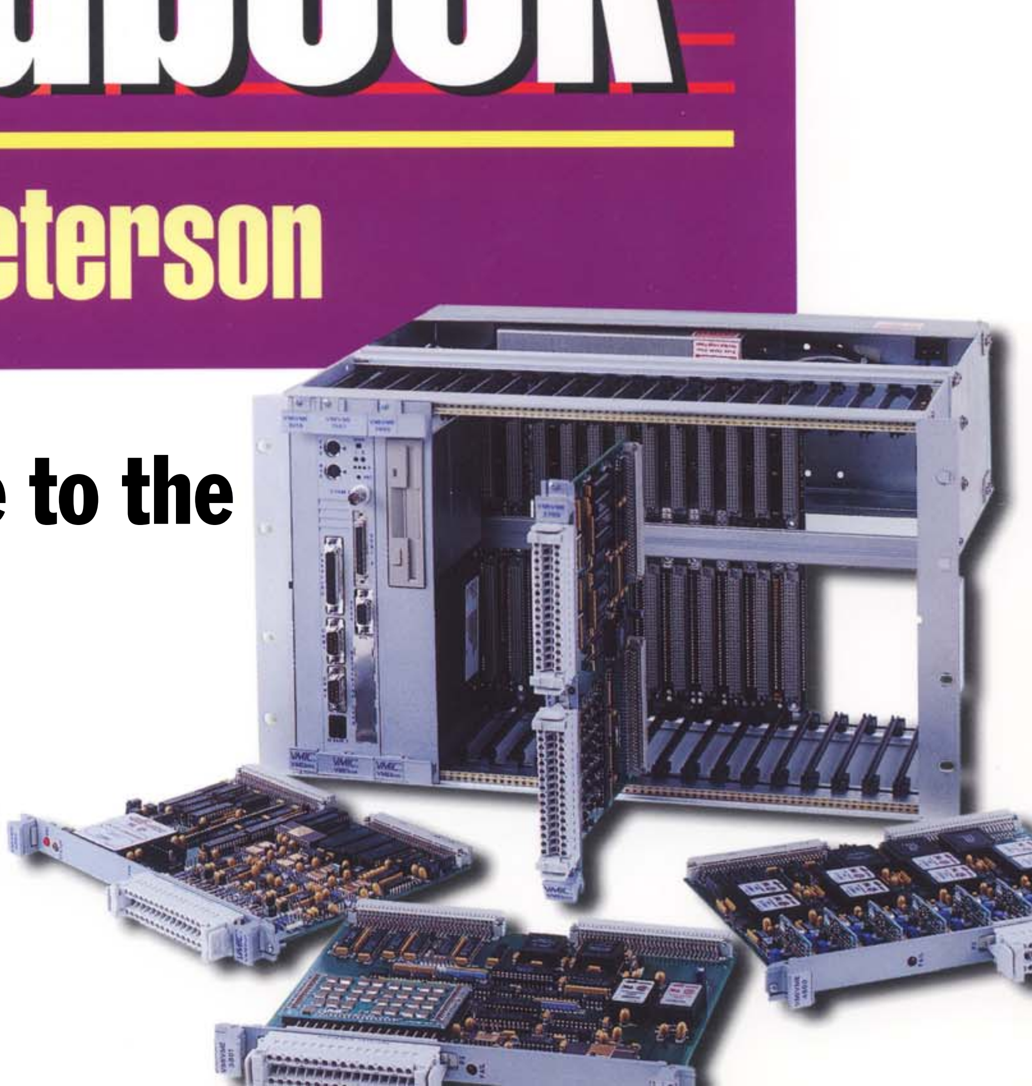
The VMEbus Handbook

4th EDITION

Wade D. Peterson



A User's Guide to the VME64 and VME64x Bus Specifications

VITA

Open Standards, Open Markets

The VMEbus Handbook

A User's Guide to the VME64 and VME64x Bus Specifications

Wade D. Peterson

Published By: VITA, www.vita.com

First Edition in 1989
Second Edition in 1991
Third Edition in 1993
Second Printing in 1996
Fourth Edition in 1997
Second Printing in 2006

DISCLAIMER

The author and VITA and all companies and individuals who contributed to the development of this document make no warranty for the use of this document, the VMEbus and related buses nor for interfacing products to the buses, and they assume no responsibility for consequential damages nor for errors appearing in this document. Every effort has been made to make this book as accurate as possible. However, it is impossible to find every error (both content and typographical) in an original work of this sort. The reader is advised to double check any information found herein.

The author and publisher reserve the right to make changes to any part of this book without notice.

Cover Photography: courtesy of VMIC (VME Microsystems International Corporation). Used with permission.

ISBN 1-885731-08-6

Acknowledgements

Like any great technical project, the *VMEbus Handbook, Fourth Edition* could not have been completed without the help of many people. The author wishes to thank the following for their ideas, suggestions and contributions:

Ray Alderman (VITA)
Dick Anderson (Hewlett Packard)
Chris Barecki (Xycom)
Debbie Beagan (Schroff)
Mark Bemis (MTS Systems)
Rob Bigelow (Bigelow Photography)
John Black (VMEbus Systems Magazine)
Fred Bode (Bode Enterprises)
Rick Brown (Motorola)
William Cabral (Schroff)
Cheryl Cook (VITA)
Ken Covey (Motorola MCD)
Dick DeBock (Matrix Corporation)
Suzanne Dodson (Dawn VME Products)
Bob Downing (VIPA SIG)
Kerry Fisher (VMEtro)
Dave Flessert (Marquip)
Wayne Fischer (Force Computers)
Cary Fox (Technico)
Ernst Godsey (Mizar Digital Systems)
Wil Hamsher (AMP Incorporated)
Billy Harris (National Instruments)
Betty Hevle (VITA)
Micheal Humphrey (BICC-VERO / VITA)
Zoltan Hunor (VITA)
Steve Johnson (Microware)
Tad Kubic (Dawn VME Products)
Mary Klang (Matrix Corp.)
Takashi Kusanagi (Ampere Inc.)
Dr. Joel Libove (Ultraview)
Tom Lundell (MTS Systems)
John McCann (VITA)
Jonathan Morris (Tundra Semiconductor)
Dave Nelson (MTS Systems)
Jack O'Connel (Mupac Corporation)
Bill O'Brien (CCT Technology Inc.)
Doug Patterson (Radstone Technology)
Randy Ridenour (PEP)
John Rynearson (VITA & elsewhere)
Sue Smeltzer (DY-4)
Barbara Sproull (VITA)
Morton Thayer (Augat)
Phillip Vukovic (Bit-3 Computer)
Eike Waltz (Rittal)
Lollie Wheeler (VITA)
Joel Witt (VMElaboratories)
Todd Wynia (Heurikon)
Yoshio Yamaguchi (VME Member)

I would also like to thank several people who put in an extra effort on this project: Lollie Wheeler for editing the fourth edition, John Rynearson for his numerous suggestions, Ray Alderman for his continued support, John McCann for the cover layouts and advertisements and Lym Hevle for his pioneering spirit on the original edition.

Others include: Tony Simons at Mizar who corrected the draft copy of the first edition and allowed me to bounce some crazy ideas off his head, Al Middleton at Interphase who slaved through endless hours of proof reading and still maintained a sense of humor, Dale White at Lee Data who provided excellent technical and grammatical suggestions and David Haworth at Tektronix who contributed to the text in Chapter 9.

Finally, I would like to thank my students at the UNDERSTANDING VME64 seminars...because they ask some really good questions.

Without the help of these people, this project just simply couldn't have been done.

Wade Peterson
November 1997

Preface

The typical reader of *The VMEbus Handbook, Fourth Edition* is a system integrator using VMEbus for the first time. System integrators encompass a broad range of people such as software engineers, hardware engineers, technicians, project managers, mechanical designers, service engineers and production people.

The subject matter of The VMEbus Handbook includes a practical discussion of VMEbus functional modules, sub-busses, mechanical hardware, integration techniques and so on. Software issues are included only to illustrate other bus concepts.

Circuit examples are given whenever possible, and are intended to guide users interested in designing their own bus modules. Some people have the opinion that system integrators buy all of their modules. However, custom modules are often used. One recent survey showed that over 60% of all VMEbus systems include at least one custom module.

All *circuit examples* in The VMEbus Handbook have been built and tested by the author. Illustrations labeled as *circuit ideas* have not been tested. These relied on information from the manufacturer's data sheets and/or conversations with various individuals.

In all cases the IEEE 1014-1987, ANSI/VITA 1-1994 (VME64) and VITA 1.1-1997 (VME64x) versions of the VMEbus specification are used as the basis for this book. Whenever possible, the differences between the various versions of the specification have been noted.

The *VMEbus Handbook, Fourth Edition* can be thought of as a companion to the bus specification. Chapters 1 - 7 are organized like the VMEbus specification. Chapter 8 covers system integration and Chapter 9 discusses VXIbus.

Newer versions of the bus specification are identified throughout the book. New features in the VME64 standard are marked with a single dagger (†), new features new in VME64x are indicated by a double dagger (††) and references to proposed new standards are indicated by a triple dagger (†††). Users should be aware that emerging new standards may have changed since this handbook was printed.

The fourth edition of the book coincides with its sixth printing and its eighth anniversary. Major new topics introduced since the third edition include: final updating to the ANSI/VITA 1-1994 (VME64) and VITA 1.1-1997 (VME64x) versions of the VMEbus specification, additional circuit ideas, updating the VXIbus chapter to include the Revision 1.4 and IEEE versions, 9U VMEbus modules and the High Availability VME64x standard. More text, illustrations, and photos were added, and all material was brought up to date.

Chapter 9 on the Futurebus+/VME64 Bridge has been removed. It was originally included in the book because, at one time, it appeared that Futurebus+ was going to replace VMEbus. However, wide acceptance of Futurebus+ never materialized, and the chapter was removed.

Various internet (e-mail or world wide web) addresses have also been included in the book. Undoubtedly, these will change over time, as this form of communication is still evolving. The author apologizes in advance for any inconvenience this may cause.

Comments or suggestions about *The VMEbus Handbook* are welcome, and should be directed to: *info@vita.com*.

Table of Contents

Chapter 1
Introduction To VMEbus

VMEbus is a computer architecture. The term 'VME' stands for *VERSAmodule Eurocard* and was first coined in 1980 by the group of manufacturers who defined it. This group was composed of people from Motorola, Mostek and Signetics corporations who were cooperating to define the standard. The term 'bus' is a generic term describing a computer data path, hence the name *VMEbus*.

Actually, the origin of the term 'VME' has never been formally defined. Other widely used definitions are *VERSAbus-E*, *VERSAmodule Europe* and *VERSAmodule European.* However, the term *Eurocard* tends to fit better, as VMEbus was originally a combination of the VERSAbus electrical standard, and the Eurocard mechanical form factor.

VERSAbus was first defined by Motorola Corporation in 1979 for its 68000 microprocessor. Initially, it competed with other buses such as Multibus™, STD Bus, S-100 and Q-bus. However, it is rarely used anymore.

The microcomputer bus industry began with the advent of the microprocessor, and in 1980 many buses were showing their age. Most worked well with only one or two types of microprocessors, had a small addressing range and were rather slow. The VMEbus architects were charged with defining a new bus that would be microprocessor independent, easily upgraded from 16 to 32-bit data paths, implement a reliable mechanical standard and allow independent vendors to build compatible products. No proprietary rights were assigned to the new bus, which helped stimulate third party product development. Anyone can make VMEbus products without any royalty fees whatsoever.

Since much work was already done on VERSAbus it was used as a framework for the new standard. In addition, a mechanical standard based on the Eurocard format was chosen. *Eurocard* is a term which loosely describes a family of products based around the DIN 41612 and IEC 603-2 connector standards, the IEEE 1101 PC board standards and the DIN 41494 and IEC 297-3 rack standards. When VMEbus was first developed, the Eurocard format had been well established in Europe for several years. A large body of mechanical hardware such as card cages, connectors and sub-racks was readily available. The pin and socket connector scheme is more resilient to mechanical wear than older printed circuit board edge connectors.

The marriage of the VERSAbus electrical specification and the Eurocard format resulted in VMEbus Revision A. It was released in 1981.

The VMEbus specification has since been refined through revisions B, C, C.1, IEC 821, IEEE 1014-1987 and ANSI/VITA 1-1994. The ANSI, VITA, IEC and IEEE standards are important because they make VMEbus a publicly defined specification. Since no proprietary rights are assigned to it, vendors and users need not worry that their products will become obsolete at the whim of any single manufacturer.

The most recent version of VMEbus is the ANSI/VITA 1-1994, which is also known as VME64. Since its approval in 1995, a variety of boards and chips have been introduced to support the new standard. As the next generation architecture for VMEbus, VME64 promises to extend the life of VMEbus well into the 21st century. The new standard offers a much-needed 'face-lift', with enhancements such as higher bandwidths, larger address spaces and easier-to-use cards.

As shown in Figure 1-1, VME64 is a mechanical and electrical 'superset' of the original IEEE 1014-1987 standard. It offers new features such as:

- Larger, 64-bit data path for 6U boards.
- Larger, 64-bit addressing range for 6U boards.
- 32-bit data and 40-bit addressing modes for 3U boards.
- Twice the bandwidth (up to 80 Mbytes/sec).
- Lower noise connector system.
- Cycle retry capability.
- Bus LOCK cycles.
- First slot detector.
- Automatic 'plug-and-play' features.
- Configuration ROM / CSR capability.
- Re-definition of SERCLK and SERDAT pins.

The VME64 standard also provided a foundation for another standard called the VMEbus Extensions, or VME64x. VME64x adds new capabilities such as:

- A new 160 pin connector family.
- A 95 pin P0/J0 connector.
- 3.3 V power supply pins.
- More +5 VDC power supply pins.
- Geographical addressing.

- Higher bandwidth bus cycles (up to 160 Mbytes/sec).
- 141 more user-defined I/O pins.
- Rear plug-in units (transition modules).
- Live insertion / hot-swap capability.
- Injector / ejector locking handles.
- EMC (ElectroMagnetic Compatible) front panels.
- ESD (Electrostatic Discharge) features.

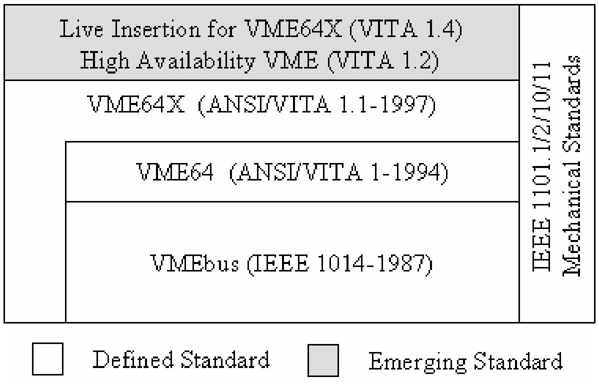


Figure 1-1 Important VMEbus Standards

The VME64x standard also lays the groundwork for the High Availability and Live Insertion (Hot Swap) VME64x standards. These are emerging standards, and will not be covered in great detail in this book.

Actually, the term 'VME64' is something of a misnomer, as all VMEbus modules that conform to the IEEE-1014-1987 are now considered to be VME64 compliant (regardless of their data transfer capability). For example, a 16 or 32-bit CPU board designed under the older specification can be (correctly) identified as *VME64 compatible*.

All of the enhancements under VME64 and VME64x are optional. New products work in conjunction with older 'legacy' boards, thereby providing a smooth upgrade path for system integrators.

In all cases the ANSI/VITA 1-1994 (VME64) version of the VMEbus specification is used as the basis for this book. However, features and options that are new to VME64 are identified with a single daggar (†), and those new to VME64x are identified with a double daggar (††). Proposed new enhancements to the standards, which are not yet complete, are identified with a triple daggar (†††).

Since its introduction, VMEbus has generated thousands of products and attracted hundreds of manufacturers of boards, mechanical hardware, software and bus interface chips. It continues to grow and support diverse applications such as industrial controls, military, telecommunications, office automation and instrumentation systems.

1.1 System Integration

At the beginning of any project the system integrator must navigate through a myriad of design choices. These depend upon the objectives of the project, which are often answered by the following questions:

- What technologies do we use?
- Do we make or buy the major components?
- Does it need to be software compatible with our last (or our competitor's) system?
- How many are we going to make?
- How fast does the project need to be completed?
- What is the total cost of the system?
- How reliable is it going to be?
- Does it need to conform to any regulations (CE, UL, CSA etc.)?

Before answering these questions, we must first define the objectives of the product development process.

1.1.1 Product Development Objectives

As shown in Figure 1-2, there are three major goals in any new product development: cost, performance and time-to-market. These can be broken down into sub-objectives (such as reliability, features and so on), but all major product decisions eventually break down to these three objectives.

Very often there is a tendency to optimize all of these goals in a product. For example, it may be expected to minimize cost, maximize performance and be brought quickly to market. However, this rarely happens. A well designed product can be expected to achieve one or two of these goals, but rarely (if ever) all three of them.

There are many examples of this in products we use everyday. In automobiles, for example, the performance of a Mercedes Benz could probably be delivered at the cost of a Ford Escort, but it would take a long time to develop the product. A goal like that would take major advancements in design, manufacturing and technology. Therefore, most car manufacturers offer a range of products, with each offering a different mix of cost, performance and time-to-market.

Computer system integrations are no different. The system can achieve ultimate performance, but only by adding cost or development time. It can be cheap, but it won't get done very fast or operate at the cutting edge of performance. The product can be developed in a hurry, but it's going to cost a lot or it won't perform very good.

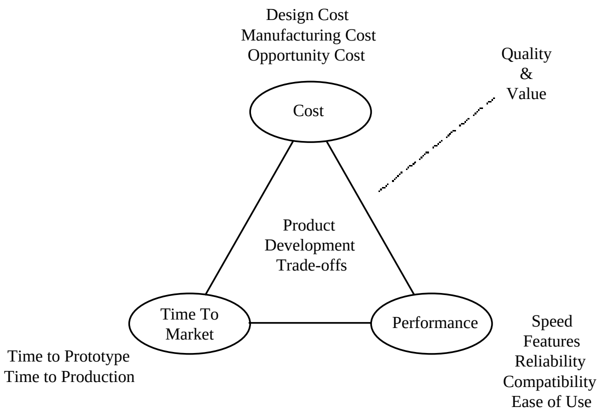


Figure 1-2 Product Development Trade-offs

A good project manager will build a consensus on these issues at the beginning of any project. He or she will get every member of the project team to 'buy into' whatever goals are needed. That way, all members of the team will be working together with a common purpose. When this happens, the project will run much smoother.

1.1.2 The Make Vs. Buy Decision

One major decision in any project is whether to make or buy the major components. Ultimately, this must be handled in terms of the *core competencies of the corporation*. Every company (or any *organization* for that matter) has core competencies, and its employees must take the initiative to identify, practice and protect them. This allows the company to dedicate scarce resources (such as labor or capital) to the core business, and delegate unrelated aspects to outside suppliers.

The microcomputer board industry is a classic example: companies who buy these products are generally those whose core competency lies outside of the hardware or operating system design. For example, a company that specializes in sawmill controls might very well decide that their core competency lies in their ability to solve automation problems, and provide good customer service. Since computer parts (such as VMEbus boards), operating systems, cables and other items can be purchased from outside, the company can divest itself from these unrelated activities.

This sounds good in principle, but in practice it is sometimes hard to do. There is always a tendency to bring too many aspects of a product 'in-house'. Companies often feel there's a need to keep one's technology proprietary, regardless of the consequence to the customer. This is a very complex problem. Oftentimes, a company may decide that their technology (design, manufacturing, etc.) is too valuable to send outside. Other times there's a feeling that 'nobody else can do this better than we can.'

