

## **PCI to VME Interface**

---

## **User's Manual**

## General Remarks

The only purpose of this manual is a description of the product. It must not be interpreted a declaration of conformity for this product including the product and software.

**W-IE-Ne-R** revises this product and manual without notice. Differences of the description in manual and product are possible.

**W-IE-Ne-R** excludes completely any liability for loss of profits, loss of business, loss of use or data, interrupt of business, or for indirect, special incidental, or consequential damages of any kind, even if **W-IE-Ne-R** has been advises of the possibility of such damages arising from any defect or error in this manual or product.

Any use of the product which may influence health of human beings requires the express written permission of **W-IE-Ne-R**.

Products mentioned in this manual are mentioned for identification purposes only. Product names appearing in this manual may or may not be registered trademarks or copyrights of their respective companies.

No part of this product, including the product and the software may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means with the express written permission of **W-IE-Ne-R**.

VMEMM3, VMEMM6 and PCIADA are designed by ARW Elektronik, Germany.

**Table of contents:**

**1. PCI-VME INTERFACE: GENERAL DESCRIPTION ..... 1**

**2. PCI INTERFACE CARD PCIADA ..... 2**

    2.1. OPERATION..... 2

    2.2. MAJOR LCREGISTERS ..... 2

    2.3. TIMEOUT..... 4

    2.4. BLOCK DIAGRAM ..... 4

**3. THE VMEMM BOARD ..... 5**

    3.1. JUMPER SETTINGS ..... 6

    3.2. OPERATION..... 7

    3.3. VMEMM AREA..... 7

    3.4. VMEMM CONTROL AND STATUS REGISTERS ..... 8

    3.5. VIC68A-REGISTER ..... 10

    3.6. INTERRUPTS..... 11

**4. OPERATING THE INTERFACE..... 13**

    4.1. INSTALLATION..... 13

    4.2. INITIALIZING THE INTERFACE ..... 13

    4.3. VME ACCESS ..... 14

    4.4. INTERRUPT HANDLING..... 15

    4.5. UNINSTALLING THE INTERFACE ..... 16

**APPENDIX A : LAYOUT OF PCIADA AND VMEMM ..... 17**

**APPENDIX B : POWER REQUIREMENTS OF PCIADA: ..... 18**

**APPENDIX C : ACCESS TIMES..... 18**

**List of figures:**

FIGURE 1: FRONT PANEL OF VMEMM6 AND VMEMM3 ..... 5

FIGURE 2: LOCATION OF JUMPERS OF VMEMM ..... **ERROR! BOOKMARK NOT DEFINED.**

FIGURE 3: INITIALIZATION OF THE VME INTERFACE ..... 14

FIGURE 4: STEPS TO ACCESS THE VME BUS. .... 15

FIGURE 5: HANDLING OF AN INTERRUPT I ..... 16

FIGURE 6: STEPS TO UNINSTALL THE INTERFACE..... 16

FIGURE 7: LAYOUT OF VMEMM..... 17

FIGURE 8: LAYOUT OF THE PCIADA BOARD ..... 18

**Legend:**

Symbols for jumper settings:

- I           =     jumper is installed
- :           =     jumper is not installed

## 1. PCI-VME Interface: General Description

With the help of the PCI-VME interface which consists of the PCI card PCIADA and the VME main board VMEMM users of the VME bus profit of the technical success which takes place in the IBM/PC<sup>1</sup> development.

A huge variety of software is available for a PC. In parallel it's performance was constantly improved. Modern operating systems were developed which turn the PC into a powerful workstation.

On the other hand the number of UNIX<sup>2</sup> workstations which are equipped with PCI increases strongly. The PCI-VME interface can be used in this systems, too.

Drivers and programming tools for PCI-VME are provided for different operating systems and programming languages.

A driver for Windows offers an easy access to the VME bus. Using `pvmon.exe` a quick check of your VME hardware is possible.

For C and Turbo Pascal users the Pascal and c libraries provide easy routines to operate VME modules.

Using the Linux driver all advantages of a multi user and multi tasking operating system can be used for VME operations.

VMEMM is available as a 3U (VMEMM-3) and 6U VME module (VMEMM-6). It is linked via a standard SCSI 2 cable to the PCIADA card. Address modes A16, A24, A32 / D8, D16, D32 (A16, A24 / D8, D16 for VMEMM-3) are possible.

PCIADA supports 8, 16 and 32 – bit PCI bus slave access. It provides programmable interrupt generation on PC.

---

<sup>1</sup> IBM/PC is a trademark of IBM Corporation

<sup>2</sup> UNIX is a trademark of AT&T

## 2. PCI interface card PCIADA

### 2.1. Operation

The major part of the PCIADA card represents the interface chip PCI9050 by *PLX-Technology*. It manages PCI bus communications.

At booting time necessary settings are configured by PCI auto setup. No jumpers have to be set on the interface card. All PCIADA information such as memory and IRQ requirements are stored in an EEPROM called the PCI Configuration Register (PCR). Some IDs can be read out from PCR:

**Table 1:** Basic values of the PCI Configuration Register (PCR).

ID	fixed values
Vendor ID	0x10B5
Device ID	0x9050
Subsystem Vendor ID	0x9050
Subsystem Device ID	0x1167

During PCI setup PCIADA calls for three different memory areas:

1. 54 Byte Local Configuration Register (LCR) of the I/O area. Different control status register and base addresses are available in the LCR. *Note:* If no LCR is available in the I/O area please refer to the LCR in the memory area.
2. LCR in the memory area.
3. 8 kByte of the memory area for direct access to VME and the local VMEMM memory area.

