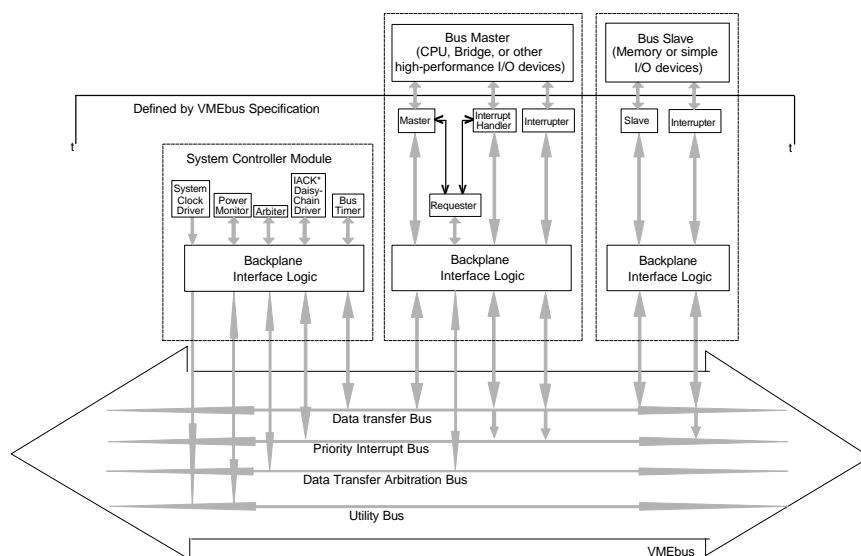


VMEbus Basics

The VMEbus is a 32-bit bus that is widely used worldwide in industrial, commercial and military applications. An abundance of VMEbus cards are available to perform a wide range of tasks, from digital image processing to disk controllers. The VMEbus supports multiple bus masters and high data transfer rates. Most VMEbus cards are configured via a combination of hardware jumpers, card-specific software configuration, and setting parameters in non-volatile memory.

A VMEbus chassis consists of a card cage with 1 - 21 slots, a backplane with two connectors and, normally, five jumpers per slot. The slots are numbered from 1 - 21. Slot 1 is the System Controller slot. Cards with different functions are inserted in the slots to form a customized VMEbus chassis.

Diagram of VMEbus Components



Many devices implement more than one component shown in the diagram above. Most single board computers include the VMEbus System Controller Module, bus master and bus slave on a single card.

The VMEbus System Controller Module provides arbitration and monitors the system's state.

A bus master is any device that can initiate transactions (read, write, Interrupt Acknowledge (IACK), BLT, and MBLT) on the bus. All CPUs are bus masters, as are most disk or network devices. All DMA capable devices are bus masters. In most cases, cards capable of being bus masters can also respond as bus slaves.

A bus slave is any device that can respond to transactions within its Memory Window. All memory devices and most graphics devices are bus slaves.

Some VMEbus signal lines fit into more than one category; therefore, are listed as part of more than one bus. For example, the BERR* line is considered part of both the data transfer bus and the utility bus.

Data Transfer Bus

➔ An asterisk (*) indicates an active low signal.

The data transfer bus consists of data lines (D31 - D00), addressing lines (A31 - A01, AM0 - AM5, LWORD*, DS0*, and DS1*), and control lines (AS*, WRITE*, DTACK*, BERR*, RETRY*, DS0*, and DS1*).

Some data transfer bus lines are used for more than one purpose. Please refer to the VMEbus specification if greater detail is required.

Addressing lines are driven by a bus master and monitored by a bus slave. The master drives AS* (Address Strobe) to indicate a valid address on the bus.

Data lines are used to transfer information across the bus. Transfer size (8-bit, 16-bit, 32-bit) is determined by the state of the LWORD*, DS0*, and DS1* lines. Direction of the transfer is determined by the WRITE* line's state.

All data transfers are terminated by asserting one of the following signals: DTACK* (a data transfer acknowledgment indicating a successful transfer), BERR* (a Bus Error indicating a bus timeout or an error), or VME64 RETRY* (retry in which the master should automatically release the bus and try the transfer again later).

