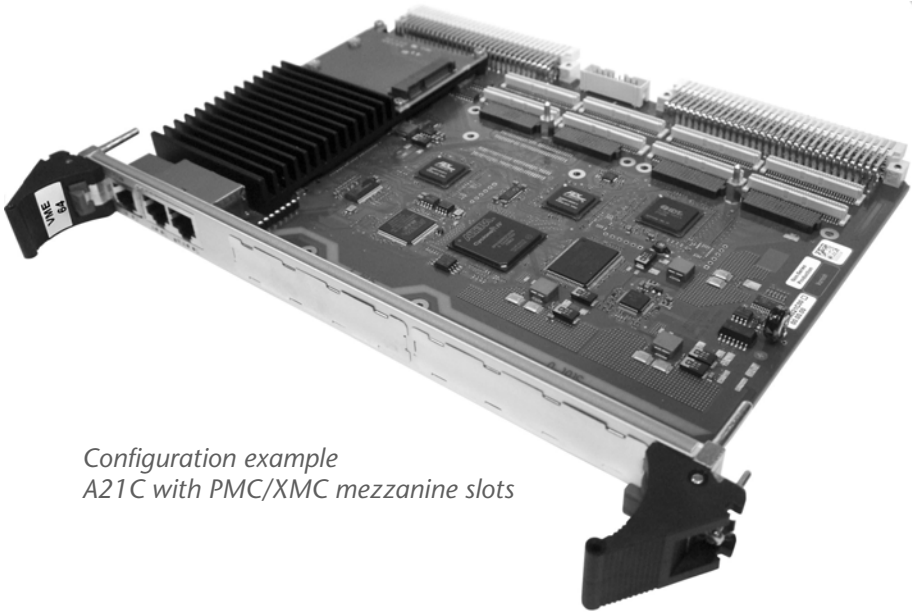
Embedded Solutions

A21B/A21C – 6U VME64 QorIQ™ P1013/P1022 CPU Boards with Mezzanines



Configuration example
A21C with PMC/XMC mezzanine slots

User Manual

mikro elektronik
gmbh · nürnberg

A21B/A21C – 6U VME64 QorIQ™ P1013/P1022 CPU Boards with Mezzanines

The A21 is a Freescale™ QorIQ™ based single-board computer for embedded industrial applications. The SBC features full VME64 support and can be used as a master or a slave in a VMEbus environment. The A21 provides 1 MB local dual-ported SRAM for slave access and communication between the local CPU and another VMEbus master.

The CPU card comes with a single-core P1013 or dual-core P1022 QorIQ™ processor with up to 1.067 GHz clock frequency and a serial communication architecture. With two Gigabit Ethernet ports and one RS232 COM at the front, and DDR3 SDRAM with ECC, Flash and FRAM, the board offers the crucial basics of an industrial computer. To satisfy your needs for mass storage, you can use microSD™ cards and mSATA plug-in modules.

In addition, the A21 can be equipped with mezzanine modules: the A21C accommodates up to two XMC or PMC mezzanine cards on shared sites, for functions such as graphics, mass storage, or further Ethernet. The two PMC slots support modules up to 64-bit/133-MHz PCI-X, while the XMC slots are powered by two PCI Express® x1 links each. Its sister card, the A21B, offers three M-Module™ slots, which are ideal for real-world I/O like analog/binary process and instrumentation input/output.

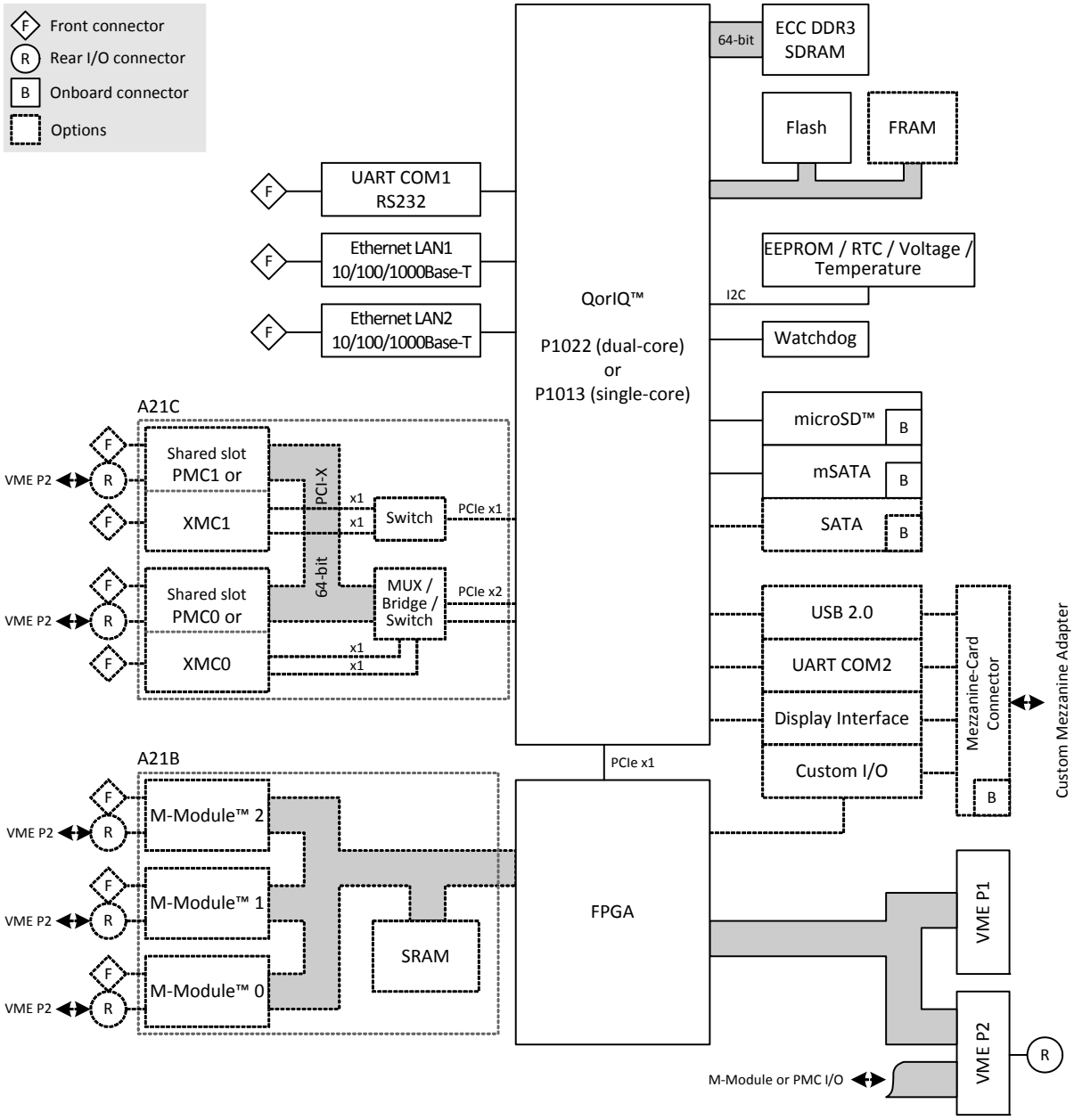
The supported mezzanine types provide both front (all module types) and rear I/O (PMCs and M-Modules™). The modular combination of I/O functionality on a single-board computer allows to build up tailored control systems which appear as customized solutions based on standard components.

Where there's a need for even more or other I/O, the A21 also includes a custom mezzanine-card option that reduces the board by one regular PMC/XMC/ M-Module™ slot but provides interfaces like USB 2.0, COM or even custom I/O controlled by the onboard FPGA. The mezzanine card is always an entirely customized adapter PCB, including front I/O, and makes the A21 a semi-custom solution.

The A21 supports operation in a -40°C to +85°C temperature range, and the board withstands shock and vibration.

The CPU board is supported by the U-Boot Universal Boot Loader, which can be used for bootstrapping operating systems, for hardware testing, or for debugging applications without running any operating system.

Block Diagram



Technical Data

CPU

- Freescale™ QorIQ™ P1022 or P1013, dual or single core
  - 600 MHz, 800 MHz or 1.067 GHz
  - Two/one high-performance Power Architecture e500v2 cores
  - Double precision floating point support and signal processing engine (SPE) APU

Memory

- 32 KB L1 instruction cache and 32 KB L1 data cache per processor core
- 256 KB L2 cache with ECC
- Up to 2 GB SDRAM system memory
  - Soldered
  - DDR3 with ECC support
  - 333 MHz memory bus clock frequency (667 MT/s, 5.33 GB/s data rate)
- Up to 64 MB boot/program Flash
- 128 KB non-volatile FRAM
- Serial EEPROM 4 kbits for factory settings

Mass Storage

- microSD™ card slot
  - Directly accessible on the board
  - Connected to SDHC controller
- mSATA disk slot
  - Directly accessible on the board (via small adapter card)
  - Connected via one SATA channel

I/O

- Ethernet
  - Two 10/100/1000Base-T Ethernet channels at the front
  - RJ45 connectors at front panel
  - Two front LEDs for each port to signal LAN link and activity status
- One RS232 UART (COM1)
  - RJ45 connector at front panel
  - Data rates up to 230.4 kbit/s
  - 16-byte transmit/receive buffer
  - Handshake lines: CTS, RTS

Front Connections

- Two Ethernet (RJ45)
- One RS232 COM (RJ45)
- A21B: M-Module™ front I/O if populated
- A21C: PMC / XMC front I/O if populated

Rear I/O

- A21B: M-Module™ 0, 1 and 2
- A21C: PMC 0 and 1

Mezzanine Extensions A21B

- Three M-Module™ slots
  - Compliant with M-Module™ standard
  - Characteristics: A08, A24, D16, D32, INTA, INTC, TRIGI, TRIGO

Mezzanine Extensions A21C

- Two slots usable for PMC or XMC
- XMC slots
  - Compliant with XMC standard VITA 42.3-2006
  - Two x1 PCI Express® links for slot 0, data rate 250 MB/s per link in each direction (2.5 Gbit/s per lane)
  - Two x1 PCI Express® links for slot 1, data rate 125 MB/s per link in each direction (1.25 Gbit/s per lane)
  - PCIe® 1.0a support (PCI Express® Base Specification)
- PMC slots
  - Compliant with PMC standard IEEE 1386.1
  - PCI / PCI-X 32/64 bits, 33/66/133 MHz, 3.3 V V(I/O)
  - PMC I/O module (PIM) support through J4 for both slots

Miscellaneous

- Real-time clock, buffered by a supercapacitor or battery (optional)
  - Data retention of supercapacitor: typically up to one week
- Watchdog
- Voltage monitor and temperature sensor
- Reset button and status LEDs at the front panel

VMEbus

- Compliant with VME64 Specification
- Slot-1 function with auto-detection
- Master
  - D08(EO):D16:D32:D64:A16:A24:A32:ADO:BLT:RMW
- Slave
  - D08(EO):D16:D32:D64:A16:A24:A32:BLT:RMW
- 1 MB shared fast SRAM
- DMA
- Mailbox functionality
- Interrupter D08(O):I(7-1):ROAK
- Interrupt handler D08(O):IH(7-1)
- Single level 3 fair requester
- Single level 3 arbiter
- Bus timer
- Location Monitor

- Performance
  - Coupled read/write D32 non-block transfer rate tbd. MB/s
  - DMA read/write D32 BLT transfer rate tbd. MB/s
  - DMA read/write D64 MBLT transfer rate tbd. MB/s

Electrical Specifications

- Supply voltage/power consumption:
  - +5 V (-3%/+5%), 1.3 A typ.
  - +3.3 V (-3%/+5%), 1 A typ.
  - ±12 V (-5%/+5%), only provided for mezzanines that need 12 V

Mechanical Specifications

- Dimensions: standard double Eurocard, 233.3 mm x 160 mm
- Weight (without mezzanines):
  - A21B: 428 g
  - A21C: 412 g

Environmental Specifications

- Temperature range (operation):
  - -40..+85°C (screened)
  - Airflow: min. 1.0 m/s
- Temperature range (storage): -40..+85°C
- Relative humidity (operation): max. 95% non-condensing
- Relative humidity (storage): max. 95% non-condensing
- Altitude: -300 m to +3000 m
- Shock: 50 m/s², 30 ms (EN 61373)
- Vibration (function): 1 m/s², 5 Hz – 150 Hz (EN 61373)
- Vibration (lifetime): 7.9 m/s², 5 Hz – 150 Hz (EN 61373)
- Conformal coating on request

MTBF

- A21B: 346 417 h @ 40°C according to IEC/TR 62380 (RDF 2000)
- A21C: 286 910 h @ 40°C according to IEC/TR 62380 (RDF 2000)

Safety

- PCB manufactured with a flammability rating of 94V-0 by UL recognized manufacturers

EMC Conformity

- EN 55022 (radio disturbance)
- IEC 61000-4-2 (ESD)
- IEC 61000-4-3 (electromagnetic field immunity)
- IEC 61000-4-4 (burst)
- IEC 61000-4-5 (surge)
- IEC 61000-4-6 (conducted disturbances)

BIOS

- U-Boot Universal Boot Loader

Software Support

- Linux
- VxWorks®
- OS-9®
- QNX® (on request)
- For more information on supported operating system versions and drivers see the A21B and A21C pages on MEN's website.

Configuration Options

CPU

- QorIQ™ P1022 or P1013
  - P1022: dual core
  - P1013: single core
- All processors available with 600 MHz, 800 MHz or 1.067 GHz

Memory

- System RAM
  - 1 GB or 2 GB
- Boot/program Flash
  - 32 MB or 64 MB
- FRAM
  - 0 KB or 128 KB

Mass Storage

- Serial ATA (SATA)
  - Onboard SATA connector for one additional port possible
  - SATA Revision 2.x support
  - Transfer rates up to 300 MB/s (3 Gbit/s)
  - For connection of an in-system hard-disk drive

I/O

- Various additional I/O possible using onboard mezzanine card
  - Partly fixed set of interfaces, plus 16 pins for custom I/O
  - One USB 2.0 port, EHCI implementation
  - Additional UART COM interface
  - Display interface
  - Custom I/O functions can be implemented as FPGA IP cores (16 pins usable)
  - Occupies the space of M-Module™ slot 3 (A21B) or PMC/XMC slot 1 (A21C)
  - Please note that the mezzanine card is always completely customized, including front I/O, no standard cards are available.

Mezzanine Slots

- 3 M-Module™ slots with A21B
- 2 PMC / XMC slots with A21C

Miscellaneous

- Back-up battery holder for real-time clock (RTC) (may be in mechanical conflict with mezzanine module slot 0)

Software Support

- QNX®

Please note that some of these options may only be available for large volumes. Please ask our sales staff for more information.

For available standard configurations see the A21B and A21C pages on MEN's website.

Product Safety

Electrostatic Discharge (ESD)

Computer boards and components contain electrostatic sensitive devices. Electrostatic discharge (ESD) can damage components. To protect the board 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.
- Store the board only in its original ESD-protected packaging. Retain the original packaging in case you need to return the board to MEN for repair.

About this Document

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

It describes the hardware functions of the board, connection of peripheral devices and integration into a system. It also provides additional information for special applications and configurations of the board.

The manual does not include detailed information on individual components (data sheets etc.). A list of literature is given in the appendix.

The two varieties, A21B and A21C, are generally referred to as A21, and are fully named only where there are differences.

History

Issue	Comments	Date
E1	First issue	2012-07-31
E2	General update, minor errors corrected, various additions	2013-04-25

Conventions

This sign marks important notes or warnings concerning the use of voltages which can lead to serious damage to your health and also cause damage or destruction of the component.

This sign marks important notes or warnings concerning proper functionality of the product described in this document. You should read them in any case.

italics — Folder, file and function names are printed in *italics*.

bold — **Bold** type is used for emphasis.

`monospace` — A `monospaced` font type is used for hexadecimal numbers, listings, C function descriptions or wherever appropriate. Hexadecimal numbers are preceded by "0x".

comment — Comments embedded into coding examples are shown in green color.

hyperlink — Hyperlinks are printed in blue color.

The globe will show you where hyperlinks lead directly to the Internet, so you can look for the latest information online.

IRQ#
/IRQ — Signal names followed by "#" or preceded by a 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 "coming from it".

Vertical lines on the outer margin signal technical changes to the previous issue of the document.

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.

Trademarks

All products or services mentioned in this publication are identified by the trademarks, service marks, or product names as designated by the companies which market those products. The trademarks and registered trademarks are held by the companies producing them. Inquiries concerning such trademarks should be made directly to those companies.

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
Fax +49-911-99 33 5-901
E-mail info@men.de
www.men.de

France
MEN Mikro Elektronik SA
18, rue René Cassin
ZA de la Châtelaine
74240 Gaillard
Phone +33 (0) 450-955-312
Fax +33 (0) 450-955-211
E-mail info@men-france.fr
www.men-france.fr

USA
MEN Micro, Inc.
24 North Main Street
Ambler, PA 19002
Phone (215) 542-9575
Fax (215) 542-9577
E-mail sales@menmicro.com
www.menmicro.com

Contents

Figures

Tables

1 Getting Started

This chapter gives an overview of the board and some hints for first installation in a system.