### 2.2. Major LCR Registers

Base Addresses for the memory areas described in section 2.1 will be available after booting of the computer. They can be obtained via BIOS from the PLX chip. Most important are:

The **Interrupt Control / Status Register (INTCSR)** handles interrupts of the VIC68A chip (installed on VMEMM), interrupts of PCIADA itself, PCI interrupts and software interrupts. Sources for the VIC68 A interrupt which are summarized in the INTERRUPT 1 bits are:

1. seven interrupt requests of the VME bus
2. Mailbox interrupts in the VIC068A chip by a second VME-Master
3. interrupts caused by a successful interrupt generation
4. interrupts caused by SYSFAIL or ACFAIL
5. bus timeout interrupt
6. arbitration timeout interrupt
7. VIC68A timer interrupts
8. interrupt caused by a manual reset

The INTERRUPT2 bits controls local events of the PCIADA card. Interrupts are caused by

1. Timeout while accessing the VME bus. Timeout is fixed to about 35  $\mu$ s.
2. Cable is not connected.
3. The VME power supply is not switched on.
4. VMEMM access has not yet been switch on. Before any access to VMEMM user bit I/O 2 has to be set.

Table 1 describes all bits of the INTCSR.

**Table 2:** Contents of the Interrupt Control / Status Register (INTCSR). Base address: LCR base address + 0x4c

Bit		RD	WR	after Init
0	Local Interrupt 1 enable / 1 = enable / 0 = disable / source = VMEMM	yes	yes	1
1	Local Interrupt 1 polarity / 1 = active high / 0 = active low	yes	yes	0
2	Local Interrupt 1 status / 1 = active. 0 = not active	yes	no	0
3	Local Interrupt 2 enable / 1 = enable / 0 = disable / Source = PCIADA	yes	yes	1
4	Local Interrupt 2 polarity / 1 = active high / 0 = active low	yes	yes	0
5	Local Interrupt 2 status / 1 = active. 0 = not active	yes	no	0
6	PCI Interrupt enable / 1 = enable / 0 = disable (Source = global)	yes	yes	0
7	Software Interrupt / 1 = generate Interrupt	yes	yes	0
16..8	not used	yes	no	00000000 b

The **User I / O Register (CNTRL)** is divided into three parts which are summarized in Table 3. Part one (USER I/O1) **must not** be modified. The register USER I/O2

1. locks VMEMM during boot time. to prevent any access of the Operating System.
2. resets Interrupt 2 if it is deleted.

Register USER I/O3 monitors the status of the VME crate (on / off) and the cable (connected / not connected).

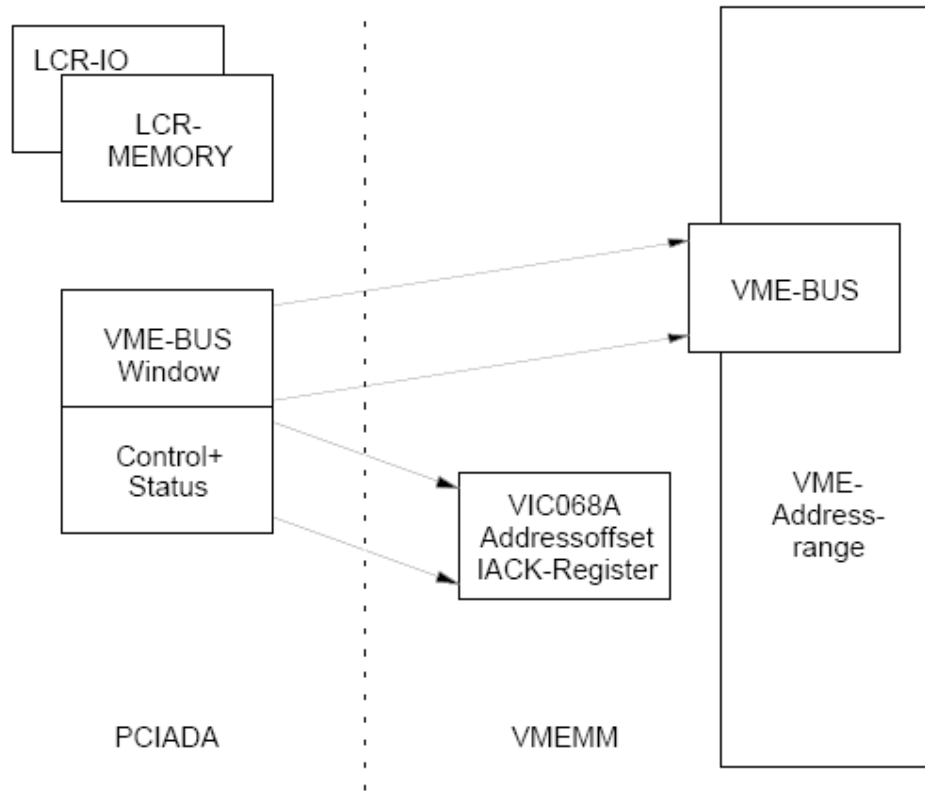
**Table 3:** The User I / O Register (CNTRL). LCR base address + 0x50. To enable access to VMEMM write 0x4184 into CNTRL, 0x4084 to disable.