VME64 multiplexes the addressing and data lines. When using A64 (long addressing) modes, D31 - D00 are also used as addressing lines. When using D64 transfers, the A31 - A01 and LWORD* lines are also used as data lines.

Priority Interrupt Bus

The priority interrupt bus consists of interrupt request lines (IRQ7* - IRQ1*), the IACK* line, and IACK daisy-chain lines (IACKIN* and IACKOUT*). This bus is used to assert an interrupt. It is also used in conjunction with the data transfer bus to acknowledge an interrupt.

Data Transfer Arbitration Bus

The data transfer arbitration bus is used by a bus master to request permission from the System Controller to use the bus. The arbiter in the System Controller determines which master will be granted the bus.

Utility Bus

The utility bus contains all system monitoring lines, including SYSRESET*, BERR*, SYSFAIL* and ACFAIL*. SYSRESET* is used to reset the VMEbus system. BERR* is used to terminate a bus cycle. SYSFAIL* is a utility signal that can be used for system diagnostics. ACFAIL* is asserted when the AC line voltage stops.

VMEbus Specifications

The VMEbus Specification was introduced in 1981 and has been updated several times. Each revision clarified and added features to the bus, and is 100% upwardly compatible from the older version.

The first revision, Revision A, was based on the earlier VERSAbus specification developed by Motorola Microsystems. It changed the form factor of the cards to 3U, 6U, and 9U sizes.

Revision A, Revision B, and Revision C laid the foundation for versions of the specification that are used in today's advanced systems: Revision C.1, IEEE-1014-87, and VME64.

Revision C.1

Revision C.1, approved in 1985, clarified the bus specification and removed some incompatibilities found in earlier revisions. No new features were added or removed from the earlier Revision C.

IEEE-1014-87

Approved in 1987, IEEE-1014-87 primarily clarified the various VMEbus options to insure compatibility. Changes included:

- Redundant PERMISSIONS were removed.
- Addition of the FAIR requester.
- Addition of the J1/J2 monolithic backplane.
- Addition of Appendix D that describes metastability and sample circuits.
- Addition of Appendix E that describes permissible options.

VME64 Specification (ANSI/VITA 1-1994)

The current revision, VME64, provides several major changes including:

- Addition of the 64-bit Multiplexed Block Transfer (MBLT) option.
- Addition of the 64-bit address option.
- Designated a RESERVED pin to a bus RETRY* function (RETRY* support is optional).
- Addition of a control status register function (CR/CSR) allowing automatic configuration.
- Implemented a detector that automatically detects when a card is in slot 1, and therefore, is the System Controller.
- Permitted use of shielded DIN connectors.
- Addition of rescinding DTACK* option.

VMEbus Specific Terminology

Mnemonic symbols are commonly used to describe the many interface options offered by VMEbus. When integrating a VMEbus system, use these mnemonics to make sure all cards in the system will work together reliably.

Options

When selecting modules, make sure options from one module are compatible with the other selected options. For example, D08(EO), D16, D32 and RMW all specify types of data transfer cycles that can be generated by a master. In this case, the bus interface options would be described as:

D08(EO):D16:D32:RMW.

Slave modules in a system with this master may or may not be able to handle all of its cycle types. For example, consider a slave with the following options:

D08(EO):D16:RMW.

If the master generates D08(EO), D16 or RMW bus cycles, this slave would accept them. If the master generates a D32 cycle, the slave does not have the capability to handle it. This does not mean the slave should not be used in the system, rather that you must ensure the master does not generate a D32 cycle to this slave (usually by software control).

Address Information Mnemonics

ADO

Devices with ADO (ADdress Only) capability allow masters to generate and slaves to accept address-only cycles. Using ADO cycles can enhance system performance by allowing simultaneous slave and master memory decoding.

A16

VMEbus masters with A16 capability can generate bus cycles using 16-bit addresses (Short addresses). A16 capable slaves can accept these cycles. A16 addresses are often used as an I/O space.

A24

VMEbus masters with A24 capability can generate bus cycles using 24-bit addresses (Standard addresses). A24 capable slaves can accept these cycles.

A32

VMEbus masters with A32 capability can generate bus cycles using 32-bit addresses (Extended addresses). A32 capable slaves can accept these cycles.

