PCI-Bus to VMEbus Interface
Type: **PCI-VME**
Version 1.1

ARW - Elektronik
Schlehenweg 1
D-69168 Wiesloch
Germany
Tel. 06222/50816

Veränderungen

20.07.1998 Erstellung des Dokuments.
19.08.1998 Korrekturen rund um SYSRESET und Subsystem Device-ID.

Legende

I	**=**	**Steckbrücke gesteckt**
:	**=**	**Steckbrücke offen**

Absolute Adressen werden hexadezimal entweder mit einem führenden „$“- Zeichen oder mit einer führendem „0x“ wie in C-Notation üblich dargestellt.

Beispiele: $00AA entspricht 0x00AA entspricht dezimal 170.

„don’t care“ Bedingungen werden mit einem „X“ verdeutlicht. Dann ist der Inhalt unbedeutend.

Inhaltsverzeichnis

1. VORWORT

Mit dem von der Firma ARW-Elektronik entwickelten PCI-VME Interface kann sehr effizient und schnell vom einem PC (welcher mit PCI Steckplätzen ausgestattet ist) direkt auf den VMEbus zugegriffen werden. Es sind A16, A24 und A32 Adresszugriffe sowie D8, D16 und D32 breite Datentransfers vom PCI-BUS zum VME-BUS möglich.

Das Interface besteht aus einer PCI-Adapter-Karte (nachfolgend PCIADA genannt), und einem VME-Master-Modul (nachfolgend VMEMM genannt).

Für den Kompakt-PCI-Bus ist eine Karte (CPCIADA) mit den gleichen Eigenschaften von PCIADA verfügbar.

Die Datenübertragung erfolgt mittels differentiellen Low-Level-Leitungstreiber (LVDS) und einem 50pol. geschirmten Kabel von bis zu 2 m Länge. Das Kabel ist ein handelsübliches SCSI2-Kabel mit 50 pol. Feinpitch Stecker auf beiden Seiten. Optional sind Verlängerungskabel von jeweils 2 m lieferbar. Für eine zuverlässige Übertragung sollte die Gesamtlänge 10 m Distanz zwischen den beiden Adaptern nicht überschreiten. Durch die differentielle Übertragungstechnik und das durch mit „Hardware-Handshake“ verriegelte parallele Übertragungsverfahren sind hohe Übertragungsraten bei gleichzeitig stark verbessertem Störverhalten erreicht worden.

Die VMEMM-Karte ist sowohl als 6-HE Einsteckkarte mit einem A32/D32 Interface als auch als 3HE- Einsteckkarte mit einem A24/D16 Interface erhältlich. Diese Beschreibung gilt für beide Varianten, Abweichung der Varianten sind gekennzeichnet.

Obwohl die Beschreibung sich stark an der PC-Technik und den MS-Betriebssystemen anlehnt, ist das Interface für alle Rechner mit 33 MHz PCI-BUS verwendbar. Auch ist als zugehöriges Host-Betriebssystem jedes andere auf solchen Rechnern lauffähige Betriebssystem denkbar. (Beispiele: PowerPC, Sparc, Alpha und LINUX, UNIX, MacOS, Solaris etc.)