Byte		RD	WR	after Init
5..0	do not modify !	yes	yes	000100 b
6	USER I/O2 Type, must always be 0	yes	yes	0
7	USER I/O2 Direction, must always be 1 = output	yes	yes	1
8	USER I/O2 output, 0 = disable access to VMEMM, 1 = enable access	yes	yes	0
9	USER I/O3 Type, must always be 0	yes	yes	0
10	USER I/O3 Direction, must always be 0 = input	yes	yes	0
11	USER I/O3 Input, 0 = VMEMM failed , 1 = VMEMM OK	yes	no	0
15..12	do not modify	yes	yes	0100 b

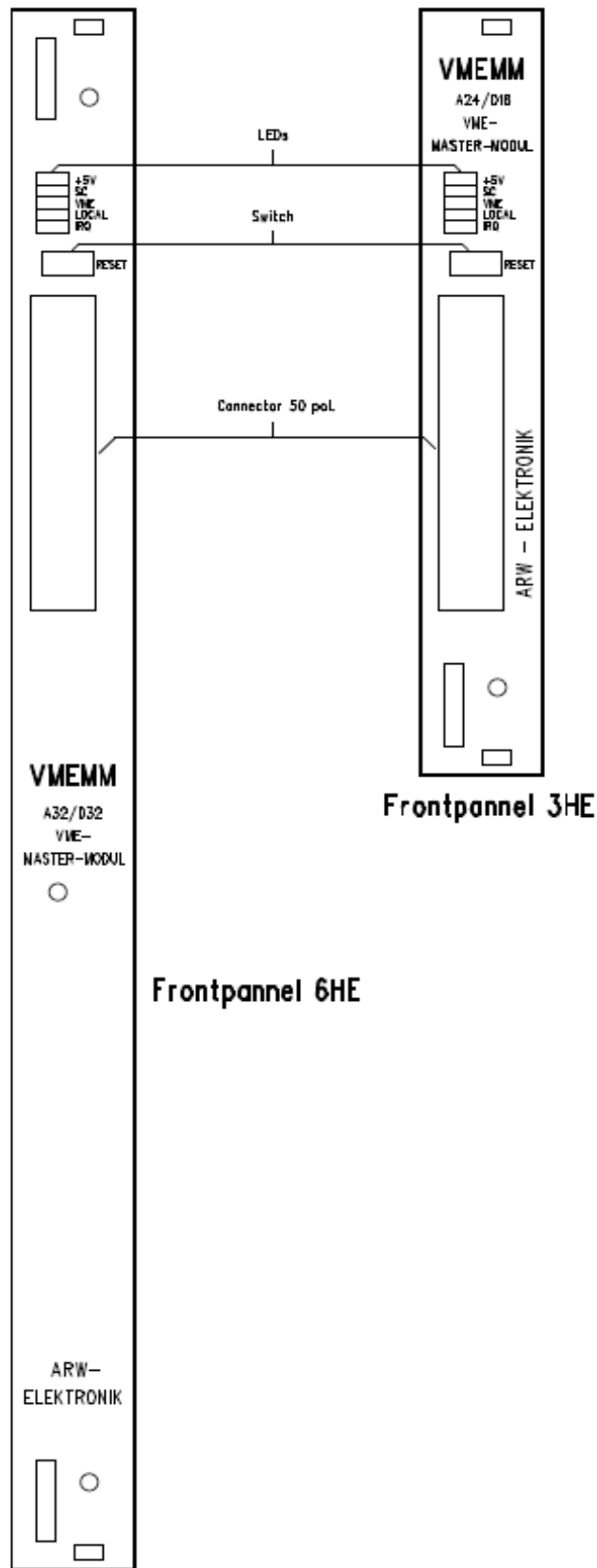
### 2.3. Timeout

An **Access Timeout** is implemented in the PCIADA interface, which is currently set to about 35  $\mu$ s. If the VMEMM is switched of or disconnected during an access timeout causes an PCIADA Interrupt 2.

### 2.4. Block Diagram



3. The VMEMM board



**Figure 1:** front panel of VMEMM6 and VMEMM3

### 3.1. Jumper Settings

VMEMM can be operated as a VME system controller module which is controlled by the **SC jumper** (Table 4). In this case it provides slot #1 features like BUS arbitration, BUS timeout and system reset to VMEbus. Insert VMEMM into slot 1 if it should work as VME system controller. Jumper position can be read out by VMEMM-STATUS.

**Table 4:** Settings of the SC jumper (readable by VMEMM-Status)

J402	Function
I	VMEMM works as System Controller (default)
:	VMEMM works as Slave

Choose different **Board Numbers** which are selected via jumper J304 to J301 if more than

**Table 5:** Settings of the **Board Number BN** Jumpers, which can be read out by the register VMEMM-Status

J304	J303	J302	J301	Function
BN3	BN2	BN1	BN0	Bits of BN (BN0: Low Bit)
I	I	I	:	Default value BN = 1

one PCIADA is installed in one PC.

VMEMM is available as a 6U and a 3U board. Long word access is not possible for the 3U version (J2 / P2 does not exist). The VIC068 A chip checks the WORD Jumper before a long word access. It is the task of the supporting software to split longword accesses to WORD data paths into 2 WORD accesses. The VMEbus is not capable of the autosize feature provided by some 68XXX family members.

VMEbus allows multiple masters to share the data transfer bus which needs a special communication between all master boards to organize data transfer. Inter process communication are important for this communication. The address of this register is set by the **VME Short Address Jumper**.