1.1 Map of the Board

Figure 1. *Map of the board – A21B front panel and top view (M-Modules)*

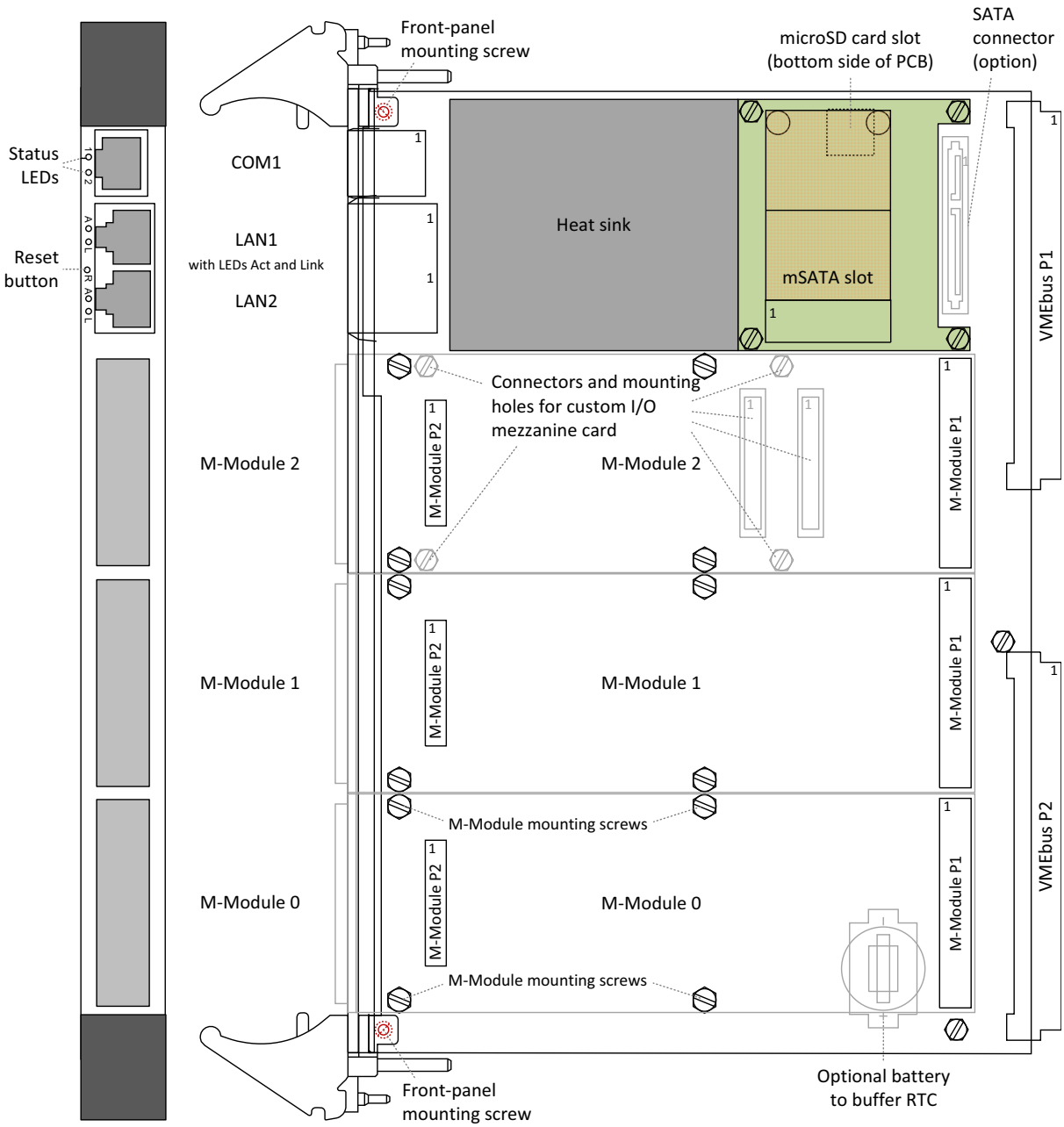
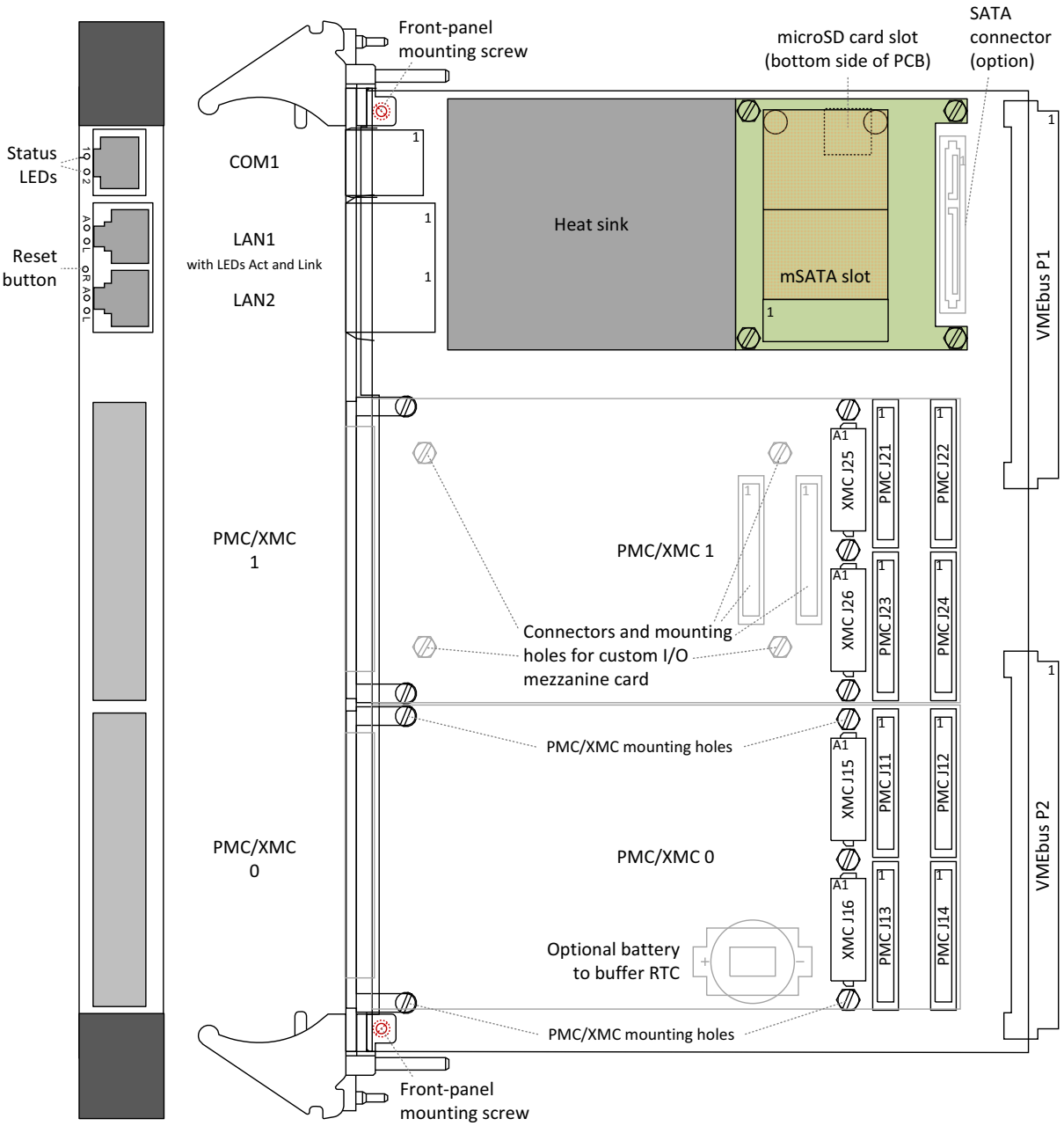


Figure 2. *Map of the board – A21C front panel and top view (PMC/XMC modules)*



1.2 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 when the board is mounted in a rack.

The following check list gives an overview on what you might want to configure.

☑ M-Modules

Refer to Chapter 2.9.3 Installing an M-Module Mezzanine Module on page 38 for a detailed installation description.

☑ PMC modules

Refer to Chapter Both slots support rear I/O via VMEbus connector P2, see Table 28, Pin assignment of VMEbus rear I/O connector P2 – A21C – PMC on page 76. on page 39 for a detailed installation description.

☑ microSD

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

Refer to Chapter 2.6.1 microSD Card Slot on page 27 for more information on installation of the card.

☑ mSATA disk

The board is shipped without an mSATA disk. You should check your needs and install a suitable disk.

Refer to Chapter 2.6.2 mSATA Slot on page 29 for more information on installation of the disk.

☑ SATA hard disk

If an **optional** SATA connector is implemented on the board, you can install a SATA hard disk inside the system. However, since this is only an option and the hard disk cannot be mounted directly on the A21, there is no ready-to-use adapter or other mounting material available.

Refer to Chapter 2.6.3 SATA Interface Option on page 30 for details on the SATA interface.

1.3 Integrating the Board into a System

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

We recommend to perform the following procedure without any mezzanine module installed.

- ☑ Power down the system.
- ☑ Remove all boards from the VMEbus system.
- ☑ Insert the A21 into slot 1 of the system, making sure that the connectors are properly aligned.
- ☑ Connect a terminal to the RS232 interface COM1 (RJ45 connector).
- ☑ 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.
- ☑ The terminal displays a message similar to the following:

```
U-Boot 2012.04-rc1 (Apr 10 2012 - 18:05:15)MEN_14A021-00_0.2_beta (standard)

CPU0:  P1022E, Version: 1.1, (0x80ee0011)
Core:  E500, Version: 5.1, (0x80211151)
Clock Configuration:
       CPU0:800  MHz, CPU1:800  MHz,
       CCB:400  MHz,
       DDR:333  MHz (666 MT/s data rate) (Asynchronous), LBC:25   MHz
L1:    D-cache 32 kB enabled
       I-cache 32 kB enabled
Board: MEN A021
Wdog:  init by SW, disabled
Reset Cause: 0 (Power On Reset)
I2C:   ready
DRAM:         (memory mapped)
       Detected UDIMM
1 GiB (DDR3, 64-bit, CL=6, ECC on)
Testing 0x00000000 - 0x3fffffff
Remap DDR
POST memory PASSED
Flash: 32 MiB
L2:    256 KB enabled
MMC:  FSL_SDHC: 0
PCIe1: Root Complex of XMC Slot 1, no link, regs @ 0xff70a000
PCIe1: Bus 00 - 00

...Further test output...

Board: name=A021-00 rev=00.00.00 ser#=7
Net:   eTSEC1, eTSEC2 [PRIME]
Hit any key to stop autoboot:  0
A21=>
```

☑ Now you can make configurations for your operating system in the U-Boot boot loader. See the detailed description in Chapter 3 U-Boot Boot Loader on page 78.

☑ Observe the installation instructions for the respective software.

1.4 Installing Operating System Software

The board supports Linux, VxWorks, OS-9 and QNX (on request).

By default, no operating system is installed on the board. Please refer to the respective manufacturer's documentation on how to configure your operating system image!

The U-Boot Chapter of this manual describes the first steps of how to get your operating system running, see Chapter 3.2 Getting Started: Setting Up Your Operating System on page 79.

You can find any software available in the A21B and A21C pages on MEN's website.

1.5 Installing Driver Software

For a detailed description on how to install driver software, e.g., for mezzanine modules, please refer to the respective documentation.

2 Functional Description

The following describes the individual functions of the board and their configuration on the board. There is no detailed description of the individual controller chips and the CPU. They can be obtained from the data sheets or data books of the semiconductor manufacturer concerned (Chapter 6.1 Literature and Web Resources on page 106).

Please note that the board BSPs for the different operating systems may not support all the functions of the A21. For more information on hardware support please see the respective BSP data sheet in the A21B and A21C pages on MEN's website.

2.1 Power Supply

The A21 is supplied with +5 V, ± 3.3 V and ±12 V via the VMEbus. However, ±12 V may be required only by some mezzanine modules. The board is designed for VME64 operation, which also uses 3.3 V. If the 3.3-V pins are not connected, the voltage is generated on the board.

2.2 Board Supervision and Reset Behavior

2.2.1 Temperature and Voltage

The board features a temperature sensor and voltage monitor.

A voltage monitor supervises all used voltages. The CPU is held in reset condition until all supply voltages are within their nominal values.

2.2.2 Watchdog

The A21 has an integrated watchdog that must continuously be triggered. After configuration the CPU serves the watchdog.

The watchdog can be enabled or disabled and can be triggered by a software application. This function is supported by U-Boot and normally also by the board support package (see Chapter Table 34. U-Boot – Environment variables – Other on page 98 and BSP documentation).

There are two possible modes of operation, i.e. timeouts, for the watchdog:

- Slow mode: The watchdog resets the CPU when served less than every 30 s (±30%).
- Fast mode: The watchdog resets the CPU when served less than every 1 s (±30%).

Once the watchdog is enabled, it cannot be disabled at runtime. The same is true for the operating mode: Once you have set the watchdog to fast mode, you cannot change it at runtime anymore.

At every reset of the board, the watchdog is disabled. If the U-Boot setting says to enable the watchdog, it will be enabled by U-Boot and will run in slow mode again.

2.2.3 Reset Behavior

The board is reset if one of the following occurs:

- Normal power on
- Watchdog reset
- Local power bad
- Push button reset
- CPU reset request
- VMEbus reset request

See also Chapter 2.12 Reset Button and Status LEDs on page 47.

2.3 Real-Time Clock

The board includes an RA8581 real-time clock. For data retention during power off the RTC must be supplied with 5 V via the standby voltage of the VME backplane (pin *+5VSTDBY* on P1). It is buffered using an onboard supercapacitor (data retention typically up to 1 week).

The RTC can optionally be buffered by a battery mounted directly on the board. However, the battery holder is assembled only as an option and is in mechanical conflict with PMC/XMC slot 0 (A21C) or M-Module slot 0 (A21B). It may collide with a plugged mezzanine module.

For the exact type and position of the battery holder please see Chapter 5.1 Lithium Battery on page 105.

For more information please contact MEN's sales team.

2.4 Processor Core

The board is equipped with a multi-core Freescale QorIQ P1013 or P1022 processor up to 1.067 GHz, which includes one or two 32-bit Power Architecture e500v2 cores.

2.4.1 General

The QorIQ processors available for A21 combine processing performance, high-speed connectivity and advanced power management for high energy efficiency.

The P1022 and P1013 are identically built. Their only difference is the number of e500 cores. They both have a 64-bit DDR2/3 memory controller with ECC and integrate high-speed serial interfaces, with Gigabit Ethernet, PCI Express 1.x and SATA Revision 2.x support. The selection of two different QorIQ-family processor types usable on the A21 allows tailoring of the board for different application requirements.

Table 1. *Processor core options on A21*

Processor Type	Core Frequency	Cores
P1022	600 MHz, 800 MHz or 1.067 GHz	2
P1013		1

Typical power dissipation of the processor is around 4 W.

2.4.2 Thermal Considerations

A suitable heat sink is provided to meet thermal requirements.

Please note that 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 A21 may cease. If you have any questions or problems regarding thermal behavior, please contact MEN.

2.5 Memory

2.5.1 DRAM System Memory

The board provides up to 2 GB onboard, soldered DDR3 SDRAM with ECC (error-correcting code) on one bank. The memory bus is 72 bits wide including ECC and operates at 333 MHz physical clock frequency (667 MT/s, 5.33 GB/s data rate).

ECC memory provides greater data accuracy and system uptime by protecting against soft errors in computer memory.

2.5.2 NOR Flash

The board includes up to 64 MB soldered NOR Flash memory controlled by the enhanced local bus controller of the processor. The data bus is 16 bits wide.

Flash memory contains the U-Boot/operating system bootstrapper and U-Boot environment variables. It can also contain the operating system and application software. See also Chapter 3.5 Updating the Boot Flash on page 89.

2.5.3 FRAM

The board has up to 128 KB non-volatile FRAM memory connected to the enhanced local bus controller of the processor.

The FRAM does not need a back-up voltage for data retention.

2.5.4 EEPROM

The board has a 4-kbit serial EEPROM for factory data.

2.6 Mass Storage

The A21 offers the possibility to connect a standard mSATA disk and a standard microSD card directly on the board. It also includes an option for a SATA hard-disk connector.

The board supports two SATA channels directly controlled by the processor. One is used for control of the onboard mSATA slot, and the other can control an optional onboard SATA connector for implementation of an additional in-system hard disk.

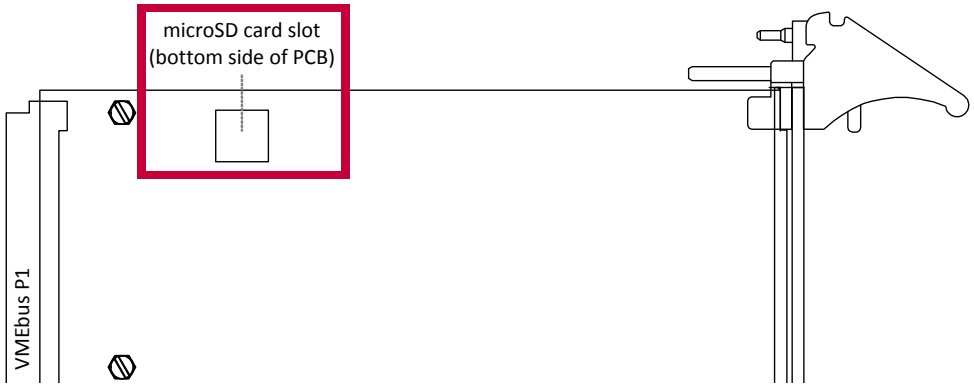
The SATA interfaces are compliant with SATA Revision 2.x and support transfer rates of 3 Gbit/s.

2.6.1 microSD Card Slot

The A21 provides an onboard microSD card slot on the bottom side of the board. It is directly controlled by the processor's SDHC interface and is ready-to-use. The A21 is shipped without a microSD card installed.

Please see the A21B and A21C pages on MEN's website for ordering options.

Figure 3. *Position of the microSD card slot on the bottom side of A21*



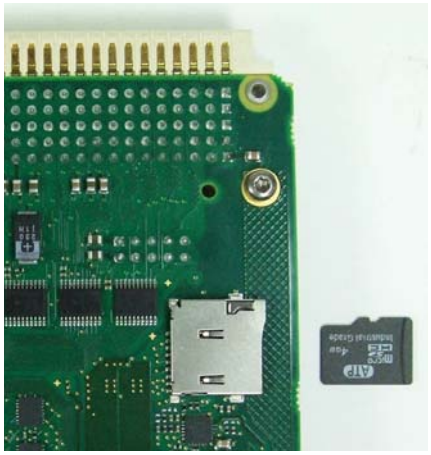
2.6.1.1 Inserting and Extracting a microSD Card

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

☑ Power down your system and remove the A21 from the system.

☑ Put the board on a flat surface carefully with the bottom side facing up.

☑ Insert the microSD card into the slot with the contacts facing down.



☑ Make sure that it clicks into place properly.

Apart from being held by the slot mechanics, the microSD card is firmly held in place by the VMEbus guide rails as well.

☑ For extracting the card push it into the slot a little bit more. It is then released and you can pull it out.

2.6.2 mSATA Slot

The onboard mSATA disk slot of A21 is ready-to-use and is located on a small adapter card in the heat sink area which is assembled by standard. The A21 is shipped without an mSATA disk installed.

Please see the A21B and A21C pages on MEN's website for ordering options.

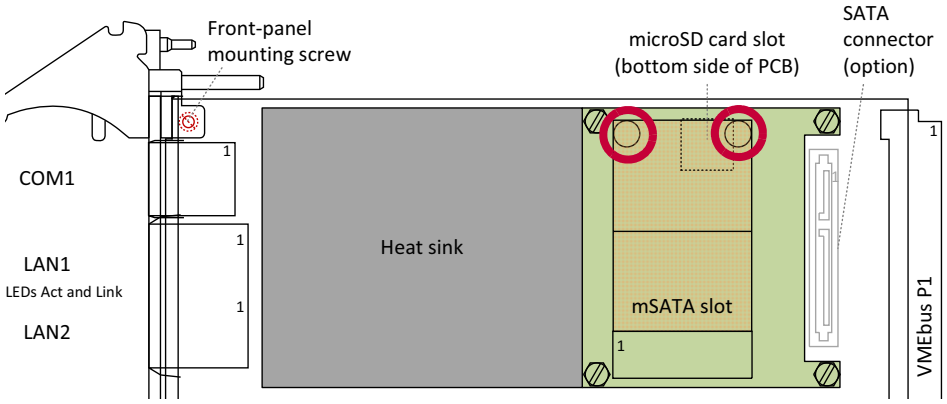
2.6.2.1 Installing an mSATA Disk

To install an mSATA disk, please stick to the following procedure.

- ☑ Power down your system and remove the A21 from the system.
- ☑ Put the board on a flat surface.
- ☑ Insert the mSATA disk carefully in a 30° angle.

- ☑ Make sure that all the contacts are aligned properly and the card is firmly connected with the card connector.

☑ Fix the card using two M2.5 x4 screws (highlighted in red). Suitable screws are supplied with the A21 in an extra bag.

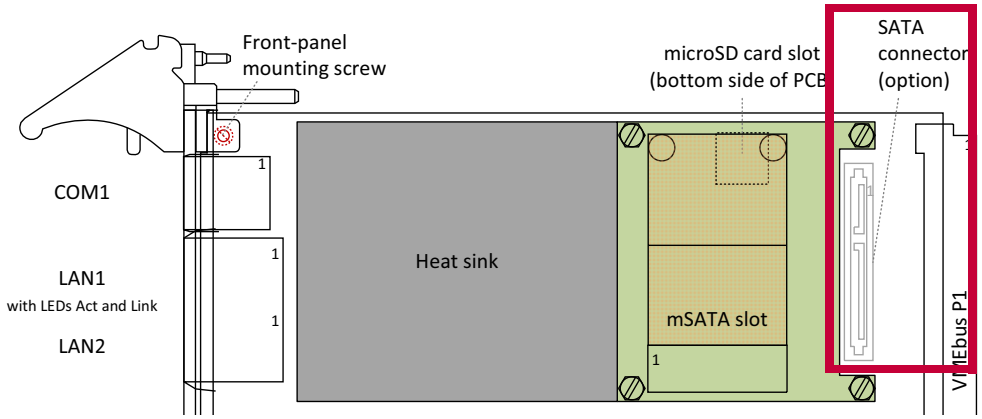


2.6.3 SATA Interface Option

Another processor-controlled SATA interface can be made available through an optional onboard SATA connector fully equipped with all signals, including power. You can connect a hard-disk drive or SSD drive via cable. The disk itself is then located on a second slot.

As this additional connector is only an option, there is no standard mounting kit available for A21.

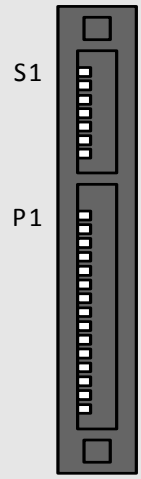
Figure 4. *Position of optional onboard SATA connector*



Connector type:

- 7- & 15-pin SATA receptacle connector, 1.27 mm pitch
- Mating connector:
  7- & 15-pin SATA plug connector, 1.27 mm pitch

Table 2. *Pin assignment of optional 7- & 15-pin SATA connector*



S1	GND
S2	SATA_TX+
S3	SATA_TX-
S4	GND
S5	SATA_RX-
S6	SATA_RX+
S7	GND
Key and spacing, separate signal and power segments	
P1	+3.3V
P2	+3.3V
P3	+3.3V
P4	GND
P5	GND
P6	GND
P7	+5V
P8	+5V
P9	+5V
P10	GND
P11	Reserved
P12	GND
P13	+12V
P14	+12V
P15	+12V

Table 3. *Signal mnemonics of optional SATA connector*

Signal	Direction	Function
+12V	out	+12V power supply
+3.3V	out	+3.3V power supply (optional)
+5V	out	+5V power supply
GND	-	Digital ground
SATA_RX+, SATA_RX-	in	Differential pair of SATA receive lines
SATA_TX+, SATA_TX-	out	Differential pair of SATA transmit lines

2.7 Ethernet Interfaces

The A21 has two Ethernet interfaces controlled by the CPU. Both channels are available at the front panel and support up to 1000 Mbit/s and full-duplex operation.

LEDs for each channel indicate the Ethernet link and activity.

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 A21 are:

- LAN1: `0x 00 C0 3A B1 80 00 - 0x 00 C0 3A B1 9F FF`
- LAN2: `0x 00 C0 3A B1 A0 00 - 0x 00 C0 3A B1 BF FF`

where "00 C0 3A" is the MEN vendor code. The last six digits depend on the interface and the serial number of the product. The serial number is added to the offset, for example for LAN1:

Serial number 0042 (`0x2A`): `0x 80 00 + 0x 00 2A = 0x 80 2A.`

(See Chapter 6.2 Finding out the Product's Article Number, Revision and Serial Number on page 108.)