The best way to differentiate between core and non-core competencies is look at the uniqueness of the business. For example, in 1980, there were very few choices in terms of off-the-shelf hardware and software. Companies needing this technology were forced to develop their own. The design and fabrication of hardware and software components was part of their core competencies because they were unique to the business.

Today the situation has changed dramatically. In the VMEbus realm, there are hundreds of hardware and software vendors who make this type of equipment. Products are extremely flexible, and are no longer unique to niche markets.

There are many good examples of this in other areas of the computer business. At one time it made sense to make your own circuit boards, power supplies and integrated circuits. Many companies had no choice because there weren't any good alternatives...they had to be *horizontally* integrated. Except in very large volume products (like consumer electronics) horizontal integration makes little sense anymore.

In small and medium size corporations, *vertical integration* tends to work much better. In those organizations items are purchased whenever they lie outside of their core competencies. This makes for a much more efficient (and profitable) company.

1.1.3 Microcomputer Bus Trade-Offs

System integrators have three alternatives to choose from when developing a system: build it from scratch, adapt a standard computer or use microcomputer boards. Again, this choice is affected by cost, performance and time to market. Figure 1-3 shows these trade-offs.

The full custom system is the costliest in terms of development time and money. Its main advantage is that it can be exactly tailored to an application and can be cheaply produced in large volumes. It might also be the only choice if no other solution exists. Full custom might be a poor choice if a simple machine control must be built in six weeks, but it would probably be the right choice if thousands of units are needed, and a year or two were available for product development.

Adapting a standard system (such as an off-the-shelf IBM-PC™) is the most cost effective in modest volumes. The major drawbacks of this approach are inflexibility and (perhaps) lack of ruggadized hardware. If special functions, I/O or mechanical packaging are needed, the standard system might be too inflexible. For example, a personal computer can be adapted to perform bench top data acquisition, but might not be able to handle a sophisticated control application.

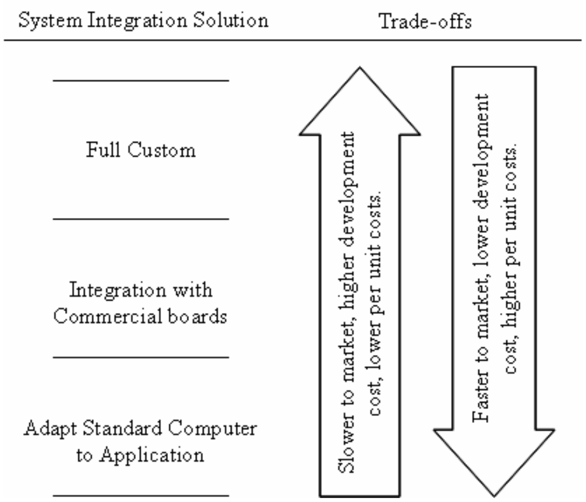

System Integration Solution
Trade-offs
Full Custom
Integration with Commercial boards
Adapt Standard Computer to Application
Slower to market, higher development cost, lower per unit costs.
Faster to market, lower development cost, higher per unit costs.

Figure 1-3 System Integration Trade-Offs

Integration with microcomputer boards is a compromise between custom and standard systems. They can be pieced together with available parts and minimum tooling costs. Generally they are used for small and moderate volume applications where up to 1000 units are built. In many cases the system can be integrated by people with little or no background in electronic hardware.

Another advantage to microcomputer boards is flexibility. At the onset of product development the requirements of the computer system are only generally known. As the project progresses more of the application becomes obvious, and the computer architecture must adjust accordingly. This is especially true of exotic or first generation products. As the product becomes more refined, so does the computer architecture. In many cases the microcomputer bus is the most flexible because standard boards can be swapped out at any time.

1.1.4 Economies Of Scale

Economies of scale are also a big part of the make vs.buy issue. If you make a few hundred or thousand of something it's tough to compete with a similar technology where millions are made. When raw materials are plentiful, a large-volume competitive environment always leads to lower overall cost. If ten companies are competing to make the same card cage, then you will always get better pricing (and quality). It's Darwin's theory in action...survival of the fittest.

This is an area where standards work in favor of the small user. Standard products or services such as VMEbus boards, racks, software and human resources make fewer, larger markets out of many, smaller ones. Niche markets are converted into mainstream ones.

Economies of scale also make it much cheaper to stay on the knee of the so-called *technology curve*. There are many examples of this: minicomputers replaced mainframes, workstations replaced minicomputers, and desktop PC's are now replacing workstations. In each of these cases, users found it much cheaper to stay up on the technology curve with the higher volume products.

Pricing in the microcomputer board business is mainly one of economies of scale, and volume is an important catalyst. For example, you can walk into any computer store and buy an IBM-PC parallel port card for $US 40 (full retail). The same type of VMEbus board may cost about $US 200. Why the big difference? Volume! You can make some arguments that the VMEbus parts cost more (connector, front panel, bus interface, etc.), but in reality they cost about the same in small quantities. The big difference is volume.

In terms of price, VMEbus products will never be able to compete with PC buses like ISA or PCI. However, these other products have not been tailored to the niche markets served by VMEbus. They will never thrive in applications like robust industrial control, demanding military systems or telephone switches.

1.1.5 Estimate The Total Cost Of The Project

Very often the make vs. buy issue comes down to cost. When most people figure out product price, they compare parts costs. This is very short-sighted, as there are many other costs involved in making something:

- Time to market (opportunity costs).
- Inventory (interest costs).
- Lead-times (for replenishing inventory).
- Design.
- Test engineering (fixtures, labor, etc.).
- Manufacturing tooling, personnel and floor space.
- Direct labor to manufacture and test.
- User manuals.
- Test manuals.
- ISO 9000 procedures.
- Common knowledge (i.e. are the idiosyncrasies well known).
- Warranty repair costs.

Although microcomputer boards (such as VMEbus) may seem to cost more than other solutions, the actual cost of custom products is much higher.

1.2 Advantages of VMEbus

Once it is decided to use a microcomputer bus, the choice of which one to use must be made. A wide variety of buses are available, with new ones introduced regularly. Table 1-1 shows a comparison between VMEbus and some other popular buses. The choice depends on application, and is affected (again) by cost, performance and time to market.

1.2.1 VMEbus Features

A few features of VMEbus are listed in Table 1-2. Noteworthy functions include (up to) 64-bit address and data buses, multiprocessing capability and seven level interrupt protocol. Both the address and data buses can be dynamically configured (i.e. they change widths automatically). This allows system expansion as microcomputer technology grows. It also handles data transfer speeds to 160 Mbytes/second.

VMEbus uses a master-slave architecture. Functional modules called *masters* transfer data to and from functional modules called *slaves*. Since many masters can reside on the bus it is called a *multiprocessing* bus. Before a master can transfer data it must first acquire the bus using a central arbiter. This arbiter is part of a module called the *system controller*. Its function is to determine which master gets access to the bus. All bus activity takes place on the four sub-buses shown in Figure 1-4. These are called the *Data Transfer Bus*, the *Data Transfer Arbitration Bus*, the *Priority Interrupt Bus* and the *Utility* bus.

VMEbus is an asynchronous bus. That means that there are no clocks used coordinate data transfers. Data is passed between modules using interlocked handshaking signals. The cycle speed of each transfer is set by the slowest module participating in the cycle.

The maximum speed of asynchronous buses is limited by the propagation delay of signals across backplanes and through buffer ICs. A VMEbus backplane can be up to 500 mm (19.68") in length and can have relatively high inductive and capacitive loads on the signal traces. If VMEbus were synchronous, it would probably have a system clock speed of around 10 MHz. This allows about 100 nanoseconds for a signal to propagate from a master at one end of a bus, through the backplane and interface ICs, and then back again.

This speed has been doubled with the 2eVME protocol. This bus cycle uses new transciever technology and cycle timing to increase VMEbus bandwidth from 80 Mbyte/second to 160 Mbyte/second.

Table 1-1 Features of Some Popular Microcomputer Buses

Bus	Sync (S) or Async (A)	Multiplexed	Data bus width (bits)	Address bus width (bits	Interrupts (levels)	Multiprocessing	Card size (MM)	Connector style	Governing body
CompactPCI	S	Y	32/64	32	Y	?	160 x 100 or 233 Eurocard	2mm Metric	PICMG
IBM-PC	A	N	8	20	Y (6)	N	335 x 106	Card edge	IBM Microsoft
Multibus	A	N	8, 16	8, 16, 20, 24	Y (8)	Y	305 171	Card edge	IEEE 796
Multibus II	S	Y	8, 16, 24, 32	32	N	Y	233 x 220 Eurocard	DIN 41612	IEEE 1296
Nubus	S	Y	32	32	N	Y	233 x 160, 220 x 280 Eurocard	DIN 41612	TI
Q-Bus	A	Y	8, 16	16, 18, 22	Y (4)	Y	214 x 132, 257 or 393	Card edge	Digital Equipment Corp.
STD Bus	A	N	8	16	Y (2)	Y	114 x 165	Card edge	IEEE 961
S-100	A	N	8, 16	16, 24	Y (8)	Y	254 x 130	Card edge	IEEE 696
VERSAbus	A	N	8, 16, 32	16, 24, 32	Y (7)	Y	368 x 235	Card edge	IEEE 970
VMEbus	A	NY	8, 16, 24, 32, 64	16, 24, 32, 64	Y (7)	Y	160 x 100 or 233 Eurocard	DIN 41612	ANSI VITA 1-1994

Notes: All dimensions rounded and presented in millimeters for comparison.

Y = Yes, N = No

Table 1-2 VMEbus Features

Item	Specification	Notes
Architecture	Master/Slave	
Transfer mechanism	Asynchronous, both multiplexed (†), and non-multiplexed.	No central synchronization clock
Addressing range	16-bit (short I/O) 24-bit (standard) 32-bit (extended) 64-bit (long)	Address range selected dynamically
Data path width	8, 16, 24, or 32-bit 64-bit (†)	Data path width selected dynamically
Unaligned data transfers	Yes	Compatible with most popular microprocessors
Error detection	Yes	Using BERR* (optional)
Data transfer rate	0 – 40 Mbyte/sec 0 – 80 Mbyte/sec (†) 0 – 160 Mbyte/sec (††)	Experimental VME320 backplane to 500 Mbyte/sec.
Interrupts	7 levels	Priority interrupt system with STATUS/ID
Multiprocessing capability	1-21 processors	Flexible bus architecture
System diagnostic capability	Yes	Using SYSFAIL*
Hot-swap capability	Yes (†††)	High Availability VME64 & BLLI Standards (†††)
Control & Status Registers	Yes (†) (††)	VME64 VME64x
Conduction Cooled (military) version	Yes	IEEE1101.2
Geographic Addressing	Yes (††)	VME64x
Mechanical standard	3U Single height 6U Double height 9U Triple height (†††)	160 X 100 mm Eurocard 160 X 233 mm Eurocard 367 X 400 mm Eurocard (†††)
International standards	Yes	ANSI/VITA 1-1994
(†) - Enhancement under VME64 (††) - Enhancement under VME64x (†††) - Proposed VMEbus enhancement		

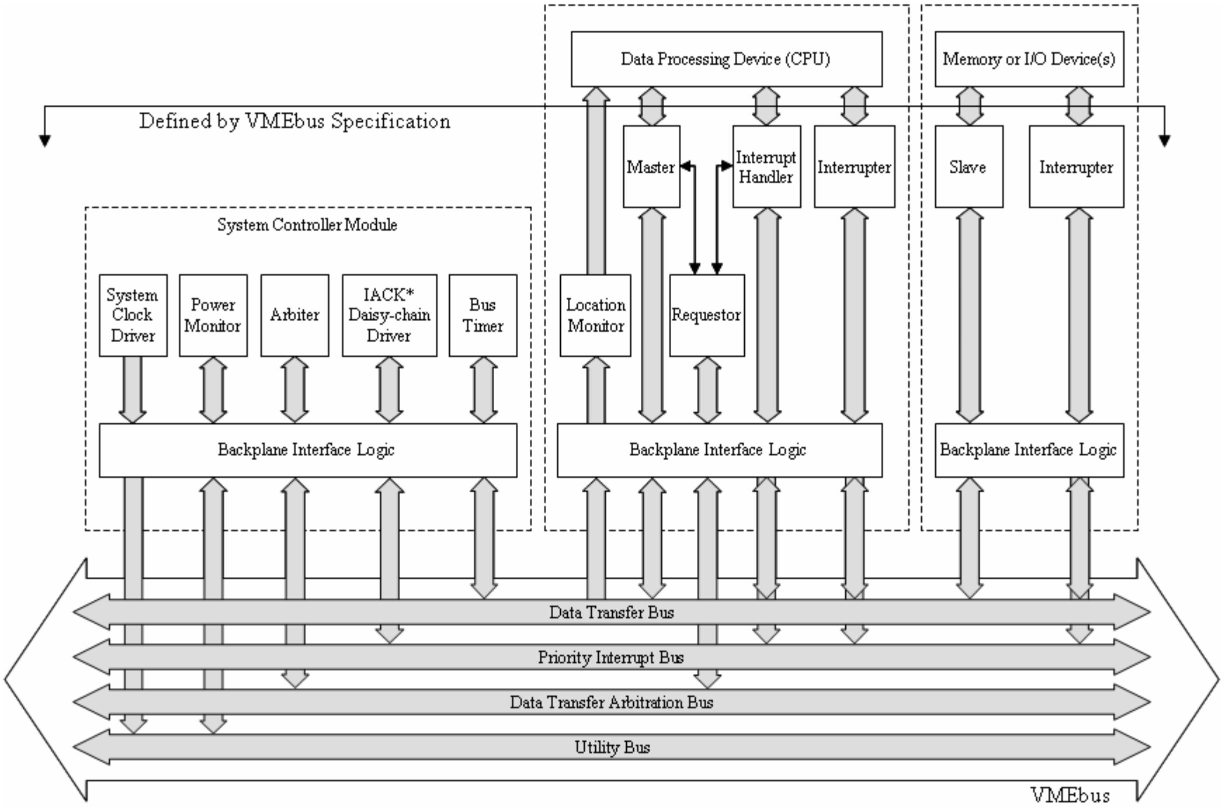
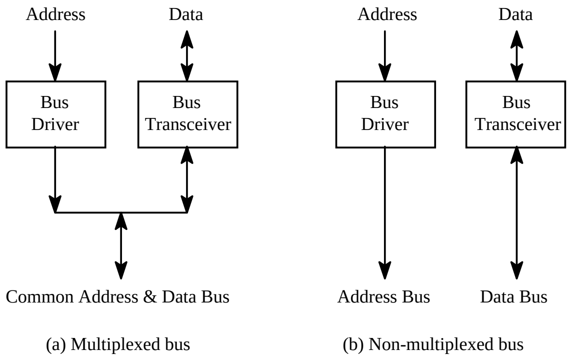


Figure 1-4 Functional Block Diagram of VMEbus

1.2.1.1 Multiplexed And Non-multiplexed Architectures

Microcomputer buses usually fall into one of two general categories: multiplexed and non-multiplexed. As Figure 1-5 shows, multiplexed buses share the same set of pins for address and data lines. A two part bus cycle is required to transfer data. During the first portion of the cycle an address is passed over the pins. Data is then transferred during the second half of the cycle. Non-multiplexed buses have separate pins for both address and data lines. Unfortunately this also means that more bus interface circuitry and signal pins are required.



(a) Multiplexed bus

(b) Non-multiplexed bus

Figure 1-5 Multiplexed and Non-multiplexed Buses

VMEbus uses both multiplexed and non-multiplexed architectures. For address and data transfers of up to 32-bits, VMEbus uses a non-multiplexed architecture. For 64-bit transfers, it uses a multiplexed architecture.

All bus cycles in the IEEE 1014-1987 bus specification are non-multiplexed. That standard uses separate 32-bit address and data paths. However, the VME64 and VME64x specifications allow special multiplexed transfers using the MBLT, MD32 and 2eVME cycles.

1.2.1.2 Backplanes

VMEbus modules are connected together by a backplane. Each backplane can support between one and twenty-one bus modules. One or two backplanes may be used depending upon the system configuration. A *J1* backplane handles 24 address bits, 16 data bits, some power pins and most of the control signals. A second backplane, called *J2*, adds an additional eight address lines and sixteen data lines. It also has more power and user-defined I/O pins.

The two backplanes can be mounted separately or as a single unit. Figure 1-6 shows a 12 slot combination (monolithic) J1/J2 backplane mounted in a sub-rack (the J1 portion of the backplane is at the top).

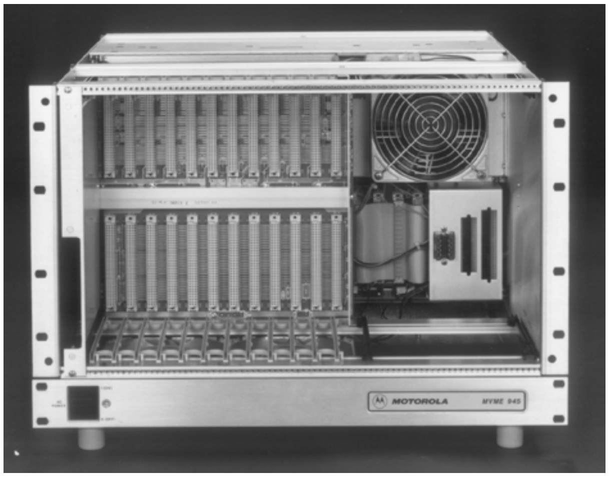


Figure 1-6 Sub-rack with 12 slot combination J1/J2 backplane

This backplane is shown with standard DIN 41612 connectors.

Photo courtesy of Motorola Computer Group.

Most of the signals on the VMEbus backplane are bussed and terminated. Active or passive termination technologies can be used.

VMEbus uses the conventional jack and plug connector designation scheme. Jack connectors J1 and J2 are located on the backplane, and plug connectors P1 and P2 are located on the boards.

VMEbus modules are located on 0.8" centers. This is sometimes called the *board pitch*.

Backplanes which are VME64x compliant must include some special features. For example, a 6U VME64x backplane must have:

- 160 pin connectors.
- All defined pins must be connected.
- Monolithic PCB (i.e. it must not use separate J1 and J2 backplanes).
- Geographic address pins.
- Must route and terminate all VME64 and VME64x bused signal lines.
- Power connection and distribution for +5V, +3.3V, +/- 12V, +/- V1/2 and VPC.
- If rear (user defined) I/O pins are supported, then the rear connections must comply with the IEEE 1101.11 rear I/O transition board standard.

1.2.1.3 Styles Of Bus Modules

There are two styles of VMEbus modules. These are called single and double height modules. The smallest is the single height module, and connects to the P1/J1 connector. Traditionally, these modules can generate or accept up to 24-bit address and 16-bit data transfers. However, larger address and data transfers can be acheived with MD32 and 2eVME cycles (which are allowed under VME64 and VME64x standards).

Single height modules are commonly used if space is limited. Because of their size, they are also more resilient to shock and vibration than double height boards. Figure 1-7 shows a single height module.

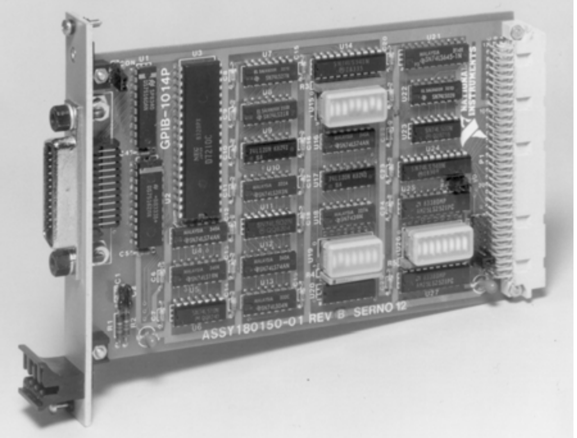


Figure 1-7 Single Height (3U) Module

This module is shown with a standard DIN 41612 connector.