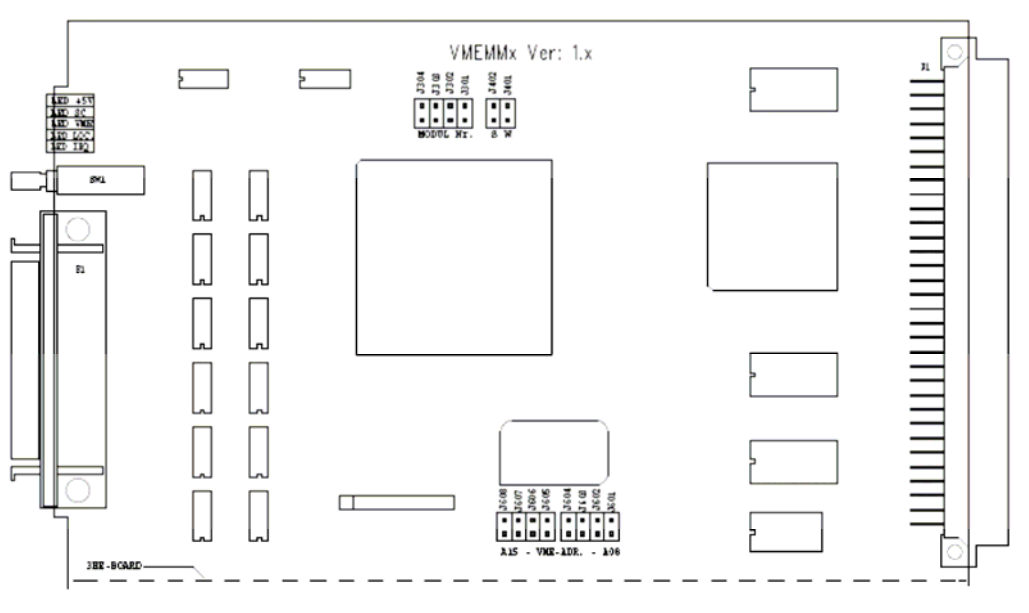


Figure 2: **Location of jumpers of VMEMM.**

**Note:** Watch the setting of the VME Short Address. The address range can not be masked. An access by VMEMM itself will cause a `_BERR*` and will not be ended by `_DTACK`.

**Table 6:** Settings of the WORD jumper. It is readable via VMEMM-Status.

J401	Function	
I	VIC-Chip in word mode	(VMEMM3 default)
:	VIC-Chip in long word-mode	(VMEMM6 default)

**Table 7:** VME Short Address Jumper. Note: An inserted jumper means bit = 0.

J608	J607	J606	J605	J604	J603	J602	J601	Function
A15	A14	A13	A12	A11	A10	A9	A8	VME Short Address
:	I	:	I	:	I	:	I	VME Short = 0xAA00 (default)

### 3.2. Operation

PCIADA accesses a region of the VMEMM memory directly. This region is divided into four blocks:

1. **VMEMM:** VME addresses A11 to A31 and control signal RMC are stored here and VME reset is activated. Signals VMEMM-Stat, IRQ Status and A11 to A31 are readable.
2. **VIC068A:** This is the part of the VMEbus Interface Controller which controls the VMEbus.
3. **Interrupt Vector (INTVEC):** An interrupt acknowledge cycle is stated if one byte of this area is accessed.
4. **VME:** Every time this area is accessed a VMEbus transfer will be started. Address lines A11 to A1 equals PCI address lines A11 to A1. VME address lines A31 to A12 are stored in the VMEADR register.

Following sections describe each memory block in details.

### 3.3. VMEMM area

The size of this block is 8 kByte and it is mapped into the PCI memory. The PCI base address is defined by the BIOS during boot time.

**Table 8:** Contents of VMEMM which is mapped into the PCI memory area.

Base-Address	region / access	WR-function	RD-function
+ \$1FFF to + \$1000	VME / byte A0 = 0 VME / byte A0 = 1 VME / word VME / lword	D07..D00 > VME-D15..D08 D07..D00 > VME-D07..D00 D15..D00 > VME-D15..D00 D31..D00 > VME-D31..D00	D15..D00 < VME-D15..D08 D07..D00 < VME-D07..D00 D15..D00 < VME-D15..D00 D31..D00 < VME-D31..D00
VMEBASE			
+ \$080F	INTVEC / byte	-	D07..D00 < Iack 7 Vector
+ \$080D	INTVEC / byte	-	D07..D00 < Iack 6 Vector
+ \$080A	INTVEC / byte	-	D07..D00 < Iack 5 Vector
+ \$0809	INTVEC / byte	-	D07..D00 < Iack 4 Vector
+ \$0807	INTVEC / byte	-	D07..D00 < Iack 3 Vector
+ \$0805	INTVEC / byte	-	D07..D00 < Iack 2 Vector
+ \$0803	INTVEC / byte	-	D07..D00 < Iack 1 Vector
VECBASE			
+ \$04E3 to + \$0403	VIC / byte VIC / byte VIC / byte	VIC068A Reg. VIC068A Reg. VIC068A Reg.	VIC068A Reg. VIC068A Reg. VIC068A Reg.
VICBASE			
(+ \$000a)	VMEMM / word	VMEADDRReg high D15..D00 > VME-A31..A16	VMEADRStat high D15..D00 = VME-A31..A16
(+ \$0008)	VMEMM / word	VMEADDRReg low D15..D12 > VME-A15..A12 D11..D00 > x	VMEADRStat low D15..D12 = VME-A15..A12 D11..D00 = 0
+ \$0008	VMEMM / lword	VMEADDRReg D31..D12 > VME-A31..A12 D11..D00 > x	VMEADRStat D31..D12 = VME-A31..A12 D11..D00 = 0
+ \$0004	VMEMM / word	VMEVICReset D15..D00 > h000a 1* D15..D00 > h0005 2*	VMEMMIRQStat D0 = IRQ > 1= aktiv D1 = IPL0 D2 = IPL1 D3 = IPL2 D15..D04 = 0
+ \$0000	VMEMM / word	VMEVICReg D0 > RMC-VIC048A	VMEMMStat D0 < RMC-VIC068A 3* D1 = BLT-VIC068A 4* D2 < WORD Jumper sel. D3 < SC Jumper sel. D07..D04 = Modul-Number D11..D08 = FPGA-Revision D15..D12 = module-Type
PCI-MEMBASE			