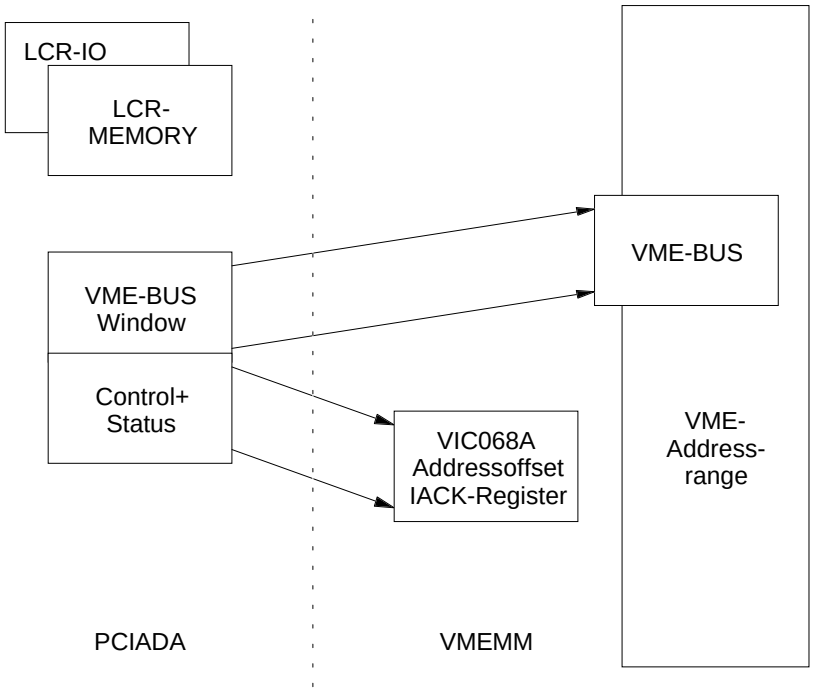


Bild 1: Adreßfenster von PCIADA und VMEMM

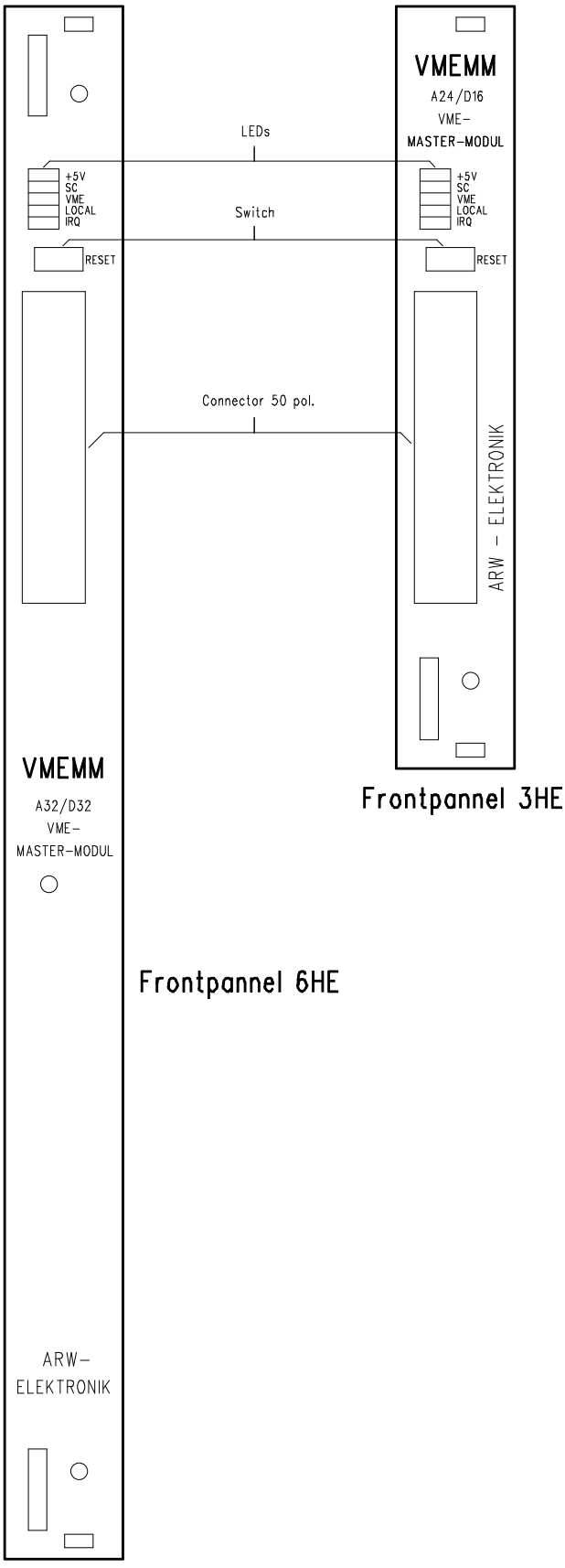
1.1. Eigenschaften PCIADA / CPCIADA

- PCI-Interface mit Standard PCI Baustein **PLX9050** von *PLX-Technology*
  (siehe beiliegende Kurzbeschreibung oder unter **www.plxtech.com.**)
- Es werden 8, 16 u. 32 Bit PCI-BUS Slave Zugriffe unterstützt.
- Direkte Datenabbildung in den Zieladreßbereich mit automatischer „low-big-endian“ Konversion.
- Das Interface belegt 8K Speicher-Adressbereich unterhalb von 1MB und ist damit auch unter MS-DOS adressierbar.
- Es wird je eine maskierbare Interruptquelle von VMEMM und eine lokale „Timeout“-Interruptquelle unterstützt.
- Die Verwendung differentieller Signaltreiber und Empfänger bürgt für störsichere, u. schnelle Datenübertragung.

1.2. Eigenschaften VMEMM

- VME-Interface-Modul als 6HE Einsteckkarte (A16, A24, A32 / D8, D16, D32) oder als 3HE Einsteckkarte (A16, A24 / D8, D16).
- Mit Standard-VMEbus Interface Controller **VIC068A** der Fa. *Cypress*
  (siehe beiliegende Kurzbeschreibung oder unter **www.cypress.com**)
- Konfigurierbarer VME-Master u. Systemcontroller (SYSCLK, SYSRESET, Arbiter, IACK-Driver, BGI-Driver).
- Programmierbare BUS-Timeout-Funktion.
- Konfigurierbarer BUS-Arbiter (SGL, PRIO und RRS, „Arbitration Timeout“).
- Konfigurierbarer BUS-Requester.
- Konfigurierbarer Interrupt-Handler auf allen 7 VME-Interrupt-Ebenen.
- Konfigurierbarer Interrupt-Generator auf allen 7 VME-Interrupt-Ebenen.
- Unterstützung der Interprozess-Kommunikationsregister mit Mailbox-Interrupts.
- Unterstützung ununterbrechbarer VME-Zugriffe. (RMC)
- Unterstützung und Detektion von VME-SYSFAIL und ACFAIL.
- Softwareauslösbarer VME-Reset.
- Reset-Taster an der Frontseite (erzeugt Interrupt / Software-Reset)
- LED-Anzeige für +5V, System-Controller-Funktion, VME-Zugriff, lokaler VMEMM-Zugriff und VME-Interrupt.
- Keine Beeinflussung des PCs beim Ein- oder Ausschalten der VME-Seite (Bedingung: Interrupt disabled). Beim Ein- oder Ausschalten des PCs wird auf dem VMEbus ein SYSRESET-Zyklus aktiviert.

! Blocktranfers werden nicht unterstützt.


VMEMM
A24/D16
VME-
MASTER-MODUL
LEDs
+5V
SC
VME
LOCAL
IRQ
Switch
RESET
Connector 50 pol.
ARW - ELEKTRONIK
Frontpannel 3HE
VMEMM
A32/D32
VME-
MASTER-MODUL
Frontpannel 6HE
ARW-
ELEKTRONIK

1.4. Installation PCIADA + VMEMM

1.4.1. Allgemeine Hinweise

Hinweis: Sorgen sie vor dem Auspacken für eine elektrostatisch entladene Arbeitsumgebung. Fassen sie nie direkt auf die elektronischen Bauteile. Elektrostatische Entladung kann zu Schäden an den elektronischen Bauteilen führen.

Hinweis: Versichern sie sich, daß sich der Quellrechner und der VMEbus-Einschubrahmen auf gleichem Spannungspotential befinden. Unerwartete Ausgleichsströme können zu Schäden an den elektronischen Bauteilen führen.

Hinweis: Die Karten dürfen nie bei eingeschalteter Spannungsversorgung eingesteckt oder ausgezogen werden. Unerwartete Seiteneffekte bei Einstecken unter Spannung können elektronische Bauteile zerstören.

Hinweis: Bitte beachten sie, daß der Stecker der VMEMM Adapterkarte am VME P2/J2-Steckverbinder nach VME-Standard belegt ist. Verwenden sie nur Steckplätze, die diesem Standard folgen oder keine Belegung aufweisen. Unsachgemäße Anschlüsse können zur Zerstörung von elektronischen Bauteilen führen.

1.4.2. Schrittweise Installation

1. Vor der Installation sind die Steckbrücken der VMEMM Karte zu prüfen bzw. entsprechend der gewünschten Funktion zu ändern. (siehe **1.5. Steckbrücken VMEMM**). Auf der Steckkarte PCIADA gibt es keine Benutzer-konfigurierbare Steckbrücken.
2. Stecken sie die PCIADA-Karte in einen verfügbaren PCI-Steckplatz des PCs und schrauben sie die Karte fest.
3. Stecken sie die VMEMM-Karte in den VMEbus-Einschubrahmen und schrauben sie mittels Sicherungsschraube(n) an der Frontplatte fest.
4. Schließen sie das 50 pol. Verbindungskabel zwischen PCIADA und VMEMM an.
5. Schalten sie sowohl den PC als auch den VME-Einschubrahmen ein. Beim erstmaligen Starten von Windows95 erscheint die Meldung, daß eine neue Hardware erkannt wurde. Aktivieren sie „keinen Treiber installieren (keine erneute Aufforderung zur Installation)“ und bestätigen dies mit „OK“.
6. Installieren sie entsprechend dem Handbuch PCI-VME WIN95 Treiber die Software.
7. Für die Pascal - Software müssen sie in CONGIG.SYS den EMM386.EXE (entsprechend der Beschreibung unter A:\DOS\PASCAL\Readme.txt) einbinden.
8. Testen sie die Installation mit der mitgelieferten Prüf-Software.