Photo courtesy of National Instruments.

The larger and more popular size VMEbus board is the double height module. These are electrically compatible with single height boards (which use only the P1/J1 connector). Traditionally, 6U modules can generate or accept up to 32-bit address and 32-bit data transfers. However, the VME64 and VME64x specifications have introduced wider, 64-bit transfers in the MBLT and 2eVME cycles. Figure 1-8 shows a double height module.

Single and double height modules are sometimes referred to as 3U and 6U boards. The 'U' is a unit of measure for the front panel, where each 'U' is equal to 1.75 inches.

VMEbus closely follows the *IEEE Standard for Mechanical Core Specifications for Microcomputers* (IEEE 1101). This specification shows the dimensions for a variety of board sizes (up to 12U x 400 mm), as well as sub-racks and other enclosures. The IEEE 1101 was later superseded by the IEEE 1101.1-1991.

Some manufacturers also offer triple height (9U) boards that use three connectors. Strictly speaking, these modules are not supported by the VME64 specification. However, some users choose to mount 3U, 6U and 9U boards into a single sub-rack. Most Eurocard packaging systems will support multiple card sizes in a single chassis. For example, Figure 1-9 shows how an adapter is used to install a 6U board into a 9U card slot. A 9U form factor is also offered in the VXIbus specification.

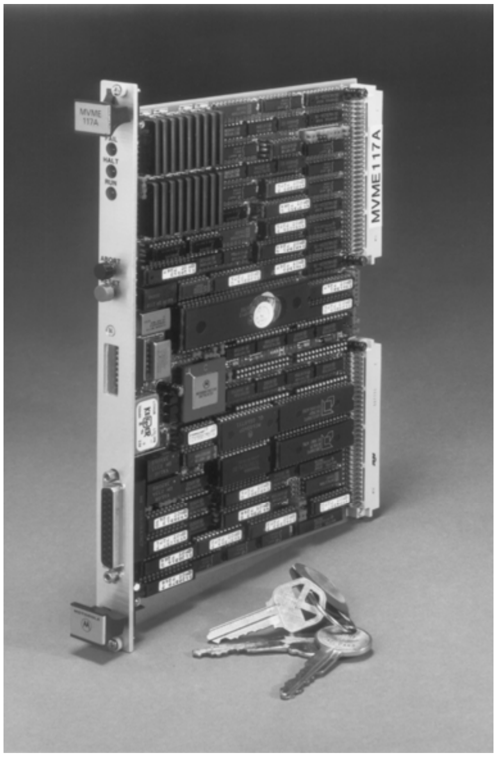


Figure 1-8 Double Height (6U) Module

This module is shown with standard DIN 41612 connectors.

Photo courtesy of Motorola Computer Group.

ANSI/VITA 1.3 is for 9U x 400mm size Eurocards.

The use of two standard card sizes has proven to be one of the biggest features of VMEbus. Users that have tight space requirements or severe shock & vibration constraints may choose the single height module. Users that have the space can go with the larger and more popular 6U card.

Another popular VMEbus card style is the conduction cooled module. These are used mainly in military applications where convection cooling cannot be used. Figure 1-10 shows a conduction cooled module. These allow heat to conduct through the printed circuit board or through a conduction plate on the backside. Special expanding card guides then transfer the heat through rails and then out to the chassis.

Figure 1-9 9U Adapter with 6U Bus Module

This unit allows standard VMEbus modules to be used with Sun workstations.

Photo courtesy of National Instruments.

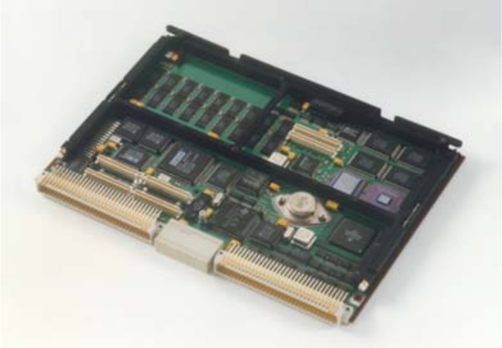


Figure 1-10 A Military-Style Conduction Cooled VMEbus Module

Note that the module has 160 pin P1/P2 connectors, as well as a 95 pin J0 connector.

Photo courtesy of Radstone Technology PLC.

1.2.1.4 160 Pin Connector

Perhaps the most radical change in the VME64 and VME64x standards is a new 160 pin connector. This connector is shown in relation to the standard DIN 41612 connector in Figure 1-11. Figure 1-12 shows the 160 pin backplane connector as well.

As shown in Figure 1-13, the 160 pin connector has thirty-two additional pins located on both sides of the original DIN 41612 connector. These are designated as the 'z' and 'd' rows. The new connector was added for two reasons:

- Additional signal connections allow more backplane functions.
- Additional ground pins suppress ground-bounce and cross-talk.

Four pre-charge pins are provided on the connector. These form a make-first, break-last connection. This feature is intended for hot-swap boards, and are used to pre-charge bus interface circuitry.

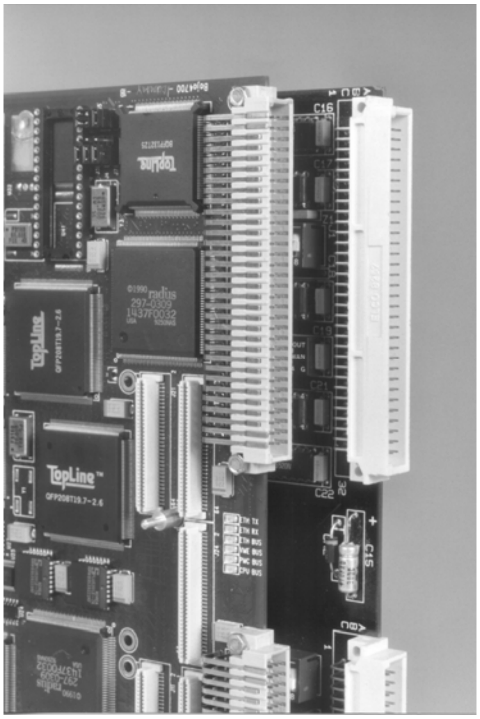


Figure 1-11 Standard DIN 41612 (right) and 160 Pin (left) connectors

Photo by Wade Peterson.

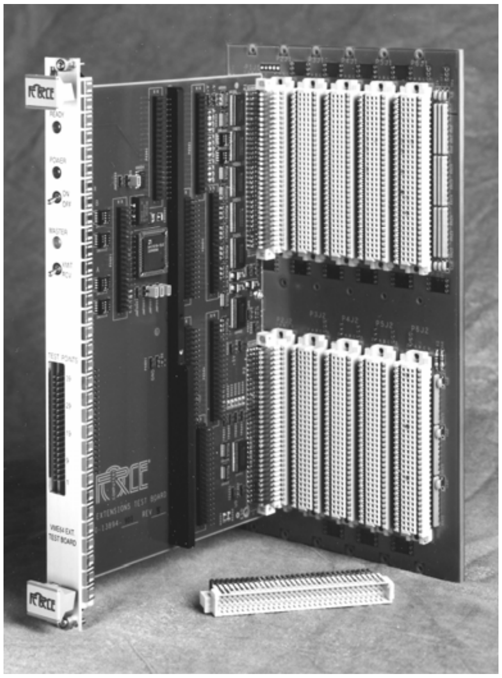


Figure 1-12 Bus Module and Backplane with 160 Pin Connectors

Photo courtesy of Force Computers Inc.

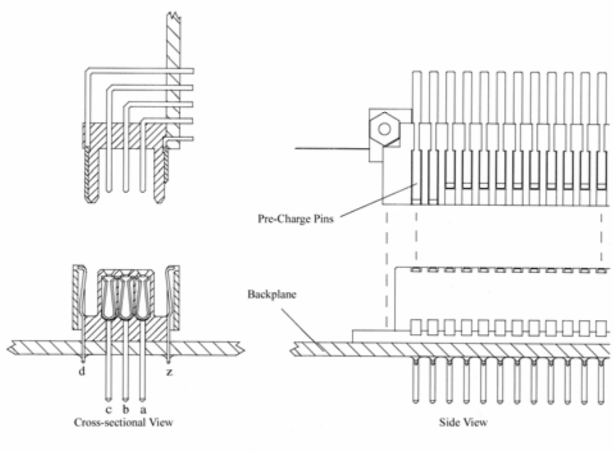


Figure 1-13 Key Features of the 160 Pin Connector

The 160 pin connector is completely optional in the VME64 standard. The connector itself is not needed to support any of the newly defined functions. For example, it is not needed to support the new sixty-four bit address and data cycles. However, in VME64x boards and racks the new connector is needed to support additional pins such as +3.3V power and geographical addressing.

The 160 pin connector is designed to allow full forward and backward compatibility: older legacy boards will fit into new backplanes (and vice-versa).

1.2.1.5 P0/J0 (95 Pin) Connector (††)

The VME64x standard allows the use of a 95 pin P0/J0 connector. This connector, which is shown in Figure 1-10, is placed between the P1/J1 and P2/J2 connectors.

The P0/J0 connector was added because of the higher I/O demands now placed on VMEbus systems. This is especially true in military and telecom applications. For example, in many applications the use of front panel cables are frowned upon because:

- There is not very much room for connectors on the front panels of conduction cooled boards.
- Front panel cables are often susceptible to shock and vibration problems.
- Bus modules are harder to replace when cables are attached to the front panel.

The connector system for the P0/J0 connector conforms to the IEC 1076-4-101 standard. These connectors are available from AMP Inc., and ERNI components. Unlike the DIN 41612 connector family, they include specifications for mating impedance and maximum capacitance. This makes them an excellent choice for high speed signals.

The VME64x standard also allows the use of custom connectors in the region between the P1/J1 and P2/J2 connectors. For example, a coaxial cable or fiber-optic connector could be used. This practice is not recommended, as the board may not be compliant with VME64x backplanes. However, if a custom connector must be used, then the VME64x standard recommends that the front panel keying mechanism be installed to prevent installation of an incompatible module.

I/O from the P0/J0 connector can be routed:

- Out the rear of the backplane using ribbon cable connectors.
- Onto a rear transition module (plug-in unit).
- Bussed across the backplane.

Since the P0/J0 connector signals are user defined, they can be used as needed. However, users should be aware that some other standards have assigned pinouts for this connector. One example is ANSI/VITA 31.1. This standard defines a pin assignment and interconnection methodology for implementing a 10/100/1000BASE-T Ethernet switched network on a VME64x backplane.with a P0/J0 connector.

Rear I/0 for the P0/J0 user defined pins are handled in much the same way as it is on the P2/J2 user defined pins. For example, in the case of the ATM Cells Bus a mezzanine backplane can be placed on the rear of the VMEbus backplane. This has been done before on sub-buses such as VSB and SCSA. In these cases the mezzanine backplane has all necessary bussed and terminated signals. Similarly, in the future users will probably have the option of buying custom backplanes with these signals already bussed and terminated, thereby saving the cost of the mezzanine backplane.

Other standards also map to the P0/J0 user defined pins. One example is user I/O from IP and CMC/PMC mezzanine modules.

At first glance it would appear that all of these options on the P0/J0 connector could lead to incompatibilities. However, this problem should be no worse than the present P2/J2 user defined pins. Also, VME64x module keying can also be used to insure that VMEbus modules are placed into compatible backplane slots.

1.2.1.6 Compatibility

An important part of a reliable bus architecture is card compatibility. This is sometimes a problem in the microcomputer board industry because many modules are developed and tested independently. Boards are often used together for the first time by the end user. Due to the large number of products it is impossible to verify that every module will work with all others on the market. A bus specification must be robust enough to insure that products developed independently (past and future) will work together.

Card compatibility on asynchronous buses was a major concern when VMEbus was first designed. However, thousands of products made by hundreds of vendors have since proven that these modules can be made to be compatible. Some problems occasionally crop up, but these are usually caused by software glitches or incompatible option mixes. VMEbus modules will work together when properly designed.

VMEbus has seen numerous bus specification changes and enhancements since its introduction in 1981. However, the VITA Standards Organization (VSO) has always made board compatibility a major goal whenever it considers these changes. In general, all changes to the bus specification are both forward and backward compatible. However, newer products sometimes have features that are not supported by older 'legacy' boards. For example, VME64 CPU modules that support 64-bit data transfers must also support 8, 16, 24 and 32-bit data transfers. This allows them to coexist with both the older and the newer technologies.

All legacy VME and VME64 modules are forward compatible to the VME64x backplanes and subracks. That means that older bus modules can be plugged inoto newer systems. In general, the reverse is also true. Bus modules designed to the VME64x standard are backward compatible with older backplanes and subracks. For example, the 160 pin connectors can be plugged into an older backplane. However, there are a few exceptions to this. For example, if a board requires the new +3.3 VDC power supplies, then in will not work in an older backplane (which does not have those power pins).

1.2.1.7 Bandwidth

At present, VMEbus can achieve a data throughput in excess of 160 Mbytes/second (using the 2eVME bus cycle). Just as important, however, is its ability to let fast processors transfer data to peripherals of any speed. This allows users to tailor their systems to specific needs based on the cost and availability of modules.

1.2.2 Functional Modules

The VMEbus specification uses functional modules to describe its operation. A functional module is a conceptual tool used to describe each part of the bus. Figure 1-4 shows the various types. Circuits on VMEbus modules may or may not be designed as modular functional modules. They are used only as a conceptual tool.

1.2.2.1 Master

A *master* is a functional module that can initiate data transfer bus cycles. CPU modules and peripherals with DMA controllers are examples of masters.

1.2.2.2 Slave

Slave modules detect bus cycles generated by masters, and participate in the cycles if they are selected. Examples of slaves include memory and I/O modules.

1.2.2.3 Location Monitor

The *location monitor* watches the bus and asserts an on-board signal if certain addresses are selected. It also allows messages to be broadcast to all of the bus modules simultaneously.

An example of a module with a location monitor is a *bus analyzer*. These boards are available from several vendors, and operate much like a special purpose VMEbus logic analyzer. They monitor the VMEbus backplane and generate timing and state information.

1.2.2.4 Bus Timer

The *bus timer* measures how long each data transfer takes. If it takes too long it asserts BERR* to terminate the cycle. It is used to prevent lockups during memory sizing or system failures, and is usually located on the slot 01 system controller module.

1.2.2.5 Interrupter

An *interrupter* generates interrupt requests to an interrupt handler. During an interrupt acknowledge cycle, the interrupter passes a STATUS/ID word (8, 16 or 32-bit) to the interrupt handler. Interrupters are sometimes called *interrupt requesters*.

An example of an interrupter would be a serial I/O module that requests an interrupt every time it receives a character.

1.2.2.6 Handler

An *interrupt handler* responds to requests from interrupters. These modules must be capable of obtaining the data transfer bus, generating an interrupt acknowledge cycle and reading a STATUS/ID word from the interrupter. Interrupt handlers are often found on CPU modules.

1.2.2.7 IACK* Daisy-Chain Driver

During an interrupt acknowledge cycle, the *IACK* daisy-chain driver* initiates activity on the IACKIN*/IACKOUT* daisy-chain. It insures that only one interrupter responds with a STATUS/ID word, and provides the correct timing for the daisy-chain. It resides on the slot 01 system controller.

1.2.2.8 Requester

Bus masters and interrupt handlers use a *requester* to obtain ownership of the data transfer bus. The requester uses the data transfer arbitration bus to handshake with the arbiter. The arbiter grants the bus to the requester, which allows the master to use the bus. The requester is sometimes called a *bus requester*.

1.2.2.9 Arbiter

The *arbiter* monitors bus requests (from requesters) and grants control of the data transfer bus, one master at a time. It resides on the slot 01 system controller.

1.2.2.10 System Clock Driver

The *system clock driver* provides a stable 16 MHz utility clock (SYSCLK) to all bus modules. Since VMEbus is asynchronous, it has no relationship to other bus timing.

1.2.2.11 Serial Clock Driver

The *serial clock driver* is an obsolete functional module. In revisions A, B, C, C.1, IEC 821 and IEEE 1014-1987 of the VMEbus specifications it was used to generate the SERCLK signal. SERCLK is a clocking signal used on a serial bus called VMSbus.

The ANSI/VITA 1-1994 (VME64) specification redefined the SERCLK and SERDAT* pins as *SERA* and *SERB*. These are now user defined to support a wide variety of serial buses. However, VMSbus can still be used on them.

1.2.2.12 Power Monitor

The *power monitor* is responsible for generating system reset and monitoring the system's AC power source. The power monitor asserts the SYSRESET* and (optional) ACFAIL* signals.

1.2.3 Sub-Buses

The VMEbus specification groups all of the bus signals into four sub-buses. Figure 1-4 shows the relationship of these buses to the various functional modules. [Note: some people also consider the VME serial bus to be a fifth sub-bus].

1.2.3.1 Data Transfer Bus

The *data transfer bus* is used by masters to move data to and from slaves. It is also used by interrupt handlers to fetch STATUS/ID bytes from interrupters during interrupt acknowledge cycles. This sub-bus is composed of address lines, data lines and control signals. The following signals are part of the Data Transfer Bus:

Address	**Data**	**Control**
A01-A31	D00-D31	AS*
AM0-AM5		DS0*
DS0*		DS1*
DS1*		BERR*
LWORD*		DTACK*
		RETRY* (†)
		RESP* (††)
		WRITE*

1.2.3.2 Data Transfer Arbitration Bus

The *data transfer arbitration bus* is used by masters and interrupt handlers to obtain ownership of the data transfer bus. A functional module called the arbiter is used in conjunction with the data transfer arbitration bus to determine which master or interrupt handler is granted the bus. Signals in the data transfer arbitration bus include:

BR0*	BG0IN*	BG0OUT*	BBSY*
BR1*	BG1IN*	BG1OUT*	BCLR*
BR2*	BG2IN*	BG2OUT*	
BR3*	BG3IN*	BG3OUT*	

1.2.3.3 Priority Interrupt Bus

System interrupts are performed over the *priority interrupt bus*. Up to seven levels of interrupts can be used. This bus uses the following signals:

IRQ1*	IRQ5*	IACK*
IRQ2*	IRQ6*	IACKIN*
IRQ3*	IRQ7*	IACKOUT*
IRQ4*		

1.2.3.4 Utility Bus

The *utility bus* is a collection of signals used for system reset, periodic timing, system diagnostics and power failures. Signals of the utility bus include:

SYSCLK*	MPR (††)	GAP (††)
SYSRESET*	MCLK (††)	GA0*-GA4* (††)
SYSFAIL*	MSD (††)	LI/I* (††)
ACFAIL*	MMD (††)	LI/O* (††)
	MCTL (††)	

1.2.4 Data Transfer Cycles

VMEbus offers eight types of data transfer bus cycles. This variety of cycles allow the bus to adapt to the changing requirements of the system.

1.2.4.1 Read/Write Cycle (R/W Cycle

The Read/write cycle is used to transfer 8, 16, 24 or 32-bits of data between bus modules. 64-bits of data cannot be transferred with this cycle. The cycle begins when a master broadcasts an address and an address modifier code. When selected, slaves capture the address and respond to the cycle by asserting DTACK*, BERR* or RETRY*. Read/write cycles support 16, 24 or 32-bit addressing modes.

Read/write cycles do not support the 64-bit MBLT or 2eVME cycles.