1\* VIC „Global Reset“ ( refer to VIC068A Manual )

2\* VIC „Internal Reset“ ( refer to VIC068A Manual )

3\* Read Modify Cycle (refer to VIC068A Manual )

4\* Blocktransfer = fix disabled

### 3.4. VMEMM Control and Status registers

You find important information about configuration and controlling of the interface in the VMEMM Control and Status registers.

Register **VMEADDR** contains high bytes of VMEbus addresses (A31 to A12) of the following accesses. Data bit D11 to D0 are ignored during write and replied as '0' during read.

**Note:** Addresses A31 to A24 (A31 to A16) have no meaning for standard (short I/O) VMEbus accesses.

**Note:** VMEADDR is read- and writeable in word or long word mode.

**Table 9:** Contents of the **VMEADDR** register for word or long word read- write access.

Bit	extended	RD	WR	after Init
31..12	most significant address bit A31 .. A12	yes	yes	X
11..0	write: no meaning, read returns 0	yes	yes	0
Bit	Standard	RD	WR	after Init
31..24	no meaning	yes	yes	X
23..12	most significant address bit A23 .. A12	yes	yes	X
11..0	write: no meaning, read returns 0	yes	yes	0
Bit	Short	RD	WR	after Init
31..16	no meaning	yes	yes	X
15..12	most significant address bit A15 .. A12	yes	yes	X
11..0	write: don't care, read returns 0	yes	yes	0

Reset functions are handled by the **VMEReset** register. Write data word 0x000a into **VMER** to perform a VMEbus reset and a global reset of the interface which equals a power up reset. After this a reinitialization of **VMEMM** is mandatory.

The performing reset needs about 300 ms to finish its operation. During this time it is not possible read back any **VIC68A** register. This feature can be used to check the end of the internal reset. For example the software writes 0xff into the **AMSR** register of the **VIC68A**. Then reset is raised and after this the software polls the **AMSR** register until 0x00 is read back. This tells you the end of the internal reset.

Pushing the reset button on the front panel will cause an interrupt. Further actions have to be preformed by software. This method allows an individual programming of the button.

Interrupt information of the **VIC068A** chip are stored in the **VMEMMIRQ** status register. An interrupt request sets D0 to '1'. The request which has highest priority is encoded in D3 to D1.

**Note:** The index of the corresponding interrupt vector can be calculated very fast since no shift of **VMEMMIRQ** shift is necessary.

**Table 10:** Register **VMEMMIRQ**Status.

bit		RD	WR	after Init
15..4	0	yes	no	0
3..1	interrupt level	yes	no	0
0	0 = no interrupt pending, 1 = interrupt pending	yes	no	0

The **VMEVIC**Register used to realize uninterruptible cycles. If **RMC** (= D0) is set signal **\_AS** (Address Strobe) will remain active for all following VMEbus accesses. Bits D15 to D1 must be set to '0'.

**Note:** VMEbus allows only single address uninterruptible cycles. The address must not be changed during the uninterruptible cycle.

**Table 11:** Register **VMEVIC**.

Bit		RD	WR	after Init
15..1	must be 0	no	yes	X
0	<b>RMC</b> , 0 = No Read-Modify-Write-Cycle, 1 = Read-Modify-Write-Cycle	no	yes	0

Various settings of VMEMM are stored in register **VMEMMStatus**. Refer to section 3.1 for WORD and Module Number

**Table 12:** Register **VMEMMStatus**.

Bit		RD	WR	after Init
15..12	module type Identification, 0001b for VMEMM-Modul	yes	no	0001b
11..8	FPGA-Revision	yes	no	xxxx
7..4	module number, coding of Jumpers J304..J301	yes	no	Jumpers
3	SC – status of system controller Jumper J402	yes	no	Jumpers
2	WORD – status of word-path jumper J401	yes	no	Jumpers
1	BLT (fix 0)	yes	no	0
0	RMC (status of VMEVICReg RMC)	yes	no	0

### 3.5. VIC68A-Register

This section gives a short description of some VIC68 A registers. For further information please refer to the Cypress VIC068 A data sheet which is distributed with this interface.

Registers can be accessed in byte mode. A 4 byte offset has to be considered for address calculations (e. g. VMEMM\_base + 0x403 + 0x0, VMEMM\_base + 0x403 + 0x4, ...).

The Address Modifier for the next VMEbus access is stored in the Address Modifier Source Register (**AMSR**). It does not influence Interrupt Acknowledge Cycles.

**Table 13:** Register **AMSR**. VIC base address + 0xB7 (byte access only).

Bit		RD	WR	after Init
7	must be 0	yes	yes	0
6	must be 0	yes	yes	0
5..0	AM5 .. AM0	yes	yes	00000

The VIC chip generates a SYSFAIL after each reset of the VMEbus which is deactivated by the Interprocessor Communication Register number 7 (**ICR7**).