1.5. Steckbrücken VMEMM

1.5.1 Systemcontroller

Mit Steckbrücke SC wird festgelegt, ob VMEMM als Systemcontroller arbeiten soll. Wenn dies der Fall ist, muß VMEMM in den Steckplatz **1** (Steckplatz links) eingesteckt werden!

Die Einstellung kann über **VMEMM-Status** gelesen werden.

J402	Funktion
I	VMEMM ist Systemcontroller (werkseitige Einstellung)
:	VMEMM ist nicht Systemcontroller

1.5.2 Modulnummer

Mit dieser einstellbaren Nummer kann die Software eines PCs mehrere gleichzeitige PCI-VME-Interface Anschaltungen unterscheiden.

Hinweis: Bei gleicher Einstellung mehrerer VMEMM Karten kann nicht zwischen den Anschaltungen unterschieden werden. Dies kann nicht vorhersagbare Effekte zur Folge haben.

Die Einstellung kann über das Register **VMEMM-Status** gelesen werden.

J304	J303	J302	J301	Funktion
X	X	X	X	Kennung bei Multi PCI-VME installation
I	I	I	:	(werkseitige Einstellung, Nummer 1)

1.5.3 „WORD“-Datenpfad

Der Steckbrücke WORD zeigt dem VIC068A IC, ob nur Wortzugriffe auf dem VMEbus erlaubt sind (z.B. wenn nur ein P1/J1-BUS existiert).

Hinweis: Diese Steckbrücke muß bei einer 3HE VME-Einsteckkarte immer gesetzt sein.

Die Einstellung kann über das **VMEMM-Status** gelesen werden.

J401	Funktion
I	VIC-Chip im Word-Mode (werkseitige Einstellung bei VMEMM3)
:	VIC-Chip im Longword-Mode (werkseitige Einstellung bei VMEMM6)

1.5.4 Adresse der Interprozess-Kommunikationsregister

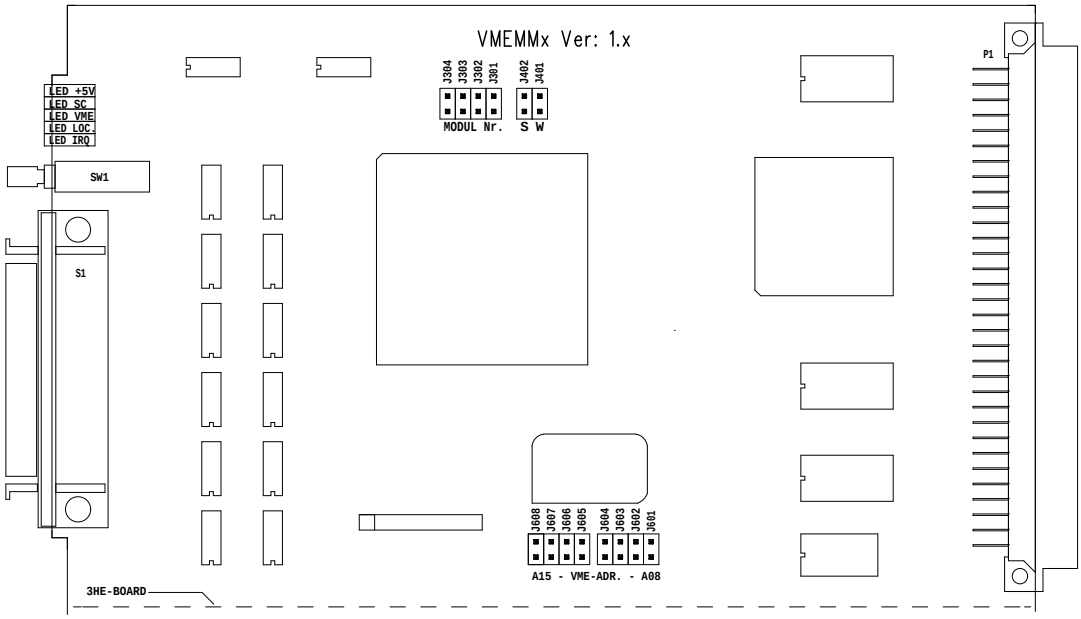
Mit den Steckbrücke J601...J607 wird die (Slave-) Basisadresse der Interprozess-Kommunikationsregister einer VMEMM-Einsteckkarte eingestellt. Diese Adresse befindet sich im „User-Short-IO“-Adressbereich des VMEbus und belegt 256 Bytes dieses Adressraums. Diese Register können zur Kommunikation zwischen mehreren Master-Anschaltungen auf dem VMEbus verwendet werden.

Hinweis: Bitte beachten sie, daß der Adressbereich eindeutig nur einer Baugruppe zugeteilt wird. Dieser Bereich ist nicht ausblendbar und muß daher zugeteilt werden. Unvorhersagbare Seiteneffekte können die Folge von falscher Einstellung sein. Wenn dieser Adressbereich von VMEMM selbst angesprochen wird, dann wird der VME-Zyklus nicht mit _DTACK. sondern mit einem _BERR beendet.

Hinweis: Eine gesteckte Steckbrücke entspricht einem Low-Pegel. (D.h. die werkseitig eingestellte Basisadresse ist $AA00.)

J608	J607	J606	J605	J604	J603	J602	J601	Funktion
A15	A14	A13	A12	A11	A10	A9	A8	VME-Short-Adresse
:	I	:	I	:	I	:	I	(werkseitige Einstellung)

1.6. Jumperplan



1.7. Zugriffszeiten

Typische Zugriffszeiten mit einem Pentium 133MHz einschließlich Progamm-Code (Turbo-Pascal) lesen:

Zugriff	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

Ein nicht optimierter, kompletter VME-A32/D32 Write dauert ca. 1,8 µs und setzt sich aus folgender Sequenz zusammen:

1. VME-Modifier in VIC schreiben (byte)
2. VME-ADR A12...A31 in Adr. Reg laden. (2 * word oder 1 * lword)
3. VME-Zugriff ausführen (byte, word oder lword)

2. PCIADAPTER KARTE

2.1. Übersicht/Funktion

Hinweis: Im weiteren wird vorausgesetzt, daß der verwendete PC ein PCI-BIOS mit seinen Autokonfigurationsfunktionen besitzt. Manche ältere Rechner weisen diese Eigenschaft nicht auf.

Um eine sichere Anpassung an den PCI-Bus zu gewährleisten, wurde der PCI-Bus Target-Interface-Chip **PCI9050** der Fa. *PLX-Technology* ausgewählt. Die PCIADApterkarte konfiguriert sich entsprechend dem Inhalt des **PCI CONFIGURATION REGISTER** (**PCR**) beim Booten des Rechners selbst. Die Inhalte des **PCR**egisters sind in einem EEPROM abgelegt und werden beim Einschalten des PCs automatisch in den **PCI9050** übernommen. Es werden 3 unterschiedliche Adressbereiche von PCIADA angefordert. Diese sind wie folgt:

1. 54 Byte **LOCAL CONFIGURATION REGISTER** (**LCR**) im I/O-Bereich. Im **LCR** sind die Basisadressen, sowie diverse Control-Status-Register verfügbar. (siehe Datenblatt **PCI9050**)

Hinweis: Das LCR im I/O-Bereich ist nicht auf allen Rechnerplattformen mit PCI-Bus verfügbar. Es sollte dann die inhaltsgleichen LCRegister im Memory-Bereich verwendet werden.