1.2.4.2 Read-Modify-Write Cycle (RMW cycle

The read-modify-write cycle permits indivisible bus cycles. This cycle is especially useful for arbitrating shared resources in multiprocessor or multiuser systems.

1.2.4.3 Block Transfer Cycle (BLT cycle

The block transfer cycle moves a block of data between masters and slaves. Up to 256 bytes of data may be transferred with the cycle. It is faster than READ/WRITE cycles because slaves are addressed only once (at the beginning of the cycle). After that, counters on both the master and the slave keep track of the local address. This cycle supports transfer rates up to 40 Mbyte/second.

The block transfer cycle quickly moves large blocks of data. These cycles are most often used to move block oriented data, such as between a CPU and a disk controller or network port.

1.2.4.4 Multiplexed Block Transfer Cycle (MBLT cycle

The Multiplexed Block Transfer (MBLT) cycle is similar to the BLT cycle, except the address and data lines are multiplexed to form 64-bit paths. This cycle supports transfer rates up to 80 Mbyte/second. MBLT cycles support A24, A32 and A64 addressing modes.

The MBLT cycle was first introduced in the ANSI/VITA 1-1994 (VME64) specification.

1.2.4.5 Multiplexed Data Thirty-two Cycle (MD32 cycle

Functionally, the Multiplexed Data Thirty-two (MD32) cycle is similar to the MBLT cycle. It permits 32-bit data and 40-bit address transfers over the P1/J1 connector. It was included in the bus specification to support 3U bus modules (which only have a single backplane connector).

The MD32 cycle was first introduced in the ANSI/VITA 1-1994 (VME64) specification.

1.2.4.6 Two Edge VMEbus Cycle (2eVME Cycle

The Two Edge VMEbus Cycle (2eVME) allows data to be transferred at twice the rate of the MBLT Cycle. It is the fastest bus cycle on VMEbus, and allows data to be moved at 160 MByte/sec. It is called a two edge cycle because each data 'beat' only requires two edges: one data strobe edge and one DTACK* edge. All other VMEbus cycles require four edges: two data strobe edges and two DTACK* edges.

This 2eVME cycle was first introduced in the VITA 1.1-1997 (VME64x) specification.

1.2.4.7 Address-Only Cycle (ADO Cycle)

During the address-only cycle a master generates a valid address, but slaves do not respond. Also, the master does not assert either data strobe. It allows slaves to decode an address at the same time that a master's on-board memory does, and speeds up the system. Sometimes this cycle is called a 'do-nothing' cycle.

Some IC chips spontaneously generate address-only cycles. For example, a memory management unit (MMU) circuit may generate a new system address before it is done with a previous cycle. If the new cycle is aborted, then the MMU terminates it and generates a spontaneous address-only cycle.

1.2.4.8 Address-Only With Handshake Cycle (ADOH Cycle

The Address-only with Handshake (ADOH) cycle is intended for use with the LOCK commands. An address is broadcast, but no data is transferred. The ADOH cycle has the same protocol as the address phase of the MBLT cycle.

This cycle was first introduced in the ANSI/VITA 1-1994 (VME64) specification.

1.2.5 Interrupt Acknowledge Cycle (IACK* Cycle

The *interrupt acknowledge cycle* is initiated by interrupt handlers in response to interrupt requests. This cycle performs two functions: it passes STATUS/ID bytes and arbitrates the interrupt sources.

1.2.6 Arbitration Cycle (ARB Cycle

The arbitration cycle takes place during bus arbitration. It begins when a requester generates a bus request to the central arbiter. The arbiter grants the bus to the requester when the bus is vacant, and the requester acquires the bus by asserting the bus busy signal (BBSY*). The requester releases the bus by negating BBSY*.

1.2.7 Signal Summary

Table 1-3 shows pin assignments for the P1/J1 and P2/J2 connectors. That table reflects the pinout described in the VME64x specification. Table 1-4 shows the pin assignments for the P0/J0 connector.

Under the IEEE-1014-1987 specification the three row DIN 41612 connector is used for P1/J1 and P2/J2. The P1/J1 connector carries 24 address lines, 16 data lines, all control signals and some of the power and ground traces. The P2/J2 connector carries the extra eight address and 16 data lines, with additional power and ground pins. All of the defined P2/J2 signals are located on the center row of pins. The 'a' and 'c' rows of that connector are user defined, and can be used for any purpose. In general they are used for I/O signals or for a side bus such as VSBbus.

Also note that both the 3U and 6U VMEbus modules have a P1/J1 connector. That connector has all of the control signals and a subset of the address and data lines. 6U modules, which have a P2/J2 connector, can utilize the additional address and data lines on the second connector.

The VME64 specification redefines the former RESERVED pin (J2/B3) as a RETRY* pin. That specification also renamed the SERCLK and SERDAT* pins (J1/B21 and J1/B22) as SERA and SERB. Although the 'z' and 'd' rows of the 160 pin connector are included in the VME64 specification, they were only defined as ground or reserved pins.

The VME64x specification redefines the 'z' and 'd' rows of pins that were RESERVED under the VME64 specification. The new pins add additional enhancements to the connectors such as geographical addressing, +3.3 VDC power and more user defined pins.

VMEbus uses an asterisk (*) following a signal name to show that its active (valid) state is logic zero. The absence of an asterisk means that its active state is logic one. When a signal is edge sensitive the asterisk shows that it is valid during the high to low transition of the signal.

Each VMEbus signal falls into one of five classes of electrical specification. These classes are called standard three-state, high current three-state, standard totem-pole, high current totem-pole and open collector signals. Chapter 6 describes the characteristics of these classes in detail.

A sixth class of electrical specification was defined in the VME64x standard. This is called ETL or Enhanced Transisitor Logic, and is used by the 2eVME cycle.

1.2.7.1 A01-A31

The address bus [A01-A31] is driven by masters and interrupt handlers. During data transfer cycles they are used to broadcast short I/O (16-bit), standard (24-bit) and extended (32-bit) addresses. The width of the address bus is selected by masters with the address modifier code AM0-AM5.

Note that there is no address line A00. The zeroith address bit is encoded into data strobes DS0* and DS1*.

The address lines can also be used to transfer a portion of the data during MBLT, MD32 and 2eVME cycles. During these cycles, the zeroith address bit is carried by the LWORD* signal.

Interrupt handlers use A01-A03 to broadcast the level of interrupt being acknowledged during interrupt acknowledge cycles.

A01-A31 are standard three-state class signals.

1.2.7.2 ACFAIL

The AC power fail signal [ACFAIL*] is driven by the power monitor. When asserted, it signals to all modules that the system power supply is about to quit. The use of ACFAIL* is optional. ACFAIL* is an open-collector class signal.

1.2.7.3 AM0-AM5

The address modifier code [AM0-AM5] is driven by masters. It accompanies an address and indicates both the size and type of address transfer. It is used by slaves to determine which address lines should be decoded. AM0-AM5 are standard three-state class signals.

1.2.7.4 AS*

Address strobe [AS*] is driven by masters and interrupt handlers. When asserted, it indicates that a valid address and address modifier code have been placed onto the bus. The signal also qualifies other signals such as IACK*. AS* is a high current three-state signal.

1.2.7.5 BBSY*

Bus busy [BBSY*] is driven by masters and interrupt handlers when they are using the data transfer bus. The arbiter monitors this signal to determine when it should grant the bus to another master or interrupt handler. BBSY* is an open-collector class signal.

1.2.7.6 BCLR*

Bus clear [BCLR*] is driven by the bus arbiter. When asserted it informs the current master or interrupt handler that another module is requesting the bus. The conditions under which it is asserted depend upon the arbitration method being used (see Chapter 3). BCLR* is a high current totem-pole class signal.

1.2.7.7 BERR*

Bus error [BERR*] is driven by slave or bus timer modules. A slave asserts BERR* if an error has occurred during the bus cycle. The bus timer asserts it when the bus has locked-up (such as during faults, memory sizing or card counting). BERR* is an open-collector class signal.

1.2.7.8 BG0IN* - BG3IN* / BG0OUT* - BG3OUT*

The bus grant signals [BG0IN* - BG3IN* and BG0OUT* - BG3OUT*] are part of the bus grant daisy-chain and are driven by arbiters and requesters. The slot 01 arbiter asserts a bus grant in response to a bus request on the same level [BR0* - BR3*]. The bus grant daisy-chain starts at the slot 01 system controller and propagates from module to module until it reaches the module that initially requested the bus. Each VMEbus module has a bus grant input and a bus grant output. They are standard totem-pole class signals.

1.2.7.9 BR0* - BR3*

Bus requests [BR0* - BR3*] are asserted by a requester whenever its master or interrupt handler needs the bus.

Before accepting the bus, the master waits until the arbiter grants the bus by way of the bus grant daisy-chain [BG0IN* - BG3IN*]. They are open-collector class signals.

1.2.7.10 D00-D31

Data bus [D00-D31] is driven by masters, slaves or interrupters. These are bi-directional signals and are used for data transfers. Different portions of the data bus are used depending upon the state of DS0*, DS1*, A01 and LWORD* pins. They are standard three-state signals.

The data lines can also be used to transfer a portion of the address during MD32, MBLT and 2eVME cycles.

Table 1-3 VMEbus P1/J1 and P2/J2 Pin Assignments

P1/J1 Pin Assignments					
Pin	Row z (††)	Row a	Row b	Row c	Row d (††)
1	MPR	D00	BBSY*	D08	VPC (§)
2	GND	D01	BCLR*	D09	GND (§)
3	MCLK	D02	ACFAIL*	D10	+V1
4	GND	D03	BG0IN*	D11	+V2
5	MSD	D04	BG0OUT*	D12	RsvBus
6	GND	D05	BG1IN*	D13	-V1
7	MMD	D06	BG1OUT*	D14	-V2
8	GND	D07	BG2IN*	D15	RsvBus
9	MCTL	GND	BG2OUT*	GND	GAP*
10	GND	SYSCLK	BG3IN*	SYSFAIL*	GA0*
11	RESP*	GND	BG3OUT*	BERR*	GA1*
12	GND	DS1*	BR0*	SYSRESET*	+3.3V
13	RsvBus	DS0*	BR1*	LWORD*	GA2*
14	GND	WRITE*	BR2*	AM5	+3.3V
15	RsvBus	GND	BR3*	A23	GA3*
16	GND	DTACK*	AM0	A22	+3.3V
17	RsvBus	GND	AM1	A21	GA4*
18	GND	AS*	AM2	A19	+3.3V
19	RsvBus	GND	AM3	A20	RsvBus
20	GND	IACK*	GND	A18	+3.3V
21	RsvBus	IACKIN*	SERA (†)	A17	RsvBus
22	GND	IACKOUT*	SERB (†)	A16	+3.3V
23	RsvBus	AM4	GND	A15	RsvBus
24	GND	A07	IRQ7*	A14	+3.3V
25	RsvBus	A06	IRQ6*	A13	RsvBus
26	GND	A05	IRQ5*	A12	+3.3V
27	RsvBus	A04	IRQ4*	A11	LI/I*
28	GND	A03	IRQ3*	A10	+3.3V
29	RsvBus	A02	IRQ2*	A09	LI/O*
30	GND	A01	IRQ1*	A08	+3.3V
31	RsvBus	-12 VDC	+5VSTDBY	+12 VDC	GND (§)
32	GND	+5 VDC	+ 5VDC	+5 VDC	VPC (§)

P2/J2 Pin Assignments					
Pin	Row z (††)	Row a	Row b	Row c	Row d (††)
1	UsrDef	UsrDef	+5 VDc	UsrDef	UsrDef (§)
2	GND	UsrDef	GND	UsrDef	UsrDef (§)
3	UsrDef	UsrDef	RETRY* (†)	UsrDef	UsrDef
4	GND	UsrDef	A24	UsrDef	UsrDef
5	UsrDef	UsrDef	A25	UsrDef	UsrDef
6	GND	UsrDef	A26	UsrDef	UsrDef
7	UsrDef	UsrDef	A27	UsrDef	UsrDef
8	GND	UsrDef	A28	UsrDef	UsrDef
9	UsrDef	UsrDef	A29	UsrDef	UsrDef
10	GND	UsrDef	A30	UsrDef	UsrDef
11	UsrDef	UsrDef	A31	UsrDef	UsrDef
12	GND	UsrDef	GND	UsrDef	UsrDef
13	UsrDef	UsrDef	+5 VDC	UsrDef	UsrDef
14	GND	UsrDef	D16	UsrDef	UsrDef
15	UsrDef	UsrDef	D17	UsrDef	UsrDef

16	GND	UsrDef	D18	UsrDef	UsrDef
17	UsrDef	UsrDef	D19	UsrDef	UsrDef
18	GND	UsrDef	D20	UsrDef	UsrDef
19	UsrDef	UsrDef	D21	UsrDef	UsrDef
20	GND	UsrDef	D22	UsrDef	UsrDef
21	UsrDef	UsrDef	D23	UsrDef	UsrDef
22	GND	UsrDef	GND	UsrDef	UsrDef
23	UsrDef	UsrDef	D24	UsrDef	UsrDef
24	GND	UsrDef	D25	UsrDef	UsrDef
25	UsrDef	UsrDef	D26	UsrDef	UsrDef
26	GND	UsrDef	D27	UsrDef	UsrDef
27	UsrDef	UsrDef	D28	UsrDef	UsrDef
28	GND	UsrDef	D29	UsrDef	UsrDef
29	UsrDef	UsrDef	D30	UsrDef	UsrDef
30	GND	UsrDef	D31	UsrDef	UsrDef
31	UsrDef	UsrDef	GND	UsrDef	GND (§)
32	GND	UsrDef	+ 5VDC	UsrDef	VPC (§)
Notes:	(†) - Pin(s) redefined under the VME64 specification. (††) - Pin(s) redefined under the VME64 specification. (§) - Elongated (mate first, break last) connector contact.				

Table 1-4 P0/J0 Pin Assignments

P0/J0/RJ0/RP0 Connector Pinout							
Pos.	Row f	Row c	Row d	Row c	Row b	Row a	Row z
1	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
2	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
3	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
4	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
5	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
6	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
7	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
8	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
9	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
10	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
11	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
12	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
13	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
14	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
15	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
16	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
17	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
18	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND
19	GND	UsrDef	UsrDef	UsrDef	UsrDef	UsrDef	GND

1.2.7.11 DS0*-DS1*

Data strobes DS0* and DS1* are driven by masters and interrupt handlers. These signals serve not only to qualify data, but also to indicate the size and position of the data transfer.

When combined with LWORD* and A01, the data strobes indicate the size and type of data transfer. DS0* - DS1* are high current three-state class signals.

1.2.7.12 DTACK*

Data transfer acknowledge [DTACK*] is driven by slaves or interrupters. During write cycles DTACK* is asserted by a slave after it has latched data. During read and interrupt acknowledge cycles, DTACK* is asserted by a slave after data is placed onto the bus. DTACK* can be an open-collector or a high current three-state class signal.

1.2.7.13 GA0* - GA4*

The geographical address [GA0*-GA4*] is a binary code that indicates the slot number of the backplane. They are open collector signals, and were added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.14 GAP* (††)

The geographical address parity [GAP*] is tied high or floating, depending upon the parity of the geographical address lines [GA0*-GA4*]. It is an open collector signal, and was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.15 GND

Ground [GND] is used both as a signal reference and a power return path.

1.2.7.16 IACK*

Interrupt acknowledge [IACK*] is driven by interrupt handlers in response to interrupt requests. It is connected to IACKIN* at slot 01 (on the backplane), and used by the IACK* daisy-chain driver to start propagation of the [IACKIN* - IACKOUT*] daisy-chain. IACK* can be either an open-collector or a standard three-state class signal.

1.2.7.17 IACKIN*-IACKOUT*

The interrupt acknowledge daisy-chain [IACKIN* - IACKOUT*] is driven by the IACK* daisy-chain driver. These signals are used both to indicate that an interrupt acknowledge cycle is in progress, and to determine which interrupters should return a STATUS/ID. They are standard totem-pole class signals.

1.2.7.18 IRQ1*-IRQ7*

Priority interrupt requests [IRQ1*-IRQ7*] are asserted by interrupters. Level seven is the highest priority, and level one the lowest. They are open-collector class signals.

1.2.7.19 LI/I* (††)

The live insertion input [LI/I*] signal is used to carry hot swap (live insertion) control information. It is a three state driven signal and was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.20 LI/O* (††)

The live insertion output [LI/O*] signal is used to carry hot swap (live insertion) control information. It is a three state driven signal and was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.21 LWORD*

Long word [LWORD*] is driven by masters. It is used in conjunction with A01, DS0* and DS1* to indicate the size of the current data transfer. LWORD* is a standard three-state class signal.

During 64-bit *address* transfers, LWORD* doubles as address bit A00. During 64-bit *data* transfers, LWORD* doubles as a data bit.

1.2.7.22 MCLK (††)

The module clock [MCLK] signal is part of the IEEE 1149.5 MTM bus. It is a three-state driven signal which was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.23 MCTL (††)

The module control [MCTL] signal is part of the IEEE 1149.5 MTM bus. It is a three-state driven signal which was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.24 MMD (††)

The module data [MMD] signal is part of the IEEE 1149.5 MTM bus. It is a three-state driven signal which was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.25 MPR (††)

The bus pause request [MPR] signal is part of the IEEE 1149.5 MTM bus. It is a three-state driven signal which was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.26 MSD (††)

The slave data [MSD] signal is part of the IEEE 1149.5 MTM bus. It is a three-state driven signal which was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.27 RESERVED (†)

The RESERVED signal pin is obsolete and is no longer used. Under the IEEE 1014-1987 version of the bus specification there was a single reserved pin. This pin was redefined under VME64 as a RETRY* pin.

The VME64x specification uses the names RsvB and RsvU for reserved pins.

1.2.7.28 RESP* (††)

The response [RESP*] signal is used to carry the information as defined by the 2eVME protocol. It was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.29 RsvB (††)

The reserved bussed [RsvB] signal should not be used. VME64x backplanes must bus and terminate this signal. It was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.30 RsvU (††)

The reserved unbussed [RsvU] signal should not be used. VME64x backplanes must not bus or terminate this signal. It was added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.31 RETRY* (†)

[RETRY*], together with [BERR*], can be asserted by a slave to postpone a data transfer. The master must then attempt the cycle again at a later time. The retry cycle prevents deadlock (deadly embrace) conditions in bus-to-bus links and secondary buses. RETRY* is a standard three-state signal.

The [RETRY*] signal was added in the ANSI/VITA 1-1994 (VME64) version of the bus specification. This pin was RESERVED in earlier versions. However, boards that support [RETRY*] should work just fine with older backplanes, as they were required to bus and terminate this signal line.

1.2.7.32 SERA & SERB (†)

The [SERA] and [SERB] signals are used for an (optional) serial bus such as the AUTOBAHN (IEEE 1394) or VMSbus.

Under the ANSI/VITA 1-1994 (VME64) bus specification, these pins can be used for any user defined serial bus. Earlier versions of the VMEbus specification defined these pins as [SERCLK] and [SERDAT*], which were originally intended for a serial bus called VMSbus. However, they were rarely used for that purpose.

1.2.7.33 SERCLK & SERDAT* (†)

The [SERCLK] and [SERDAT*] signals were made obsolete under the ANSI/VITA 1-1994 (VME64) bus specification. Refer to [SERA] and [SERB] for more details.

1.2.7.34 SYSCLK

16 MHz utility clock [SYSCLK] is driven by the slot 01 system controller. This clock can be used for any purpose, and has no timing relationship to other VMEbus signals. SYSCLK* is a high current totem-pole class signal.