A64

VMEbus masters with A64 capability can generate bus cycles using 64-bit addresses (Long addresses). The 32 data lines are used to supplement the normal address lines in this mode. A64 capable slaves can accept these cycles.

Data Transfer Mnemonics

BLT

Block Transfer (BLT) allows a block transfer cycle to be generated by a master or accepted by a slave. This cycle can be faster than normal read/write cycles.

MBLT

Multiplexed Block Transfer (MBLT), allows a block transfer cycle with 64 bits of data to be generated by a master or accepted by a slave. This cycle is a faster version of the BLT cycle.

D08(O)

D08(O) capable slaves can accept 8-bit data transfers at odd addresses. All masters must generate D08(O) cycles because they are a subset of the D08(EO) capability. Slaves with D08(O) are most often used on I/O devices with 8-bit integrated circuits.

D08(EO)

Masters or slaves with D08(EO) capability can generate or accept 8-bit bus cycles at even or odd addresses (but not simultaneously).

D16

Masters and slaves with D16 capability can generate or accept 16-bit transfers.

D32

Masters and slaves with D32 capability can generate or accept 32-bit transfers.

D64

Masters and slaves with D64 capability can generate or accept 64-bit transfers. Unaligned transfers are not permitted. Thirty-one address lines, 32 data lines and LWORD* are used to pass the data.

RMW

Read-Modify-Write (RMW) capable masters can generate RMW cycles. Slaves can accept them. RMW is primarily used in multiprocessing systems to allow arbitration of shared system resources. It guarantees that the value at the slave cannot be modified between the read and the write cycle.

UAT

An UnAligned Transfer (UAT) can be generated by a master and accepted by a slave. It allows 32 bits of data to be transferred at unaligned address boundaries in two bus cycles instead of three.

VMEbus System Controller Module

BTO(x)

Bus Time Out (BTO(x)) indicates a bus timer functional device. This device monitors data strobes DS0* - DS1* and asserts BERR* if the strobes are low for more than "x" microseconds. This terminates the transaction. The bus timer is used to prevent system lock-ups from happening when bus masters are probing for devices or bus failures.

I(x)

The I(x) (Interrupter) option applies to interrupters that generate interrupt requests on IRQx*.

IH(x)

Devices that support the Interrupt Handler (IH(x)) option are interrupt handlers that can monitor interrupt requests on level IRQx* only. For any interrupt level it is monitoring, the interrupt handler generates an interrupt acknowledge (IACK) cycle in response to the requests.

IH(x-y)

Devices with the Interrupt Handler (IH(x-y)) option are interrupt handlers that can monitor interrupt requests on multiple levels. For example, IH(3-5) can handle interrupts on IRQ3*, IRQ4*, and IRQ5*. For any interrupt levels it is monitoring, the interrupt handler generates an interrupt acknowledge cycle in response to the requests.

IRQx*

Any of the VMEbus IRQ1*, IRQ2*, IRQ3*, IRQ4*, IRQ5*, IRQ6*, or IRQ7* lines. Used as an abbreviation when making a general statement about a VMEbus interrupt line. IRQ7* receives highest priority and IRQ1* receives lowest priority.

ROAK

The Release-On-AcKnowledge (ROAK) option describes an interrupter that removes its interrupt request shortly after an interrupt acknowledge cycle. Usually, hardware automatically removes the interrupt.

RORA

The Release-On-Register-Access (RORA) option describes an interrupter that removes its interrupt request when a master accesses a register on the interrupter. Usually, the software instructs hardware to remove the interrupt.

VMEbus System Controller Module Arbitration

PRI

PRIority (PRI) applies to bus arbiters that use a priority scheduling algorithm in which bus requesters on level BR3* have the highest priority. Bus requesters on level BR0* have the lowest priority.

ROR

Bus requesters that have the Release-On-Request (ROR) option relinquish the data transfer bus when it is requested by another VMEbus device.

RRS

The Round Robin Select (RRS) option applies to bus arbiters that use a round robin scheduling algorithm in which the bus is granted on a rotating basis. Bus masters and interrupt handlers are granted the bus according to equal priority.

