBD***	
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data	-				*USR2*	*USR1*	*HTSWP*	*STA*

Bit Field	Description
USR2:1	User outputs 1 and 2
HTSWP	Hot swap LED
STA	Status LED at front panel

5.3.2 Example BMC API Usage

See Chapter 2.6.1 Accessing Board Management Functions on page 19 for how to access board management functions under Linux.

6 Maintenance

6.1 Optional Lithium Battery

This board might contain a lithium battery. There is a danger of explosion if the battery is incorrectly replaced!

- Replace only with the same or equivalent type.
- Manufacturer: Renata
- Type: CR1025
- Capacity: 30 mAh
- The battery has to be UL listed.

Used batteries have to be disposed of according to the local regulations concerning the disposal of hazardous waste.

Figure 5. Position of optional lithium battery on G52A