1.2.7.35 SYSFAIL*

System fail [SYSFAIL*] can be asserted or monitored by any module. It indicates that a failure has occurred in the system. Implementation of [SYSFAIL*] is user defined, and its use is optional. SYSFAIL* is an open-collector class signal.

1.2.7.36 SYSRESET*

System reset [SYSRESET*] can be driven by any module and indicates that a reset (such as power-up) is in progress. SYSRESET* is an open-collector class signal.

1.2.7.37 UD or UsrDef (††)

Pins that are user defined [specified as UD *or* UsrDef] can be specified by the user. Generally, they are routed directly through the backplane so that they can be connected to cables or to rear I/O transition modules.

1.2.7.38 VPC (††)

Voltage pre-charge [VPC] pins forma a 'make first / break last' contact. They are intended to be used as pre-charge power sources for live insertion logic. These pins were added to the 160 pin P1/J1 and P2/J2 connectors in the VME64x specification. The VPC pins are connected to the +5 VDC power supply on VME64x backplanes. These pins may also be used as additional +5 VDC power pins in boards that do not support live insertion.

1.2.7.39 V1/V2 Auxiliary Power (††)

The [+/- V1/V2] power pins supply 38 - 75 VDC to the bus module. They are also known as the auxiliary power pins, and were originally intended to be used as 48 VDC battery supplies in Telecom systems. However, they can be used for any purpose. These pins were added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.40 WRITE*

The read / write signal [WRITE*] is driven by masters. It indicates the direction of data transfer over the bus. It is asserted during a write cycle and negated during a read cycle. WRITE* is a standard three-state class signal.

1.2.7.41 +5V STDBY

[+5V STDBY] is an optional +5 VDC standby power supply. This power pin is often connected to a rechargable battery. This eliminates the need for individual batteries on VMEbus modules. Individual batteries are often used for real time clock and static RAM chips.

1.2.7.42 +3.3 V (††)

Main +3.3 VDC power source. These pins were added to the 160 pin P1/J1 connector in the VME64x specification.

1.2.7.43 +5 VDC, +12 VDC, -12 VDC

The main system power supplies are [+5 VDC], [+12 VDC] and [-12 VDC].

1.3 Bus Options and Mnemonics

VMEbus offers many interface options. Mnemonic symbols have been defined to describe these options. When integrating a VMEbus system, use these mnemonics to insure that all of the boards work together.

1.3.1 Options

When selecting VMEbus modules make sure that the options on each module are compatible with all others in the system. For example, D08(EO), D16, D32 and RMW all specify possible types of data transfer cycles that a master can generate. In this case the bus interface options would be described as:

D08(EO):D16:D32:RMW

Other boards in the system may or may not be compatible with this board. 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 would reject it. This is not to say that the slave couldn't be used in the system, but rather the user must make sure that the master does not generate a D32 cycle to that particular slave.

1.3.2 Mnemonics

1.3.2.1 A16

Bus masters with A16 capability can generate bus cycles using 16-bit addresses. Slaves and location monitors with A16 capability can accept these cycles. The 16-bit addresses are sometimes called *short I/O addresses* because they are often used as an I/O space.

1.3.2.2 A24

Bus masters with A24 capability can generate bus cycles with 24-bit addresses. Slaves and location monitors with A24 capability can accept these cycles. 24-bit addresses are sometimes called *standard addresses*.

Beginning with the IEEE 1014-1987 version of the bus specification, A24 masters are also required to generate A16 bus cycles. This requirement was optional under older versions. The rule does not apply to slaves or location monitors.

1.3.2.3 A32

Bus masters with A32 capability can generate bus cycles with 32-bit addresses. Slaves and location monitors with A32 capability can accept these cycles. 32-bit addresses are sometimes called *extended addresses*.

Beginning with the IEEE-1014-1987 version of the bus specification, A32 masters are also required to generate A16 and A24 bus cycles (but not A40). Older versions of the bus specification did not require this. This rule does not apply to slaves or location monitors.

1.3.2.4 A40

Bus masters with A40 capability can generate bus cycles with 40-bit addresses. Slaves and location monitors with A40 capability can accept these cycles.

A40 capability was first introduced in the ANSI/VITA 1-1994 (VME64) version of the bus specification. In general, this addressing mode was introduced to support larger addressing spaces for 3U modules. The bus specification is somewhat unclear as to which bus cycles actually support the A40 addressing mode. The author interprets the specification to mean:

AM code 0x37: A40BLT ---> MD32 Cycle

AM code 0x35: A40 Lock Command ---> ADOH Cycle

AM code 0x34: A40 access ---> A40 8/16 bit data transfer

1.3.2.5 A64

Bus masters with A64 capability can generate bus cycles with 64-bit addresses. Slaves and location monitors with A64 capability can accept these cycles. 64-bit addresses are sometimes called *long addresses*.

64-bit address transfers were first permitted under the ANSI/VITA 1-1994 (VME64) version of the bus specification. Thirty-one address lines, thirty-two data lines and LWORD* are used to pass the address.

A64 masters are also required to generate A32 and A24 bus cycles, but are not required to generate A40 cycles.

1.3.2.6 ADO

Bus modules with ADO [ADdress Only] capability can generate or accept address-only cycles. A master can begin the address phase of a bus cycle, but it can terminate it before the data phase begins. This means that the address portion of the cycle 'leaks' onto the bus, resulting in an ADO cycle.

ADO capability was first required under the IEEE 1014-1987 version of the VMEbus specification. This bus cycle is optional for masters, but all slaves must tolerate them.

1.3.2.7 ADOH (†)

Bus modules with ADOH [ADdress Only with Handshake] capability can generate or accept ADOH bus cycles. ADOH cycles are used to generate LOCK commands, and are functionally the same as the address phase of an MBLT cycle.

ADOH capability was first allowed under the ANSI/VITA 1-1994 (VME64) version of the specification.

1.3.2.8 AUX (††)

Bus modules or backplanes with AUX [AUXiliary power] capability use or provide the VME64x auxiliary power pins.

1.3.2.9 BLT

The BLT [BLock Transfer] option specifies that a block transfer cycle can be generated by a master, or accepted by a slave or location monitor. The BLT cycle can be faster than normal read/write cycles.

1.3.2.10 BTO(x)

The BTO(x) [Bus Time Out] option indicates a bus timer is present on the module. The bus timer monitors data strobes DS0* and DS1*, and asserts BERR* if they are low for more than 'x' microseconds. The bus timer is sometimes called a watchdog timer. It is used to prevent system lock-ups during memory sizing, card counting or bus failures.

1.3.2.11 D08(O)

Slaves with D08(O) capability can accept 8-bit data transfers at odd addresses. Simple D08(O) *only* slaves are often found in the market. These boards usually have simple 8-bit I/O circuits, such as on low-speed serial boards.

1.3.2.12 D08(EO)

Masters, slaves or location monitors with D08(EO) capability can generate or accept 8-bit bus cycles at even or odd addresses.

1.3.2.13 D16

Masters, slaves or location monitors with D16 capability can generate or accept 16-bit transfers.

1.3.2.14 D32

Masters, slaves or location monitors with D32 capability can generate or accept 32-bit transfers.

The IEEE-1014-1987 version of the bus specification first required that D32 masters, slaves and location monitors include the D16 and D08(EO) capabilities. This was optional under earlier versions of the bus specification.

The IEEE-1014-1987 version of the bus specification also required that D32 slaves and location monitors must accept unaligned (UAT) data transfers. D32 masters are not required to generate these cycles, however. The UAT function was optional for slaves and location monitors under earlier versions of the bus specification.

1.3.2.15 ESD (††)

Bus modules or subracks with ESD [ElectroStatic Discharge] capability support the VME64x ESD scheme.

1.3.2.16 EMC (††)

Bus modules or subracks with EMC [ElectroMagnetic Conformance] capability support the VME64x EMC scheme.

1.3.2.17 ETL (††)

Bus modules with ETL [Enhanced Transceiver Logic] capability us ETL devices for bus transceivers.

1.3.2.18 FAIR

Bus modules with FAIR capability support the FAIR bus request mode.

1.3.2.19 GA (††)

Bus modules or backplanes with GA [Geographical Addressing] capability support the VME64x geographical addressing pins.

1.3.2.20 GAP (††)

Bus modules or backplanes with GAP [Geographical Address Parity] capability support the VME64x geographical address parity pin.

1.3.2.21 I(x)

The I(x) option applies to interrupters that generate interrupt requests on IRQx*, where 'x': $1 \leq x \leq 7$.

1.3.2.22 IE/HDL (††)

Bus modules with IE/HDL [Injector Extractor / HanDLe] capability use front panels with the VME64x injector / extractor handles.

1.3.2.23 IEL/HDL (††)

Bus modules with IEL/HDL [Injector Extractor Locking / HanDLe] capability use front panels with the VME64x injector / extractor locking handles.

1.3.2.24 IE/COMB (††)

Sub racks with IE/COMB [Injector Extractor / COMB] capability include the front rail comb for support of the VME64x injector / extractor handles.

1.3.2.25 IH(x)

Modules with the IH(x) option are interrupt handlers that monitor interrupt requests on level IRQx* only, where 'x': $1 \leq x \leq 7$. The interrupt handler generates an interrupt acknowledge cycle in response to these requests.

1.3.2.26 IH(x-y)

Modules with the IH(x-y) [Interrupt Handler] option are interrupt handlers that monitor interrupt requests on multiple levels. For example an IH(3-5) will handle interrupts on IRQ3*, IRQ4* and IRQ5*. The interrupt handler generates an interrupt acknowledge cycle in response to these requests.

1.3.2.27 KEY (††)

Bus modules or sub racks with KEY capability include provisions for the VME64x keying mechanism.

1.3.2.28 LCK

The LCK [LoCK] mnemonic indicates that a module can generate or accept LOCK commands. The LOCK command can be used to access a peripheral at it's maximum bandwidth. Oftentimes, a peripheral has multiported (or shared) resources, where one port is on VMEbus. This command 'locks out' these other ports, and grants VMEbus full bandwidth on the module. The LOCK command itself is embedded in the ADOH cycle.

Generally, this mnemonic is used in conjunction with an address size. For example, an A24:LCK slave will accept 24-bit LOCK cycles.

LOCK commands were first permitted under the ANSI/VITA 1-1994 (VME64) version of the bus specification.

1.3.2.29 MBLT

Bus masters with MBLT capability can generate Multiplexed BLock Transfer cycles. These cycles always transfer 64-bits of data in a block fashion, using A24, A32 or A64 addressing modes. MBLT cycles can only be used with 6U bus modules.

MBLT cycles were first permitted under the ANSI/VITA 1-1994 (VME64) version of the bus specification.

1.3.2.30 MD32

Bus masters with MD32 capability can generate Multiplexed D32 cycles. These cycles always transfer 32-bits of data in a block fashion, using the A40 addressing mode. MD32 cycles were originally intended to support a wider data path on 3U bus modules. However, the cycle can also be used with 6U bus modules.

MD32 cycles are similar in operation to MBLT cycles. Both cycles combine the address and data lines to form wider address and data paths.

MD32 cycles were first permitted under the ANSI/VITA 1-1994 (VME64) version of the bus specification.

1.3.2.31 PRI

The PRI [PRIority] option applies to bus arbiters. PRI arbiters employ a priority scheduling algorithm where bus requesters on level BR3* have the highest priority, and those on BR0* the lowest.

1.3.2.32 RETRY (†)

Bus modules with RETRY capability generate or accept the VME64 RETRY* signal. The RETRY* signal is asserted by a slave when it wants the master to retry a bus cycle.

1.3.2.33 RESP (††)

Bus modules with RESP capability generate or accept the VME64x RESP* signal. The RESP* signal is used during 2eVME bus cycles.

1.3.2.34 RMW

Masters with the RMW [Read Modify Write] option can generate read-modify-write cycles. Slaves and location monitors can accept them. This cycle is primarily used in multiprocessing or multiuser systems to allow (software) arbitration of shared system resources.

1.3.2.35 ROAK

The ROAK [Release-On-AcKnowledge] option describes an interrupter that removes its interrupt request during an interrupt acknowledge cycle. This is opposed to the RORA interrupter which releases its interrupt only after an on-board register is accessed by the interrupter (in response to an interrupt).

1.3.2.36 ROR

Bus requesters with the ROR [Release-On-Request] option will give up the data transfer bus when another VMEbus module requests it.

1.3.2.37 RORA

The RORA [Release-On-Register-Access] option describes an interrupter that removes its interrupt request when a master accesses an on-board status or control register (during an interrupt service routine). This is opposed to the ROAK interrupter, which releases its interrupt automatically during the interrupt acknowledge cycle.

1.3.2.38 RRS

The RRS [Round Robin Select] option applies to bus arbiters. These arbiters employ a round robin scheduling algorithm where the bus is granted on a rotating basis. With this scheme, bus masters and interrupt handlers are granted the bus in an equal priority fashion.

1.3.2.39 RWD

The RWD [Release-When-Done] option describes a requester that releases the data transfer bus when it is done with it. This is opposed to the ROR requester which releases the bus whenever another master or interrupt handler requests it.

1.3.2.40 SGL

The SGL [SinGLe-level] option applies to bus arbiters. SGL arbiters grant the bus on level 3 only. This is opposed to PRI and RRS arbiters which can use up to four bus request levels.

1.3.2.41 UAT

The UAT [UnAligned Transfer] option specifies that an unaligned transfer can be generated by a master, accepted by a slave or monitored by a location monitor. Unaligned transfers can speed up a VMEbus system by allowing 32-bits of data to be transferred at unaligned address boundaries in two bus cycles instead of three.

1.3.2.42 2eVME

Modules with 2eVME capability can participate in two edged VME cycles. These cycles always transfer 64-bits of data in a block fashion. On 6U bus modules, the 2eVME cycle can support A32 or A64 addressing. On 3U modules the 2eVME cycle can support A32 or A40 addressing.

2eVME cycles were first permitted under the VITA 1.1-1997 (VME64x) version of the bus specification. The two edged protocol was introduced to increase the bandwidth of the bus.

1.4 The VMEbus Specification

The VMEbus specification is an indispensable reference for system integrators and board designers alike. However, it was written in a formal style that may cause problems for some newcomers. Understanding the organization of the specification, along with the differences between versions, will help the reader to understand it better.

Throughout this text only three versions of the VMEbus specification are recognized. These are the IEEE 1014-1987, the ANSI/VITA 1-1994 (VME64) and the VITA 1.1-1997 (VME64x). For example, when describing the 'ADO' mnemonic, the author makes the statement:

> "ADO capability was first required under the IEEE 1014-1987 version of the VMEbus specification. This bus cycle is optional for masters, but all slaves must tolerate them."

Actually, the ADO capability was first introduced in the revision C VMEbus specification. However, the revision C specification wasn't around very long, so the change is attributed to the IEEE 1014-1987 revision.

There were (and probably will be) numerous *draft* versions of the VMEbus specification. These are not listed here, as very few boards actually claim conformance to them.

1.4.1 History

The VMEbus specification has been updated several times since its introduction in 1981. These changes have been made under the auspices of the VITA, IEC-821 and IEEE-1014 technical committees. Every version of the specification has clarified or added features to the standard. Every new version is 100% upward compatible with the older versions.

1.4.1.1 Revision A

The first version of the specification was called Revision A. It was taken from an earlier spec for VERSAbus, which was developed by Motorola Microsystems (Phoenix, Arizona USA) during the late 1970's. In 1980 a group of engineers from Motorola Microsystems (Munich, Germany) proposed the marriage of the VERSAbus specification and the popular Eurocard packaging format into what they called VERSAbus-E.

A Manufacturers group was formed and composed of members from Mostek, Motorola and Signetics corporations, who later named the new bus VMEbus. In 1981 the group released a public domain specification which they called Revision A. This first version was used for public comment, and was not intended for design purposes.

1.4.1.2 Revision B

In 1982 Revision B of the specification was released by the manufacturers group. This second version refined the specifications for the signal line drivers, and brought the mechanical specifications into line with the IEC 297-3 Eurocard standard.

During 1982 the French delegation to the IEC (International Electrotechnical Commission) proposed VMEbus as a standard. The IEC formed a committee to start formal standardization of the bus calling it the IEC-821 bus.

In 1983 the IEEE also formed a committee to standardize VMEbus. The IEEE version was called the IEEE-P1014.

1.4.1.3 Revision C

Both the IEC and the IEEE distributed copies of the Revision B specification for comment, and both received requests for changes. These changes were incorporated into the Revision C specification which was released in February 1985. The specification was also known as the IEEE P1014/D1.0.

Revision C clarified the Revision B specification by removing ambiguities, making technical descriptions better and improving compliance. The language of the spec was clarified by adding RULES, RECOMMENDATIONS, PERMISSIONS and so on. It also added some new features to the bus including:

- The Location Monitor functional module.
- IACK* daisy-chain driver functional module.
- Bus cycles with unaligned data transfers.
- Address-only cycles.
- Limited the size of block transfers to 256 bytes.
- Extended read-modify-write cycles to 2 and 4 bytes.
- Allowed removal of bus requests during arbitration.
- Extended the STATUS/ID transfers to 4 bytes.
- Better defined the interrupt release mechanism.

1.4.1.4 Revision C.1

In 1985 the IEEE-P1014 committee met and approved additional changes to the Revision C specification. These changes further clarified the bus specification and removed some incompatibilities. Although there are subtle differences between Revision C and Revision C.1, no features were added or removed from the specification. The C.1 version is the same as the IEEE-P1014/D1.2, and the IEC-821. The IEC-821 version was approved in 1986.

The Revision C.1 document is considered to be the last version of the specification which does not have a copyright. The IEEE and ANSI hold copyrights on all later versions.

1.4.1.5 IEEE-1014-1987 (Revision C.3)

In 1987 the IEEE standards board approved a new draft called the IEEE-P1014/D2.2, which was also called the IEEE-1014-1987 (and sometimes Revision C.3). The changes between the Revision C.1 specification and the IEEE version are mostly clarifications of the various options of the bus. Until this draft, the IEEE standards board was concerned mostly with module compatibility (because of the many options). The new version attempted to clarify the various options to insure compatibility. Some of the changes included:

- Editorial changes clarifying the various options.
- Details of options were moved next to their descriptions.
- Redundant PERMISSIONS were removed.
- Addition of the FAIR requester.
- Addition of the J1/J2 monolithic backplane.
- Appendix D, which describes metastability and sample circuits.
- Appendix E, describing permissible options was added.

1.4.1.6 Revision D

In 1991 work began on the Revision D version of the VMEbus specification. The concepts in Revision D were later moved to the ANSI accredited VITA Standards Organization (VSO). These concepts were combined with wording from the (non-copywritten) Revision C.1 bus specification, and eventually became the ANSI/VITA 1-1994 standard.

1.4.1.7 ANSI/VITA 1-1994 (VME64)

The ANSI/VITA 1-1994 (also known as VME64) is the most recent version of the bus specification.

Suppliers of VMEbus boards and chips have introduced a variety of new products that conform to the new VME64 standard. As the next generation architecture for VMEbus, VME64 promises to extend the life of VMEbus well into the 21st century. The new standard offers a much-needed 'face-lift', with enhancements such as higher bandwidths, larger address spaces and easier-to-use cards.

As shown in Figure 1-1, VME64 is a mechanical and electrical 'superset' of the original standard. It offers new features such as:

- Larger, 64-bit data path for 6U boards.
- Larger, 64-bit addressing range for 6U boards.
- 32-bit data and 40-bit addressing modes for 3U boards.
- Twice the bandwidth (up to 80 Mbytes/sec).
- Lower noise connector system.
- Cycle retry capability.
- Bus LOCK cycles.
- First slot detector.
- Automatic 'plug-and-play' features.
- Configuration ROM / CSR capability.
- Auto-slot identification.
- Re-definition of SERCLK and SERDAT pins.