Note: With prototypes of A21C (hardware revision 00.xx), only LAN2 is operable. LAN1 does not work. Both LAN interfaces work properly from hardware revision 01.xx of A21C and on all hardware revisions of A21B.

2.7.1 Front-Panel Connection

Two standard RJ45 connectors are available at the front panel. There are two status LEDs for each channel at the front panel.

The pin assignment corresponds to the Ethernet specification IEEE802.3.

Connector types:

- Modular 8/8-pin mounting jack according to FCC68
- Mating connector:
  Modular 8/8-pin plug according to FCC68

Table 4. *Pin assignment and status LEDs of Ethernet front-panel connectors*

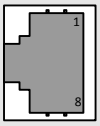
				1000Base-T	10/100Base-T
Off: No activity **Flashing**: Transmit or receive activity	A ●		1	BI_DA+	TX+
			2	BI_DA-	TX-
			3	BI_DB+	RX+
			4	BI_DC+	-
On: Link on (any speed) **Off**: Link off	L ●		5	BI_DC-	-
			6	BI_DB-	RX-
			7	BI_DD+	-
			8	BI_DD-	-

Table 5. *Signal mnemonics of Ethernet front-panel connectors*

Signal	Direction	Function
BI_Dx+/-	in/out	Differential pairs of data lines for 1000Base-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

2.8 RS232 COM1 Interface

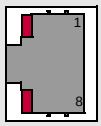
The A21 provides one RS232 COM interface at the front panel. It is controlled by the processor and supports data rates up to 230.4 kbit/s, a 16-byte transmit/receive buffer and handshake lines CTS and RTS.

A second UART COM port can be made available with a customized physical layer through the board's mezzanine-card option, see Chapter 2.11 I/O Extension on page 46.

Connector types:

- Modular 8/8-pin mounting jack according to FCC68
- Mating connector:
  Modular 8/8-pin plug according to FCC68

Table 6. *Pin assignment of RJ45 RS232 connector (COM1)*



1	-
2	-
3	-
4	GND
5	RXD
6	TXD
7	CTS
8	RTS

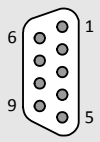
Table 7. *Signal mnemonics of RJ45 RS232 connector (COM1)*

Signal	Direction	Function
GND	-	Ground
CTS	in	Clear to send
RTS	out	Request to send
RXD	in	Receive data
TXD	out	Transmit data

MEN offers an RS232 interface cable that leads the RJ45 connector to a standard 9-pin D-Sub plug.

The D-Sub connector has the following pinout:

Table 8. *Pin assignment of 9-pin D-Sub plug connector for RS232 COM1*



6	-	1	-
7	RTS	2	RXD
8	CTS	3	TXD
9	-	4	-
		5	GND

Please see the A21B and A21C pages on MEN's website for ordering options.

2.9 M-Module Slots (A21B)

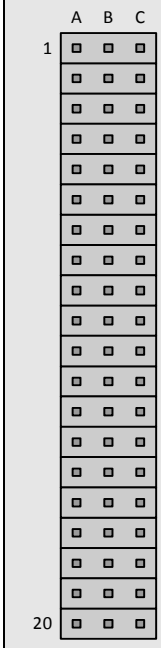
M-Module slots enable the user to add a number of I/O functions to the CPU board. The wide range of standardized M-Modules includes not only process I/O modules but also interface extensions, network boards (such as Profibus, CAN bus etc.), DSP and transputer modules and special-purpose functions.

The A21B has three M-Module slots and supports the following M-Module characteristics: A08, A24, D16, D32, INTA, INTC, TRIGI, TRIGO.

2.9.1 Connection

The signals from the CPU board are fed to the M-Module via three 20-pin plug connector rows. These connectors correspond to connectors on the M-Module. The pin assignment corresponds to the M-Module specification (see Chapter 6.1 Literature and Web Resources on page 106).

Table 9. *Pin assignment of the 60-pin M-Module plug connectors (A21B)*



	A	B	C
1	CS#	GND	AS#
2	A01	+5V	D16
3	A02	+12V	D17
4	A03	-12V	D18
5	A04	GND	D19
6	A05	DREQ#	D20
7	A06	DACK#	D21
8	A07	GND	D22
9	D08/A16	D00/A08	TRIGA
10	D09/A17	D01/A09	TRIGB
11	D10/A18	D02/A10	D23
12	D11/A19	D03/A11	D24
13	D12/A20	D04/A12	D25
14	D13/A21	D05/A13	D26
15	D14/A22	D06/A14	D27
16	D15/A23	D07/A15	D28
17	DS1#	DS0#	D29
18	DTACK#	WRITE#	D30
19	IACK#	IRQ#	D31
20	RESET#	SYSCLK	DS2#

M-Module I/O signals are also led to the VMEbus P2 connector for rear I/O connection using the 24-pin M-Module P2 connector. The pin-out depends on the implementation of the respective M-Module. Please see the M-Module's user manual for more information. Table 26, Pin assignment of VMEbus rear I/O connector P2 – A21B – M-Modules on page 74 gives the assignment of pins from the M-Module's 24-pin P2 connector to VMEbus P2.

2.9.2 Addressing the M-Modules

The processor can address M-Modules via a dedicated interface implemented in the FPGA. The three M-Modules are mapped within the PCI target using PCI BAR 0 as shown in the following table. The address determines the access mode in which the respective M-Module is addressed. The address spaces mirrored in the address map can be accessed in each range.

The M-Module access data width depends on the access initiated by the CPU. Byte accesses will result in D8, word accesses in D16, long-word accesses in D32 M-Module accesses. If a long-word access is performed to A08 mode window, the controller will separate it to two D16 M-Module accesses.

The address is the value of BAR 0 (e.g., `0x F000 0000`) plus the offset (e.g., `0x 1705` for the Trigger Data Register), e.g., resulting in address `0x F000 1705`.

The interrupt of each M-Module can be handled in the Interrupt Control Register. The interrupts of all M-Modules are summarized in the bridge as the PCI interrupt of this target device. M-Modules use MSI vector 1 for interrupts.

Table 10. *M-Module address map (A21B)*

M-Module Slot	Mode	Offset Address Range	Size	Function
0	A08	0x 0000 1400..0000 14FF	256 bytes	M-Module Slot 0 A08 D8/D16
		0x 0000 1500..0000 1503	4 bytes	M-Module Slot 0 IACK access
		0x 0000 1701		Interrupt Control Register
		0x 0000 1705		Trigger Data Register
		0x 0000 1707		Trigger Direction Register
1	A08	0x 0000 1800..0000 18FF	256 bytes	M-Module Slot 1 A08 D8/D16
		0x 0000 1900..0000 1903	4 bytes	M-Module Slot 1 IACK access
		0x 0000 1B01		Interrupt Control Register (mirrored)
		0x 0000 1B05		Trigger Data Register (mirrored)
		0x 0000 1B07		Trigger Direction Register (mirrored)
2	A08	0x 0000 1C00..0000 1CFF	256 bytes	M-Module Slot 2 A08 D8/D16
		0x 0000 1D00..0000 1D03	4 bytes	M-Module Slot 2 IACK access
		0x 0000 1F01		Interrupt Control Register (mirrored)
		0x 0000 1F05		Trigger Data Register (mirrored)
		0x 0000 1F07		Trigger Direction Register (mirrored)
0	A24	0x 0100 0000..01FF FFFF	16 MB	M-Module Slot 0 A24 D8/D16/D32
1	A24	0x 0200 0000..02FF FFFF	16 MB	M-Module Slot 1 A24 D8/D16/D32
2	A24	0x 0300 0000..03FF FFFF	16 MB	M-Module Slot 2 A24 D8/D16/D32

Interrupt Control Register (0x1701, 0x1B01, 0x1F01) (read/write)

15..4	3	2	1	0
-	TIME_ OUT	-	IEN	IRQ

TIME_OUT Time-out of M-Module access
1 = Time-out has occurred. Write 1 to clear.

IEN Interrupt enable bit
0 = Disable interrupt
1 = Enable interrupt

IRQ Interrupt pending
1 = Interrupt pending (reflects active MSI interrupt vector number 1)

Trigger Data Register (0x1705, 0x1B05, 0x1F05) (read/write)

15..2	1	0
-	TRIGB _DAT	TRIGA _DAT

TRIGx_DAT Trigger A or B data output/input value
0 = With *TRIGx_DIR*=0: *TRIGx* was detected as 0.
With *TRIGx_DIR*=1: The value driving on *TRIGx* is 0.
1 = With *TRIGx_DIR*=0: *TRIGx* was detected as 1.
With *TRIGx_DIR*=1: The value driving on *TRIGx* is 1.

Trigger Direction Register (0x1707, 0x1B07, 0x1F07) (read/write)

15..2	1	0
-	TRIGB _DIR	TRIGA _DIR

TRIGx_DIR Trigger A or B direction
0 = *TRIGx* signal will be read
1 = *TRIGx* signal will be driven with the value of *TRIGx_DAT*

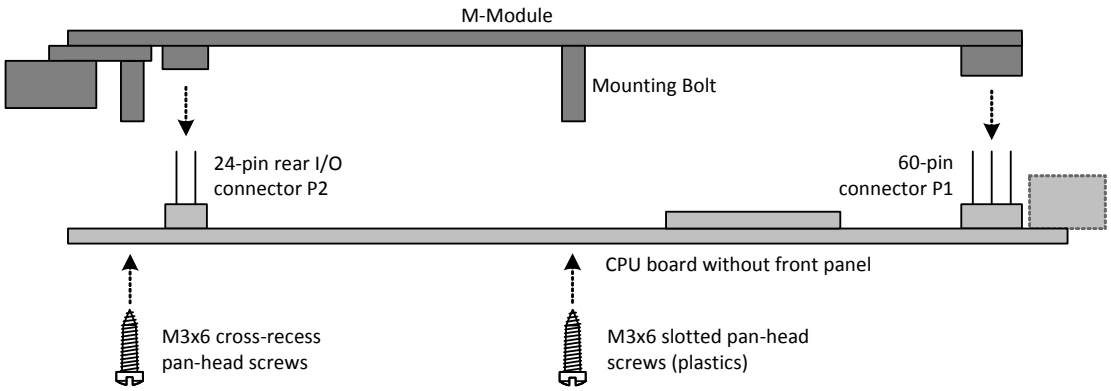
2.9.3 Installing an M-Module Mezzanine Module

Perform the following steps to install an M-Module:

- ☑ Power down your system and remove the CPU board from the system.
- ☑ Remove the filler panel from the board's front M-Module slot, if installed.
- ☑ Loosen the two front-panel mounting screws at the bottom side of the CPU board and remove the whole front panel (see Figure 1, Map of the board – A21B front panel and top view (M-Modules), on page 18).
- ☑ Hold the M-Module over the target slot of the CPU board with the component sides facing each other.
- ☑ Align the 60-pin connectors of the M-Module and carrier board.
- ☑ Press the M-Module carefully but firmly onto the CPU board, making sure that the connectors are properly linked.
- ☑ Turn the CPU board upside down and use four M-Module mounting screws to fasten the M-Module on the solder side of the board.
- ☑ Re-install the front panel.

You can order suitable mounting screws from MEN. Please see the A21B and A21C pages on MEN's website for ordering options.

Figure 5. *Installing an M-Module mezzanine module (A21B)*

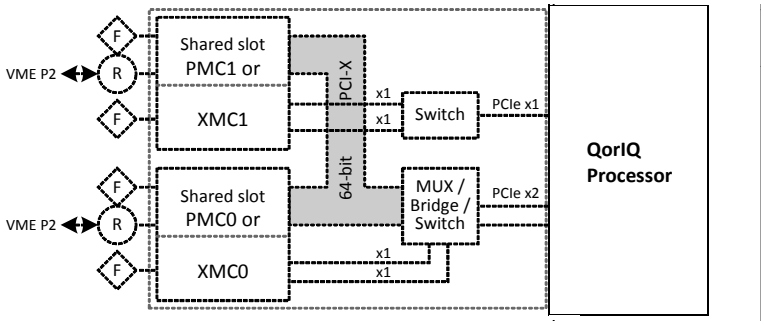


2.10 PMC/XMC Slots (A21C)

The A21C provides two PMC and two XMC mezzanine slots on shared sites.

Both the PMC and XMC slots are connected to the processor using PCI Express links (supporting PCIe 1.0a) to allow fast data rates for both module types. Please see Chapter 2.10.1 PMC Slots on page 39 on how you can combine the use of both module types and which data rates can be achieved.

Figure 6. *PCI Express connection of PMC and XMC slots (A21C)*



2.10.1 PMC Slots

The A21C provides two PMC slots for extension such as graphics, Fast Ethernet, SCSI etc. The market offers lots of different PMC mezzanines.

The signaling voltage is set to 3.3 V, i. e. the CPU board has a 3.3-V voltage key (see Figure 7, Installing a PMC mezzanine module (A21C), on page 40) and can only carry PMC mezzanines that support this keying configuration. Mezzanine cards may be designed to accept either or both signaling voltages (3.3 V / 5 V).

The PMC slots support 32-bit and 64-bit PCI-X bus operation at 33 MHz, 66 MHz or 100 MHz.

The connector layout is fully compatible to the IEEE1386 specification, including PCI-X capability. For connector pinouts please refer to the specification (see Chapter 6.1 Literature and Web Resources on page 106).

See Chapter 2.10.3 Support of Combined PMC/XMC Usage on page 44 for support of different PMC/XMC configurations and resulting data rates.

2.10.1.1 Connection

Connector types:

- 64-pin, 1-mm pitch board-to-board receptacle according to IEEE 1386
- Mating connector:
  64-pin, 1-mm pitch board-to-board plug according to IEEE 1386

Both slots support rear I/O via VMEbus connector P2, see Table 28, Pin assignment of VMEbus rear I/O connector P2 – A21C – PMC on page 76.

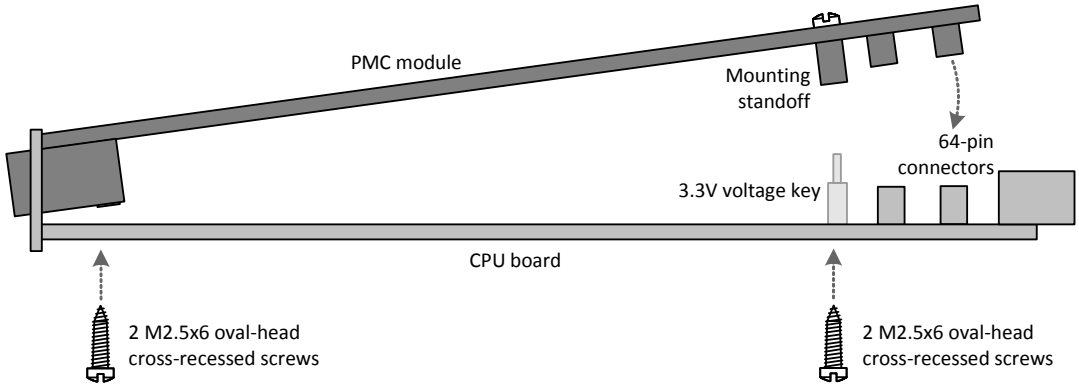
2.10.1.2 Installing a PMC Mezzanine Module

Perform the following steps to install a PMC module:

- ☑ Make sure that voltage keying of your PMC module matches the CPU board.
- ☑ Power down your system and remove the CPU board from the system.
- ☑ Remove the filler panel from the board's front PMC slot, if installed.
- ☑ The PMC module is plugged on the board with the component sides of the PCBs facing each other.
- ☑ Put the PMC module's front connector through the front slot at a 45° angle.
- ☑ Carefully put it down, making sure that the connectors are properly aligned.
- ☑ Press the PMC module firmly onto the board.
- ☑ Make sure that the EMC gasket around the PMC front panel is properly in its place.
- ☑ Screw the PMC module tightly to the CPU board at the bottom side of the PCB using four oval-head cross-recessed screws of type M2.5x6.

You can order suitable mounting screws from MEN. Please see the A21B and A21C pages on MEN's website for ordering options.

Figure 7. *Installing a PMC mezzanine module (A21C)*



2.10.2 XMC Slots

The A21C board provides two single-width XMC slots for extension such as high-speed graphics, Ethernet etc.

XMC modules have the same form factor as PMC modules, however they do not use a PCI bus but a high-speed PCI Express connection and therefore have a different carrier board connector.

The pinout of the XMC connectors conforms with VITA 42.3 for dual-lane PCIe interfaces. Both XMC slots offer two connectors with two x1 PCI Express links each for data rates up to 500 MB/s. (See also Chapter 6.1 Literature and Web Resources on page 106.)

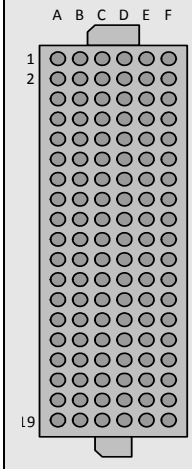
See Chapter 2.10.3 Support of Combined PMC/XMC Usage on page 44 for support of different PMC/XMC configurations and resulting data rates.

2.10.2.1 Connection

Connector types:

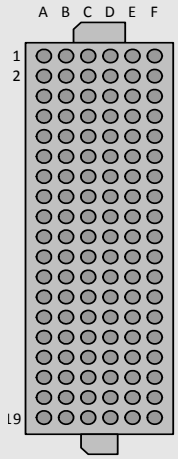
- 114-pin XMC receptacle connector
- Mating connector:
  114-pin XMC plug connector, e. g. SAMTEC ASP105885-01

Table 11. *Pin assignment of 114-pin XMC connectors J15/J25 (A21C)*



	A	B	C	D	E	F
1	PER0p0	PER0n0	+3.3V	-	-	+5V
2	GND	GND	-	GND	GND	MRSTI#
3	-	-	+3.3V	-	-	+5V
4	GND	GND	-	GND	GND	MRSTO#
5	-	-	+3.3V	-	-	+5V
6	GND	GND	-	GND	GND	+12V
7	-	-	+3.3V	-	-	+5V
8	GND	GND	-	GND	GND	-12V
9	-	-	-	-	-	+5V
10	GND	GND	-	GND	GND	GA0
11	PET0p0	PET0n0	-	-	-	+5V
12	GND	GND	GA1	GND	GND	MPRESENT#
13	-	-	+3.3V	-	-	+5V
14	GND	GND	GA2	GND	GND	MSDA
15	-	-	-	-	-	+5V
16	GND	GND	MVMRO	GND	GND	MSCL
17	-	-	-	-	-	-
18	GND	GND	-	GND	GND	-
19	REFCLKAp	REFCLKAn	-	WAKE#	ROOT0#	-

Table 12. *Pin assignment of 114-pin XMC connector J16/J26 (A21C)*



	A	B	C	D	E	F
1	PER1p0	PER1n0	-	-	-	-
2	GND	GND	-	GND	GND	-
3	-	-	-	-	-	-
4	GND	GND	-	GND	GND	-
5	-	-	-	-	-	-
6	GND	GND	-	GND	GND	-
7	-	-	-	-	-	-
8	GND	GND	-	GND	GND	-
9	-	-	-	-	-	-
10	GND	GND	-	GND	GND	-
11	PET1p0	PET1n0	-	-	-	-
12	GND	GND	-	GND	GND	-
13	-	-	-	-	-	-
14	GND	GND	-	GND	GND	-
15	-	-	-	-	-	-
16	GND	GND	-	GND	GND	-
17	-	-	-	-	-	-
18	GND	GND	-	GND	GND	-
19	REFCLKBp	REFCLKBn	-	-	-	-

Table 13. *Signal mnemonics of 114-pin XMC connectors J15/J16 (A21C)*

	Signal	Direction	Function
Power	+12V, -12V	out	+12V supply voltage
	+3.3V	out	+3.3V supply voltage
	+5V (VPWR)	out	+5V supply voltage
	GND	-	Ground
PCIe	PER[0:1]p/n0	out	PCI Express x1 link 0 and 1, differential receive
	PET[0:1]p/n0	in	PCI Express x1 link 0 and 1, differential transmit
	REFCLK[A:B]p/n	out	Differential reference clock, link 0 (A) and 1 (B)
	ROOT0#	out	Root Complex enabling, link 0
	WAKE#	out	Reactivation of power rails and reference clocks, link 0
Other	GA[0..2]	out	I2C channel select
	MBIST#	in	XMC built-in self test
	MPRESENT#	in	Module present. Allows the A21 to determine whether an XMC is present.
	MRSTI#	out	XMC reset in
	MRSTO#	in	XMC reset out (not used)
	MSCL	out	SMBus clock
	MSDA	in/out	SMBus data
	MVMRO	out	XMC EEPROM write prohibit

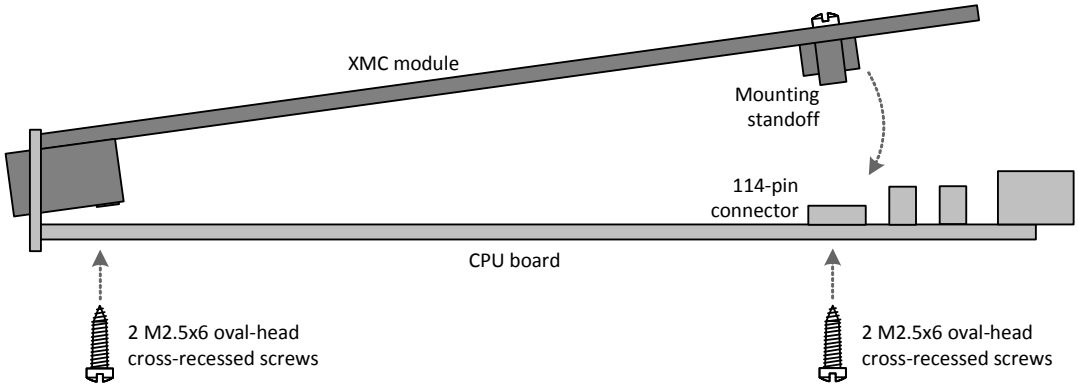
2.10.2.2 Installing an XMC Mezzanine Module

Perform the following steps to install an XMC module:

- ☑ Power down your system and remove the CPU board from the system.
- ☑ Remove the filler panel from the board's front XMC slot, if installed.
- ☑ The XMC module is plugged on the CPU board with the component sides of the PCBs facing each other.
- ☑ Put the XMC module's front connector through the CPU board's front slot at a 45° angle.
- ☑ Carefully put it down, making sure that the connectors are properly aligned.
- ☑ Press the XMC module firmly onto the CPU board.
- ☑ Make sure that the EMC gasket around the XMC front panel is properly in its place.
- ☑ Screw the XMC module tightly to the CPU board using the two mounting standoffs and four matching oval-head cross-recessed screws of type M2.5x6.