2. **LCR** im Memory-Bereich.
3. **8Kbyte Memory-Bereich** für Zugriff auf den lokalen VMEMM-Adressraum und den direkten Durchgriff auf den VMEbus.

2.2 Wichtige PCRegister

Zum Identifizieren der Hardware erfragt das PCI-BIOS die verschiedenen IDs der PCI-Anschaltung. Die ID sind wie folgt festgelegt:

ID	festgelegte Konstante
Vendor ID	$10B5
Device ID	$9050
Subsystem Vendor ID	$9050
Subsystem Device ID	$1167

2.3. Wichtige LCRegister

Andere als die hier erklärten Register müssen in dem Baustein nicht programmiert werden. Die Basisadressen der LCRegister und des VMEMM Adressbereichs werden gemäß der PCI-Spezifikation beim Booten des PCs dynamisch zugeteilt. Sie müssen aus dem PLX-Baustein gelesen oder über das PCI-Bios erfragt werden (siehe 4.x Software). Sonstige Funktionen sind dem Datenblatt des PCI9050 Bausteins zu entnehmen.

2.3.1 Interrupt Control/Status Register

Das INTERRUPT1 Statusbit wird bei allen vom VIC68A erzeugten Interrupts auf 1 gesetzt. Hierzu müssen die zugehörigen Interrupt-Anfragen zugelassen sein. Folgende Interrupt-Quellen werden hier zusammengefaßt:

1. Alle 7 VME-BUS Interrupt-Anfragen.
2. Mailbox-Interrupts in den VIC068A-IC durch einen 2. VME-Master.
3. Interrupts einer erfolgreichen Interrupt-Generierung.
4. SYSFAIL, ACFAIL Interrupts.
5. Interrupts aufgrund eines Bus-Timeouts.
6. Interrupts aufgrund eines Arbitration-Timeouts.
7. VIC68A Timer-Interrupts.
8. Interrupt aufgrund von betätigen des Reset-Tasters an der Frontseite.

Das INTERRUPT2 Statusbit wird durch lokale Ereignisse der PCIADA aktiv. Hierzu gehören:

1. Zugriffs-Timeout bei einem Zugriff auf den VME-BUS. (ca. 24 µs Timeout fest eingestellt.)
2. Nicht angeschlossenes Datenkabel.
3. Nicht eingeschalteter VME-Einschubrahmen.
4. Das PCI-VME Interface ist nicht freigeschaltet. (Das Bit USER I/O2 Output muß vor einem Zugriff auf VMEMM gesetzt sein!).

INTCSR; LCR-Baseadr + $4c (by word-access)

Byte		RD	WR	after Init
0	Local Interrupt 1 enable / 1 = enable / 0 = disable / Quelle = VMEMM	yes	yes	1
1	Local Interrupt 1 polarity / 1 = activ high / 0 = activ 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 / Quelle = PCIADA	yes	yes	1
4	Local Interrupt 2 polarity / 1 = activ high / 0 = activ 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 (Quelle = global)	yes	yes	0
7	Software Interrupt / 1 = generate Interrupt	yes	yes	0
16..8	not used	yes	no	00000000 b

2.3.2 User I/O Register

USER I/O2:

1. Um zu vermeiden, daß beim Booten des PCs bzw. beim Starten des PC-Betriebssystems in den VMEMM-Bereich gegriffen wird, wird der USER I/O2 Ausgang zum Sperren des VMEMM-Bereichs benutzt.
2. Ein gelöschtes Bit „USER I/O2“ setzt auch den „Interrupt 2“ zurück.

USER I/O3:

Ein gesetztes Bit „USER I/O3“ zeigt an, daß der VME-Einschubrahmen eingeschaltet und das Verbindungskabel gesteckt ist.

CNTRL; LCR-Baseadr + $50 (by word-access)

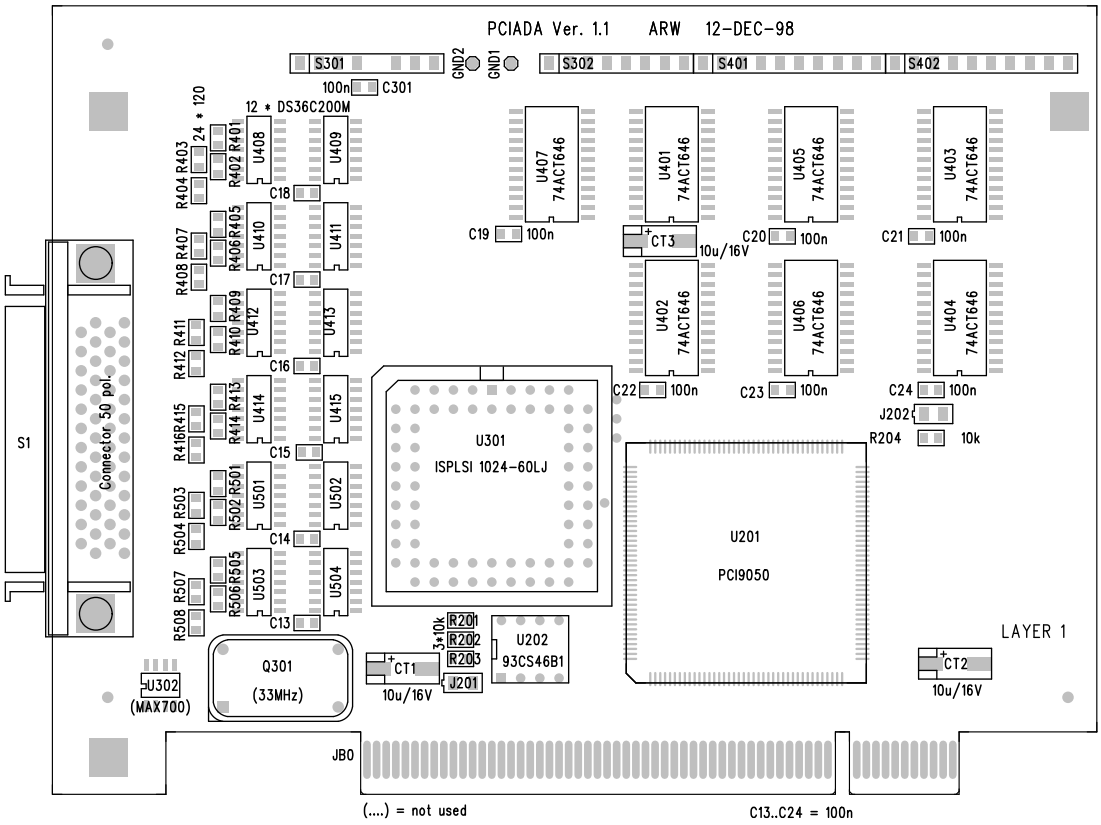
Byte		RD	WR	nach Init
5..0	do not modify !	yes	yes	000100 b
6	USER I/O2 Type, must always be 0	yes	yes	0
7	USER I/O2 Direction, must always be 1 = output	yes	yes	1
8	USER I/O2 output, 0 = disable access to VMEMM, 1 = enable access	yes	yes	0
9	USER I/O3 Type, must always be 0	yes	yes	0
10	USER I/O3 Direction, 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