Actually, the term 'VME64' is something of a misnomer, as all VMEbus modules that conform to the Revision C.1 specification are now considered to be VME64 compliant (regardless of their data transfer capability). For example, 16 or 32-bit CPU boards designed under the old specification can be (correctly) identified as VME64 compatible modules.

All of the enhancements in VME64 are optional. New products work in conjunction with older 'legacy' boards, thereby providing a smooth upgrade path for system integrators.

1.4.1.8 VITA 1.1-1997 (VME64x)

In 1997 the VITA Standards Organization (VSO) adopted a superset to the VME64 standard. The new standard is called the VME64 Extensions (VME64x). It adds many new features to the VME64 architecture such as:

- A new 160 pin connector family.
- A new 95 pin P0/J0 connector.
- 141 more user defined I/O pins.
- Rear plug-in units (transition modules).
- Injector/extractor locking handles.
- EMC (electromagnetic conformance) front panels.
- Better ESD (electrostatic discharge) protection.
- Card keying capability.
- 3.3 VDC power supply pins.
- More +5 VDC power supply pins.
- Geographical addressing.
- Expanded CR/CSR register definitions.
- A new 2eVME bus cycle (up to 160 Mbyte/sec).
- Support for live-insertion (hot-swappable) bus modules.
- A test and maintenance bus.

All legacy VME and VME64 modules are forward compatible to VME64x backplanes and subracks. That means that older bus modules can be plugged into newer systems.

In general, the reverse is also true. Bus modules designed to the VME64x standard are also backward compatible with older backplanes and subracks. For example, the new 160 pin connectors can be plugged into an older backplane. However, there are a few exceptions to this. For example, if a board requires the new +3.3 VDC power supplies, then it will not work in an older backplane (which does not have these power pins).

The VME64x standard describes many optional features. However, the standard insists that a minimum set of features be present on boards and backplanes before they are considered to be VME64x compliant. All of the other features in the standard are considered optional.

For example, the minimum features that must be present on *6U modules* include:

- 160 pin connectors.
- All defined grounds must be connected (row ‘d’ is optional).

The minimum features that must be present on a *6U backplane* include:

- 160 pin connectors.
- All defined grounds must be connected.
- Monolithic PCB (i.e. must include both J1 and J2 connectors).
- Geographical address pins.
- Must route and terminate all VME64 and VME64x bused signal lines.
- Power connection and distribution for +5V, +3.3V, +/- 12V, +/- V1, +/- V2 and VPC.
- If rear (user defined) I/O pins are supported, then the rear connections must comply with the IEEE 1101.11 rear I/O transition board standard.

1.4.2 VMEbus Terminology

The VMEbus specification uses special terminology to describe what can and cannot be done. The concepts of functional modules and sub-buses improve the clarity of the specification. In addition, written rules are used to clearly define the various modes of operation. These are categorized using five key words: *rule*, *recommendation*, *suggestion*, *permission* and *observation*.

1.4.2.1 Rule

A *rule* is a precise statement of what you can and cannot do. Rules must always be followed to insure compatibility with other VMEbus modules. For example:

> RULE 2.3:
> VMEbus Slave boards MUST NOT respond to the reserved address-modifier codes.

1.4.2.2 Recommendation

A *recommendation* is less severe than a *rule*, and is based on the experience of the VMEbus architects. Recommendations prevent slow or awkward system operation, and are generally good advice. For example:

> RECOMMENDATION 2.1:
> VMEbus users MAY tailor the use of the user-defined address-modifier codes to their own needs and decode them in a flexible way on Slave boards. Users can then configure the board to give any decoding required for their system.

1.4.2.3 Suggestion

A *suggestion* provides board designers with a clear understanding of how to make VMEbus modules. For example:

> SUGGESTION 2.5a
> Design Slaves to respond to data transfers when one or both of DS0* and DS1* are low and when DTACK* and BERR* are both high, instead of a simultaneous low level on AS* and one or both of DS0* and DS1*.

1.4.2.4 Permission

A *permission* is granted when a rule does not specifically prohibit a design technique. Permissions are especially helpful when the description of a bus function is somewhat ambiguous. For example:

> PERMISSION 2.3
> User-defined codes MAY be used for any purpose which board vendors or board users deem appropriate (page switching, memory protection, master or task identification, privileged access to resources, etc.).

1.4.2.5 Observation

An *observation* does not offer specific advice. It shows the implications of rules, recommendations, suggestions or permissions. For example:

> OBSERVATION 2.5
> Reserved address modifier codes are for future enhancements. If slave boards respond to these codes, incompatibilities might result at some future date, when usage of these codes is defined.

1.4.2.6 Data Strobe Nomenclature

In the VMEbus specification there are two forms of data strobe nomenclature:

- DSA*: First data strobe to fall (or rise).
- DSB*: Second data strobe to fall (or rise).
- DS0*: Name for data strobe zero.
- DS1*: Name for data strobe one.

In popular articles (such as the trade press) the term 'DSX' will sometimes be used. This term refers to either data strobe. For example: "When DSX is asserted, the data is valid."

1.4.3 Diagrams

Three types of diagrams are used to clearly describe the bus specification. These are called timing, sequence and flow diagrams.

1.4.3.1 Timing Diagrams

Timing diagrams show the timing relationships between the various bus signals. However, the first time user can get confused when reading the timing parameters in the

VMEbus specification. Once understood, they become a powerful tool for quickly evaluating and designing bus modules. To illustrate their use consider the address broadcast timing diagram (VME64 specification Figure 2-12) shown in Figure 1-14.

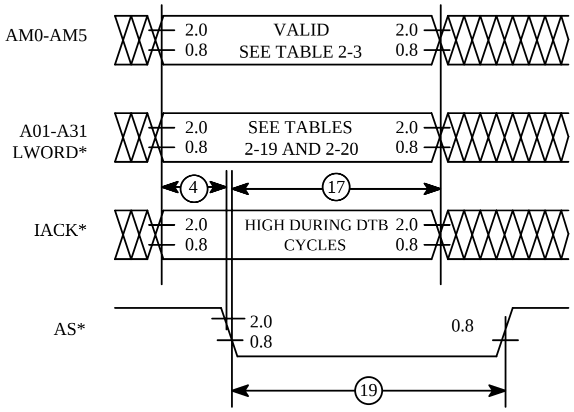


Figure 1-14 Address Broadcast Timing - All Cycles

For example, if we needed the time between address lines A01-A31 valid to AS* asserted we would use timing parameter (4). The exact time for (4) would then be determined by looking at the timing parameter table reproduced here in Figure 1-15.

PARAMETER	MASTER		SLAVE		LOC. MON.	
NUMBER	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
4	35		10		10	
17	40		30		30	
19	40		30		30	

Figure 1-15 Timing Parameters are Presented for Masters, Slaves and Location Monitors

The table presents three values of timing parameter (4): masters, slaves and location monitors. The parameters can be different depending upon the module type. Masters and slaves have different parameters because they reside at different locations on the backplane. Since signals take time to propagate, they may get skewed between the master and the slave. The table adds timing adjustments for this skew. If the timing for a master module is needed: use 35 ns. If it is a slave or location monitor: use 10 ns.

When verifying bus compatibility, always measure the timing parameters at the appropriate spot on the backplane (master, slave or location monitor).

Figure 1-14 also shows how timing diagrams are cross referenced to other information. For example, where the address modifier code AM0-AM5 is valid, the diagram directs the reader to Table 2-3 in the specification. Table 2-3 is partially reproduced in Figure 1-16 and defines the valid states of the address modifier.

Hex	ADDRESS MODIFIER						
Code	5	4	3	2	1	0	FUNCTION
3F	H	H	H	H	H	H	A24 supervisory block transfer (BLT)
3E	H	H	H	H	H	H	A24 supervisory program access

Figure 1-16 VMEbus Table 2-3 Shows the Valid states of the Address Modifier Code

To aide the user in determining bus compatibility, the timing parameters are cross referenced to additional rules, recommendations, suggestions, permissions and observations. Again, these are categorized throughout the VMEbus specification according to the module type (master, slave or location monitor). For example, timing parameter (4) is cross referenced to observation 2.52 for a slave as Figure 1-17 shows. This further clarifies the timing parameter.

4. Rule 2.30:
   The Master MUST NOT drive AS* low until IACK* has been high, and the required lines among A[31..1], AM[5..0], and LWORD* have been valid for this minimum time.

Figure 1-17 Rules, Recommendations, Suggestions, Permissions And Observations Are Cross Referenced To The Timing Parameters

1.4.3.2 Sequence Diagrams

Sequence diagrams show the timing relationships between various bus signals. They don't specify timing relationships, but they do help clarify the order of events. Figure 1-18 shows an example of a sequence diagram.

1.4.3.3 Flow Diagrams

Flow diagrams are another way to show the sequence of events during a bus cycle. They describe the operation of the bus in a sequential manner, and also show the interaction of the various functional modules. Figure 1-19 shows an example of a flow diagram.

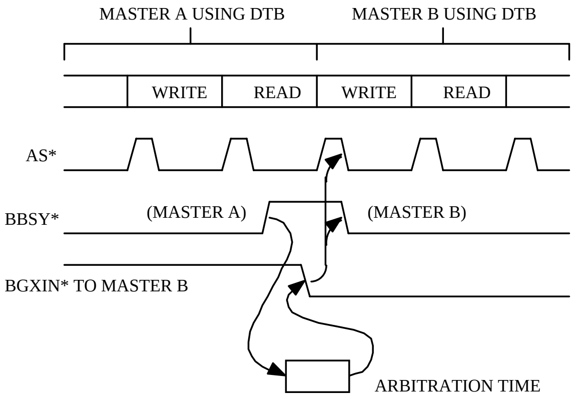


Figure 1-18 Sequence Diagram

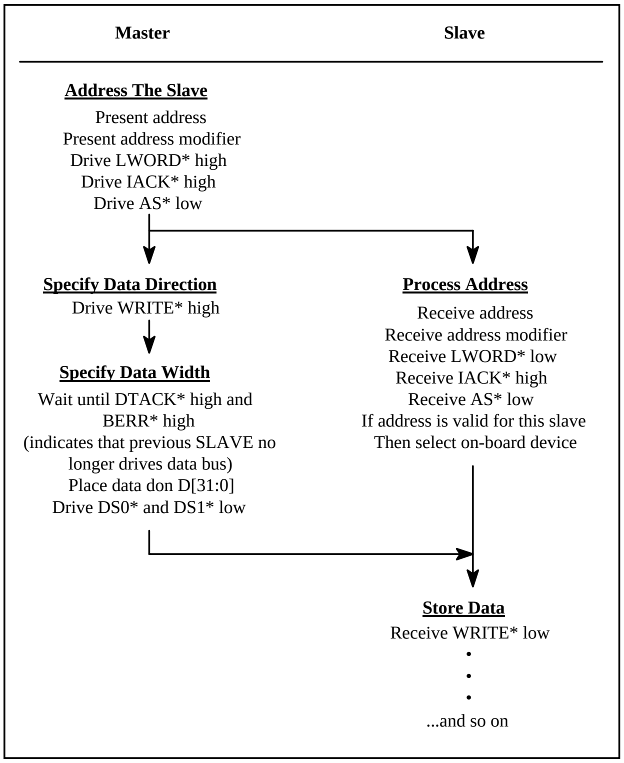


Figure 1-19 Flow Diagram

1.5 VMEbus Interface ICs

Table 1-5 is a list of interface ICs available for VMEbus. More detailed information is available from the IC manufacturers:

Cypress Semiconductor www.cypress.com
Tundra Semiconductor Corporation www.tundra.com
PLX Technology www.plxtech.com

1.6 VXIbus Interface ICs

Interface ICs for VXIbus are available from:

Interface Technology, Inc.
www.interfacetech.com

1.7 Where to Find Information About VMEbus and VXIbus

There are many sources of information about VMEbus and VXIbus. Those listed below provide a wealth of information on the topic.

1.7.1 The Internet

The VMEbus community is very active on the internet. Probably the best source of information is the world-wide-web, where many sites are available. Just query any of the popular search engines using the keywords of *VMEbus* or *VXIbus*. Most of the major VMEbus manufacturers offer web pages as well. VITA also offers an excellent site, with links to its member organizations and companies. Their web page can be found at: <www.vita.com>.

Another good source of information is the VMEbus USENET News Group. This group has on-going discussions on various topics. Users can post questions, comments and suggestions. The VMEbus USENET News Group can be found at: <comp.arch.bus.vmebus>.

There are other internet services for VMEbus users as well. For example, various *list servers* are available. Much of the standards activity now takes place in this way. For more information on the list servers contact VITA at <info@vita.com>.

Table 1-5 VMEbus Interface ICs.

VMEbus Interface ICs		Function											
Company	Device	Master	Slave	Location Monitor	Bus Timer	Interrupter	Handler (Interrupt)	Requester (bus)	Arbiter	System Clock Driver	Power Monitor	64-bit Support	Other
Cypress Semiconductor	CY7C960		●			●						●	●
	CY7C961	●	●			●		●				●	●
	VIC068	●	●	●	●	●	●	●	●	●	●		●
	VAC068												●
PLX Technology	VME12XX	●						●	●	●	●		
	VME2000		●										
	VME3000				●								
	VME4000					●							
Tundra Semiconductor Corporation	Universe™	●	●	●	●	●	●	●	●	●	●	●	●
	Trooper	●	●	●	●	●	●	●	●	●	●	●	●
	CA91C014					●	●	●	●	●	●		
	CA91C015	●	●	●									●
	CA91C064	●	●	●									●

1.7.2 VITA

Information about VMEbus is available from VITA. VITA is an incorporated, non-profit organization of vendors and users having common market interest. VITA's activities are international in scope. The functions performed by VITA are primarily technical, promotional and user related. They are aimed at increasing the total market size, providing vendors greater market exposure, and users more timely technical information than they could achieve with their own efforts.

Founded in 1984, VITA believes in -- and champions -- open system architectures as opposed to proprietary system architectures. The association, through the efforts and hard work of member companies, has been successful in changing the way computing is done at all levels.

VITA's mission includes not only promoting VMEbus, but promoting the very concept of open technology as embodied in the many standards currently under development within the VITA Standards Organization (VSO). The VSO is an ANSI accredited standards developer.

VITA's continuing goal is to unite manufacturers and users through the acceptance and implementation of open technology standards. Members come together with different views and ideas and use VITA as a forum for all.

VMEbus technology specifications are available from VITA.

For more information about VITA contact:

VITA
email: info@vita.com
www.vita.com

1.7.3 VIPA (VITA International Physics Association)

The VITA International Physics Association (VIPA) is a VITA special interest group that represents the international physics community in the VSO. VIPA has membership in VITA through the Organization for Nuclear Research (CERN - Geneva, Switzerland), the

Fermilab National Accelerator Laboratory (Batavia, IL USA) and the JVP Working Group (Japan).

VIPA publishes guidelines for using VMEbus within the physics community. This document, known as VME-P, presents guidelines for grounding, high frequency signaling, low noise analog and other areas of interest to the physics community.

For further information about VIPA contact VITA (at one of the addresses above). VIPA also has extensive information and standards available on the internet, especially the world-wide-web. For more information, look under 'VIPA' on one of the web search engines.

1.7.4 VMEbus Systems Magazine

Users may wish to subscribe to VMEbus Systems magazine. This bi-monthly journal is free to those who specify, buy, use or are considering the use of VMEbus products. It has a lot of information on new trends, products and ideas. VXIbus users may wish to subscribe to VMEbus Systems. For more information contact:

VMEbus Systems Magazine
30233 Jefferson, St. Clair Shores, MI USA 48082
TEL: (586) 415-6500
FAX: (586) 415-4882
www.opensystems-publishing.com

1.7.5 VXIbus Consortium

Information about VXIbus can be obtained from The VXIbus Consortium. This organization is established to maintain, develop and promote VXIbus. They also publish an annual product directory listing available VXIbus products. For more information contact their representative at:

Bode Enterprises/VXI Consortium
2525 Camino del Rio South, Suite 340
San Diego, CA USA 92108
TEL: (619) 297-1213
email: fbode@vxinl.com

1.7.6 Bus Specifications

Users and board designers alike will want to obtain a copy of the VMEbus specification. Several versions are available. The C.1 version has been distributed as a promotional give-away by some vendors. The most current version is the ANSI/VITA 1-1994 and is available through VITA (at the address above).

ANSI also publishes the ANSI/VITA 1-1994, as well as other specifications. Their address is:

American National Standards Institute (ANSI)
11 West 42nd Street
New York, NY USA 10036
TEL: (212) 642-4900
FAX: (212) 398-0023
www.ansi.org/home.html

The IEEE 1014-1987 specification is available from the IEEE. Copies can be obtained from:

IEEE Service Center
Publications Sales Department
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ USA 08855-1331
800-678-4333
www.ieee.org

Some IEEE documents that have been withdrawn (such as the original IEEE 1101 Mechanical Specification) are available from Global Engineering Documents. Their address is:

Global Engineering Documents
15 Inverness Way East
Englewood, CO USA 80112
TEL: (303) 792-2181

1.8 References

Black, John A. Jr. and Shlomo Pri-Tal. "From B to C: A Summary of Changes in VMEbus Specifications" *VMEbus Systems* Fall 1985

Di Giacomo, Joeseph. Digital Bus Handbook McGraw-Hill 1990

IEEE Standard for Mechanical Core Specifications for Microcomputers Using IEC 603-2 Connectors / IEEE Std 1101.1-1991 IEEE, New York NY 1992

IEEE Standard for Mechanical Core Specifications for Conduction-Cooled Eurocards / IEEE Std 1101.2-1992 IEEE, New York, NY 1992

IEEE Standard for Additional Mechanical Specifications for Microcomputers Using the IEEE Std 1101.1-1991 Equipment Practice / IEEE Std 1101.10-1996 IEEE, New York, NY 1997

IEEE Standard for Mechanical Rear Plug-in Units Specifications for Microcomputers Using the IEEE 1101.1 and the IEEE 1101.10 Equipment Practice / IEEE Std P1101.11-1996 (draft 3.1) IEEE, Piscataway, NJ 1997

IEEE Standard for VMEbus Extensions for Instrumentation: VXIbus / IEEE Std 1155-1992 IEEE, New York, NY 1993

Industry Bus Configurations Manual Hybricon Corporation, Ayer MA 1987

PCI Local Bus Specification, Revision 2.0 PCI Special Interest Group 1993

Peterson, Wade D. "VME64: Taking The VMEbus Architecture Into The 21st Century" *VMEbus Systems* August 1996

Peterson, Wade D. "VME64x: The VME64 Extensions Standard" *VMEbus Systems* August 1997

Pri-Tal, Shlomo. "IEEE 1014-1987: How Does It Differ From Rev C.1?" *VMEbus Systems* July-August 1987

Prahalad, C.K. and Gary Hamel. "The Core Competence Of The Corporation" *Harvard Business Review* May-June 1990

VMEbus Specification / Revision C.1. PRINTEX, Phoenix AZ 1985.

VMEbus Specification / ANSI/IEEE STD1014-1987. VITA, Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA, Scottsdale, AZ 1995

VME64 9U x 400mm Format Draft Standard / VITA 1.3-199x (Draft 0.3) VITA, Scottsdale, AZ 1995

VME64x Specification / VITA 1.1-1997 (Draft 2.0) VITA, Scottsdale, AZ 199

Chapter 2
Data Transfers

VMEbus data is transferred over the Data Transfer Bus (DTB). Masters use the DTB to move data to and from slaves, and interrupters use it to pass STATUS/ID words to handlers.