RWD

The Release-When-Done (RWD) option describes a requester that relinquishes the data transfer bus each time it is done using it.

SGL

SiNGle-Level (SGL) applies to bus arbiters. SGL arbiters only grant the bus to bus requesters on level three (BR3*).

FAIR requester

A requester that provides equal access to all masters requesting the bus on its level. This is done by the requester not requesting the bus again until all other bus masters have released their request.

VMEbus Addressing And Address Modifiers

VMEbus memory is divided into A16, A24, A32 and A64 address spaces. The names indicate how many address bits are used when a data transfer is directed toward that memory space. For example, address bits A15 - A01 are used for a data transfer to A16 address space (VMEbus does not have an A00 bit).

The VMEbus address modifier determines which memory space is addressed. It tells the card being addressed how many address lines to look at and the type of memory cycle.

➔ When accessing the VMEbus, you must know which of the four address spaces you want to access: A16, A24, A32, or A64.

The VMEbus specification defines four types of address spaces: Long (A64), Extended (A32), Standard (A24), and Short (A16). Long (A64) addressing multiplexes the data lines to use 64 address bits. Extended (A32) addressing uses 32 address bits. Standard (A24) addressing uses 24 address bits. Short (A16) addressing uses 16 address bits. The VMEbus uses the address modifier lines to select which type of address space is being referenced.

Each address space is independent of the other address spaces and can be thought of as a logically separate address bus. A32 addressing uses address lines A31-01. A24 uses A23-01, and A31-24 are unused. A16 uses A15-A01, and A31-16 are unused. The VMEbus does not have an address 0 (A00) line. Byte addressing is controlled by the signals DS0*, DS1* and LWORD*. Address lines A03-A01 are used during Interrupt ACKnowledge cycles (IACK cycles).

The following table summarizes use of the address bus.

ACTIVE PORTION OF ADDRESS BUS - ADDRESS ROUTING				
A31 - A24	A23 - A16	A15 - A04	A03 - A01	Address Modifier Codes (hex)
A31 -----A01				Extended (32-bit) 08 - 0F
	A23-----A01			Standard (24-bit) 38 - 3F
		A15-----A01		Short I/O (16-bit) 29, 2D
			A03 - A01	Interrupt Acknowledge

■ = Unused portion of address bus.

A64 addressing uses the whole address bus for lines A31 - A01 and the data lines (D31 - D00) to provide the upper (A63 - A32) address bits.

The value of the address modifier lines, AM[0..5], determines which address space is used. The VMEbus master is responsible for supplying the proper address modifier at the same time it drives the address lines. Slaves are designed to respond to cycles with a particular address modifier; however, most slaves are capable of responding to several address modifiers.

The most commonly used address modifier codes are listed below.

ADDRESS MODIFIER (HEX)	NUMBER OF ADDRESS BITS	TRANSFER TYPE
2D	16	Short supervisory access
3D	24	Standard supervisory data access
3F	24	Standard supervisory BLT
3C	24	Standard supervisory D64-MBLT
0D	32	Extended supervisory data access
0F	32	Extended supervisory BLT
0C	32	Extended supervisory D64-MBLT

Complete list of VMEbus address modifiers:

ADDRESS MODIFIER (HEX)	# OF ADDRESS BITS	TRANSFER TYPE
3C	24	Standard supervisory D64-MBLT
3F	24	Standard supervisory BLT
3E	24	Standard supervisory program access
3D	24	Standard supervisory data access
3C	24	Standard supervisory D64-MBLT
3B	24	Standard non-privileged BLT
3A	24	Standard non-privileged program access
39	24	Standard non-privileged data access
38	24	Standard non-privileged D64-MBLT
2D	16	Short supervisory access
29	16	Short non-privileged access
10 - 1F	undefined	User defined
0F	32	Extended supervisory BLT
0E	32	Extended supervisory program access
0D	32	Extended supervisory data access
0C	32	Extended supervisory D64-MBLT
0B	32	Extended non-privileged BLT
0A	32	Extended non-privileged program access
09	32	Extended non-privileged data access
08	32	Extended non-privileged D64-MBLT
07	64	Long supervisory BLT
06	64	Long supervisory program access
05	64	Long supervisory data access
04	64	Long supervisory 64-bit BLT
03	64	Long non-privileged BLT
02	64	Long non-privileged program access
01	64	Long non-privileged data access
00	64	Long non-privileged 64-bit BLT
don't care state	3	IACK cycle (uses A01-A03)