You can order suitable mounting screws from MEN. Please see the A21B and A21C pages on MEN's website for ordering options.

Figure 8. *Installing an XMC mezzanine module (A21C)*



2.10.3 Support of Combined PMC/XMC Usage

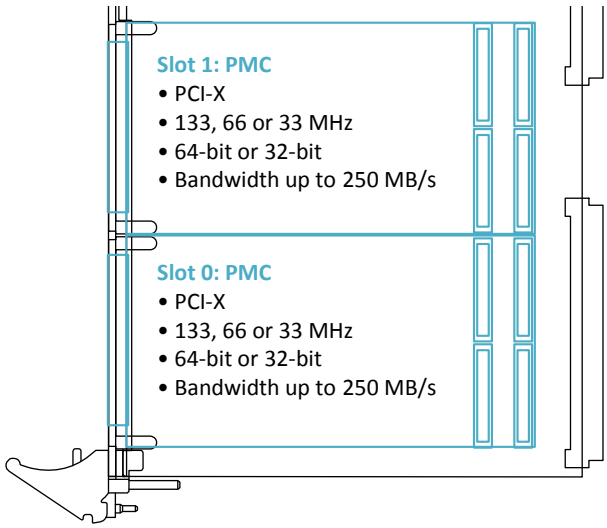
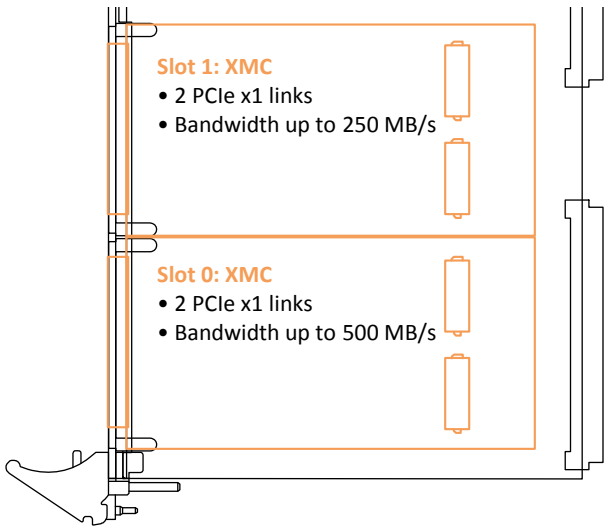
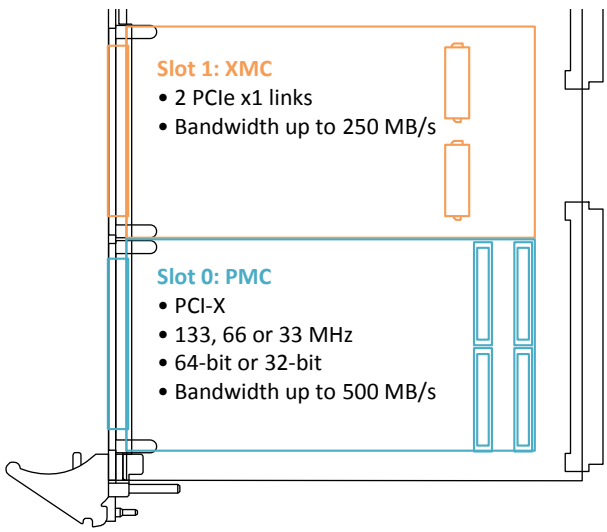
Technically, the A21C has four mezzanine slots with different features regarding connection speeds and modes of operation. You can combine one PMC and XMC module, each, but not in any desired slot.

When a PMC and XMC are combined in the two slots, the board automatically configures its internal routing accordingly.

Please note that, because of internal PCIexpress sharing, it is not possible to combine an XMC module in slot 0 and a PMC module in slot 1.

The following table gives an overview of the supported combination possibilities.

Table 14. *Supported PMC/XMC Combinations and Characteristics*

2 PMCs	
2 XMCs	
1 PMC, 1 XMC	

2.11 I/O Extension

Apart from its standardized mezzanine slots, the A21 also offers the option of integrating a completely customizable mezzanine card. The card takes the place of one of the PMC/XMC or M-Module slots and can include front I/O.

A number of different interfaces are already routed to a prepared connector, i.e. the signals would always be implemented with a mezzanine card connector. These interfaces comprise:

- One USB 2.0 port, EHCI implementation
- Additional UART COM interface
- Display interface

The A21 also provides the option of integrating customized functions in its onboard FPGA. 16 pins are available for this, supported by an Altera Cyclone IV device. It is connected directly to the CPU via a PCIexpress x1 link (supporting PCIe 1.0a).

While some interface types are prepared already, the implementation of I/O connectors remains free for the user's requirements.

The position, connectors and mounting facilities for the mezzanine card are already fixed, too. They differ slightly for A21B and A21C. Please see Chapter 1.1 Map of the Board.

For more information and custom solutions please contact MEN's sales team.
You can find more details on FPGA technology on our web page "User I/O in FPGA".

2.12 Reset Button and Status LEDs

2.12.1 Front Panel

The A21 has a reset button and two general status LEDs at the front panel.

The reset button is marked "R" and is recessed within the front panel. It requires a tool, e.g., a paper clip to be pressed, preventing the button from being inadvertently activated.

The status LEDs are marked "1" and "2". "1" is a red/green bicolor type, "2" is yellow. The following status indications are preset:

Table 15. *Status indications by front-panel LEDs*

LED	Indication	Description
1	Red on	Reset active; LED remains on while the reset is active (approx. 1 second)
	Green flashing	Heartbeat
2	Yellow flashing	Application problem
	Yellow on	Watchdog expired

The two LEDs are controlled through GPIOs of the processor. This allows further flashing codes for detailed user information about the status of the board. The board support package usually supports control of the LEDs. Please refer to your operating system BSP documentation for more information.

2.12.2 Onboard Development LED

For system integration or software development purposes, the A21 also has a bicolored onboard LED related to the FPGA. The LED signals two independent functions.

Table 16. *Status indications by bicolored onboard development LED*

Indication	Description
Green on	VMEbus slot1 was detected. LED stays on permanently.
Red on	PCIe link from CPU to FPGA is being established. The LED normally lights up briefly and then goes out again when the link is up. If the LED is active for a longer time or repeatedly, this indicates that the link between the CPU and FPGA is not stable.

Note: If both colors are active, they appear as a mixture between green and red.

Figure 9. *Position of onboard development LED on A21B*

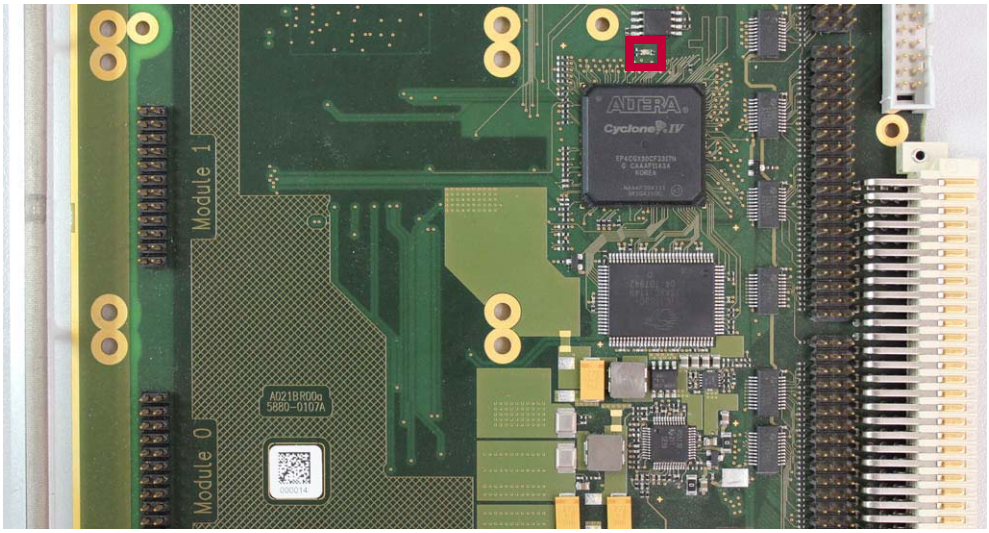
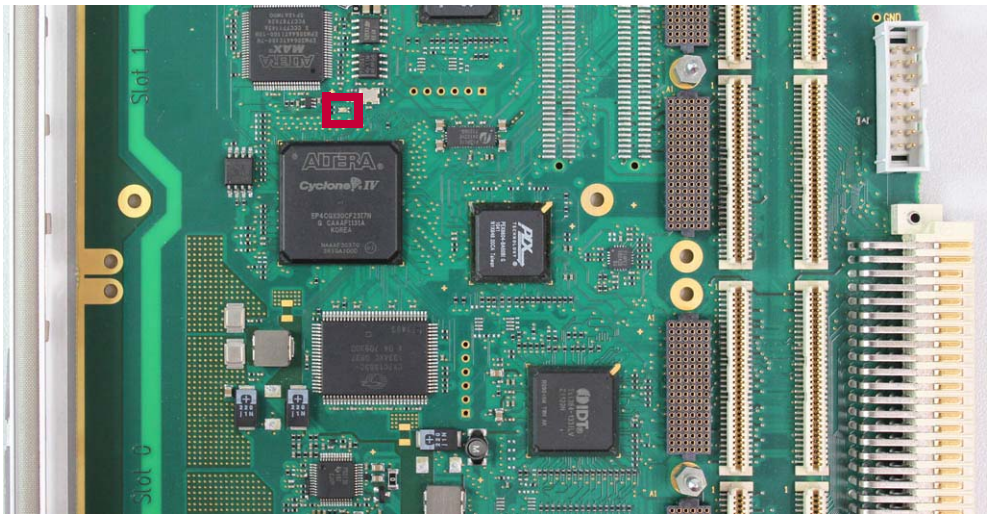


Figure 10. *Position of onboard development LED on A21C*



2.13 VMEbus Interface

The A21's VMEbus interface conforms to the VMEbus specification. It has the following features:

- Slot-1 functionality with auto-detection
- Wide range of VMEbus address and data transfer modes
  - Master D08(EO):D16:D32:D64:A16:A24:A32:BLT:RMW
  - Slave D08(EO):D16:D32:D64:A16:A24:A32:BLT:RMW
- Interrupter: 7-level, D08(O):I(7-1):ROAK
- Interrupt handler: 7-level, D08(O):IH(7-1)
- Single level 3 fair requester
  - ROR (release on request)
  - RWD (release when done)
- Single level 3 arbiter
- Multi-master capabilities
- BTO bus timeout
- ADO address only cycles
- Mailbox functionality
- Location monitor A16, A24, A32
- DMA controller with scatter-gather capabilities (A32/D64)
- Dual-ported system registers
- Utility functions
- Access to 1 MB dual-ported fast SRAM or to PCI space

2.13.1 PCI Configuration Space Registers

The Configuration Registers from 0x00 to 0x3C conform with the PCI Device Configuration Header Format.

Table 17. *VMEbus interface PCI configuration space registers*

Address	Byte			
	3	2	1	0
0x00	Device ID (0x4D45)		Vendor ID (0x1A88)	
0x04	Status Register		Command Register	
0x08	Class Code (0x068000)			Revision ID (currently 0x01)
0x0C	BIST	Header Type	Latency Timer	Cache Line Size
0x10	Base Address Register 0			
0x14	Base Address Register 1			
0x18	*This register is always 0.*			
0x1C	*This register is always 0.*			
0x20	*This register is always 0.*			
0x24	*This register is always 0.*			
0x28	Card Bus CIS Pointer			
0x2C	Subsystem Device ID (0x5A91)		Subsystem Vendor ID (0xAB)	
0x30	Expansion ROM Base Address Register			
0x34	Reserved			
0x38	Reserved			
0x3C	Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line

2.13.2 Runtime Registers

The registers are accessible both from the PCI and the VMEbus side.

From the PCI-bus side the address is the value of BAR 0 (e.g., 0x F000 0000) plus an offset (0x 0080 0000) plus the register address (e.g., 0x 0008 for ISTAT), e.g. resulting in address 0x F080 0008.

From the VMEbus side, the registers are accessible in A16 mode only. (E.g. slave base address A16 = 0x00001000, plus register address).

The Mailbox Data Registers and the DMA Buffer Descriptors are stored in the SRAM and can be accessed via the offset address of the SRAM. See Chapter 2.13.3 VMEbus Master Mapping on page 52, Chapter 2.13.4 VME Slave Mapping on page 53, Chapter 2.13.5 SRAM on page 53 and Table 19, Mailbox and DMA registers in SRAM, on page 52.

Note: If an application uses the whole SRAM space, the DMA and mailbox functions will not work!

Table 18. *VMEbus bridge control registers*

Address	D31..D0
0x 0000	INTR – VME Interrupt Control Register (r/w)
0x 0004	INTID – VME Interrupt STATUS/ID Register (r/w)
0x 0008	ISTAT – Interrupt Status Register (r)
0x 000C	IMASK – Interrupt Mask Register (r/w)
0x 0010	MSTR – Master Control Register (r/w)
0x 0014	SLV24 – Slave Control Register A24 for SRAM Access (r/w)
0x 0018	SYSCTL – System Control Register (r/w)
0x 001C	LONGADD – Upper 3 Address Bits for A32 (r/w)
0x 0020	MAIL_IRQE – Mailbox Interrupt Enable Register (r/w)
0x 0024	MAIL_IRQ – Mailbox Interrupt Request Register (r/w)
0x 0028	PCI_OFFSET – PCI Offset Address for VME to PCI Access (r/w)
0x 002C	DMASTA – DMA Status Register (r/w)
0x 0030	SLV16 – Slave Control Register A16 for SRAM Access (r/w)
0x 0034	SLV32 – Slave Control Register A32 for SRAM Access (r/w)
0x 0038	LOCSTA_0 – Location Monitor Status Register (r/w)
0x 003C	LOCSTA_1 – Location Monitor Status Register (r/w)
0x 0040	LOCADDR_0 – Location Monitor Address Register (r/w)
0x 0044	LOCADDR_1 – Location Monitor Address Register (r/w)
0x 0048	SLV24_PCI – A24 Slave Base Address for PCI (r/w)
0x 004C	SLV32_PCI – A32 Slave Base Address for PCI (r/w)

Table 19. *Mailbox and DMA registers in SRAM*

Address	D31..D0
0x FF800	MAILBOX_0 – Mailbox Data Register (r/w)
0x FF804	MAILBOX_1 – Mailbox Data Register (r/w)
0x FF808	MAILBOX_2 – Mailbox Data Register (r/w)
0x FF80C	MAILBOX_3 – Mailbox Data Register (r/w)
0x FF900..FF90C	DMA_BD#1 – DMA Buffer Descriptor (r/w)
0x FF910..FF91C	DMA_BD#2 – DMA Buffer Descriptor (r/w)
.. 0x FFFF0..FFFFF	Further Buffer Descriptors

2.13.3 VMEbus Master Mapping

The PCI to VMEbus bridge uses BAR0 to access the VMEbus. BAR1 is used for VME A32/D32 accesses.

Table 20. *VMEbus master address map*

PCI BAR	Block	Address Offset (BAR view)	Size	Function
0	VME A16	0x0001 0000..0001 FFFE	64 KB	VME A16/D16 (short) space
		0x0002 0000..0002 FFFC	64 KB	VME A16/D32 (short) space
	SRAM	0x0040 0000..004F FFFF	1 MB	Local SRAM
		0x004F F800	4 bytes	MAILBOX_0 register
		0x004F F804	4 bytes	MAILBOX_1 register
		0x004F F808	4 bytes	MAILBOX_2 register
		0x004F F80C	4 bytes	MAILBOX_3 register
		0x004F F900..004F F90F	16 bytes	DMA_BD#1 buffer descriptor
		0x004F F920..004F F91F	16 bytes	DMA_BD#2 buffer descriptor
		0x004F F920..004F F92F	16 bytes	DMA_BD#3 buffer descriptor
		...		
		0x004F FFF0..004F FFFF	16 bytes	DMA_BD#256 buffer descriptor
	Control Registers	0x0080 0000..0080 004F	80 bytes	VMEbus bridge control registers (see Table 18, VMEbus bridge control registers on page 51)
	VME IACK	0x00C0 0000	16 bytes	VME IACK space
	VME A24	0x0400 0000..04FF FFFE	16 MB	VME A24/D16 (standard) space
		0x0500 0000..05FF FFFC	16 MB	VME A24/D32 (standard) space
1	VME A32	0x0000 0000..1FFF FFFC	512 MB	VME A32/D32 (long) space; offset can be changed in register LONGADD in order to access the entire 4-GB address space

2.13.4 VME Slave Mapping

The PCI space, 1 MB SRAM and all registers are accessible by the VMEbus. The registers are a 4-kB block, which is accessible in A16 mode only. This block can be mapped to the slave base address for A16 mode. The SRAM can be mapped to the A24 and A32 base address, whereas the SRAM is mirrored in A32 mode. The PCI space can also be mapped to the A24 and A32 space. The size of the A24/A32 windows is configurable in the corresponding Mask Registers.

Table 21. *VMEbus slave address windows*

VME Slave Access	Address	Size	Function
A16	0x 000..04F	72 bytes	VME Bridge Control Registers
A24	0x 0 0000..F F7FF	1022 KB	Local SRAM
	0x F F800..F FFFC	2 KB	Mailbox Data Registers and DMA Buffer Descriptors (included in SRAM)
	0x 0 0000..F FFFC	64 KB.. 1 MB	PCI Space
A32	0x 0 0000..F F7FF	1022 KB	Local SRAM
	0x F F800..F FFFC	2 KB	Mailbox Data Registers and DMA Buffer Descriptors
	0x 0000 0000..000F FFFC	1 MB.. 256 MB	PCI Space

Note: The registers for mailbox and DMA are located in the SRAM (0xF F800 to 0xF FFFF). If an application uses the whole SRAM space, the DMA and mailbox functions will not work!

2.13.5 SRAM

The SRAM is accessible from the PCI and VMEbus side. From the VMEbus side, the base address is defined in the Slave Control Registers for A24 or A32 (in A32 mode the SRAM is mirrored!).

The SRAM has a size of 1 MB. It is accessible via block and standard transfers (user and supervisor space).

2.13.6 Slot-1 Function

The A21 supports slot 1 functionality with auto-detection upon power-up. For this, the bus arbitration daisy chain is used, in order to examine the location of the board within the VMEbus. The slot 1 status can be read from the SYSCTL register (bit *SYSCON*).

If the A21 samples *BG3IN#* after *SYSRES#* has gone high, the slot 1 function is enabled. As specified in the VMEbus specification all *BG* lines should be high after reset. Therefore *BG3IN#* can be sampled low only in slot 1.

After slot 1 is detected the functions listed below are enabled.

- Generation of *SYSRES#*
- Generation of *SYSCLK* (not provided during slot 1 detection cycle)
- Level 3 arbitration
- Bus arbitration timeout 250 µs
- Bus transfer timeout 60 µs (indicated by assertion of signal *BERR#*)

The timeout values cannot be changed.