Write $4184 for enable access to VMEMM
Write $4084 for disable access to VMEMM

2.4 Zugriffs-Timeout

In PCIADA ist ein globaler Zugriffs-Timeout zum Absichern der Zugriffe zu und über VMEMM realisiert. Beim Ablauf dieses Timeouts wird der laufende Zugriff ordnungsgemäß abgebrochen und der INTERRUPT2 ausgelöst. Dieser Timeout ist ergänzend zu dem VME-BUS Timeout eingeführt um Zugriffe auf abgeschaltete oder nicht angeschlossene VMEMM Interface abzufangen. Die Timeout Zeit ist fest auf ca. 24 usec eingestellt.

2.5. Bestückungsplan



2.6. Stromaufnahme

Spannung	Stromaufnahme	Leistung
+5V	ca. 0,5 A	ca. 2,5W

3. VMEMM KARTE

3.1. Übersicht/Funktion

Der Adressraum der VMEMM-Karte (sichtbar für den PCI-Master) ist in 4 Blöcke unterteilt:

1. **VMEMM** ⇒ Hier werden die VME-Adressen A11...A31 und das Steuersignal RMC gespeichert, sowie ein VME-Reset aktiviert. Der Status der Signale VMEMMStat, IRQ-Status u. A11...A31 können aus dem Bereich **VMEMM-Status** ausgelesen werden.

2. **VIC068A** VMEbus Interface Controller ⇒ Er übernimmt die komplette VMEbus Steuerung.

3. **Interrupt-Vector** (INTVEC) ⇒ Das Lesen eines Bytes in diesem Adressbereich führt auf dem VMEbus oder mittels VIC068A einen Interrupt-Acknowledge Zyklus aus.

4. **VME** 4k-Byte Memorybereich ⇒ Ein Zugriff auf diesen Speicherbereich führt immer einen VMEbus Transfer aus. Die VME-Adressleitungen A11..A1 entsprechen den PCI-Adreßleitungen A11..A1. Die VME-Adreßleitungen A31..A12 entsprechen dem Inhalt der VMEADR-Register.

3.2. Adressbereich VMEMM

Dieser Adressbereich wird als 8 kByte umfassender Raum in den PCI-Adressraum eingeblendet. Die Basisadresse im PCI-Adressraum wird dynamisch festgelegt.

Die Aufteilung ist wie folgt:

Basis-Adresse	Bereich /Zugriff	WR-Funktion	RD-Funktion
+ $1FFF bis + $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	VIC / byte	VIC068A Reg.	VIC068A Reg.
bis	VIC / byte	VIC068A Reg.	VIC068A Reg.
+ $0403	VIC / byte	VIC068A Reg.	VIC068A Reg.
VICBASE			
(+ $000a)	VMEMM / word	**VMEADRReg high** D15..D00 > VME-A31..A16	**VMEADRStat high** D15..D00 = VME-A31..A16
(+ $0008)	VMEMM / word	**VMEADRReg low** D15..D12 > VME-A15..A12 D11..D00 > x	**VMEADRStat low** D15..D12 = VME-A15..A12 D11..D00 = 0
+ $0008	VMEMM / lword	**VMEADRReg** 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 = Modul-Type
PCI-MEMBASE			

1* VIC „Global Reset“ (siehe VIC068A Manual)

2* VIC „Internal Reset“ (siehe VIC068A Manual)

3* Read Modify Cycle (siehe VIC068A Manual)

4* Blocktransfer = fix disabled

3.3 VMEMM-Control und -Statusregister

In die VMEMM-Control-Register werden besondere Steuerungsinformationen zur Konfiguration des Interface geschrieben. Aus den VMEMM-Status-Registern können Informationen zum Zustand der Konfiguration und des Interface erfragt werden.

3.3.1 VMEADDR-Register

Das VMEADDR-Register enthält den höherwertigen Anteil der Adressen der folgenden VME-BUS Zugriffe (A31..A12). Der Inhalt der Datenbits D11..D0 wird beim Schreiben ignoriert und beim Lesen als '0' zurückgelesen.

Hinweis: Für Zugriffe in den Standard- oder Short-I/O Adressbereich des VME-BUS sind die Adressen A31..A24 bzw. A31..A16 als „don't care" zu betrachten.

Hinweis: Das Register lässt sich im Wort- und Langwort-Zugriff beschreiben und lesen.

VMEADDR-Register (word or lword read- write access)

Bit	Extendet	RD	WR	after Init
31..12	most significant address bit A31 .. A12	yes	yes	X
11..0	write: don't care, read returns 0	yes	yes	0

Bit	Standard	RD	WR	after Init
31..24	don't care	yes	yes	X
23..12	most significant address bit A23 .. A12	yes	yes	X
11..0	write: don't care, read returns 0	yes	yes	0

Bit	Short	RD	WR	after Init
31..16	don't care	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

3.3.2 VMEReset-Register

Durch Beschreiben des VMEReset-Registers kann eine globale Rücksetzfunktion ausgeführt werden:

1. Das Beschreiben des VMEReset-Registers mit dem Dateninhalt $000a löst einen VMEbus-Reset und einen globalen Reset des VMEMM Interface aus. Dies entspricht einem 'Power-up'- Reset.
2. Nach dieser Aktion muß das VMEMM-Interface neu initialisiert werden.

Der Reset Taster an der Frontplatte löst nur einen Interrupt aus. Die Ausführung der eigentlichen Rücksetzfunktion bleibt dann der steuernden Software überlassen. Damit kann benutzerdefiniert auch ein manueller Reset unterdrückt werden.

3.3.3 VMEMMIRQStatus

Durch Lesen dieses Registers kann abgefragt werden, ob eine Interrupt-Anfrage des VIC068A Bausteins ansteht. Steht eine Anfrage an, dann ist das Bit 'D0' gesetzt. In den Datenbits D3..D1 steht kodiert die Anfrage mit der zum Lesezeitpunkt höchsten Priorität.

VMEMMIRQStatus

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

Hinweis: Durch die Anordnung der Datenbits kann das gelesene Datenwort direkt als Indizierungs-Offset im INTVEC Bereich des VMEMM verwendet werden. Dadurch ist ein schnelles Lesen des zugehörigen Interrupt-Vektors möglich.