**Table 14:** Register **ICR7**. VIC-base address + 0x7F (byte-access only).

Bit		RD	WR	after Init
7	SYSFAIL-MASK, 1 deaktiviert SYSFAIL	yes	yes	0
6..0	further functions are described in the VIC68 A manual	yes	yes	00x0000

Register **TTR** handles the VMEbus timeout. Values smaller than the PCIADA timeout e. g. 4 or 16  $\mu$ s should be inserted.

**Table 15:** Timeout coding.

Timeout / $\mu$ s	Bit-Code (Bits 7..5 or 4..2)
4	0
16	1
32	2
switched off (infinite)	7

**Table 16:** Register TTR. VIC base address + 0xA3 (byte access only)

Bit		RD	WR	after Init
7..5	VME-BUS Timeout Period (see table below)	yes	yes	32 usec
4..2	Local BUS Timeout Period (see table below)	yes	yes	32 usec
1	Arbitration Timeout detected; 1 = detected	yes	no	
0	With VME-BUS Acquisition Time included	yes	yes	not incl.

Events VMEbus timeout (local interrupt number 7) and “front panel button pushed” (local interrupt number 6) are processed via local interrupts. Timeout causes an interrupt number 7 and the button causes interrupt 6. For this reason it is necessary to setup registers **LICRx** (Local Interrupt Control Register) and **LIVBR** (Local Interrupt Vector Base Register). The upper 5 bits of the interrupt vector are stored in LEIVBR. Lower bits codes the source layer of the causing interrupt. Please refer to section 3.6 for further information.

**Table 17:** Register LICR. VIC base address 0x3B / 0x3F (byte access only).

Bits		RD	WR	after Init
7	Interrupt Mask; 0 = clear, 1 = masked off	yes	yes	1
6	Polarity of input; 1 = high or rising edge; use always a falling edge	yes	yes	
5	Edge or level sensitive input; 1 = edge sensitive input; use always edge	yes	yes	
4	Autovector enable; Must be 1; LIVBR supplies vector	yes	yes	
3	Interrupt status; 0 = interrupt is asserted	yes	no	
2..0	Interrupt level to map; always map to the same level as the input	yes	yes	

### 3.6. Interrupts

VMEbus accesses causing a VMEbus timeout or a BERR\* signal are treated as a normal access by VMEMM. But they result in a local interrupt number 7 which has to be handled by software.

Pushing the front panel button causes only a local interrupt number 6. All specific reset functions have to be defined in the software.

For programming the VIC68A a falling slope at the input LIRQ7 (LICR6) has to be considered.

It is mandatory to convert PCIADA local interrupts into vectors before any other action is performed. Usually internal events which are caused by the VME interface itself generate vectors between 0x00 to 0x0F. Other vectors (0x40 to 0xff) are reserved to be sourced from VMEbus hosted peripherals.

**Note:** Local interrupts expect VMEMM to generate an interrupt vector.

**Note:** The PCIADA timeout interrupt has to be mirrored in interrupt vector number 1.

**Table 18:** Coding of interrupt signals.

<b>Interrupt source</b>	<b>vector no.</b>
Interrupt caused by PCIADA (timeout)	1
Clock tick interrupt generator	2
Reset button at front panel	6
VMEbus Timeout (Bus-Error)	7
Interprocess communication global switch #0	8
Interprocess communication global switch #1	9
Interprocess communication global switch #2	10
Interprocess communication global switch #3	11
Interprocess communication module switch #0	12
Interprocess communication module switch #1	13
Interprocess communication module switch #2	14
Interprocess communication module switch #3	15
ACFAIL asserted	16
Write post Fail	17
Arbitration Timeout	18
SYSFAIL asserted	19
VMEbus Interrupter acknowledge	20

## 4. Operating the Interface

### 4.1. Installation

**Note:** To protect interface and your computer against damage from static electricity follow some major precautions:

1. Before unpacking the interface be sure that your working area is discharged.
2. Switch off and unplug computer and VME crate.
3. Be sure that VME crate and Computer are on the same electrical potential.

For the installation procedure stay close to the following procedure:

1. Check all jumper settings. Refer to section 3.1 for VMEMM. There are no jumpers on PCIADA.
2. Insert PCIADA into your Computer and fix it by a screw.
3. Insert VMEMM into your VME Crate and tight the screws.
4. Connect the interface cable to PCIADA and VMEMM.
5. To install the PCI-VME Windows95<sup>3</sup> driver follow instructions of the manual.
6. Windows95 will recognize PCIADA as new hardware and will ask for a driver. Activate "do not install a driver" and press "o.k."
7. If you use Pascal drivers you have to edit the `emm386.exe` in `config.sys`. You will find details in `d:\dos\pascal\readme.txt`.
8. Check your interface using the supplied test programs.