SYSCTL – System Control Register (0x0018) (read/write)

7..3	2	1	0
-	ATO	SYSRES	SYSCON

ATO Monitor for arbitration timeout signal (read only)

0 = No arbitration timeout (default)
1 = Arbitration timeout occurred (cleared by writing 1)

SYSRES Reset VMEbus; generates *SYSRES#* on the VMEbus

0 = *SYSRES#* active (can be activated when *SYSCON*=1)
1 = *SYSRES#* not active

SYSCON System controller status (depends on slot 1 auto-detection after reset). May be overridden by software. Independent of the *SYSCON* bit, a *SYSRES* on the VMEbus will result in a reset, which will be routed to the local CPU.

0 = Read: Slot 1 function disabled
Write: Overwrites slot 1 auto detection result and disables slot 1 functions
1 = Read: Slot 1 function enabled
Write: Overwrites slot 1 auto detection result and disables slot 1 functions

2.13.7 VMEbus Master Interface

The VMEbus controller supports several access types to the VMEbus. Depending on the type of backplane connection, the data width can be chosen between D8/D16 (one VME connector) and D32 (two VME connectors). D64 transfers are supported by the DMA, when two VME connectors are used.

The VMEbus is separated into several address spaces which can be accessed by the VMEbus controller: A16, A24 and A32.

If one of the VME master windows is accessed by a read or write, the VMEbus access will be requested on level 3. If the bus was granted by activation of the arbitration daisy chain input (signal *BG3IN#*), the VME master will perform the access.

The **bus requester schemes** 'Release When Done' and 'Release On Request' can be selected (VMEbus Release Mechanism).

Access type '**Read-Modify-Write**' can be selected by setting the *RMW* bit before the access. If the read access to one VME master window is performed then, the bus will be blocked until the write access follows. No other VME master can perform any VME transmission between these two accesses!

Access type '**Address Only**' can be selected by setting the *ADO* bit before a byte or word read access follows. During this access, no data will be transmitted, but VMEbus participants can monitor (e.g., through the Location Monitor) the bus and can detect this message. No other access types are allowed if the *ADO* bit is set.

If a **bus error** occurs, bit *BERR* is set. An interrupt is triggered if the *IBERREN* bit is set. The timeout for *BERR* is 60 µs. The bus error can be cleared by writing 1 to the *BERR* bit.

MSTR – Master Control Register (0x0010) (read/write)

7..6	5	4	3	2	1	0
-	ADO	POSTWR	IBERREN	BERR	REQ	RMW

ADO Enable single address only cycle

0 = Normal cycle (default)
1 = Adress only cycle

The master does not transfer data with this cycle. Other slaves can detect this cycle to generate local interrupts.
Only 8-bit or 16-bit read cycles are allowed for address-only cycles!
This bit will be reset after the transmission was done.

POSTWR Posted Write Access to VMEbus (not supported)

0 = Delayed write access to VMEbus (default)
1 = Posted write access to VMEbus

IBERREN Interrupt Bus Error Enable

0 = Local interrupt disabled if VMEbus bus error occurs (default)
1 = Local interrupt enabled if VMEbus bus error occurs

BERR Monitor for VMEbus *BERR#* signal

0 = No VMEbus error (default)
1 = VMEbus error occurred; *BERR#* signal asserted. Cleared by writing 1.

REQ Set VMEbus requester scheme

0 = Request scheme for VMEbus master and interrupt handler is set to Release On Request (ROR) (default)
1 = Request scheme is set to Release When Done (RWD)

If this bit is changed from 0 to 1, i. e. from ROR to RWD, and there were previous accesses over the master interface, it is recommended to do a dummy read to free the bus.

RMW Enable single Read-Modifiy-Write cycle

0 = Normal cycle (default)
1 = RMW cycle. Master keeps *AS#* asserted during back-to-back read/ write cycle.

This bit is automatically cleared after the RMW cycle and must be set for the next RMW cycle again.

The following master/slave RMW accesses are allowed:

- Byte or word access in D16 mode
- Byte, word or long access in D32 mode

Note: During RMW cycles all interrupts on the host CPU should be masked.

In order to do a RMW cycle, the right order must be considered: First set the RMW bit in the MSTR register. Now, a read transaction on VMEbus can be done (no long access on D16 spaces!). When the read was done, the bus is blocked until the write access will be performed. In order not to interfere with the RMW order, all other VME master actions should be blocked by software (DMA should not be active!).

Transfers in D32 mode are done within one VME cycle. If a D32 access is performed to a D16 address space, the VMEbus master will do two D16 accesses in order to transmit 32 bits. These two accesses are not dividable by other VME masters!

The following table lists all valid combinations for byte enables supported by the bridge. Note that even within a PCI burst transfer the byte enables may change from one data portion to the next. This must be taken into account when exchanging data with the SRAM.

Table 22. *VMEbus interface valid combinations for byte enables supported by PCI-to-VME bridge*

PCI Bus (always dword aligned address)				Byte Lane Mapping 32-bit PCI 64-bit VME	VMEbus (always word aligned address)			
Byte Enables								
3	2	1	0		DS1	DS0	A1	A0[1]
D16								
1	1	1	1	No transfers	x	x	x	x
1	1	1	0	D7..D0 ⇔ D15..D8	0	1	0	1
1	1	0	1	D15..D8 ⇔ D7..D0	1	0	0	1
1	0	1	1	D23..D16 ⇔ D15..D8	0	1	1	1
0	1	1	1	D31..D24 ⇔ D7..D0	1	0	1	1
1	1	0	0	D7..D0 ⇔ D15..D8 D15..D8 ⇔ D7..D0	0	0	0	1
0	0	1	1	D23..D16 ⇔ D15..D8 D31..D24 ⇔ D7..D0	0	0	1	1
1	0	1	0	First: D23..D16 ⇔ D15..D8 Second: D7..D0 ⇔ D15..D8	0 0	1 1	1 0	1
0	1	0	1	First: D31..D24 ⇔ D7..D0 Second: D15..D8 ⇔ D7..D0	1 1	0 0	1 0	1
0	1	1	0	First: D31..D24 ⇔ D7..D0 Second: D7..D0 ⇔ D15..D8	1 0	0 1	1 0	1
1	0	0	1	First: D23..D16 ⇔ D15..D8 Second: D15..D8 ⇔ D7..D0	0 1	1 0	1 0	1
1	0	0	0	First: D23..D16 ⇔ D15..D8 Second: D7..D0 ⇔ D15..D8 D15..D8 ⇔ D7..D0	0 0	1 0	1 0	1
0	0	0	1	First: D23..D16 ⇔ D15..D8 D31..D24 ⇔ D7..D0 Second: D15..D8 ⇔ D7..D0	0 1	0 0	1 0	1
0	1	0	0	First: D31..D24 ⇔ D7..D0 Second: D7..D0 ⇔ D15..D8 D15..D8 ⇔ D7..D0	1 0	0 0	1 0	1
0	0	1	0	First: D31..D24 ⇔ D7..D0 D23..D16 ⇔ D15..D8 Second: D7..D0 ⇔ D15..D8	0 0	0 1	1 0	1
0	0	0	0	First: D23..D16 ⇔ D15..D8 D31..D24 ⇔ D7..D0 Second: D7..D0 ⇔ D15..D8 D15..D8 ⇔ D7..D0	0	0	1 0	1

PCI Bus (always dword aligned address)				Byte Lane Mapping 32-bit PCI 64-bit VME	VMEbus (always word aligned address)			
Byte Enables								
3	2	1	0		DS1	DS0	A1	A0[1]
D32								
1	0	0	1	D15..D8 ⇔ D23..D16 D23..D16 ⇔ D15..D8	0	0	1	0
0	0	0	1	D15..D8 ⇔ D23..D16 D23..D16 ⇔ D15..D8 D31..D24 ⇔ D7..D0	1	0	0	0
1	0	0	0	D7..D0 ⇔ D31..D24 D15..D8 ⇔ D23..D16 D23..D16 ⇔ D15..D8	0	1	0	0
0	0	0	0	D7..D0 ⇔ D31..D24 D15..D8 ⇔ D23..D16 D23..D16 ⇔ D15..D8 D31..D24 ⇔ D7..D0	0	0	0	0
D64								
0	0	0	0	D7..D0 ⇔ D63..D56 D15..D8 ⇔ D55..D48 D23..D16 ⇔ D47..D40 D31..D24 ⇔ D39..D32 D39..D32 ⇔ D31..D24 D47..D40 ⇔ D23..D16 D55..D48 ⇔ D15..D8 D63..D56 ⇔ D7..D0	0	0	0	0

[1] Signal *LWORD*.

2.13.7.1 Address Modifiers

Table 23. *VMEbus master Address Modifier codes*

Hex Code	AM						Function
	5	4	3	2	1	0	
0x08	L	L	H	L	L	L	A32 non-privileged 64-bit block transfer (MBLT)
0x09	L	L	H	L	L	H	A32 non-privileged data access
0x0B	L	L	H	L	H	H	A32 non-privileged 32-bit block transfer (BLT)
0x29	H	L	H	L	L	H	A16 non-privileged access
0x39	H	H	H	L	L	H	A24 non-privileged data access
0x3B	H	H	H	L	H	H	A24 non-privileged 32-bit block transfer (BLT)

2.13.7.2 Atomic Operations

CPU-to-SRAM Operations

Not supported.

CPU-to-VME Operations

Read-Modify-Write operations to the VMEbus can be done via bit *RMW* in the Master Control Register. Only byte and word accesses are allowed, with word accesses being made to even addresses.

VME-to-SRAM Operations

Supported.

2.13.8 VMEbus Slave Interface

The VMEbus controller supports 5 independent VME slave windows. These windows are disabled by default and can be enabled via register bits *SLENx* in the Slave Control Registers described below.

If a slave window is enabled, the base address must be set to an appropriate value. The window defined by base address and size must be unique on the VMEbus in order to prevent VMEbus signals driven by more than one slave!

Two slave windows (SLV24 and SLV32) are capable of accessing the local SRAM (VME slave base address = 0x0 of local SRAM). The local SRAM can be accessed from the CPU, DMA, mailbox and the VME slave, which must be well organized in order to prevent data mismatch.

One slave window (SLV16) is capable of accessing the VMEbus bridge control registers.

Two slave windows (SLV24_PCI and SLV32_PCI) are capable of accessing the PCI address space at an offset address defined in register PCI_Offset. The offset will be added to the VME address during each access to the PCI space.

RMW cycles are supported by the slave interface to the SRAM. During an RMW cycle to the SRAM, a PCI access to the SRAM is blocked.

PCI_Offset – PCI Offset Address for VME to PCI Access (0x0028) (read/write)

31..12	11..0
PCI_OFFSET[31:12]	0

PCI_OFFSET 4-kbyte aligned offset address on PCI bus for VME-to-PCI accesses.

The PCI space is accessible via A24 and A32 access on the VMEbus. To do this, the SLV24_PCI or SLV32_PCI control registers must be set.

SLV16 – Slave Control Register A16 for SRAM Access (0x0030) (read/write)

7..5	4	3..0
	SLEN16	SLBASE16

SLEN16 0 = Slave Unit disabled (default)
1 = Slave Unit enabled

SLBASE16 Slave's Base Address. Specifies the lowest address in the VME address range that will be decoded. Since only A[15:12] are monitored, the smallest possible address space is 4 KB.
Default: 0000

SLV24 – Slave Control Register A24 for SRAM Access (0x0014) (read/write)

15..12	11..8
SLMASK24[19:16]	SLBASE24[19:16]

7..5	4	3..0
	SLEN24	SLBASE24[23:20]

SLMASK24 Slave Base Address Mask Bits. The size of the A24 Slave window can be set to:

0000 = 1 MB
1000 = 512 KB
1100 = 256 KB
1110 = 128 KB
1111 = 64 KB

Default: 0000

SLEN24 0 = Slave Unit disabled (default)
1 = Slave Unit enabled

SLBASE24 Slave's Base Address. Specifies the lowest address in the VME address range that will be decoded. Since only A[23:16] are monitored, the smallest possible address space is 64 KB.
Default: 0000

SLV32 – Slave Control Register A32 for SRAM Access (0x0034) (read/write)

23..16
SLMASK32[27:20]

15..8
SLBASE32[27:20]

7..5	4	3..0
	SLEN32	SLBASE32[31:28]

SLMASK32 Slave Base Address Mask Bits. The size of the A32 Slave window can be set to:

00000000 = 256 MB
10000000 = 128 MB
11000000 = 64 MB
11100000 = 32 MB
11110000 = 16 MB
11111000 = 8 MB
11111100 = 4 MB
11111110 = 2 MB
11111111 = 1 MB

Default: 00000000

SLEN32 0 = Slave Unit disabled (default)
1 = Slave Unit enabled

SLBASE32 Slave's Base Address. Specifies the lowest address in the VME address range that will be decoded. Since only A[31:20] are monitored, the smallest possible address space is 1 MB.
Default: 000000000000

SLV24_PCI – Slave Control Register A24 for PCI Access (0x0048) (read/write)

15..12	11..8
SLMASK24_PCI[19:16]	SLBASE24_PCI[19:16]

7..5	4	3..0
	SLEN24_PCI	SLBASE24_PCI[23:20]

SLMASK24_PCI Slave Base Address Mask Bits. The size of the A24 PCI Slave window can be set to:
0000 = 1 MB
1000 = 512 KB
1100 = 256 KB
1110 = 128 KB
1111 = 64 KB
Default: 0000

SLEN24_PCI 0 = Slave Unit disabled (default)
1 = Slave Unit enabled

SLBASE24_PCI Slave's Base Address for PCI access. Specifies the lowest address in the VME address range that will be decoded. Since only A[23:16] are monitored, the smallest possible address space is 64 KB.
Default: 00000000

SLV32_PCI – Slave Control Register A32 for PCI Access (0x004C) (read/write)

23..16
SLMASK32_PCI[27:20]

15..8
SLBASE32_PCI[27:20]

7..5	4	3..0
	SLEN32_PCI	SLBASE32_PCI[31:28]

SLMASK32_PCI Slave Base Address Mask Bits. The size of the A32 PCI Slave window can be set to:

00000000 = 256 MB
10000000 = 128 MB
11000000 = 64 MB
11100000 = 32 MB
11110000 = 16 MB
11111000 = 8 MB
11111100 = 4 MB
11111110 = 2 MB
11111111 = 1 MB

Default: 00000000

SLEN32_PCI 0 = Slave Unit disabled (default)
1 = Slave Unit enabled

SLBASE32_PCI Slave's Base Address for PCI access. Specifies the lowest address in the VME address range that will be decoded. Since only A[31:20] are monitored, the smallest possible address space is 1 MB.
Default: 000000000000

2.13.8.1 Address Modifiers

Table 24. *VMEbus slave Address Modifier codes*

Hex Code	AM						Function
	5	4	3	2	1	0	
0x3F	H	H	H	H	H	H	A24 Standard supervisory block transfer
0x3D	H	H	H	H	L	H	A24 Standard supervisory data access
0x3B	H	H	H	L	H	H	A24 Standard non-privileged block transfer
0x39	H	H	H	L	L	H	A24 Standard non-privileged data access
0x3C	H	H	H	H	L	L	A24 supervisory 64-bit block transfer (MBLT)
0x38	H	H	H	L	L	L	A24 non-privileged 64-bit block transfer (MBLT)
0x29	H	L	H	L	L	H	A16 non-privileged access
0x2D	H	L	H	H	L	H	A16 supervisory data access
0x0F	L	L	H	H	H	H	A32 supervisory block transfer (BLT)
0x0D	L	L	H	H	L	H	A32 supervisory data access
0x0C	L	L	H	H	L	L	A32 supervisory 64-bit block transfer (MBLT)
0x0B	L	L	H	L	H	H	A32 non-privileged block transfer (BLT)
0x09	L	L	H	L	L	H	A32 non-privileged data access
0x08	L	L	H	L	L	L	A32 non-privileged 64-bit block transfer (MBLT)

2.13.9 VMEbus Requester

Bus requests are executed at bus request level 3. No other levels are available. The requester uses the Release-On-Request (ROR) or Release-When-Done (RWD) scheme. ROR should be preferred in single VMEbus master systems to increase the transfer rate. Both register schemes are implemented as fair requester.

Settings are made in the Master Control Register. See Chapter 2.13.7 VMEbus Master Interface on page 55.

Unused daisy-chain lines are passed by (*BG0IN#/OUT#, BG1IN#/OUT#, BG2IN#/OUT#*).

2.13.10 VMEbus Interrupt Handler

The VMEbus controller supports reception of VME interrupt requests on all seven levels. The Interrupt Mask Register (IMASK) is used to configure the interrupt handler. The interrupts are not prioritized.

If a VME interrupt is asserted on the bus and the handler is configured to receive this level, the request will be forwarded to the CPU. The IRQ service routine can read the VME interrupt level of the asserted IRQ in the Interrupt Status Register (ISTAT). The ISTAT register will only show interrupt requests that were enabled in IMASK!

In order to read the interrupt status/ID from the interrupter, the CPU must generate a read cycle to the IACK memory area, setting the least significant address bits A[3..1] to the corresponding interrupt level (e.g., for IRQ5# set [3..1] to [101]).

Naturally, there is no vector for *ACFAIL* interrupts. To reset such interrupts, write 1 to the *ACFST* bit in the Interrupt Status Register.

IMASK – Interrupt Mask Register (0x000C) (read/write)

7	6	5	4	3	2	1	0
IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	ACFEN

IENx If the corresponding interrupt pin *IRQx* is asserted, the bridge will signal an interrupt to the PCI side.

0 = Interrupt masked (default)
1 = Interrupt enabled

ACFEN When this bit is set, and *ACFAIL#* is detected, an interrupt on the PCI side is generated.

0 = Interrupt masked (default)
1 = *ACFAIL#* interrupt enabled

ISTAT – Interrupt Status Register (0x0008) (read/write)

7	6	5	4	3	2	1	0
I7	I6	I5	I4	I3	I2	I1	ACFST

Ix The PCI Master can read the asserted interrupts here. These are not stored here, they can just be read here. Interrupts are reset automatically if the external interrupter removes its request. Writes to these bits are ignored.

ACFST If this reads 1, an *ACFAIL#* interrupt has occurred. This interrupt is stored, because it is not static. It can be cleared by writing 1.
Default: 0

2.13.11 VMEbus Interrupter

The VMEbus controller can assert interrupt requests on the VMEbus on all seven levels, but only one at a time. The interrupt level is selected in the Interrupter Control Register. If the interrupt handler performs the IACK cycle in order to locate the source of the interrupt, the VME Interrupt STATUS/ID Register contents will be transmitted as an answer to the IACK cycle. The interrupt request is auto cleared after an IACK cycle (ROAK). The VMEbus interrupter uses MSI vector 2.

INTR – VME Interrupter Control Register (0x0000) (read/write)

7..4	3	2	1	0
-	INTEN	IL2	IL1	IL0

This register controls the internal interrupter. Interrupt levels from 1 to 7 can be set. The interrupt is generated only when the *INTEN* bit is set.

INTEN Interrupter enable

0 = Interrupt request accomplished (ROAK) (default)
1 = Interrupt request generated corresponding to the *ILx* bits. It should be set after the *ILx* bits are set to avoid glitches on the IRQ lines.

The interrupt level is set in binary code (e. g. *ILx*=011 is IRQ3).
Default: 0x0

ILx Interrupter request level

The level is set by a binary value of a 3-to-7 demultiplexer (e.g. IL[2:0]=011 is IRQ3).

Default: 0x0

INTEN is automatically cleared during the acknowledge cycle and the request is removed (ROAK). The *ILx* bits, however, remain set until they are overwritten.

Check the *INTEN* bit to verify that the interrupt has been acknowledged.

INTID – VME Interrupt STATUS/ID Register (0x0004) (read/write)

7..0
INT_ID

In this Register, the STATUS/ID of the internal interrupter is set.

INT_ID The STATUS/ID of the interrupt that the external handler reads during the IACK cycle.
Default: 0x00

If another VME master performs an IACK cycle, *INT_ID* will be transmitted in case this interrupter was the source of the request.

2.13.12 A32 Address Mode

Since the address space for BAR1 is limited to 512 MB, the upper three address bits must be set in a different way, in order to address the entire VMEbus address space. Therefore, a register contains the upper three address bits.

LONGADD – Upper 3 Address Bits for A32 (0x001C) (read/write)

7..3	2	1	0
-	ADDR31	ADDR30	ADDR29

ADDR31..ADDR29 Upper 3 address bits for A32 access. These bits will be combined with the address bits ADDR28..ADDR2 while an A32 access on the VMEbus is performed, in order to address the entire A32 space.

2.13.13 Mailbox

You can use the mailbox feature to send messages without using the slow interrupt daisy chain. Writing and/or reading one of the data registers of the mailbox from the VMEbus side generates a local CPU interrupt.

Due to the location of the data registers in the local SRAM (MAILBOX_0 to MAILBOX_3), normal A24/A32 accesses can be used in contrast to the slow daisy chain mechanism.

The VMEbus controller provides four independent 32-bit Mailbox Data Registers, which can be configured to cause an interrupt request upon read or write, or both.