VMEbus Interrupts And The IACK Cycle

Hardware devices use interrupts to indicate that they need attention or that some event has occurred. The VMEbus supports seven interrupt levels labeled IRQ1* to IRQ7*. Any number of devices can use the same interrupt; however, only one VMEbus device can respond to an interrupt level.

Interrupts are responded to with an IACK cycle. The basic interrupt process is as follows:

1. A VMEbus device asserts one of the seven interrupt lines. For this example, we assume IRQ1*.
2. The VMEbus interrupt handler for IRQ1* receives control of the VMEbus and starts an 8-, 16-, or 32-bit read with the IACK* signal asserted. This read should have address bits A03 - A01 equal to the level of the interrupt to be acknowledged. For example, IRQ1* is acknowledged by reading from location 2 (A03=0, A02=0, A01=1) with IACK* asserted.
3. The System Controller starts the IACK daisy-chain by sending IACKIN* to slot 2. This daisy-chained IACK signal is passed up the slots until it reaches the card asserting IRQ1*.
4. The device asserting IRQ1* responds to the IACK read cycle with the appropriate size data (the status/id value or IACK vector). It does not pass the IACK signal to the next slot.
5. The VMEbus interrupt handler for IRQ1* uses the IACK vector to determine which device is interrupting and makes any necessary accesses to the interrupting device's registers to release the interrupt.

Two types of interrupting devices are supported by the VMEbus: Release-On-AcKnowledgeMent (ROAK) and Release-On-Register-Access (RORA). A ROAK device removes its interrupt near the end of the IACK cycle (see step 4 above). A RORA device removes its interrupt after the IACK cycle occurred and one of the registers on that device is accessed (see step 5).

For RORA devices, software must make a register access to switch off the interrupt before exiting the Interrupt Service Routine (ISR). Otherwise, the interrupting device will not remove its interrupt and the processor may go into an endless loop handling the interrupt.

Benefits Of DMA (BLT)

Direct Memory Access (DMA) is a data transfer method used by a device to transfer data directly to or from System Memory. DMA does not use a processor to transfer the data, instead it uses a separate DMA controller. Therefore, the processor can do other tasks while the device transfers data.

When a processor transfers data, it usually puts out a new address cycle for each data transfer (single cycle transfer). In a single cycle transfer each address cycle only allows a single data cycle. Single cycle provides efficient random access to resources, however, DMA is the preferred method for transferring large blocks of consecutive data.

The VMEbus provides two special data transfer modes to provide better data throughput: BLT and MBLT. Because BLT and MBLT allow a single address cycle and multiple data cycles for each data burst, they are more efficient than single cycle transfers. BLT mode allows up to 32-bits of data to be transferred each cycle in bursts of up to 256 bytes of data. MBLT mode, part of the VME64 Specification, allows 64-bits of data to be transferred each cycle in bursts of up to 2048 bytes of data. MBLT mode can transfer 64 bits by multiplexing the 31 address lines and LWORD* line on the VMEbus for use as the upper 32-bit data lines.

Even when the VMEbus does not multiplex the address and data lines, doing multiple data transfers for a single address cycle provides substantial savings. The master is not required to remove the address, re-arbitrate for the bus, and then drive the new address on the bus. Additional time is saved because the bus slave does not need to decode the address between each data transfer to know that it is being accessed. Using single cycle transfers, the VMEbus is limited to approximately 8M Bytes/sec. BLT allows data rates up to 40M Bytes/sec. MBLT allows data rates up to 80M Bytes/sec.

A BLT or MBLT is indicated by the address modifier used for the transfer. When a slave device sees a BLT or MBLT, it must latch the address during the single address cycle. For each data transfer, the VMEbus slave device automatically increments the low bits of the address to determine where the transferred data go.