3.3.4 VMEVICRegister

In diesem Schreibe-Register ist nur das niederwertigste Datenbit (D0) relevant. Alle anderen Datenbits sollten zu 0 beschrieben werden.

Wenn das Datenbit 'RMC' gesetzt ist, dann wird für die folgenden VME-BUS Zugriffe das Signal _AS (Address-Strobe) auch über den laufenden Zyklus hinaus aktiv gehalten. Damit lassen sich ununterbrechbare Zugriffe realisieren.

Hinweis: Der VME-BUS erlaubt nur „Ein-Adress" ununterbrechbare Zyklen ähnlich dem TAS-Befehl der Motorola 68XXX Prozessoren. D.h. innerhalb eines ununterbrechbaren Lese/Schreib Zyklus darf die Zugriffsadresse nicht gewechselt werden.

VMEVICRegister (word write access only)

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

3.3.5 VMEMMStatus

Aus diesem Register kann die Einstellung des VMEMM-Interface erfragt werden.

VMEMMStat-Register (word read access only)

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 Systemcontroller 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

Hinweis: Mit Hilfe der Modul-Identifikation und der Modul-Nummer kann das angeschlossene Interface identifiziert werden. Anhand des Status „WORD“ kann ein Anwenderprogramm feststellen, ob ein D16 oder ein D32 BUS genutzt werden soll.

Hinweis: Bei einem 3HE VME-Interface muß immer ein D16-BUS gewählt sein.

3.4. VIC068A-Register

Sämtliche lokal erreichbare VIC68A-Register sind mit jeweils 4 byte Offset zueinander mit Byte-Zugriff erreichbar. (Beispiel: VMEMM_Basis + $403 + $0, VMEMM_Basis + $403 + $4 etc.)

Hier werden nur die besonderen Funktionen einiger Register erklärt. Weitere Funktionen sind dem Datenblatt des VIC068A zu entnehmen.

3.4.1 AMSR-Register

In das **A**ddress **M**odifier **S**ource **R**egister muß der für die nächsten VME-Daten-Zugriffe notwendige Address-Modifier-Code abgelegt werden. „Interrupt-Acknowledge-Cycles“ sind durch den Inhalt nicht berührt.

AMSR; VIC-Baseaddr + $b7 (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

3.4.2 SYSFAIL Rücksetzen

Nach einem Reset generiert der VIC-Chip immer SYSFAIL auf dem VMEbus. Das **I**nterprozessor **C**ommunication **R**egister Nr. **7** muß zum Deaktivieren des SYSFAIL Signals verwendet werden.

Hinweis: SYSFAIL kann von anderen Einheiten des VMEbus genutzt werden um die Bereitschaft von dem VMEMM zu erkennen.

ICR7 ; VIC-Baseaddr + $7f (byte-access only)

Bit		RD	WR	after Init
7	SYSFAIL-MASK, 1 deaktiviert SYSFAIL	yes	yes	0
6..0	sonstige Funktionen (siehe VIC68A-Beschreibung)	yes	yes	00x0000

3.4.3 VME-BUS Timeout festlegen

Mittels des TTR Registers des VIC68A sollte der VME-BUS Timeout initialisiert werden. Jedoch sind nur Timeout Zeiten von 4 oder 16 usec sinnvoll, da längere Timeout Zeiten von dem globalen Timeout des PCIADA überdeckt werden.

TTR; VIC-Baseaddr + $A3 (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 Aquisition Time included	yes	yes	not incl.

Timeout Zeit in usec	Bit-Code (Bits 7..5 oder 4..2)
4	0
16	1
32	2
ausgeschaltet (infinit)	7

3.4.4 LICR und LIVBR Register programmieren

Die Ereignisse VME-BUS Timeout und „Taste an der Frontplatte gedrückt“ werden über lokale Interrupts verarbeitet. Hierzu müssen die jeweiligen LICRx Register (Local Interrupt Conrtrol Register) und das LIVBR Register (Local Interrupt Vector Base Register) programmiert werden. Das LIVBR enthält in seinen höherwertigen 5 Bits den gewünschten Vektor, in den niederwertigen 3 Bits wird die Interrupt Ebene des auslösenden Interrupts kodiert. Siehe dazu auch unter Interrupts.

LICR; VIC-Baseaddr + $3B/$3F (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	

Hinweis: Der VME-BUS Timeout speist den lokalen Interrupt Nummer 7, der Taster an der Frontplatte speist den lokalen Interrupt Nummer 6.

3.5 Interrupts

3.5.1 VME-BUS Timeout

Wenn ein VME-BUS Zugriff mit einem Busfehler (oder BERR* Signal) aufgelöst wird dann beendet VMEMM den Zugriff so, als wäre der Zugriff erfolgreich durchgeführt. Gleichzeitig löst jedoch der lokale Interrupt Nummer 7 des VIC68A einen Interrupt zu PCIADA aus. Durch Abfangen des Interrupts in der Software kann ein fehlerhafter Zugriff dann erkannt werden.

Hinweis: Die Timeout Zeit muß im TTR Register des VIC68A eingestellt werden, zum Programmieren muß beim Auftreten des Interrupts von einer fallenden Flanke am Eingang LIRQ7 des VIC68A ausgegangen werden. Weiter betroffene Register des VIC68A: LICR7 und LVBR.

3.5.2 Taster an der Frontplatte

Der Taster an der Frontplatte löst den lokalen Interrupt Nummer 6 des VIC68A aus. In der Behandlung des Interrupts kann das Anwendungsprogramm seine spezifischen Rücksetzfunktionen durchführen.

Hinweis: Es sind die Register LICR6 und LVBR des VIC68A betroffen. Auch hier muß beim Auslösen von einer fallenden Flanke ausgegangen werden.

3.5.3 Sonstige Interrupts

Von VMEMM erzeugte oder weitergeleitete Interrupts müssen vektorisiert werden. Per Konvention werden die Interrupt Vektoren so aufgeteilt, daß die Vektoren 0x01..0x3f der internen (d.h. von einem Treiber vorbelegten) Nutzung vorbehalten sind. Die Vektoren 0x40..0xff können von Peripherie auf dem VMEbus genutzt werden.

Hinweis: Alle lokalen Interrupts von VMEMM erwarten, daß VMEMM auch den Vektor generiert.