The mailbox feature uses MSI vector 5 for interrupts.

MAIL_IRQE – Mailbox Interrupt Enable Register (0x0020) (read/write)

7	6	5	4	3	2	1	0
IEVW3	IEVR3	IEVW2	IEVR2	IEVW1	IEVR1	IEVW0	IEVR0

IEVWx VMEbus write access to mailbox register *x* generates an interrupter request on the PCI-bus side when this bit is set.

Default: 0x0 (interrupt disabled)

IEVRx VMEbus read access to mailbox register *x* generates an interrupter request on the PCI-bus side when this bit is set.

Default: 0x0 (interrupt disabled)

These bits can be set and reset from both sides (PCI side and VMEbus side A16 space). If both sides access a bit, the VMEbus side wins.

MAIL_IRQ – Mailbox Interrupt Request Register (0x0024) (read/write)

7	6	5	4	3	2	1	0
IPVW3	IPVR3	IPVW2	IPVR2	IPVW1	IPVR1	IPVW0	IPVR0

IPVWx Indicates pending interrupt on the PCI bus caused by VMEbus write access to mailbox data register *x*. Cleared by writing 1.

IPVRx Indicates pending interrupt on the PCI bus caused by VMEbus read access to mailbox data register *x*. Cleared by writing 1.

Default: 0x0 (no interrupt pending)

These bits can be set and reset from both sides (PCI side and VMEbus side A16 space). If both sides access a bit, the VMEbus side wins.

Note: Any of these bits will only be set if the corresponding enable bit is set (*IEVWx* or *IEVRx*)!

MAILBOX_0..MAILBOX_3 – Mailbox Data Register (0xFF800, 0xFF804, 0xFF808, 0xFF80C) (read/write)

31..0
Data

Writing or reading this register results in an interrupt in the Mailbox Interrupt Request Register (MAIL_IRQ), if the interrupt is enabled in the Mailbox Interrupt Enable Register (MAIL_IRQE).

This data register can be used to transmit a status or ID.

2.13.14 Location Monitor

The VMEbus controller supports two independent location monitors. Each can be used to monitor the VMEbus in order to notify the CPU about write/read accesses to a predefined VME address, even if the VME master or slave is not used for this transaction. If the ongoing address and VMEbus access is identical to the configured one, an interrupt will be generated for notification.

The location monitors use MSI vector 4 for interrupts.

LOCSTA_0 – Location Monitor Status Register 0 (0x0038) (read/write)

7	6	5	4	3	2..1	0
ADDR_4	ADDR_3	LOC_ WR_0	LOC_ RD_0	LOC_ STAT_0	LOC_AM_0	LOC_ IRQE_0

LOCSTA_1 – Location Monitor Status Register 1 (0x003C) (read/write)

7	6	5	4	3	2	0
ADDR_4	ADDR_3	LOC_ WR_1	LOC_ RD_1	LOC_ STAT_1	LOC_AM_1	LOC_ IRQE_1

ADDR_4..3 Address bits 4 and 3 of VMEbus address; stored when location monitor found hit

LOC_WR_x
0 = Write accesses are not monitored
1 = Write accesses are monitored

LOC_RD_x
0 = Read accesses are not monitored
1 = Read accesses are monitored

LOC_STAT_x Location monitor status (interrupt request if enabled): If set, the address and address mode on the VMEbus are identical to *LOCADDR_x* and *LOC_AM_x*.

Reset by writing 1.

LOC_AM_x Monitor Address Mode

0 0 = A32 - Address bits [31:10] on VMEbus will be compared with *LOCADDR_x* [31:10]
0 1 = No function
1 0 = A16 - Address bits [15:10] on VMEbus will be compared with *LOCADDR_x* [15:10]
1 1 = A24 - Address bits [23:10] on VMEbus will be compared with *LOCADDR_x* [23:10]

LOC_IRQE_x
0 = Monitor interrupt request is disabled
1 = Monitor interrupt request is enabled

LOCADDR_0 – Location Monitor Address Register (0x0040) (read/write)

31..0
ADDR[31:0]

ADDR[31:0] Compare address for location monitor

The desired address bits depend on *LOC_AM_0*:

LOC_AM_0 = 0 0: *LOCADDR_0* [31:10]
LOC_AM_0 = 1 0: *LOCADDR_0* [15:10]
LOC_AM_0 = 1 1: *LOCADDR_0* [23:10]

LOCADDR_1 – Location Monitor Address Register (0x0044) (read/write)

31..0
ADDR[31:0]

ADDR[31:0] Compare address for location monitor

The desired address bits depend on *LOC_AM_1*:

LOC_AM_1 = 0 0: *LOCADDR_1* [31:10]
LOC_AM_1 = 1 0: *LOCADDR_1* [15:10]
LOC_AM_1 = 1 1: *LOCADDR_1* [23:10]

2.13.15 DMA Controller

The VMEbus controller has a DMA controller for high performance data transfers between the SRAM, PCI space and VMEbus. It is operated through a series of registers that control the source and destination for the data, length of the transfer and the transfer protocol (A24 or A32) to be used. These registers are not directly accessible, but they will be loaded with the content of the buffer descriptor(s) located in the local SRAM (DMA_BD#1 to DMA_BD#256).

One buffer descriptor may be linked to the next buffer descriptor, such that when the DMA has completed the operations described by one buffer descriptor, it automatically moves on to the next buffer descriptor in the local SRAM list. The last buffer descriptor is reached, when the *DMA_NULL* bit is set in the corresponding buffer descriptor. The maximum number of linked buffer descriptors is 16.

The DMA controller supports interrupt assertion when all specified buffer descriptors are processed. It uses MSI vector 3 for interrupts.

The DMA controller is able to transfer data from the SDRAM, PCI space and VMEbus to each other. For this reason, the source and/or destination address can be incremented or not – depending on the settings. The source and destination address must be 8-byte aligned to each other.

The scatter-gather list is located in the local SRAM area, so a DMA can also be initiated by an external VME master by accessing the SRAM via A24/A32 slave and the DMA Status Register via A16 slave.

If a transfer to the PCI space has to be done, the used memory space must be allocated for this function in order to prevent data mismatch!

If DMA functionality is used, the entire local SRAM cannot be used by other functions, because the buffer descriptors are located at the end of this.

DMASTA – DMA Status Register (0x002C) (read/write)

7..4	3	2	1	0
DMA_ACT_BD	DMA_ERR	DMA_IRQ	DMA_IEN	DMA_EN

This register contains the DMA interrupt and status bits.

DMA_ACT_BD Shows the number of the active buffer descriptor (read only)
If this bit is 0 and *DMA_EN* is 0, then the DMA controller is not working.

DMA_ERR Will be asserted if a VMEbus error has occurred or if the DMA transaction has been stopped manually.

0 = DMA no error occurred
1 = DMA error occurred (reset by writing 1)

DMA_IRQ Interrupt Request bit: reflects status of interrupt request signal

0 = DMA IRQ inactive
1 = DMA IRQ set to active when *DMA_IEN* is set and when buffer descriptor with set *DMA_NULL* bit was processed or when all buffer descriptors are processed or when *DMA_ERR* bit was set) (reset by writing 1)

DMA_IEN 0 = DMA IRQ is disabled
1 = DMA IRQ is enabled

DMA_EN 0 = DMA transaction is stopped (or will be stopped if set to 0)
1 = DMA transaction is enabled (will be set to 0 when done)

DMA_BD#x – DMA Buffer Descriptors (0x0XX0..0XXC)

0x00	31..2	1..0
	DMA_DEST_ADDR	-

0x04	31..2	1..0
	DMA_SOUR_ADDR	-

0x08	31..16	15..0
	-	DMA_SIZE

0x0C	31..19	18..16	15	14..12	11..8	7..4	3	2	1	0
	-	DMA_ SOUR_ DEVICE	-	DMA_ DEST_ DEVICE	-	DMA_ VME_AM	-	INC_ SOUR	INC_ DEST	DMA_ NULL

DMA_DEST_ADDR Destination address of DMA transfer

DMA_SOUR_ADDR Source address of DMA transfer

DMA_SIZE Block size of DMA transfer in longwords: (x bytes / 4) -1

Examples:
0x0000: 4 bytes (minimum size)
0x0001: 8 bytes
0x0002: 12 bytes
0x0FFF: 16 KB
0x1000: 16 KB + 4 bytes
0xFFFF: 256 KB (maximum size)

DMA_SOUR_DEVICE 0 0 1 = Source is located in SRAM address space
0 1 0 = Source is located in VME address space
1 0 0 = Source is located in PCI address space

DMA_DEST_DEVICE 0 0 1 = Destination is located in SRAM address space
0 1 0 = Destination is located in VME address space
1 0 0 = Destination is located in PCI address space

DMA_VME_AM VME address modifier for DMA transaction

0 0 0 0 = A24 D16 (DMA_x_ADDR [23:3] is used)
0 1 0 0 = A24 D32 (DMA_x_ADDR [23:3] is used)
0 1 1 0 = A32 D32 (DMA_x_ADDR [31:3] is used)
1 1 1 0 = A32 D64 (DMA_x_ADDR [31:3] is used)

Other combinations are reserved.

INC_SOUR 0 = Increment source address during operation
1 = Keep source address

INC_DEST 0 = Increment destination address during operation
1 = Keep destination address

DMA_NULL 0 = End is not yet reached
1 = End of buffer descriptor list is reached

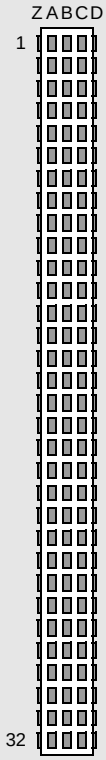
2.13.16 Connection

Connector types:

- 160-pin, 5-row plug, performance level according to DIN41612, part 5
- Mating connector:
  160-pin, 5-row receptacle, performance level according to DIN41612, part 5

The pin assignment of P1 conforms to the VME64 specification VITA 1-1994 and VME64 Extensions Draft Standard VITA 1.1-199x.

Table 25. *Pin assignment of VME64 connector P1*



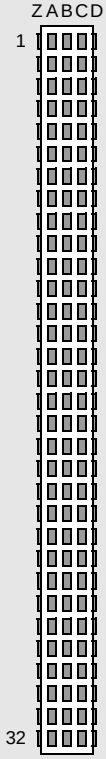
	Z	A	B	C	D
1	-	D0	BBSY#	D8	+5V
2	GND	D1	BCLR#	D9	GND
3	-	D2	ACFAIL#	D10	-
4	GND	D3	BG0IN#	D11	-
5	-	D4	BG0OUT#	D12	-
6	GND	D5	BG1IN#	D13	-
7	-	D6	BG1OUT#	D14	-
8	GND	D7	BG2IN#	D15	-
9	-	GND	BG2OUT#	GND	-
10	GND	SYSCLK	BG3IN#	SYSFAIL#	-
11	-	GND	BG3OUT#	BERR#	-
12	GND	DS1#	BR0#	SYSRESET#	+3.3V
13	-	DS0#	BR1#	LWORD#	-
14	GND	WRITE#	BR2#	AM5	+3.3V
15	-	GND	BR3#	A23	-
16	GND	DTACK#	AM0	A22	+3.3V
17	-	GND	AM1	A21	-
18	GND	AS#	AM2	A20	+3.3V
19	-	GND	AM3	A19	-
20	GND	IACK#	GND	A18	+3.3V
21	-	IACKIN#	-	A17	-
22	GND	IACKOUT#	-	A16	+3.3V
23	-	AM4	GND	A15	-
24	GND	A7	IRQ7#	A14	+3.3V
25	-	A6	IRQ6#	A13	-
26	GND	A5	IRQ5#	A12	+3.3V
27	-	A4	IRQ4#	A11	-
28	GND	A3	IRQ3#	A10	+3.3V
29	-	A2	IRQ2#	A9	-
30	GND	A1	IRQ1#	A8	+3.3V
31	-	-12V	+5VSTDBY	+12V	GND
32	GND	+5V	+5V	+5V	+5V

2.13.16.1 Pin Assignment of P2

The pin assignment of P2 depends on the type of mezzanine modules used.

Note: The 24-pin M-Module connector that leads rear I/O signals to VME P2 is also called P2. (See also Chapter Figure 1. Map of the board – A21B front panel and top view (M-Modules) on page 18.)

Table 26. *Pin assignment of VMEbus rear I/O connector P2 – A21B – M-Modules*

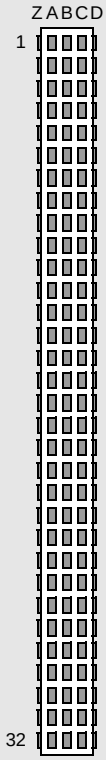


	Z	A	B	C	D
1	-	Mod2 P2-2	+5V	Mod2 P2-1	-
2	GND	Mod2 P2-4	GND	Mod2 P2-3	-
3	-	Mod2 P2-6	RETRY#	Mod2 P2-5	-
4	GND	Mod2 P2-8	A24	Mod2 P2-7	-
5	Mod2 P2-22	Mod2 P2-10	A25	Mod2 P2-9	-
6	GND	Mod2 P2-12	A26	Mod2 P2-11	-
7	Mod2 P2-23	Mod2 P2-14	A27	Mod2 P2-13	-
8	GND	Mod2 P2-16	A28	Mod2 P2-15	-
9	Mod2 P2-24	Mod2 P2-18	A29	Mod2 P2-17	-
10	GND	Mod2 P2-20	A30	Mod2 P2-19	-
11	-	Mod1 P2-1	A31	Mod2 P2-21	-
12	GND	Mod1 P2-3	GND	Mod1 P2-2	-
13	-	Mod1 P2-5	+5V	Mod1 P2-4	-
14	GND	Mod1 P2-7	D16	Mod1 P2-6	-
15	-	Mod1 P2-9	D17	Mod1 P2-8	-
16	GND	Mod1 P2-11	D18	Mod1 P2-10	-
17	Mod1 P2-22	Mod1 P2-13	D19	Mod1 P2-12	-
18	GND	Mod1 P2-15	D20	Mod1 P2-14	-
19	Mod1 P2-23	Mod1 P2-17	D21	Mod1 P2-16	-
20	GND	Mod1 P2-19	D22	Mod1 P2-18	-
21	Mod1 P2-24	Mod1 P2-21	D23	Mod1 P2-20	-
22	GND	Mod0 P2-2	GND	Mod0 P2-1	-
23	-	Mod0 P2-4	D24	Mod0 P2-3	-
24	GND	Mod0 P2-6	D25	Mod0 P2-5	-
25	-	Mod0 P2-8	D26	Mod0 P2-7	-
26	GND	Mod0 P2-10	D27	Mod0 P2-9	-
27	-	Mod0 P2-12	D28	Mod0 P2-11	-
28	GND	Mod0 P2-14	D29	Mod0 P2-13	-
29	Mod0 P2-23	Mod0 P2-16	D30	Mod0 P2-15	-
30	GND	Mod0 P2-18	D31	Mod0 P2-17	-
31	Mod0 P2-24	Mod0 P2-20	GND	Mod0 P2-19	GND
32	GND	Mod0 P2-22	+5V	Mod0 P2-21	+5V

Table 27. *Signal mnemonics of VMEbus rear I/O connector P2 – A21B – M-Modules*

	Signal	Direction	Function
Power	+5V	-	+5V power supply
	GND	-	Digital ground
VME64	A[31:24]	in	VME64 address lines
	D[31:16]	in/out	VME64 data lines
	RETRY#	out	VME64 retry for postponed data transfer
M-Module 2	MMod2 P2-*xx*	in/out	Signal *xx* from M-Module 2 24-pin rear I/O connector P2
M-Module 1	MMod1 P2-*xx*	in/out	Signal *xx* from M-Module 1 24-pin rear I/O connector P2
M-Module 0	MMod0 P2-*xx*	in/out	Signal *xx* from M-Module 0 24-pin rear I/O connector P2

Table 28. *Pin assignment of VMEbus rear I/O connector P2 – A21C – PMC*



	Z	A	B	C	D
1	PMC1 J14-2	PMC0 J14-2	+5V	PMC0 J14-1	PMC1 J14-1
2	GND	PMC0 J14-4	GND	PMC0 J14-3	PMC1 J14-3
3	PMC1 J14-5	PMC0 J14-6	RETRY#	PMC0 J14-5	PMC1 J14-4
4	GND	PMC0 J14-8	A24	PMC0 J14-7	PMC1 J14-6
5	PMC1 J14-8	PMC0 J14-10	A25	PMC0 J14-9	PMC1 J14-7
6	GND	PMC0 J14-12	A26	PMC0 J14-11	PMC1 J14-9
7	PMC1 J14-11	PMC0 J14-14	A27	PMC0 J14-13	PMC1 J14-10
8	GND	PMC0 J14-16	A28	PMC0 J14-15	PMC1 J14-12
9	PMC1 J14-14	PMC0 J14-18	A29	PMC0 J14-17	PMC1 J14-13
10	GND	PMC0 J14-20	A30	PMC0 J14-19	PMC1 J14-15
11	PMC1 J14-17	PMC0 J14-22	A31	PMC0 J14-21	PMC1 J14-16
12	GND	PMC0 J14-24	GND	PMC0 J14-23	PMC1 J14-18
13	PMC1 J14-20	PMC0 J14-26	+5V	PMC0 J14-25	PMC1 J14-19
14	GND	PMC0 J14-28	D16	PMC0 J14-27	PMC1 J14-21
15	PMC1 J14-23	PMC0 J14-30	D17	PMC0 J14-29	PMC1 J14-22
16	GND	PMC0 J14-32	D18	PMC0 J14-31	PMC1 J14-24
17	PMC1 J14-26	PMC0 J14-34	D19	PMC0 J14-33	PMC1 J14-25
18	GND	PMC0 J14-36	D20	PMC0 J14-35	PMC1 J14-27
19	PMC1 J14-29	PMC0 J14-38	D21	PMC0 J14-37	PMC1 J14-28
20	GND	PMC0 J14-40	D22	PMC0 J14-39	PMC1 J14-30
21	PMC1 J14-32	PMC0 J14-42	D23	PMC0 J14-41	PMC1 J14-31
22	GND	PMC0 J14-44	GND	PMC0 J14-43	PMC1 J14-33
23	PMC1 J14-35	PMC0 J14-46	D24	PMC0 J14-45	PMC1 J14-34
24	GND	PMC0 J14-48	D25	PMC0 J14-47	PMC1 J14-36
25	PMC1 J14-38	PMC0 J14-50	D26	PMC0 J14-49	PMC1 J14-37
26	GND	PMC0 J14-52	D27	PMC0 J14-51	PMC1 J14-39
27	PMC1 J14-41	PMC0 J14-54	D28	PMC0 J14-53	PMC1 J14-40
28	GND	PMC0 J14-56	D29	PMC0 J14-55	PMC1 J14-42
29	PMC1 J14-44	PMC0 J14-58	D30	PMC0 J14-57	PMC1 J14-43
30	GND	PMC0 J14-60	D31	PMC0 J14-59	PMC1 J14-45
31	PMC1 J14-46	PMC0 J14-62	GND	PMC0 J14-61	GND
32	GND	PMC0 J14-64	+5V	PMC0 J14-63	+5V

Table 29. *Signal mnemonics of VMEbus rear I/O connector P2 – A21C – PMC*

	Signal	Direction	Function
Power	+5V	-	+5V power supply
	GND	-	Digital ground
VME64	A[31:24]	in	VME64 address lines
	D[31:16]	in/out	VME64 data lines
	RETRY#	out	VME64 retry for postponed data transfer
PMC 1	PMC1 J14-*xx*	in/out	Signal *xx* from PMC 1 rear I/O connector J14
PMC 0	PMC0 J14-*xx*	in/out	Signal *xx* from PMC 0 rear I/O connector J14

3 U-Boot Boot Loader

3.1 General

U-Boot is the CPU board 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 operating system via Flash, TFTP or similar methods.

U-Boot for A21 consists of the standard U-Boot code and a board-specific patch. The current patch file and the complete binaries (prebuilt main U-Boot image) are available for download from the A21B and A21C pages on MEN's website.

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

For advanced developing and programming, you can also use the following resources:

- U-Boot source code on the DENX website (also includes README files): http://git.denx.de (GIT repositories) and ftp://ftp.denx.de/pub/u-boot (TAR archives)
- U-Boot mailing list: http://lists.denx.de/mailman/listinfo/u-boot

3.2 Getting Started: Setting Up Your Operating System

This chapter describes a recommended procedure of how to get your operating system running for the first time.

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 board 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 following gives an example of how to integrate and boot the example images for Linux or VxWorks provided by the MEN BSPs. In the example, the images are loaded from a host computer via TFTP.

Note: This procedure uses the U-Boot standard commands to make the individual steps clearer. For your actual application, you can use additional environment variables that the A21 U-Boot provides for booting, and which short-cut the individual steps shown below. See Chapter 3.4.2 Booting an Operating System on page 85.

☑ Connect the host computer where your boot image is located to the A21's LAN1 Ethernet port.