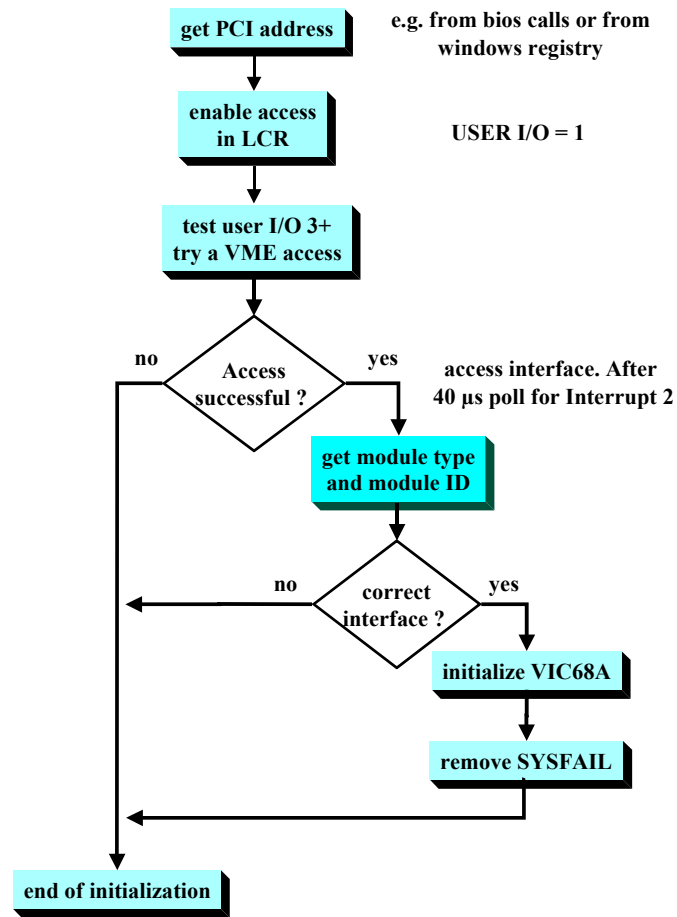
### 4.2. Initializing the interface

Necessary steps to initialize the interface are summarized in Figure 3.

For further information please refer to the supplied code written in Turbo Pascal (DOS) and C++ (Windows 95).

---

<sup>3</sup> Windos95 is a trademark of Microsoft Corporation



**Figure 3:** Initialization of the VME interface

#### 4.3. VME access

Basic steps to access the VMEbus are shown in Figure 4

A VME access which causes a timeout or a VME BUS Error ends it's cycle normally. The Timeout Interrupt can be used to identify these events. Alternatively, you can check the Interrupt 1 Status Bit.

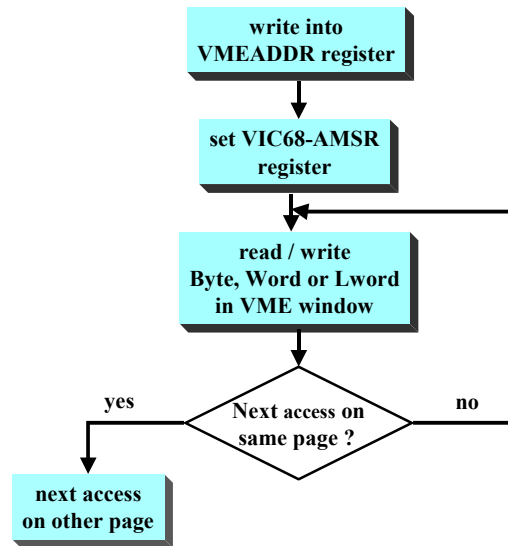


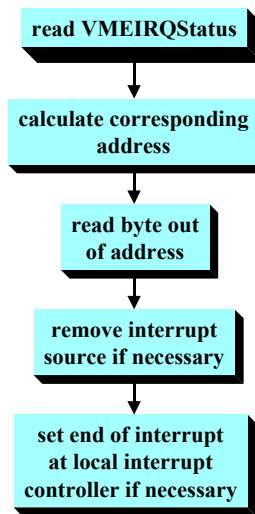
Figure 4: Steps to access the VME bus.

#### 4.4. Interrupt handling

VME Interrupts can be handled in two different ways. Either the software polls the corresponding interrupt register or they are converted into interrupts for the PCIADA host computer.

There are two different interrupt sources which cause a PCI interrupt:

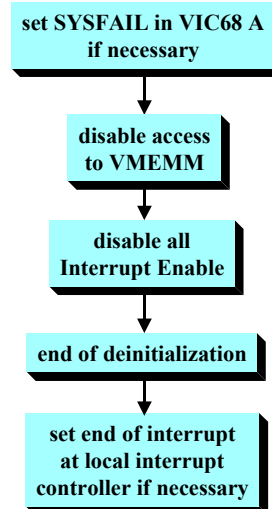
1. All VME Interrupts are summarized in Local Interrupt 1. Interrupt Priorities have to be defined in the VIC68A chip except for Interrupts with a certain VME priority which require a high local priority and the VME BUS Error which is fixed to local interrupt level 7. Each Local Interrupt 1 has to be followed by a readout of a byte interrupt vector (Figure 5).
2. The local Timeout Interrupt on PCIADA causes a Local Interrupt 2. The value is fixed on 35  $\mu$ s after the access was started. The interrupt finishes the access resulting into an indefinite value.



**Figure 5:** Handling of an Interrupt 1

#### 4.5. Uninstalling the interface

Figure 6 describes how to uninstall the interface.