2.1 The Data Transfer Bus Under VME64 And VME64x

Many changes were made to the Data Transfer Bus in theVME64 and VME64x bus specifications. These were required to keep VMEbus competitive with other technologies.

During the original development of VMEbus in 1981, 16-bit microprocessors (such as the 68000) with 24-bit addressing had only just hit the market, 8-bit peripheral chips were the state of the art, and 32-bit processors had not been developed. The original bus architects, however, had the vision to make VMEbus a flexible architecture so that it could be upgraded as greater demands were placed upon it.

As the speed of microprocessors, memory and I/O increased, the theoretical bandwidth of VMEbus has become a significant factor in overall system capability. By the mid 1980's the theoretical VMEbus data transfer rate of 40 Mbyte/second had become a reality, and in some cases was a severe limitation of the bus.

To meet this challenge, the VITA Standards Organization (VSO) accepted the task of doubling (and then quadrupling) the bandwidth of VMEbus. A variety of schemes were proposed. These ranged from changes in setup-and-hold timing, to secondary buses. While these techniques all improved system performance, they either (a) wouldn't conform to the older VMEbus standards, or (b) they created radical new changes in the system architecture.

In 1988 Performance Technologies (East Rochester, NY - USA) developed a superset to VMEbus which doubled its bandwidth. The Performance Technologies approach modified the standard block transfer cycle to combine the address and data lines into one giant 64-bit path. This approach later became the multiplexed block transfer cycles known today in VME64 as the *MBLT* and *MD32* cycles.

The attractiveness of the MBLT and MD32 cycles were their simplicity and compatibility with previous versions of VMEbus. They used similar timing and operating parameters as the older 32-bit block transfer cycle (D32BLT). The physical implementation was straightforward and did not require any special backplane modifications (such as additional connector pins, etc.).

Other features were also added to the Data Transfer Bus in the VME64 specification. For example, new RETRY* and LOCK functions were added to support complex applications such as bus-to-bus links.

In 1997 a new approach was taken by the VSO to further increase VMEbus bandwith with the two-edge VMEbus cycle (2eVME). This cycle is defined in the VME64x standard, and is designed to acheive data transfer rates of up to 160 Mbyte/second.

The 2eVME cycle uses two radical techniques to increase the operating speed of the bus: (a) it decreased then number of control transitions on the data strobes and DTACK* from four edges to two edges and (b) implemented a new class of bus transceiver logic called Enhanced Transceiver logic (or ETL).

2.2 Bus Cycles

The VME64 Data Transfer Bus allows eight types of bus cycles. These include the:

- Read/write cycle
- Read-modify-write cycle
- Address-only cycle
- Block transfer cycle
- Multiplexed block transfer (MBLT) cycle
- Multiplexed data-32 (MD32) cycle
- Address-only with handshake (ADOH) cycle
- Two-edge VMEbus (2eVME) cycle

Not all VMEbus modules are compatible with all types of bus cycles. When evaluating or designing boards, make sure that each slave is compatible with each master. For example, a master may generate an ADOH cycle, but some slaves will not support it.

2.2.1 Read/Write Cycle

The read/write cycle is the most common VMEbus cycle. Readers are urged to first understand the concepts of the read/write cycle before progressing to the more advanced bus cycles. That's because the other bus cycles are more-or-less variations of the read/write cycle.

2.2.1.1 Addressing

A master addresses a slave during every read/write cycle. This is done with address lines A01-A31, a six bit address modifier code AM0-AM5, and two control signals called IACK* and LWORD*. All of these signals are qualified by the falling edge of address strobe [AS*].

An alternative design approach, which will be discussed shortly, permits most addresses to be qualified by the falling edge of the data strobes DS0* & DS1* (rather than by the address strobe AS*). This simplifies the design of slave boards, and circumvents some tricky interface problems.

Note also that there is no 'A00' address line on VMEbus. The zeroith address bit is actually encoded into the two data strobes DS0* and DS1*. These two signals determine the byte location (within a four byte group) where data is to be accessed.

2.2.1.2 Address Sizing

VMEbus has five possible address widths as shown in Table 2-1. The address width can be changed on every bus cycle. This is sometimes called *dynamic address sizing*. The five types of address are called A16, A24, A32, A40 and A64.

Table 2-1 Dynamic address sizes

Address Modifer Type (old terminology)	Address Bits	Active Address Lines	Mnemonic
Short I/O	16	A01-A15	A16
Standard	24	A01-A23	A24
Extended	32	A01-A31	A32
-----	40	A01-A23 D00-D15	A40 (†)
Long	64	A01-A31 D00-D31	A64 (†)
(†) New in VME64			

In older versions of the VMEbus specification (and some manufacturers' data sheets), the A16, A24, A32 and A64 addressing modes are sometimes called *short I/O*, *standard*, *extended* and *long* addressing. However, this terminology was removed from the VME64 specification to avoid inconsistencies.

The most obvious advantage to the dynamic addressing scheme is to allow older boards to share the bus with newer ones. For example, a 68000 CPU module (with 24-bit addresses) can share the bus with a 68050 (which has 32-bit addresses). This approach also allows VMEbus to keep pace with the ever increasing demand for memory.

The A40 and A64 addressing modes were first adopted in the VME64 standard. They were added because some applications require very big address spaces.

2.2.1.3 Address Modifier Code

Dynamic address sizing is made possible because of a six bit *address modifier code* [AM0-AM5] which accompanies each address. This can be thought of as a *tag* that is attached to each address. There are 47 defined address modifer codes, which are summarized in Table 2-2.

During a bus cycle the VMEbus master tags each address with an address modifier code. Slaves monitor these codes so they can determine which address lines to monitor. A16 addresses are decoded from A01-A15, A24 addresses from A01-A23 and so on. The address routing is summarized in Table 2-3.

The address modifier code also indicates the type of bus transaction. It discriminates between instruction fetches, data cycles and so on. Originally, this information was intended for debugging purposes. In the early days of VMEbus it was fairly common to fetch microprocessor instructions across the backplane. This allowed logic analyzers to separate them from data cycles, and was a very powerful debugging tool.

Today, instructions are rarely fetched across VMEbus because they are generally stored in local (CPU) memory. Local memories are now fast, large and cheap, and VMEbus creates a significant bottleneck. Most people use VMEbus as an I/O channel for passing data between CPU cards and peripherals.

Some of the information in the address modifier codes is obsolete. For example, the A24 codes contain *supervisory* and *non-privileged* modes. These corresponded directly to the 68000 microprocessor *supervisor* and *user* modes, and were an early attempt at memory management. However, this function is now performed on (local) memory management IC's (MMU's), thereby rendering this data useless.

Table 2-3 also includes information about the interrupt acknowledge cycle. IACK* is a signal asserted by *interrupt handlers* to show that the current cycle is an interrupt acknowledge cycle, and negated by *masters* to show that it is a *data transfer cycle*. It is sometimes useful to think of IACK* as a seventh address modifier bit. When IACK* is asserted, the address modifier code is ignored by slaves.

Address modifier codes also simplify the design (and lower the cost) of many VMEbus modules. Modules can be as simple or as complex as the application requires. Slaves, like serial I/O modules, require only several bytes of address space and can use A16 addressing. This reduces the number of parts on a board by decreasing the number of comparator and control logic ICs. This lowers the board cost and conserves space. More complex modules, such as memory or graphic controllers, must use A24 or A32 addressing because of their large memory spaces.

The address modifiers also make single (3U) and double height (6U) modules compatible. Single height modules use only the P1/J1 connector, and can only monitor address lines A01-A23. This limits these modules to the A16, A24 and A40 cycles. Double height modules can monitor an additional eight address lines on the P2/J2 connector, so they can also perform 32 and 64-bit address transfers. Without the address modifier code, the simple P2/J2 expansion bus would have been awkward.

2.2.1.4 Address Mnemonics

A series of mnemonics have been defined to help users select compatible modules. These are shown in Table 2-4. A16, A24, A32, A40 and A64 mnemonics correspond to the number of address bits used during a transfer. For example an A16 slave decodes sixteen address bits, and will respond to bus cycles generated by an A16 master.

When selecting modules with master capability, make sure that they will generate all the cycles required by the slave boards. This is not guaranteed by all versions of the VMEbus specification.

The IEEE-1014-1987 VMEbus specification requires that A32 masters generate A24 and A16 cycles. Similarly, A24 masters must also generate A16 cycles. Slaves do not have any such requirements. Earlier versions of the VMEbus specification did not have these requirements.

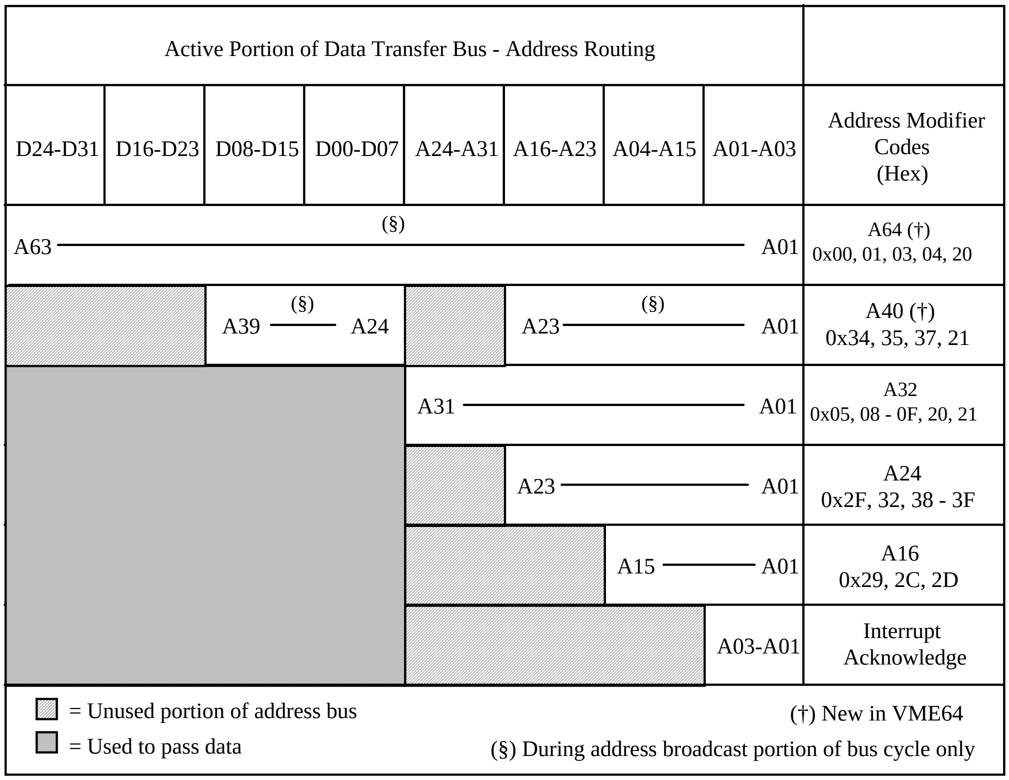
The ANSI/VITA 1-1994 VMEbus specification first offered the A40 and A64 addressing modes. Under this specification the A64 master must be A16, A24 and A32 compliant, but not necessarily A40 compatible. The A40 master must be A16 and A24 compliant, but not necessarily A32 compatible. Slaves do not have any such requirements.

Table 2-2 Address Modifier Codes

Address Modifer (Hexidecimal)	No. Addres s Bits	Description
3F	24	A24 supervisory block transfer (BLT)
3E	24	A24 supervisory program access
3D	24	A24 supervisory data access
3C (†)	24	A24 supervisory 64-bit block transfer (MBLT)
3B	24	A24 non-privileged block transfer (BLT)
3A	24	A24 non-privileged program access
39	24	A24 non-privileged data access
38 (†)	24	A24 non-privileged 64-bit block transfer (MBLT)
37 (†)	40	A40BLT (MD32)
35 (†)	40	A40 lock command (LCK)
34 (†)	40	A40 access
32 (†)	24	A24 lock command (LCK)
2F (†)	24	CR/CSR space
2D	16	A16 supervisory access
2C (†)	16	A16 lock command (LCK)
29	16	A16 non-privileged access
21 (††)	32 or 40	2eVME for 3U bus modules (address size in XAM code)
20 (††)	32 or 64	2eVME for 6U bus modules (address size in XAM code)
10-1F	--	User-defined
0F	32	A32 supervisory block transfer (BLT)
0E	32	A32 supervisory program access
0D	32	A32 supervisory data access
0C (†)	32	A32 supervisory 64-bit block transfer (MBLT)
0B	32	A32 non-privileged block transfer (BLT)
0A	32	A32 non-privileged program access
09	32	A32 non-privileged data access
08 (†)	32	A32 non-privileged 64-bit block transfer (MBLT)
05 (†)	32	A32 lock command (LCK)
04 (†)	64	A64 lock command (LCK)

03 (†)	64	A64 block transfer (BLT)
01 (†)	64	A64 single transfer access
00 (†)	64	A64 64-bit block transfer (MBLT)

Table 2-3 Address routing during various addressing modes



Active Portion of Data Transfer Bus - Address Routing								
D24-D31	D16-D23	D08-D15	D00-D07	A24-A31	A16-A23	A04-A15	A01-A03	Address Modifier Codes (Hex)
A63			(§)				A01	A64 (†) 0x00, 01, 03, 04, 20
▨	▨	A39 (§)	A24	▨	A23	(§)	A01	A40 (†) 0x34, 35, 37, 21
■	■	■	■	A31			A01	A32 0x05, 08 - 0F, 20, 21
■	■	■	■	▨	A23		A01	A24 0x2F, 32, 38 - 3F
■	■	■	■	▨	▨	A15	A01	A16 0x29, 2C, 2D
■	■	■	■	▨	▨	▨	A03-A01	Interrupt Acknowledge

▨ = Unused portion of address bus

■ = Used to pass data

(†) New in VME64

(§) During address broadcast portion of bus cycle only

Table 2-4 Mnemonics that describe the various addressing modes

Mnemonic	Description
A16	Masters, slaves and location monitors generate or accept 16-bit address transfers. Sometimes called *short I/O* addressing.
A24	Masters, slaves and location monitors generate or accept 24-bit address transfers. Sometimes called *standard* addressing. A24 masters must also be A16 compatible.
A32	Masters, slaves and location monitors generate or accept 32-bit address transfers. Sometimes called *extended* addressing. A32 masters must also be A16 and A24 compatible.
A40 (†)	Masters, slaves and location monitors generate or accept 40-bit address transfers. Primarily used with single height (3U) VMEbus modules. A40 masters must also be A16 and A24 compatible (but not necessarily A32 compatible).
A464 (†)	Masters, slaves and location monitors generate or accept 64-bit address transfers. Can only be used on double height (6U) VMEbus modules. A4 masters must also be A16, A24 and A32 compatible (but not necessarily A40 compatible). Sometimes called *long* addressing.

2.2.1.5 New VME64 Address Modifier Codes Clarified (†)

The ANSI/VITA 1-1994 (VME64) standard is somewhat ambiguous as to the use of new address modifier codes 0x00, 0x01, 0x03, 0x34 and 0x37.

When attempting to understand the differences between these codes, it's best to separate the *address phase* from the *data phase* of each bus cycle. Just remember that you can mix and match data cycles to address cycles as long as the address modifiers exists to inform the slave.

Address modifier code 0x03 (A64 block transfer) describes a 64-bit address which requires that the address and data lines are combined to form a 64-bit path. During the data phase of the cycle a block transfer (BLT) is specified. The BLT transfer can be 8, 16 or 32-bits wide.

Address modifier code 0x01 (A64 single transfer access) defines a bus cycle with a 64-bit address (as described above). However, during the data phase only a single 8, 16 or 32-bit transfer can take place. This cycle is substantially the same as that defined by the *A64 block transfer*, except that only one data transfer follows the address phase. This special cycle was created to simplify slave modules by eliminating the need for a local address counter (which is needed if multiple data transfers are supported).

Address modifier code 0x00 (A64 64-bit block transfer) refers to the MBLT cycle. It defines a bus cycle with a 64-bit address (as described above), but during the data phase the address and data lines are combined to form a 64-bit data path.

Address modifier code 0x34 (A40 access) describes a 40-bit address which requires that the address and data lines are combined to form a 40-bit path. During the data phase of the cycle a block transfer of 8 or 16-bits is supported. This cycle is intended for 3U VMEbus modules (which can only use the address and data lines on the P1/J1 connector). However, 6U modules may also support the cycle.

Address modifier code 0x37 (A40BLT) refers to the MD32 cycle. It defines a bus cycle with a 40-bit address (as described above), but during the data phase the address and data lines are combined to form a 32-bit data path. This cycle is intended for 3U VMEbus modules (which can only use the address and data lines on the P1/J1 connector). However, 6U modules may also support the cycle.

2.2.1.6 Data Transfers

VMEbus data transfers are made over the data lines D00 - D31. Data is qualified by the falling edges of data strobes DS0* and DS1*.

Sixteen of the thirty-two data lines are located on the P1/J1 connector, and sixteen are located on the P2/J2 connector. That means that single height (3U) modules can (normally) support only sixteen bit transfers, and double height (6U) modules can support thirty-two bit transfers. However, the VME64 specification now allows special multiplexed block transfer cycles that allow 3U modules to carry 32-bits, and 6U modules to carry 64-bits.

2.2.1.7 Data Sizing

The data bus is dynamically configured (just like the address bus). Data transfers of 8, 16, 24, 32 and 64-bits can be made without any software overhead whatsoever. This also makes VMEbus products compatible over a wide range of technology.

Data bus sizing is achieved by splitting the data lines into four byte-wide banks: D00-D07, D08-D15, D16-D23 and D24-D31. An additional bank on the address lines is also used for MBLT and MD32 cycles. The master signals the type and size of data transfer with strobes DS0* and DS1*, address line A01 and LWORD*.

VMEbus uses a BYTE(n) convention to specify how data is stored in memory, where (n) is the address offset from an even 32-bit boundary. Table 2-5 shows this convention.

During a data transfer the master asserts DS0*, DS1*, A01, A02 and LWORD* depending upon where it expects to read or write data. The level of these signals and the associated byte lane ordering is shown in Table 2-6.

Dynamic data sizing allows older and newer technologies to work together. For example, a CPU module with a 68000 microprocessor (8 or 16-bit data path) can share VMEbus with a 68030 based CPU (8, 16, 24 or 32-bit data path).

Table 2-5 Data organization in memory

Categories of Byte Locations			
4-Byte Group		8-Byte Group (†)	
Operand	Byte Address (Binary)	Operand	Byte Address (Binary)
BYTE(0)	XXXX....XX00	BYTE(0)	XXXX....X000
BYTE(1)	XXXX....XX01	BYTE(1)	XXXX....X001
BYTE(2)	XXXX....XX10	BYTE(2)	XXXX....X010
BYTE(3)	XXXX....XX11	BYTE(3)	XXXX....X011
Key: (X) Don't care (†) New in VME64		BYTE(4)	XXXX....X100
		BYTE(5)	XXXX....X101
		BYTE(6)	XXXX....X110
		BYTE(7)	XXXX....X111

2.2.1.8 Data Transfer Mnemonics

A series of data mnemonics have been developed to help users select compatible modules. These are shown in Table 2-7.

VMEbus offers seven styles of data bus interface. These are classified using the mnemonics D08(O), D08(EO), D16, D32, MD32, MBLT and 2eVME. Also notice that there is no D64 mnemonic. That's because 64-bit transfers are only allowed under the MBLT and 2eVME cycles.

The D08(O) slave transfers data on lines D00-D07. Transfers of eight bits can be made at odd addresses (e.g. 0x0002FF01 or 0x0002FF03). D08(O) *masters* are not explicitly allowed under the VMEbus specification. An example of a D08(O) *slave* would be an 8-bit serial I/O module.