Using BLT or MBLT mode can also significantly improve the performance of bus bridges. If the bridge can prefetch reads and pipeline writes, it can acknowledge a data transfer and start the next data cycle on one side of the bridge before the transfer has completed on the other side.

Because BLT and MBLT capability offers substantial performance benefits, be sure to check for BLT and MBLT support when selecting VMEbus cards.

RETRY*

RETRY* indicates that a requested data transfer cannot take place (usually because of deadlock) but should be attempted again by the bus master in a future bus cycle. RETRY* signal support by bus masters allows a bus bridge to transparently escape deadlock. The bus slave asserts RETRY*.

BERR* or DTACK* is used in conjunction with RETRY* so that devices that do not implement this VME64 option will terminate the cycle.

After receiving RETRY*, the bus master relinquishes bus control. The master should then wait a short time before attempting to access the bus again. The amount of time that the master waits before re-requesting the bus depends on the bus arbitration method and system architecture.

RETRY* support is optional in the VME64 specification; therefore, it may not be supported by all VME64-compliant devices.

VME64 Compatibility

ANSI/VITA 1-1994 (VME64 Specification) contains the most recent revision of the VMEbus Specification. Several features were added to this specification; all features are upwardly compatible, allowing a system to contain both VME32 and VME64 devices.

Two new features are most useful for bus bridges: MBLT and a RETRY* capability. Because both features are optional VME64 capabilities, not all VME64-compliant devices implement them.

MBLT use allows faster data transfer (see “Benefits Of DMA”). To use MBLT mode, the bus master must be able to generate the new address modifier code and the bus slave must be able to accept it. If the bus slave does not support MBLT (compliant only to Rev. C.1 or IEEE-1014-87) and data is transferred using MBLT, a BERR* will result. BERR* results even if the System Controller does not implement VME64 because the way the data strobes are used is still compatible with earlier VMEbus specifications.

Asserting RETRY* allows a bus slave or bus bridge to force a bus master to relinquish VMEbus mastership and retry the transfer. Consequently, a bus bridge can avoid potential deadlock if both sides of the bridge are simultaneously active. The retry must be on the first word of a BLT or MBLT. It is important to check that a bus master has the capabilities to accept RETRY* and relinquish the bus. Some VMEbus cards may be able to generate a RETRY* as a bus slave but cannot accept a RETRY* as a bus master. Any bus master that does not implement the VME64 RETRY* will see RETRY* as either a BERR* or DTACK*, causing it to relinquish the bus and requiring software to intervene and retry the transfer.

VMEbus System Controller Module

The VMEbus System Controller Module contains the system clock driver, power monitor, arbiter, IACK daisy-chain driver, bus timer, and backplane interface logic.

The system clock driver provides a stable 16 MHz utility clock (SYSCLK) to all devices on the bus. The VMEbus is asynchronous and the clock provides no other bus timing.

The power monitor generates system reset (asserts SYSRESET*) and monitors the system's AC power source (asserts ACFAIL*).

The arbiter monitors requests for the bus and grants control of the bus to one master at a time.

The IACK daisy-chain driver initiates activity on the IACKIN*/IACKOUT* daisy-chain during an interrupt acknowledge cycle. It makes sure only one interrupter responds and provides the correct timing for the daisy-chain.

The bus timer measures the time it takes for each data transfer. If the transfer takes longer than the time allotted, it asserts BERR* to terminate the cycle.

System Controller Operation

A card that provides System Controller functions must be installed in slot 1 of the VMEbus backplane. The System Controller (usually a processor card) provides bus arbitration, checks for timeouts, routes IACK signals, and drives the system clock and system reset signals.

A SGL arbiter only uses one arbitration level (BR3*) for all masters on the bus. SGL is the simplest arbitration method to implement on the VMEbus.

A PRI arbiter provides preferential control of the data transfer bus to the requester with the highest priority. BR3* is the highest priority, and BR0* is the lowest. When two or more requests are pending, the arbiter assigns control of the bus in the appropriate order by granting the bus according to this priority.