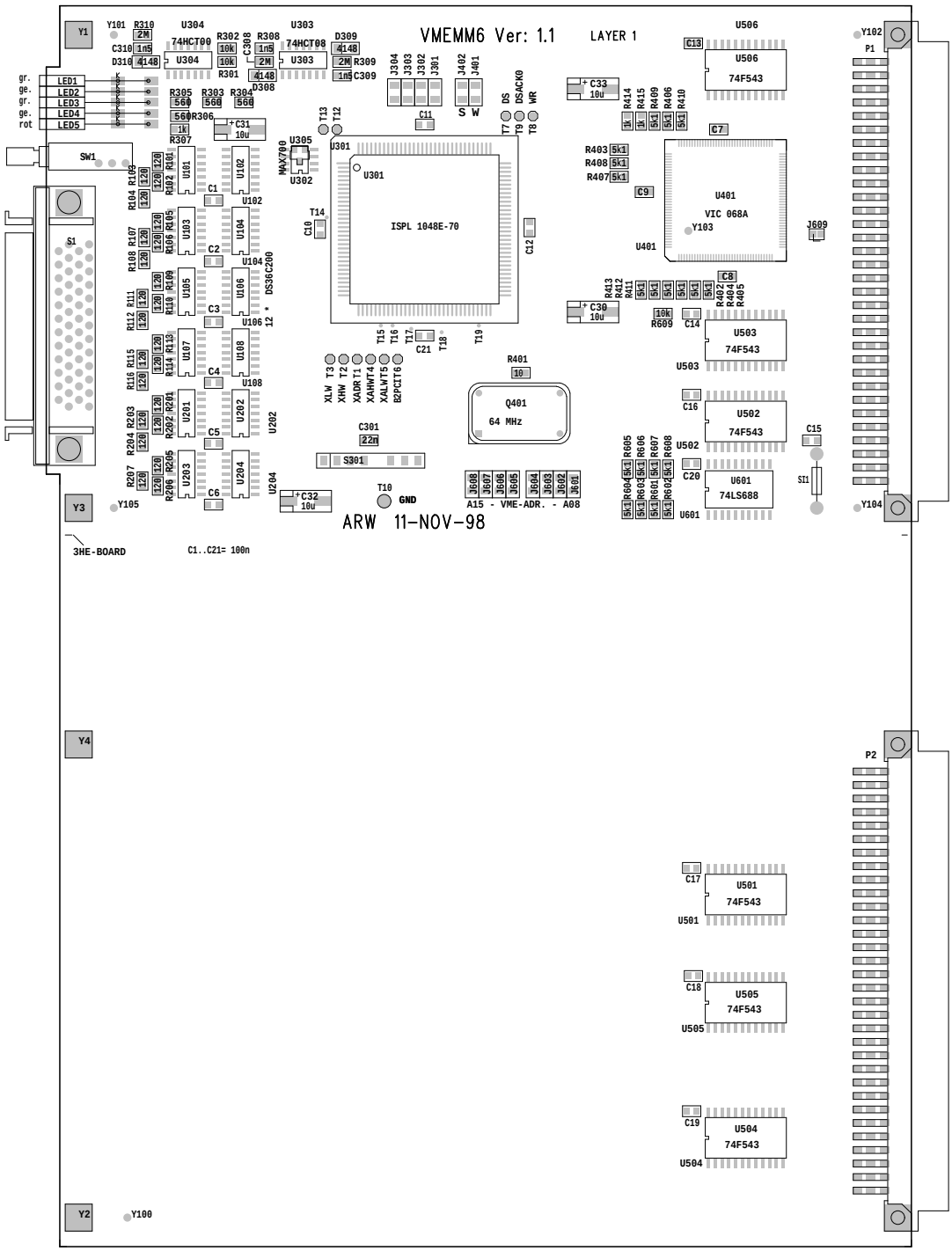
Hinweis: Der von PCIADA generierte Timeout Interrupt muß von der Software auf den Interrupt Vektor Nummer 1 gespiegelt.

Interrupt Ursache	Vektor Nr.
PCIADA erzeugter Interrupt (Timeout)	1
Clock Tick Interrupt Generator	2
Reset Taster an der Frontplatte	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

3.6. Stromaufnahme

Spannung	Stromaufnahme	Leistung
+5V	0,9 A	4,5 W

3.7. Bestückungsplan



4. SOFTWARE

4.1. Initialisierung des Interface

Es stehen Beispielprogramme in Turbo-Pascal (DOS) und C++ (Windows 95) zur Verfügung, mit denen die I/O-Basisadresse des CSR-Bereiches und die Basisadresse zur VMEMM-Karte ermittelt werden können.

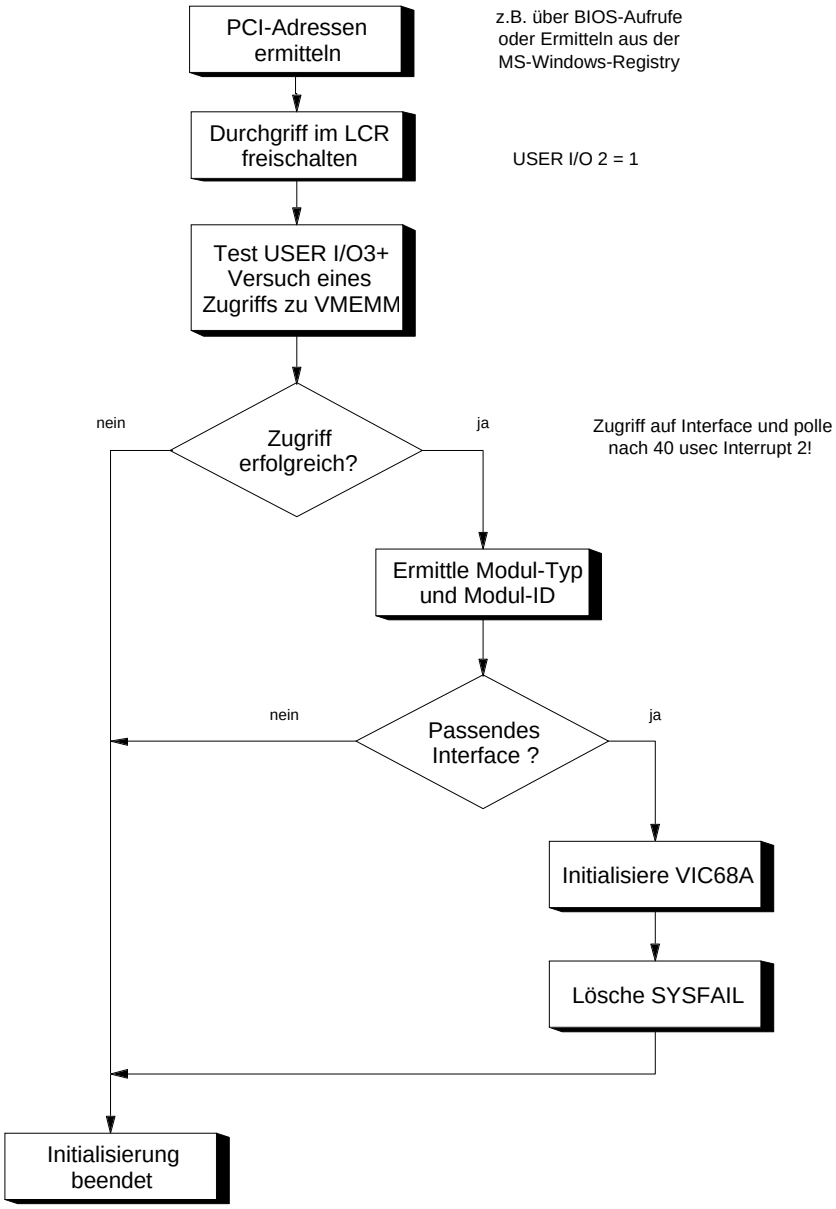


Bild 2: Initialisierung des Interface

4.2. VME Zugriffe

Es stehen Beispielprogramme in Turbo-Pascal (DOS) und C++ (Windows 95) zur Verfügung