3.2.1 Setting Up the Boot File

☑ Create a boot file for your operating system on your host computer.

3.2.2 Setting Up the Boot and TFTP Parameters

☑ Set the network parameters through the U-Boot environment variables for the network connection:

```
A21=> editenv ipaddr
edit: 192.1.1.120     Edit IP address and press <Enter>
A21=> editenv serverip
edit: 192.1.1.22      Edit TFTP server IP address and press <Enter>
A21=> editenv gatewayip
edit: 192.1.1.22      Edit IP address of the gateway and press <Enter>
A21=> editenv netmask
edit: 255.255.255.0   Edit the subnet mask and press <Enter>
```

☑ Set up the boot file through environment variable *bootfile*, e.g.:

```
Linux:
A21=> setenv bootfile /tftpboot/pMulti_A21
VxWorks:
A21=> setenv bootfile /tftpboot/vxW_A21.st
```

☑ Set the boot arguments through environment variable *bootargs*, e.g.:

```
Linux:
A21=> setenv bootargs root=/dev/ram rw
VxWorks:
A21=> setenv bootargs motetsec(0,0)${hostname}:${bootfile} e=$
{ipaddr} h=${serverip} g=${gatewayip} u=username pw=password tn= s=
                                    'u' is the FTP user name
                                                'pw' is the FTP password
```

☑ Set up the autostart script through environment variable *bootcmd*:

Linux:
```
A21=> setenv bootcmd 'tftpboot && bootm'
```
VxWorks:
```
A21=> setenv bootcmd 'tftpboot && cpenv && bootvx'
```

3.2.3 Starting Up the Operating System

☑ Save the changed environment variables.

```
A21=> saveenv
```

☑ Reset the board.

```
A21=> reset
```

☑ U-Boot restarts the board and loads the configured operating system with the settings made.

3.3 Interacting with U-Boot

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

3.3.1 Setting Up a Console Connection

To interact with U-Boot, you can use the RS232 COM1 as a serial console port. (See Chapter 2.8 RS232 COM1 Interface on page 34.)

You can select the active console by means of environment variables *stdin*, *stdout* and *stderr*, see Table 33, U-Boot – Environment variables – Console on page 98. U-Boot command *coninfo* lists all active consoles.

The default setting of the COM port is 115200 baud, 8 data bits, no parity, and one stop bit. (You can set the baud rate through environment variable *baudrate*, see description on page 98.)

3.3.2 Entering the U-Boot Command Line

After completing its initialization sequence, U-Boot continues by executing the boot script contained in the *bootcmd* variable. You can enter the U-Boot command line by aborting the OS boot process by pressing any key during U-Boot startup, i.e. before the boot script is executed.

You can insert an additional delay before the execution of the boot script by setting the *bootdelay* environment variable to the number of seconds the OS boot process shall be delayed, giving the user more time to manually abort the automated boot and entering the command line (see Table 31, U-Boot – Environment variables – OS boot on page 96).

3.3.3 User Interface Basics

3.3.3.1 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.

3.3.3.2 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*. Using command *defenv* the environment variables can be reset to default values.

Examples: Displaying environment variables

```
A21=> printenv baudrate
baudrate=115200

A21=> printenv    Print all variables
baudrate=115200
bootdelay=0
...

A21=> echo "Server IP: ${serverip}"    Shell variable expansion
Server IP: 192.1.1.22
```

Examples: Editing and saving environment variables

```
A21=> editenv ipaddr
edit: 192.1.1.023    Edit the variable in the edit line and press <Enter>

A21=> setenv ipaddr 192.1.1.123    Edit a variable directly

A21=> setenv ipaddr    Deletes a variable completely

A21=> defenv    Set all variables to default

You need to save the changes made, otherwise they will be lost after next reset:
A21=> saveenv
Saving Environment to Flash...
```

For a list of the A21 environment variables see Chapter 3.8.2 Environment Variables on page 96.

3.3.3.3 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 variable) (see also www.denx.de/wiki/view/DULG/CommandLineParsing):

```
A21=> setenv menu_script 'echo ”1=VxWorks”; echo ”2=Linux”; echo
”3=Mem test”; askenv _number; if test ${_number} = 1; then run
vxworks; elif test ${_number} = 2; then run linux; else mtest; fi'
```

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

```
A21=> saveenv
```

☑ Execute the script:

```
A21=> 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 A21 Flash and run it on the board 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:

```
me@server:/> 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:

```
A21=> tftpboot 192.1.1.22:/tftpboot/brd_info.scr
```

☑ Execute the script:

```
A21=> source ${loadaddr}
```

3.4 U-Boot Images and Start-Up

3.4.1 Images

U-Boot has a full-featured, **fallback image** mainly intended for board recovery, and a **main image** for normal operation. Both images are stored in the onboard boot Flash.

After a board reset, the CPU starts executing the fallback image. If the abort pin is not active and the main U-Boot image is valid (i.e. a valid version string is found and the image CRC is correct), the fallback image starts the main U-Boot image. This is done very early in the fallback image to provide start-up flexibility and speed.

3.4.1.1 Loading the Fallback Image

U-Boot for A21 includes the *fallback* command that can be used to start the fallback image from within the main U-Boot's command line. E.g., you can reduce the functions of the main U-Boot image for fast booting. To get the full functionality, you can call the full-featured fallback image using the *fallback* command.

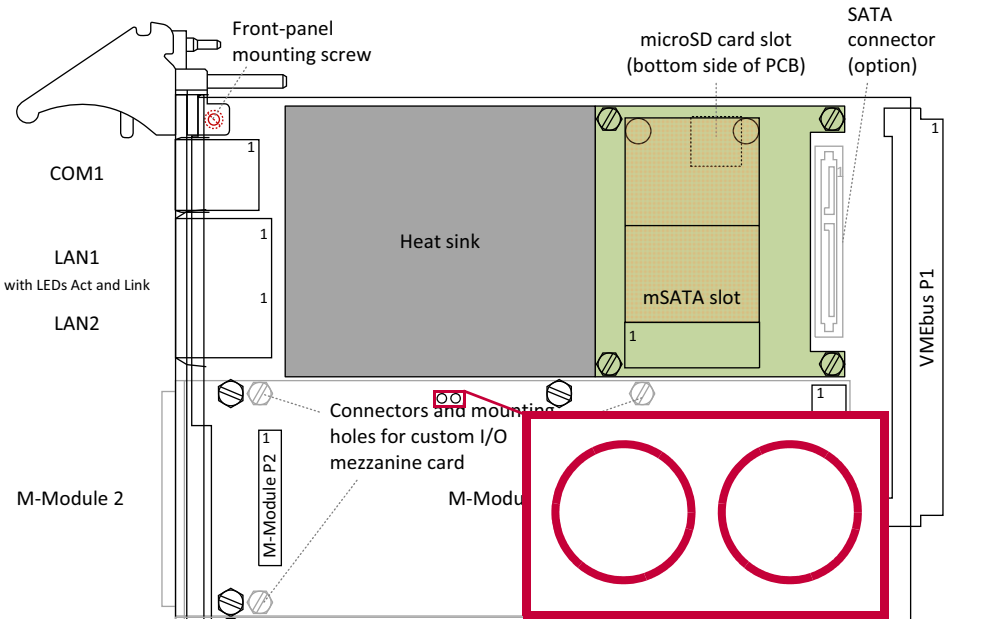
If the main image does not work, it may be necessary to force the fallback image to be loaded by hardware. Since the A21 has no real "abort" button to do this, it is simulated by connecting the abort signal to ground using a wire link. When this connection is made, only the fallback image is loaded.

Please note that both images have **their own** environment variable settings. The settings of the fallback and main image may differ.

Note: In any case, power off the system before you connect the abort pins.

The pins are accessible at the top side of the PCB. They are located in the area of M-Module slot 2 (A21B) or PMC/XMC 1 (A21C). Connect the two pins shown in the following figure using a wire link or jumper.

Figure 11. *U-Boot – Position of fallback jumper (top side)*



3.4.2 Booting an Operating System

U-Boot provides the *bootm*, *bootvx* and *bootelf* commands to support booting of Linux (using *bootm*), VxWorks (using *bootvx*), OS-9 (using *bootelf*) and QNX (using *bootelf*). It supports booting in less than 5 seconds.

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 examples in Chapter 3.2.2 Setting Up the Boot and TFTP Parameters on page 79.)

3.4.2.1 Boot Methods

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

OS Boot via Network

U-Boot command *tftpboot* allows loading of the operating system via the board's Ethernet interface using the TFTP protocol. You can find a detailed description of the necessary settings in Chapter 3.2.2 Setting Up the Boot and TFTP Parameters on page 79.

OS Boot via Boot Flash

U-Boot maps Flash memory directly into the CPU memory space to make it possible to load an operating system binary stored in the onboard boot Flash.

☑ Set the boot arguments through environment variable *bootargs*, e.g.:

```
Linux:
A21=> setenv bootargs root=/dev/ram rw
VxWorks:
A21=> setenv bootargs motetsec(0,0)${hostname}:${bootfile} e=$
{ipaddr} h=${serverip} g=${gatewayip} u=username pw=password tn= s=
                                        'u' is the FTP user name
                                                'pw' is the FTP password
```

☑ Set up the autostart script through environment variable *bootcmd*:

```
Linux:
A21=> setenv bootcmd 'bootm 0xef000000'
VxWorks:
A21=> setenv bootcmd 'cpenv && bootvx 0xef000000'
```

OS Boot via External Mass Storage Devices (SATA, microSD, USB)

U-Boot commands *ext2load*, *fatload*, *reiserload* or *usbboot* (raw) allow loading of the operating system via SATA, microSD or USB mass storage devices. The command to be used depends on the file system of the device. (See also Chapter 3.6.4 USB on page 93.)

Please note that the USB interface is implemented only as an option, see Chapter 2.11 I/O Extension on page 46. You need to activate it first before you can use USB, see Chapter Activating the USB Interface on page 94.

Example with *fatload*:

☑ Set the load address to the default DRAM value in environment variable *loadaddr*:

```
A21=> setenv loadaddr 0x02000000
```

☑ Set up the boot file through environment variable *bootfile*, e.g.:

```
Linux:
A21=> setenv bootfile /pMulti_A21
VxWorks:
A21=> setenv bootfile /vxW_A21.st
```

☑ Set the boot arguments through environment variable *bootargs*, e.g.:

```
Linux:
A21=> setenv bootargs root=/dev/ram rw
VxWorks:
A21=> setenv bootargs motetsec(0,0)${hostname}:${bootfile} e=$
{ipaddr} h=${serverip} g=${gatewayip} u=username pw=password tn= s=
                                        'u' is the FTP user name
                                                'pw' is the FTP password
```

☑ Set up the autostart script through environment variable *bootcmd* to initialize the device and load the boot file into DRAM.

For an **mSATA disk or optional SATA hard disk**:

```
Linux:
A21=> setenv bootcmd 'sata;fatload sata 0 ${loadaddr} ${bootfile};
bootm'
VxWorks:
A21=> setenv bootcmd 'sata;fatload sata 0 ${loadaddr} ${bootfile};
cpenv && bootvx'
```

For a **microSD card**:

```
Linux:
A21=> setenv bootcmd 'mmc rescan;fatload mmc 0 ${loadaddr}
${bootfile}; bootm'
VxWorks:
A21=> setenv bootcmd 'mmc rescan;fatload mmc 0 ${loadaddr}
${bootfile}; cpenv && bootvx'
```

For a **USB device** (option), this is done as follows:

```
Linux:
A21=> setenv bootcmd 'usb start; fatload usb 0:1 ${loadaddr}
${bootfile}; bootm'                              0:1 = device 0 partition 1
VxWorks:
A21=> setenv bootcmd 'usb start; fatload usb 0:1 ${loadaddr}
${bootfile}; cpenv && bootvx'                    0:1 = device 0 partition 1
```

OS Boot from Root File System on Hard Disk

As another option, the root file system can be held on a block device instead using a RAM disk in main memory. The root file system is writable and persistant much like a desktop PC. Since this option is rather untypical of embedded systems, however, the following only shows how to configure this boot method for Linux as an example, with an RFS on SATA partition 1, ext2.

☑ Set up the boot file through environment variable *bootfile*, e.g.:

```
Linux:
A21=> setenv bootfile /boot/pMulti_A21
                          Contains kernel and flattened device tree
```

☑ Set the boot arguments through environment variable *bootargs*, e.g.:

```
Linux:
A21=> setenv bootargs root=/dev/sda1
                    Linux device node for mSATA disk/partition that contains the RFS
```

☑ Set up the autostart script through environment variable *bootcmd* to initialize the device and load the boot file into DRAM.

```
Linux:
A21=> setenv bootcmd 'sata;ext2load sata o ${loadaddr} ${bootfile};
bootm'
                          Just load kernel, RFS will be mounted in Linux
```

3.4.2.2 Configuring Boot using Additional Environment Variables

Due to additional environment variables provided by the A21 U-Boot, it takes only a few steps to configure booting:

☑ Set the operating system to *linux*, *vxworks*, *os-9* or *qnx* through environment variable *os*:

```
A21=> setenv os linux
```

☑ Set the boot method to *flash*, *tftp, sata* or *mmc* through environment variable *bootmethod*:

```
A21=> setenv bootmethod tftp
```

☑ Set the boot files:

```
Linux:
A21=> editenv linux_file
VxWorks:
A21=> editenv vxworks_file
```

☑ Set the boot arguments (the defaults should work in most cases):

```
Linux:
A21=> editenv linux_setargs
VxWorks:
A21=> editenv vxworks_setargs
```

☑ Set environment variable *bootcmd*:

```
A21=> setenv bootcmd 'run ${os}'
```

☑ If you want to start the boot directly from U-Boot, execute:

```
Linux:
A21=> run linux
VxWorks:
A21=> run vxworks
```

3.5 Updating the Boot Flash

You can update U-Boot and other binaries located in the boot Flash via a serial console connection, network or mass-storage device. U-Boot provides commands specific for each medium to load a binary update file, and the general *protect*, *erase* and *cp* commands to program the boot Flash.

The A21 U-Boot supports hardware protection.

3.5.1 Update via the Serial Console

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

The terminal emulation program must be configured to start the upload via the "Ymodem" (for *loady*) or "Kermit" protocol (for *loadb* and *loads*) and send the required file.

3.5.2 Update via Network

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

3.5.3 Update via SATA, microSD or USB

You can also make a Flash update from a mass storage device, e.g., a USB Flash drive or plugged microSD card. Use the U-Boot *ext2load*, *fatload*, *reiserload* or *usbboot* (raw) command to load the binary update file.

Please note that the USB interface is implemented only as an option, see Chapter 2.11 I/O Extension on page 46. You need to activate it first before you can use USB, see Chapter Activating the USB Interface on page 94.

3.5.4 Performing an Update

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

> For instructions on how to update the U-Boot code itself, please see Chapter 3.5.5 Updating U-Boot Code on page 91.
> For a memory map of the Flash, please see Chapter 3.8.1 Boot Flash Memory Map on page 96.

☑ Download the update file:

```
Via serial console, e.g., with Y protocol:
A21=> loady

Via network, e.g.:
A21=> tftpboot 192.1.1.22:/path/file.bin

Via SATA storage, e.g.:
A21=> sata init; fatload sata 0:1 ${loadaddr} /file.bin
                                  0:1 = device 0 partition 1

Via microSD card, e.g.:
A21=> mmc rescan; fatload mmc 0:1 ${loadaddr} /file.bin
                                  0:1 = device 0 partition 1

Via USB storage, e.g.:
A21=> usb start; fatload usb 0:1 ${loadaddr} /file.bin
                                  0:1 = device 0 partition 1
```

☑ Unprotect the Flash (i.e. unlock Flash sector protection):

```
A21=> protect off 0xef000000 +${filesize}
                    0xef000000 is the update address
```

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

```
A21=> erase 0xef000000 +${filesize}
```

☑ Write the file to Flash:

```
A21=> cp.b ${loadaddr} 0xef000000 ${filesize}
```

☑ Protect the Flash again:

```
A21=> protect on all
```

If you want to do your updates in a more user-friendly way, you can create a script that includes the above steps:

☑ Create a new update script in an environment variable, e.g.:

```
A21=> setenv update_os 'setenv bootfile /tftpboot/pMulti_A21 &&
editenv bootfile && setenv loadaddr 02000000 && tftpboot && itest
${filesize} != 0 && protect off ef000000 efdfffff && erase ef000000
+${filesize} && cp.b ${loadaddr} ef000000 ${filesize}; protect on
all'
```

☑ Save the script (optional):

```
A21=> saveenv
```

☑ Start the update:

```
A21=> run update_os
```

3.5.5 Updating U-Boot Code

Updates of the A21 U-Boot (only main image) are available for download from the A21B and A21C pages on MEN's website. U-Boot's integrated Flash update functions allow you to do updates yourself. However, you need to take care and follow the instructions given here. Otherwise, you may make your board inoperable!

In any case, read the following instructions carefully!

Please be aware that you do U-Boot updates at your own risk. After an incorrect update your CPU board may not be able to boot.

Do the following to update U-Boot:

- ☑ Unzip the downloaded file, e.g., *14A021-00_01_02.zip*, into a temporary directory.
- ☑ Connect a terminal emulation program with the COM1 port of your A21 and set the terminal emulation program to 115200 baud, 8 data bits, 1 stop bit, no parity, no handshaking (if you haven't changed the target baud rate before).
- ☑ Power on your A21 and press "ENTER" immediately.
- ☑ In your terminal emulation program, you should see the U-Boot prompt.

```
A21=>
```

- ☑ Enter *run update*:

```
A21=> run update
```

- ☑ U-Boot asks you by which means you want to do the update. Type in *uart* and <*ENTER*>.

```
Please enter 'tftp,bootp or uart':uart
via UART:
## Ready for binary (ymodem) download to 0x02000000 at 115200 bps...
C(CAN) packets, 3 retries
## Total Size      = 0x00061a1c = 399900 Bytes
....... done
Un-Protected 7 sectors

.... done
Erased 4 sectors
Copy to Flash... done
Protect Flash Bank # 1
................................................................
.........................e
A21=>
```

- ☑ In your terminal emulation program, start a "YModem" download of the image file, e.g., *u-boot_A21_1.0.bin*. For example, with Windows Hyperterm, select *Transfer* > *Send File* with protocol "YModem".
- ☑ As the last step after the download, the Flash is programmed. When the update procedure has completed, reset the A21.

3.6 Accessing Devices

U-Boot provides some useful commands especially for PCI, mass storage and I2C devices.

For a complete list of available U-Boot commands, see Chapter 3.9 U-Boot Commands on page 99.

3.6.1 PCI

List PCI devices

pci [bus] [long]	List of PCI devices on bus *bus* (long = detailed)

Read (config space)

pci display[.b,.w,.l] b.d.f [addr] [no_of_objects]	Read config space
pci header b.d.f	Show header of PCI device 'bus.device.function'

Write (config space)

pci write[.b,.w,.l] b.d.f addr value	Write to config space
pci modify[.b,.w,.l] b.d.f addr	Modify config space (read, write)
pci next[.b,.w,.l] b.d.f addr	Modify config space (const. addr.)

3.6.2 SATA

SATA access must be initialized by command *sata init*.

Init and Info

sata init	Initialize SATA subsystem
sata info	Show available SATA devices
sata device [dev]	Show or set current device
sata part [dev]	Print partition table

Read/Write

sata read addr blk# cnt	Read *cnt* blocks starting at block *blk#* to memory address *addr*
sata write addr blk# cnt	Write *cnt* blocks starting at block *blk#* to memory address *addr*

3.6.3 microSD

microSD card accesses must be initialized by command *mmc rescan*.

Init and Info

mmc rescan	Initialize MMC subsystem (scan the MMC)
mmc part	List available partition on current MMC device
mmc dev [dev] [part]	Show or set current MMC device [partition]
mmc list	List available devices

Read/Write/Erase

mmc read addr blk# cnt	Read *cnt* blocks starting at block *blk#* to memory address *addr*
mmc write addr blk# cnt	Write *cnt* blocks starting at block *blk#* to memory address *addr*
mmc erase blk# cnt	Erase *cnt* blocks starting at block *blk#*

3.6.4 USB

Before you access USB devices you have to call *usb start*. At compilation time, a specific USB controller must be configured, e.g., the EHCI (high-speed) USB controller. This means that high-speed and low/full-speed devices cannot be supported at the same time.

List all devices

usb tree	Show USB device tree
usb info [dev]	Show available USB devices

List storage devices

usb storage	Show details of USB storage devices
usb part [dev]	Print partition table of one or all USB storage devices

Read RAW data

usb dev [dev]	Show or set current USB storage device
usb read addr blk# cnt	Read *cnt* blocks starting at block *blk#* to memory address *addr*

Read file system

Use commands *ext2load*, *ext2ls*, *fatinfo*, *fatload*, *fatls*, *reiserload* or *reiserls*, e.g.:

```
A21=> fatinfo usb 0:1    0:1 = device 0 partition 1
A21=> fatls usb 0:1      use usb part to find dev/part
A21=> fatload usb 0:1 ${loadaddr} /path/file
```

Activating the USB Interface

As the USB PHY must be implemented on the optional mezzanine extension card on A21, you need to activate it first before you can use USB.