The PRI arbiter must assert BCLR* when a requester of higher priority than the one in control of the bus initiates a request. When BBSY* is asserted and a request is pending, the arbiter will drive BCLR* if the pending request is of higher priority than the bus grant of the previous arbitration cycle. The current bus master is not required to relinquish control of the bus in any prescribed time limit, it can continue transferring data until it reaches an appropriate stopping point.

A RRS arbiter gives equal priority to all bus request levels. It grants control of the bus on a rotating basis. Upon release of the bus, the arbiter steps to the next request level and tests for an active request. If no request is active, it continues rotating through the levels until an active request is found. When it finds an active request, it asserts bus grant on that level.

The RRS arbiter can drive the BCLR* signal. In RRS mode, BCLR* is asserted whenever a master requests the bus on a level other than the last one granted. It does not assert BCLR* if a master on the same level requests the bus.

➔ There must be one and only one card with System Controller Mode enabled. This card must be installed in slot 1.

Backplane Jumpers

VMEbus chassis have five jumpers associated with each slot except possibly slot 1. The five jumpers pass the bus grant and interrupt acknowledge signals to the next slot. If a slot is empty, jumpers must be installed in order to pass the daisy-chained signals to the next slot. For example, if slots 1, 5 and 7 have cards installed, slots 2, 3, 4, and 6 must have backplane jumpers installed or the system will not function properly.

- ➔ If a slot has no card installed and a card is installed in higher number slot, the backplane jumpers must be installed in the empty slot.
- ➔ Backplane jumpers should never be installed in slots in which cards are installed.
- ➔ The jumpers can be on the front side, back side, or both sides of the backplane.

Some backplanes can detect when no card is installed in a slot. These backplanes are designed to automatically pass the bus grant and interrupt acknowledge to the next slot, without using jumpers.

*SYSRESET**

System reset, *SYSRESET**, is used to reset the system. When *SYSRESET** is asserted, devices on the bus initialize themselves, and bus masters must stop performing bus cycles. *SYSRESET** can be driven by any device.

*SYSFAIL**

The VMEbus has a diagnostic and system failure capability. During system start-up each device on the bus can be tested to determine if it is functioning. After successful testing, the system boots normally. If a device fails the test, the system can stop and display error messages.

*SYSFAIL** is a utility signal that can be used to facilitate system diagnostics. Although it is recommended that devices drive *SYSFAIL**, they are not required to do so. *SYSFAIL** can be driven anytime during normal operation. If a failure is detected, *SYSFAIL** shuts down the system.

*ACFAIL**

*ACFAIL** is asserted when the AC line voltage drops below a system defined level. This notifies all devices that power is about to quit so that they can shut down in an orderly manner to prevent data loss or equipment damage.

VMEbus Pin Assignments

P1/J1			
Pin #	Row A	Row B	Row C
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERA	A17
22	IACKOUT*	SERB	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12 VDC	+5VSTDBY	+12 VDC
32	+5 VDC	+5 VDC	+5 VDC

* = active low

P2/J2			
Pin #	Row A	Row B	Row C
1	User Defined	+5 VDC	User Defined
2	User Defined	GND	User Defined
3	User Defined	RETRY*	User Defined
4	User Defined	A24	User Defined
5	User Defined	A25	User Defined
6	User Defined	A26	User Defined
7	User Defined	A27	User Defined
8	User Defined	A28	User Defined
9	User Defined	A29	User Defined
10	User Defined	A30	User Defined
11	User Defined	A31	User Defined
12	User Defined	GND	User Defined
13	User Defined	+5 VDC	User Defined
14	User Defined	D16	User Defined
15	User Defined	D17	User Defined
16	User Defined	D18	User Defined
17	User Defined	D19	User Defined
18	User Defined	D20	User Defined
19	User Defined	D21	User Defined
20	User Defined	D22	User Defined
21	User Defined	D23	User Defined
22	User Defined	GND	User Defined
23	User Defined	D24	User Defined
24	User Defined	D25	User Defined
25	User Defined	D26	User Defined
26	User Defined	D27	User Defined
27	User Defined	D28	User Defined
28	User Defined	D29	User Defined
29	User Defined	D30	User Defined
30	User Defined	D31	User Defined
31	User Defined	GND	User Defined
32	User Defined	+5 VDC	User Defined