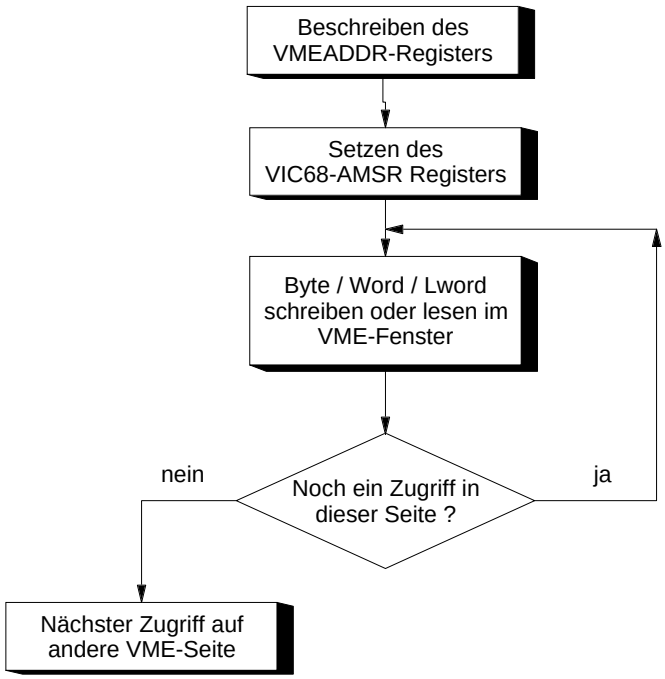


Bild 3: VME-Zugriffe

Zugriffe, die durch einen „Timeout“ oder einen VME-BUS-Error abgebrochen wurden, beenden den laufenden Zyklus normal. Zum Erkennen eines solchen Ereignisses kann ein Timeout-Interrupt ausgelöst werden. Die Annahme des Ereignisses kann auch pollend über das „Interrupt 1 Status Bit“ geschehen.

4.3. Interrupts annehmen

„Interrupts“ des Interfaces können sowohl „pollend“ als auch als wirkliche Interrupts angenommen werden. Es sind 2 Interrupt-Quellen zu unterscheiden. Beiden Quellen sind auf einen PCI-Interrupt gelegt.

1. Der „Local Interrupt 1“ faßt alle Interrupts von VMEMM kommend zusammen. Die Zusammenfassung wird durch den VIC68A-Baustein priorisiert. Die Verteilung der Interrupt-Prioritäten kann im VIC68A frei programmiert werden. Es gibt jedoch zwei Ausnahmen: Die erste Ausnahme bilden die VME-Interrupts, die gleich ihrer VME-Priorität auch lokal priorisiert werden müssen. Die zweite Ausnahme bildet der Interrupt, der bei einem VME-BUS-Error (_BERR) ausgelöst wird. Dieser ist fest auf die höchste VIC68A-lokale Interruptebene 7 verdrahtet.

Als Folge der Auslösung eines „Local Interrupt 1“ muß immer ein byte-Interrupt-Vektor gelesen werden.

2. Der „Local Interrupt 2“ wird durch einen lokalen „Timeout-Interrupt“ auf der PCIADA-Karte erzeugt. Dieser löst ca. 24 µsec generell nach Beginn eines Zugriffs aus. Gleichzeitig wird damit auch der laufende Zugriff, wenn auch mit ungültigem Ergebnis beendet.

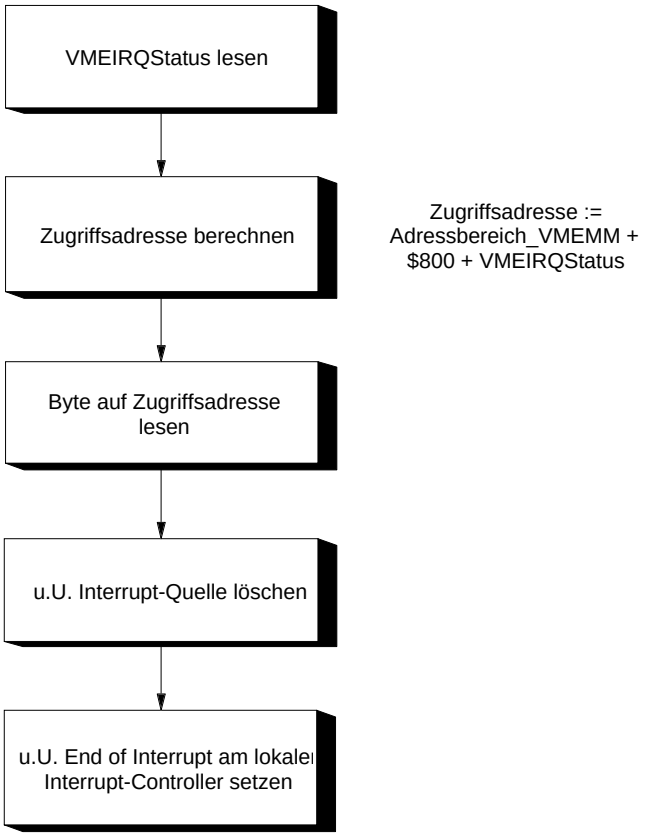


Bild 4: Reaktion auf einen „Interrupt 1“

4.4. Deinitialisierung des Interface

Hinweis: Bevor sie Ihr Anwenderprogramm verlassen, sollten sie immer die „Interrupt Enables“ für beide PCI-Interrupts sperren. Sonst kann es zu Fehlverhalten in Ihrem Rechner kommen.

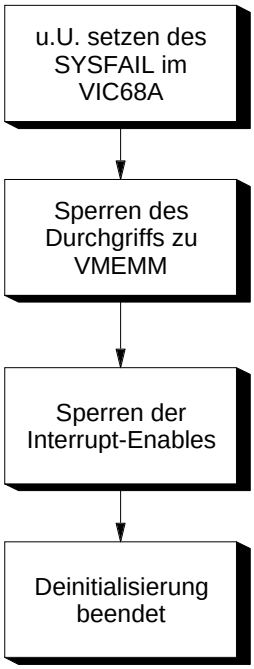


Bild 5: Deinitialisieren des Interface

5. EINSCHRÄNKUNGEN

Zur Zeit bekannte Einschränkungen:

6. ANHANG