To do this, you need to adapt the settings of environment variable *hwconfig*. Its settings on A21 are similar to this:

```
hwconfig="....;usb:dr_mode=host;..:"
```

Add the PHY type of the implemented USB interface, e.g., with a ULPI PHY:

```
hwconfig="....;usb:dr_mode=host,phy_type=ulpi;..:"
```

For further documentation on the *hwconfig* variable please refer to the U-Boot source tree in *doc/README.hwconfig*.

3.6.5 I2C

Set bus

i2c dev [dev]	Show or set current I2C bus

List

i2c probe	Show devices on the I2C bus

Read

i2c md chip addr[.0, .1, .2] [no_of_objects]	Read from I2C device
i2c loop chip addr[.0, .1, .2] [no_of_objects]	Loop reading of device

Write

i2c mm chip addr[.0, .1, .2]	Write (modify)
i2c mw chip addr[.0, .1, .2] value [count]	Write (fill)
i2c nm chip addr[.0, .1, .2]	Write (constant addr.)

3.7 Power-On Self Tests

The A21 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.
- **Image Integrity Test.** Verifies if the U-Boot image checksum is correct for both fallback and standard U-Boot images.
- **FPGA Presence Test.** Tests the presence of the A21 FPGA.

3.8 U-Boot Configuration and Organization

3.8.1 Boot Flash Memory Map

Table 30. *U-Boot – Boot Flash memory map*

CPU Address Space	Size	Description
0x ExO0 0000..EFDF FFFF[1]	Flash size - 2 MB	Available to user
0x EFE0 0000..EFE1 FFFF	128 KB	SPD data for DDR SDRAM
0x EFE2 0000..EFE3 FFFF	128 KB	Reserved for MEN production data or redundant U-Boot environment
0x EFE4 0000..EFEF FFFF	768 KB	U-Boot binary
0x EFF0 0000..EFF1 FFFF	128 KB	SPD data for DDR SDRAM
0x EFF2 0000..EFF3 FFFF	128 KB	Reserved for MEN production data or redundant U-Boot environment
0x EFF4 0000..EFFF FFFF	768 KB	U-Boot binary

[1] Depending on the Flash size, the base address differs:
16 MB: x = F
32 MB: x = E
64 MB: x = C
128 MB: x = 8
256 MB: x = 0

3.8.2 Environment Variables

U-Boot uses environment variables to configure the target. The available variables are board-specific for the A21.

Environment variables are stored in Flash. 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*. See Chapter 3.3.3.2 Configuring Your System on page 81 for command line examples.

Table 31. *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	*run ${os}*	r/w
bootdelay	Delay before the default image is automatically booted, in seconds. Set to -1 to disable autoboot	0	r/w
bootmethod[1]	Method to boot operating system Possible values: *bootp*, *flash*, *tftp*, *sata*, *mmc*	*tftp*	r/w
linux_file[1]	Linux boot file used by default *bootcmd*	*/tftpboot/pMulti_A21*	r/w
linux_setargs[1]	Used to set *bootargs* Default: *setenv bootargs root=/dev/ram rw console=ttyPSC0,${baudrate}*		r/w

Variable	Description	Default	Access
linux[1]	Example script for booting Linux Default: *cpu 0 status;setenv bootfile ${linux_file};setenv loadaddr 02000000;run linux_setargs ${bootmethod};bootm ${loadaddr}*		r/w
os[1]	Operating system to boot (e.g., *vxworks*, *linux*)	*--- do not boot ---*	r/w
vxworks_file[1]	VxWorks boot file used by default *bootcmd*	*/tftpboot/vxW_A21.st*	r/w
vxworks_setargs[1]	Used to set *bootargs* Default: *setenv bootargs motetsec(0,0)${hostname}:${bootfile} e=${ipaddr} h=${serverip} g=${gatewayip} u=${user} pw=${pwd} tn=${target} s=${script}*		r/w
vxworks[1]	Example script for booting VxWorks Default: *${vxworks_file};setenv loadaddr 02000000;run vxworks_setargs ${bootmethod};env export -b 5000;bootvx ${loadaddr}*		r/w

[1] These MEN-specific variables can be used to provide a user-friendly way to boot the operating system: Once files and arguments are set correctly, the user can boot the operating system using commands *boot*, *run linux* or *run vxworks*.

Table 32. *U-Boot – Environment variables – Network*

Variable	Description	Default	Access
bootfile	Name of the image to load through command *tftpboot*	Empty	r/w
ethact	Controls which network interface is currently active	*eTSEC1*	r/w
ethaddr *eth1addr*	MAC addresses of Ethernet interfaces determined at start-up (see Chapter 2.7 Ethernet Interfaces on page 32). *ethaddr* refers to interface LAN1, *eth1addr* refers to LAN2. You can pass MAC addresses to the OS using the *cpenv* command.	*00:c0:3a:b1:80:00*	r
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	*192.1.1.120*	r/w
loadaddr	Default load address for commands like *tftpboot*, *loadb* etc.	*0x01000000*	r/w
netmask	Subnet mask	*255.255.255.0*	r/w
pwd	Password	Empty	r/w
serverip	TFTP server IP address; needed for *tftpboot* command	*192.1.1.22*	r/w
user	User name	Empty	r/w

Table 33. *U-Boot – Environment variables – Console*

Variable	Description	Default	Access
baudrate	Baud rate for serial console Possible values: 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	*115200*	r/w
loads_echo	If set to 1, all characters received during a serial download (using the *loads* command) are echoed back. This might be needed by some terminal emulations (like *cu*), but may just take time on others.	*1*	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 34. *U-Boot – Environment variables – Other*

Variable	Description	Default	Access
failbootcmd	Code string to be executed in case of a boot failure, e.g., during power-on self tests. Default: Empty		r/w
update[1]	Example update script, see Chapter 3.5.4 Performing an Update on page 90 Default: *askenv \"tftp,bootp or uart\";if test \"tftp\" = ${tftp,bootp or uart};then setenv bootfile /tftpboot/u-boot_"CONFIG_MEN_BOARD".bin;editenv bootfile;fi;setenv loadaddr 02000000;run ${tftp,bootp or uart}&&itest ${filesize} != 0&&protect off ffe00000 ffedffff&&erase ffe00000 +${filesize}&&cp.b ${loadaddr} ffe00000 ${filesize};protect on all*		r/w
ver	U-Boot version string passed on to the operating system Default: tbd.		r/w
watchdog	Watchdog disable. If not defined, strapping option is used. If set to *dis*, the watchdog is disabled. When you change this setting, the change will become active only after the next re-boot.	Empty	r/w

[1] This example script provides a user-friendly way to update the U-Boot main image. You can start the update using command *run update*.

3.9 U-Boot Commands

The following table gives all U-Boot commands that can be entered on the A21 U-Boot prompt. The available commands are board-specific.

You can access the 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.

Table 35. *U-Boot – Command reference*

Command	Description
?	Alias for 'help'
as	Assemble memory
askenv	Get environment variables from stdin
bdinfo	Print board info structure
boot	Boot default, i.e. run *bootcmd*
bootelf	Boot from an ELF image in memory
bootm	Boot application image from memory
bootp	Boot image via network using BOOTP/TFTP protocol
bootvx	Boot VxWorks from an ELF image
break	Set or clear a breakpoint
clocks	Print clock configuration
cmp	Compare memory
coninfo	Print console devices and information
continue	Continue from a breakpoint
cp	Copy memory
cpenv	Copy environment string, may be needed to forward some additional parameters to the OS
crc32	Calculate checksum
date	Get/set/reset date & time
defenv	Set environment variables to default values (use *saveenv* to store the changes)
dhcp	Boot image via network using DHCP/TFTP protocol
ds	Disassemble memory
echo	Echo arguments to console
editenv	Edit environment variable
eeprom	EEPROM sub-system
erase	Erase Flash memory
exit	Exit script
ext2load	Load binary file from an Ext2 file system
ext2ls	List files in a directory (default /)
fallback	Start fallback (full-featured) U-Boot image

Command	Description
fatinfo	Print information about file system
fatload	Load binary file from a DOS file system
fatls	List files in a directory (default /)
fdt	Flattened device tree utility commands
flinfo	Print Flash memory information
go	Start application at address *addr*
help	Print online help
i2c	I2C sub-system
iminfo	Print header information for application image
itest	Return true/false on integer compare
lm81	Show voltages as returned by the LM81 sensor
loadb	Load binary file over serial line (Kermit mode)
loads	Load S-Record file over serial line
loady	Load binary file over serial line (Ymodem mode)
loop	Infinite read loop on address range
loopw	Infinite write loop on address range
md	Display memory
mdc	Display memory cyclically
mii	MII utility commands
mm	Modify memory (auto-incrementing address)
mtest	Simple RAM read/write test
mw	Write to memory (fill)
mwc	Write to memory cyclically
next	Single step execution, stepping over subroutines
nfs	Boot image via network using NFS protocol
nm	Modify memory (constant address)
pci	List and access PCI Configuration Space
ping	Send ICMP *ECHO_REQUEST* to network host
printenv	Print environment variables
protect	Enable or disable Flash write protection
rarpboot	Boot image via network using RARP/TFTP protocol
rdump	Show registers
reginfo	Print register information
reiserload	Load binary file from a Reiser file system
reiserls	List files in a directory (default /)
reset	Reset the CPU
run	Run commands in an environment variable

Command	Description
saveenv	Save environment variables to persistent storage
saves	Save S-Record file over serial line
setenv	Set environment variable
setexpr	Set environment variable as the result of eval expression
showvar	Print local hush shell variables
sleep	Delay execution for some time
sntp	Synchronize RTC via network (UTC time)
source	Run script from memory
step	Single step execution
temp	Display temperature
test	Minimal test like */bin/sh*
tftpboot	Boot image via network using TFTP protocol
time	Run command and output execution time
unzip	Unzip a memory region
usb	USB sub-system
usbboot	Boot from USB device
version	Print U-Boot version
where	Print the running stack

3.10 Hardware Interfaces Not Supported by U-Boot

The standard A21 U-Boot does **not** support the following hardware interfaces:

- Handshake lines for UART COM interface(s)
- OHCI for USB (only EHCI)

4 Organization of the Board

To install software on the board or to develop low-level software it is essential to be familiar with the board's address and interrupt organization.

4.1 Memory Mappings

Table 36. *Memory mappings – local address windows*

CPU Address Range	Size	Description
0x 0000 0000..End of RAM	Up to 2 GB	DDR3 SDRAM
0x 8000 0000..9FFF FFFF	512 MB	PCIe3 memory space (FPGA)
0x A000 0000..BFFF FFFF	512 MB	PCIe2 memory space (XMC0, PMC)
0x C000 0000..DFFF FFFF	512 MB	PCIe1 memory space (XMC1)
Start of Flash ..EFFF FFFF	Up to 256 MB	NOR (boot) Flash (e.g., at 0x EC00 0000 with 64 MB, 0x EE00 0000 with 32 MB)
0x F000 0000..F7FF FFFF	128 MB	PCIe3 additional memory space
0x FA00 0000..FA01 FFFF	128 KB	FRAM
0x FFC0 0000..FFC0 FFFF	64 KB	PCIe3 I/O space
0x FFC1 0000..FFC1 FFFF	64 KB	PCIe2 I/O space
0x FFC2 0000..FFC2 FFFF	64 KB	PCIe1 I/O space
0x FFE0 0000..FEFF FFFF	1 MB	CCSR (Configuration, Control and Status Registers)

Table 37. *Memory mappings – PCI/PCIe master address map*

CPU Address Space	Interface	Mapped to PCI Space	Description
0x 8000 0000..9FFF FFFF	PCIe1	0x 8000 0000..9FFF FFFF (MEM)	PCI memory space (non-prefetchable)
0x A000 0000..BFFF FFFF	PCIe2	0x A000 0000..BFFF FFFF (MEM)	PCI memory space (non-prefetchable)
0x C000 0000..DFFF FFFF	PCIe3	0x C000 0000..DFFF FFFF (MEM)	PCI memory space (non-prefetchable)
0x F800 0000..F800 3FFF	PCIe1	0x 0000..3FFF (I/O)	PCI I/O space
0x F800 4000..F800 7FFF	PCIe2	0x 4000..7FFF (I/O)	PCI I/O space
0x F800 8000..F800 FFFF	PCIe3	0x 8000..FFFF (I/O)	PCI I/O space

Table 38. *Memory mappings – PCI/PCIe slave address map*

PCI Address Space	Mapped to CPU Space	Description
0x 0000 0000..End of RAM	0x 0000 0000..End of RAM	Prefetchable, snooping enabled

4.2 PCI Devices on PMC Bus

Table 39. *PCI Devices on PMC Bus*

Device Number	Vendor ID	Device ID	Function	INTA led to
Assigned dynamically, depends on plugged PMC/XMC types	Depends on mezzanine module		PMC 0	PCI INTA
			PMC 1	PCI INTD

4.3 I2C Devices

Table 40. *I2C devices*

Address	Function
0x58	LM81 voltage monitor
0xA2	RTC
0xA8	CPU EEPROM
0xAA	XMC slot 0 EEPROM (adjusted by GA pins)
0xAC	XMC slot 1 EEPROM (adjusted by GA pins)

4.4 M-Module Addresses

You can find the M-Module address map and register description in Chapter 2.9.2 Addressing the M-Modules on page 36.

4.5 VMEbus Addresses

You can find all VMEbus address maps and register descriptions in Chapter 2.13 VMEbus Interface on page 49.

4.6 MSI Interrupts

The A21 supports PCIe MSI interrupts with the following vector mapping:

Table 41. *MSI interrupts*

Interrupt Source	MSI Vector Number
Reserved for custom FPGA function; default: 16Z127_GPIO	0
M-Module Slot 1 (Interrupt Control Register (0x1701, 0x1B01, 0x1F01) (read/write))	1
M-Module Slot 2 (Interrupt Control Register (0x1701, 0x1B01, 0x1F01) (read/write))	2
M-Module Slot 3 (Interrupt Control Register (0x1701, 0x1B01, 0x1F01) (read/write))	3
VMEbus interrupt request ACFAIL (ISTAT – Interrupt Status Register (0x0008) (read/write))	4
VMEbus interrupt request level 1 (ISTAT – Interrupt Status Register (0x0008) (read/write))	5
VMEbus interrupt request level 2 (ISTAT – Interrupt Status Register (0x0008) (read/write))	6
VMEbus interrupt request level 3 (ISTAT – Interrupt Status Register (0x0008) (read/write))	7
VMEbus interrupt request level 4 (ISTAT – Interrupt Status Register (0x0008) (read/write))	8
VMEbus interrupt request level 5 (ISTAT – Interrupt Status Register (0x0008) (read/write))	9
VMEbus interrupt request level 6 (ISTAT – Interrupt Status Register (0x0008) (read/write))	10
VMEbus interrupt request level 7 (ISTAT – Interrupt Status Register (0x0008) (read/write))	11
VMEbus bus error (MSTR – Master Control Register (0x0010) (read/write))	12
VMEbus DMA interrupt request (DMASTA – DMA Status Register (0x002C) (read/write))	13
VMEbus Location Monitor 0 request (LOCADDR_0 – Location Monitor Address Register (0x0040) (read/write))	14
VMEbus Location Monitor 1 request (LOCADDR_1 – Location Monitor Address Register (0x0044) (read/write))	15
VMEbus Mailbox request (MAIL_IRQ – Mailbox Interrupt Request Register (0x0024) (read/write))	16

5 Maintenance

5.1 Lithium Battery

This board may contain a lithium battery (available as an option). There is a danger of explosion if the battery is incorrectly replaced!

Replace only with the same or equivalent type.

- Manufacturer: Renata
- Type: CR2032
- Capacity: 220 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 12. *Position of optional lithium battery on A21B*

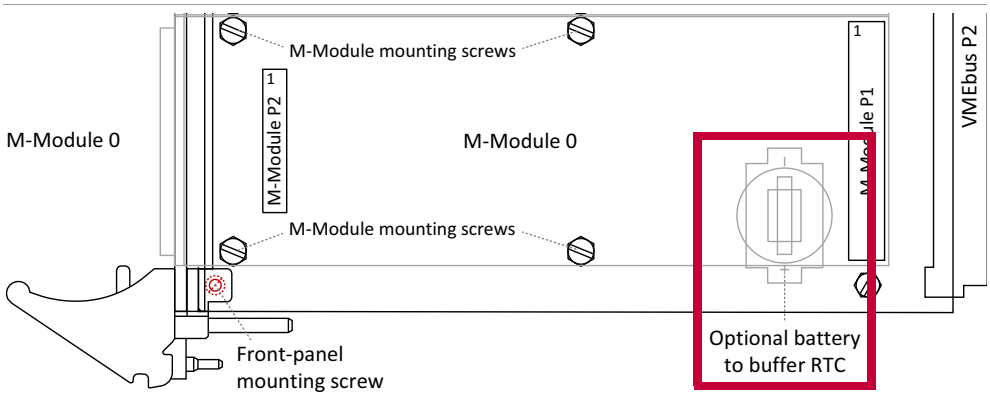
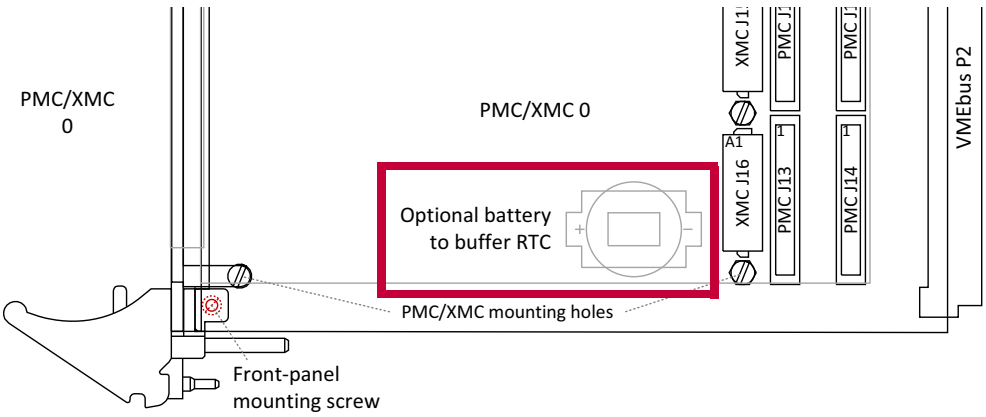


Figure 13. *Position of optional lithium battery on A21C*



6 Appendix

6.1 Literature and Web Resources

- A21B data sheet with up-to-date information and documentation:
  www.men.de/products/01A021B.html
- A21C data sheet with up-to-date information and documentation:
  www.men.de/products/01A021C.html

6.1.1 CPU

- Freescale QorIQ Processing Platforms:
  www.freescale.com/webapp/sps/site/homepage.jsp?code=QORIQ_HOME
- QorIQ P1022/P1013:
  www.freescale.com/webapp/sps/site/prod_summary.jsp?code=P1022

6.1.2 microSD (SDHC)

- SD association pages on SDHC
  www.sdcard.org/developers/tech/sdhc/
- Wikipedia article with many references
  http://en.wikipedia.org/wiki/Secure_Digital

6.1.3 SATA

- Serial ATA International Organization (SATA-IO)
  www.serialata.org

6.1.4 Ethernet

- ANSI/IEEE 802.3-1996, Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications; 1996; IEEE
  www.ieee.org
- Charles Spurgeon's Ethernet Web Site
  Extensive information about Ethernet (IEEE 802.3) local area network (LAN) technology.
  www.ethermanage.com/ethernet/
- InterOperability Laboratory, University of New Hampshire
  This page covers general Ethernet technology.
  www.iol.unh.edu/services/testing/ethernet/training/

6.1.5 M-Modules

- M-Module Standard:
  ANSI/VITA 12-1996, M-Module Specification;
  VMEbus International Trade Association
  www.vita.com

6.1.6 PMC

- PMC specification:
  Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC, 1386.1; 1995; IEEE
  www.ieee.org

6.1.7 XMC

- XMC PCI Express Protocol Layer Standard
  VITA 42.3-2006; June 2006
  VMEbus International Trade Association
  www.vita.com
- Standard for VITA 42.0 XMC
  VITA 42.0-2008; December 2008
  VMEbus International Trade Association
  www.vita.com

6.1.8 VMEbus

- VMEbus General:
  - The VMEbus Specification, 1989
  - The VMEbus Handbook, Wade D. Peterson, 1989

  VMEbus International Trade Association
  www.vita.com

6.2 Finding out the Product's Article Number, Revision and Serial Number

MEN user documentation may describe several different models and/or design revisions of the A21. You can find information on the article number, the design revision and the serial number on two labels attached to the board.

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

If you need support, you should communicate these numbers to MEN.

Figure 14. *Labels giving the product's article number, revision and serial number*