**Figure 6:** Steps to uninstall the interface

APPENDIX A: Layout of PCIADA and VMEMM

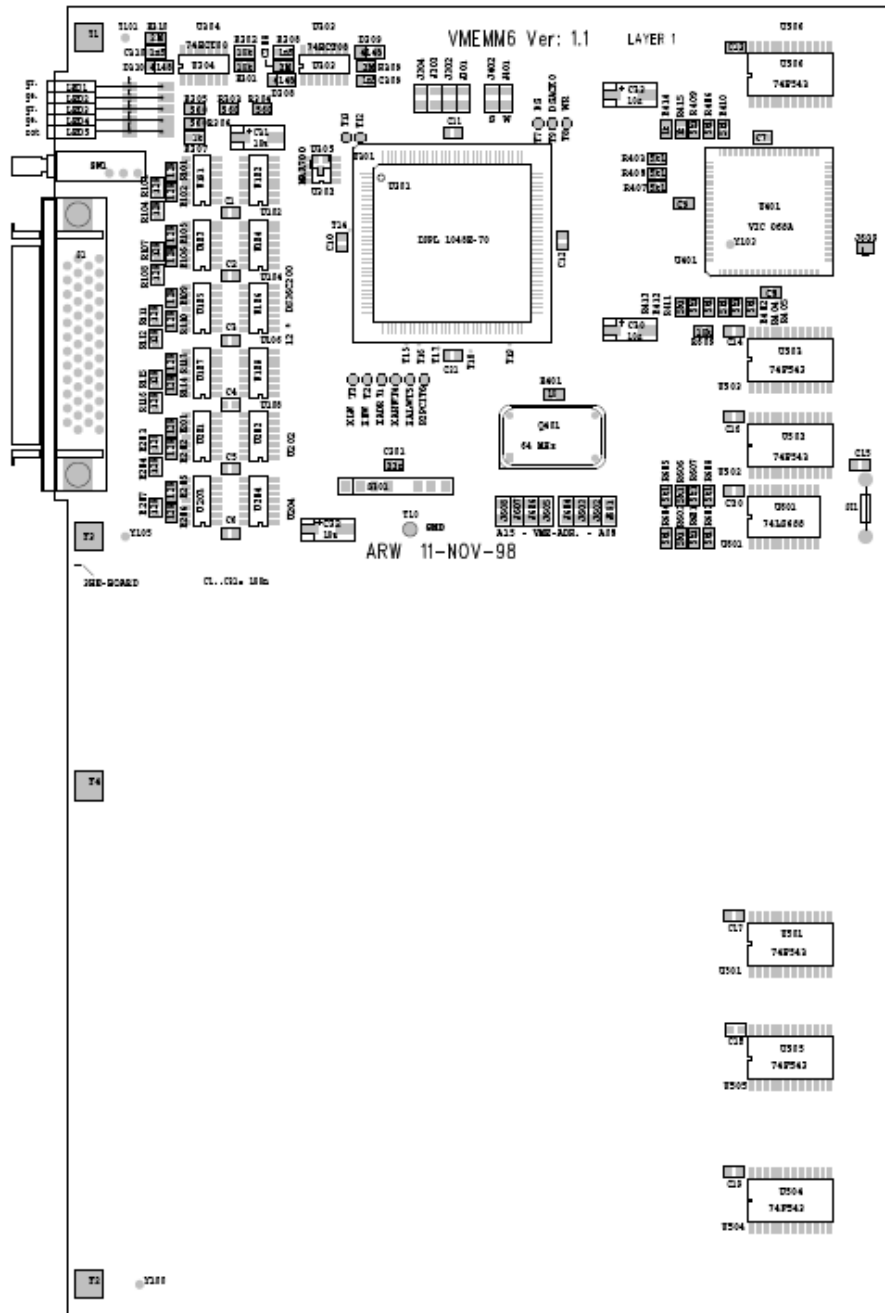
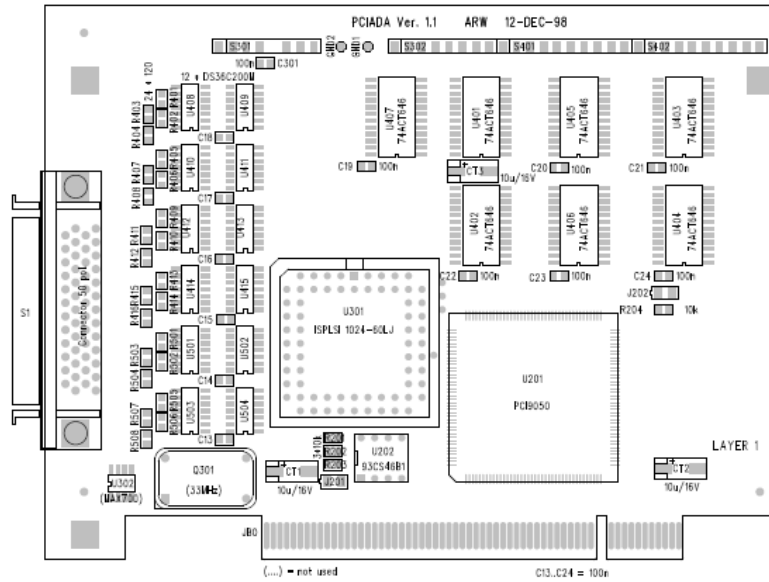


Figure 7: Layout of VMEMM



**Figure 8:** Layout of the PCIADA board

**APPENDIX B:** Power requirements of PCIADA:

- Voltage: + 5V
- Current: ≈ 0.5 A
- Power: ≈ 2.5 W

**APPENDIX C:** Access times

Access	WR-time / ns	RD-time / ns
PCI to VMEMM VIC. Reg. D8	600	1000
PCI to VMEMM Adr. Reg. D16	450	950
PCI to VMEMM Adr. Reg. D32	500	1300
PCI to VME-D8 (DS to DTACK ca. 20 ns)	660	1200
PCI to VME-D16 (DS to DTACK ca. 20 ns)	660	1200
PCI to VME-D32 (DS to DTACK ca. 20 ns)	750	1500