The D08(EO) master or slave allows bytes to be transferred at even or odd addresses. Transfers to or from these interfaces must be done eight bits at a time, and only one data strobe may go low at any time. An example of a master with this interface would be a CPU module with an 8-bit processor (such as an 68008).

The D16 master or slave allows transfers over data lines D00-D16. Transfers to or from these interfaces must be done sixteen bits at a time, and at even byte boundaries. Note that the unaligned transfers (described below) are also capable of transferring sixteen bit data words, but at other than even byte boundaries. The VMEbus specification also requires that D16 slaves also support D03(EO) capabilities. This is not required for the master.

The D32 masters and slaves allow 32-bit data transfers over data lines D00-D31. These modules always transfer data at quad byte boundaries. Some D32 modules also support unaligned transfers (a.k.a. UAT transfers - described below). Under the VMEbus specification, D32 slaves must also include the D16 and D08(EO) capabilities.

Table 2-6 Active portions of the data bus during transfers

Active Portion of Data Transfer Bus - Byte Lanes and Byte Routing								Control Signal Levels				
A24-A31	A16-A23	A08-A15	LWORD* A01-A07	D24-D31	D16-D23	D08-D15	D00-D07	DS1*	DS0*	A01	LWORD*	A02
EIGHT BYTE(0-7) - MBLT (†) BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	BYTE(4)	BYTE(5)	BYTE(6)	BYTE(7)	(Address Broadcast Phase Only) 0	0	0	0	0
▒	▒	MD32 (†) BYTE(0)	BYTE(1)	▨	▨	MD32 (†) BYTE(2)	BYTE(3)	(Address Broadcast Phase Only) 0	0	0	0	X
▒	▒	▒	▒	QUAD BYTE(0-4) BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	0	0	0	0	X
▒	▒	▒	▒	UNALIGNED(0-2) BYTE(0)	BYTE(1)	BYTE(2)	▨	0	1	0	0	X
▒	▒	▒	▒	▨	UNALIGNED(1-3) BYTE(1)	BYTE(2)	BYTE(3)	1	0	0	0	X
▒	▒	▒	▒	▨	UNALIGNED(1-2) BYTE(1)	BYTE(2)	▨	0	0	1	0	X
▒	▒	▒	▒	▨	▨	DBL BYTE(2-3) BYTE(2)	BYTE(3)	0	0	1	1	X
▒	▒	▒	▒	▨	▨	DBL BYTE(0-1) BYTE(0)	BYTE(1)	0	0	0	1	X
▒	▒	▒	▒	▨	▨	▨	SINGLE BYTE(3)	1	0	1	1	X
▒	▒	▒	▒	▨	▨	SINGLE BYTE(2)	▨	0	1	1	1	X
▒	▒	▒	▒	▨	▨	▨	SINGLE BYTE(1)	1	0	0	1	X
▒	▒	▒	▒	▨	▨	SINGLE BYTE(0)	▨	0	1	0	1	X
ILLEGAL								1 0	0 1	1 1	0 0	X X

▨ = Unused portion of data bus

▒ = Used to pass address

(X) = Don't care

(†) New in VME64

Table 2-7 Mnemonics that describe the various data transfer modes

Mnemonic	Description
D08(O)	Slaves accept eight bit data transfers at odd addresses. There is no such thing as a D08(O) *master*.
D08(EO)	Masters, slaves and location monitors generate or accept eight bit data transfers at even or odd addresss.
D16	Masters, slaves and location monitors generate or accept 16-bit data transfers. D16 slaves must also support D08(EO).
D32	Masters, slaves and location monitors generate or accept 32-bit data transfers. D32 slaves must also support D16 and D08(EO) bus interfaces. Single height (3U)bus modules cannot support D32 modes.

MD32 (†)	Masters, slaves and location monitors generate or accept 32-bit multiplexed data transfers. MD32 slaves must also support D16 and D08(EO) bus interfaces. Single height (3U)bus modules cannot support D32 modes.
MBLT (†)	Masters, slaves and location monitors generate or accept 64-bit multiplexed data transfers. Single height (3U)bus modules cannot support D32 modes.
2eVME (††)	Masters, slaves and location monitors generate or accept 64-bit multiplexed data transfers using the 2-edge-VMEbus (2eVME) cycle. 3U and 6U versions of this cycle are available.
(†) New in VME64; (††) New in VME64x	

The MD32 interface monitors or drives data lines D00-D15, as well as address lines A01-A15 and LWORD*. The address and data lines are combined to form a 32-bit data path. Since all of these signals are available on the P1/J1 connector, the MD32 cycle can be used with 3U bus modules. MD32 slaves must also include the D16 and D08(EO) capabilities. This interface was first permitted under the VME64 bus specification.

The MBLT interface monitors or drives data lines D00-D31, as well as address lines A01-A31 and LWORD*. The address and data lines are combined to form a 64-bit data path. Since some of these signals are only available on the P2/J2 connector, the MBLT cycle cannot be used with 3U bus modules. This interface was first permitted under the VME64 bus specification.

The 2eVME interface monitors or drives data lines D00-D31, as well as address lines A01-A31 and LWORD*. The address and data lines are combined to form a 64-bit data path. Since some of these signals are only available on the P2/J2 connector, there are 3U and 6U versions of the bus cycle. 2eVME cycles operate at approximately twice the speed of MBLT cycles. This interface was first permitted under the VME64x bus specification.

2.2.1.9 Unaligned Data Transfers

Unaligned data transfers are allowed under the VMEbus specification. These modules can place two, three or four bytes of data at virtually any address boundary. These are called unaligned transfers. Unaligned transfers can speed up a VMEbus system by allowing 32-bits of data to be transferred at odd addresses (in two bus cycles instead of three).

Figures 2-1 and 2-2 illustrate how 32 or 16-bits of data can be located across four byte lanes. For example, case B of Figure 2-1 shows how a four byte transfer takes place at an unaligned boundary. The master can transfer the data using one of two methods. By one method the master transfers a Single Byte(1), a Double Byte(2-3) and a Single Byte(0). This means that the whole 32-bit transfer requires three bus cycles. Using a second method the master performs an Unaligned Byte(1-3) and a Single Byte(0) transfer. This second method takes only two bus cycles. Therefore, unaligned transfers can substantially reduce the number of bus cycles required to transfer a data word.

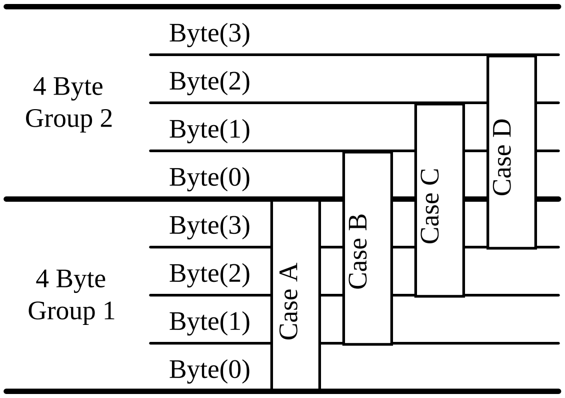


Figure 2-1 Four ways that 32-bits of data can be saved in memory

During unaligned transfers the VMEbus specification does not stipulate the order in which data is transferred to or from memory. In the example above, the Single byte(0) transfer could take place before or after the Unaligned byte(1-3) transfer. This can be important in multiprocessor systems where another master may be granted a bus cycle between two consecutive transfers. Software engineers should design flags (semaphores) accordingly.

Many software compilers offer 'soft switches' to allow data storage to be optimized for *least memory size* or for *highest speed*. For example, many ANSI-'C' compilers offer this feature. When the *least memory size* option is selected, data is stored so that the least amount of memory is used. That means that four bytes of data may be stored at other than quad byte boundaries (i.e. all memory locations within a block of memory are used). This type of storage usually requires unaligned transfers. When the *highest speed* option is selected, the software compiler automatically shifts addresses so that four bytes of data are always stored at quad byte boundaries. This eliminates the unaligned transfers, and speeds up the system.

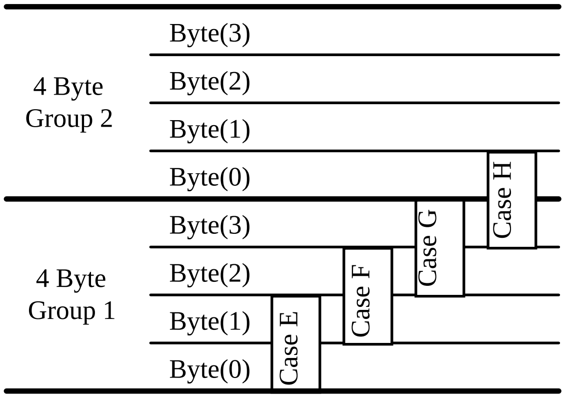


Figure 2-2 Four ways that 16-bits of data can be saved in memory

A special UAT mnemonic specifies whether an unaligned transfer can be generated by a master, accepted by a slave or monitored by a location monitor.

The IEEE-1014-1987 version of the bus specification requires that D32 slaves and location monitors must accept unaligned (UAT) data transfers. This was optional under earlier versions of the bus specification. However, the ANSI/VITA 1-1994 (VME64) specification removed this rule and, instead, only recommends this behavior.

There is no provision for MD32, MBLT or 2eVME unaligned data transfers.

2.2.1.10 Endian

Endian is a common term used in the computer industry to describe the way data is stored in memory. VMEbus is inherently *big-endian*. All modules are expected to operate in big-endian mode, regardless of the processor's domain.

This generally isn't a problem, especially with 680XX family microprocessors. However, other types, such as the Intel 80X86 families, operate in *little-endian* mode. This is also true of the PCI local bus.

When a processor or local bus operates in little-endian mode, the VMEbus interface (on that card) is expected to provide a little-endian to big-endian conversion. Many of the newer VMEbus interface ICs (such as the Tundra Universe™ chips) automatically provide this conversion.

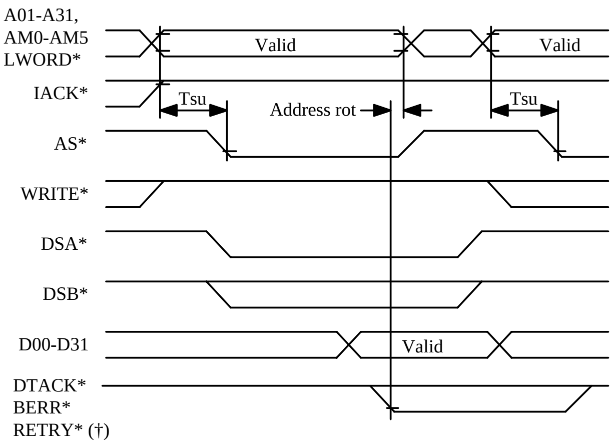
In principle, VMEbus could be operated as a little-endian bus. However, the VMEbus community (and bus specifications) have standardized on big-endian mode, and all commercial cards will conform to that convention. System integrators who build all of their own bus modules could conceivably run VMEbus in little-endian mode.

The big-endian to little-endian conversion can also be done in software. However, this is a very time consuming and cumbersome task, and is not (generally) recommended.

2.2.1.11 Typical Read/Write Cycle

During a typical read/write cycle the master first addresses a slave, and then transfers data. The data is transferred using data lines D00-D31, WRITE*, data transfer acknowledge [DTACK*], bus error [BERR*] and retry [RETRY*]. The data lines and the WRITE* signal are qualified by data strobes DS0* and/or DS1*.

The timing waveform for a typical read cycle is shown in Figure 2-3. Here a master addresses a slave by driving A01-A31, AM0-AM5, IACK* and LWORD*. These are qualified by the falling edge of AS*. The master also negates WRITE* and asserts data strobes DS0* and/or DS1*. The slave decodes the address, places data onto D00-D31 and asserts data transfer acknowledge [DTACK*]. When the master has latched the data, it informs the slave by negating the data strobe(s). The slave then negates DTACK* and the cycle is terminated.



(†) New in VME64

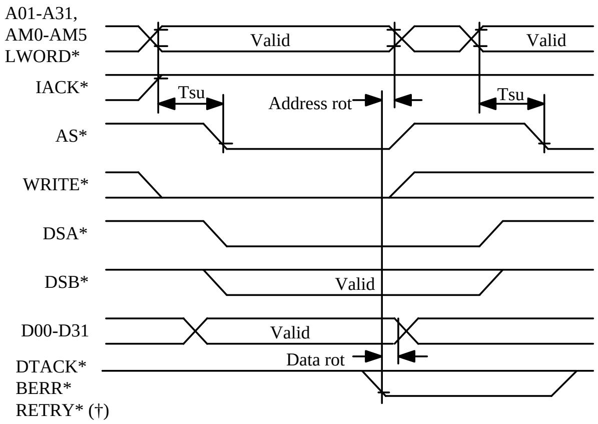
Figure 2-3 Read cycle with address pipelining

During an abnormal cycle the slave can also assert BERR* instead of DTACK*. This notifies the master that something has gone wrong. For example, a memory card can assert BERR* if a parity error occurred during the cycle.

If a slave doesn't respond during the bus cycle, such as during card-counting or memory sizing operations, then another functional module called a *bus timer* can asserts BERR*. The bus timer is a type of watchdog timer that asserts BERR* if the data strobes are asserted for too long a time. This terminates the cycle and prevents the bus from 'locking up'.

If a slave is busy it can also assert the RETRY* signal. This informs the master that the slave does not want to be interrupted, and that the cycle should be attempted again at a later time. The RETRY* signal is new in VME64, and was not defined under earlier versions of the specification.

A write cycle, which is similar to the read cycle, is shown in Figure 2-4. The main difference between the write and read cycles is that (a) the WRITE* signal is asserted and (b) the master places data onto the bus before either data strobe is asserted. Once the slave asserts DTACK*, BERR* or RETRY*, the master *can* immediately negate WRITE* and change the data lines. For this reason a slave must latch the data before it asserts DTACK*.



(†) New in VME64

Figure 2-4 Write cycle with address pipelining

2.2.1.12 Data Strobes

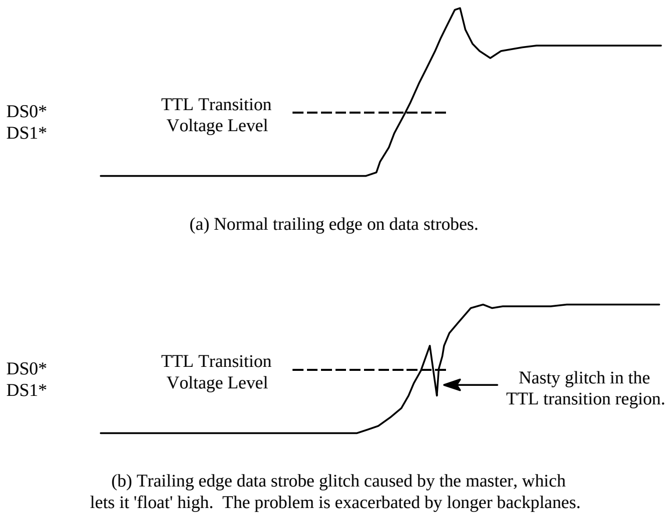
The data strobes DS0* and DS1* serve a dual function. As edge sensitive signals they qualify data, and as level sensitive signals they select which bytes are accessed.

The VMEbus specification does not use the terms DS0* and DS1* in its *timing diagrams*. Instead it refers to DSA* and DSB*. This notation is used to prevent confusion in cases where bus skew (or other propagation delays) cause one data strobe to fall or rise before the other. DSA* refers to the first data strobe to fall or rise, and DSB* is the second.

2.2.1.13 Trailing Edge Glitches On The Data Strobes

When designing or evaluating VMEbus masters, make sure that data strobes DS0* and DS1* are *driven* high at their trailing (rising) edges. Some VMEbus designers are tempted to negate the data strobes by letting them float high (i.e. by letting the backplane terminators pull them up to the logic high state).

At first glance this would seem to be a reasonable practice. After all, what difference does it make whether the data strobes are driven high, or allowed to float high? The reason for this is quite subtle. Figure 2-5(a) shows a normal data strobe signal, where the master's bus transceiver drives it high before placing it into the high impedance (three-state) condition. Figure 2-5(b) shows an abnormal data strobe signal, where the master lets the signal float high. That figure shows a glitch in the middle of the trailing (rising) edge.



(a) Normal trailing edge on data strobes.

(b) Trailing edge data strobe glitch caused by the master, which lets it 'float' high. The problem is exacerbated by longer backplanes.

Figure 2-5 Trailing edge data strobe glitch

The trailing edge data strobe glitch(es) can cause problems under the following circumstances:

1) When the master releases the data strobe, the signal interconnection is placed into a high impedance state. This makes it more susceptible to interference from other signal sources.

2) Reflections, which are generated by the signal edge at both ends of the backplane, destructively interfere with the signal edge itself. This destructive interference causes the short low-going glitch. [Remember - VMEbus terminators do not eliminate backplane reflections...they just reduce them].

3) The glitch often occurs near the TTL logic transition voltage. This is usually between 1.2 and 1.8 Volts.

4) This condition can cause some slave circuits, which are edge sensitive, to latch data twice during write cycles. That's because the glitch makes two transitions across the transition voltage level. If the magnitude of the excursions exceed the input hysteresis level, it can cause the slave to latch erroneous data. That's because VMEbus timing rules allow masters to change the data lines during the glitch conditions (on write cycles).

5) This problem is virtually non-existent on short backplanes (< 12 slots). However, on longer backplanes the problem is exacerbated, with the worst situation occurring on 21-slot backplanes.

There are no RULES in the ANSI/VITA 1-1994 bus specification requiring the master to drive the data strobes high. Some VMEbus board designers would argue that the slave must be capable of withstanding these short, trailing edge glitches. However, example protocols within the specification describe the master driving the data strobes high. In either case it is recommended by the author that: (a) slaves should be designed to withstand this glitch and (b) masters should be designed to eliminate the generation of the glitch.

2.2.1.14 Cycle Termination Using DTACK*, BERR* Or RETRY* (†)

Slaves terminate all bus cycles by asserting DTACK*, BERR* or RETRY*. DTACK* is the normal way to end the cycle. During read cycles the slave asserts DTACK* after driving the data bus, and during write cycles the slave asserts it after it has latched the data.

BERR* can be asserted by a slave or a bus timer. When it is asserted by a slave it indicates that an error has occurred during the cycle. The VMEbus specification does not say what may have caused the error nor what should be done in response to it. For example, a memory module may assert BERR* in response to a parity error.

The bus timer asserts BERR* if the bus has locked up. Bus lock-ups can be caused either by system failures or by non-responding slaves. One popular use for this feature is for self configuration of the system. During power-up initialization the number of cards can be determined by a master. However, if the master attempts to access a slave that doesn't exist on the backplane, then the bus timer will prevent the system from locking up.

RETRY* can be asserted by a slave if it is busy and doesn't want to be interrupted. It informs the master that the cycle should be attempted again at a later time.

The VME64 specification re-defined the RESERVED pin (P2/J2 pin B3) as RETRY*. This allows bus cycles to be aborted and attempted again at a later time. It is intended for use in multi-ported applications such as bus bridges and dual-ported memories. There are two ways to use the RETRY* signal:

- To prevent deadlock (deadly-embrace) situations.
- To grant maximum bandwidth to a (preferred) port.

Deadlock may occur in systems where two processes are interlinked. Bus bridges are a good example, where two VMEbus racks are tied together through a local bus. In these cases, deadlock may result if both systems attempt to access the other at the same time. If neither end of the bridge has the ability to 'back-off' from a bus cycle, then both can get stuck in a deadly embrace. RETRY* prevents this problem by allowing at least one processor to retry the cycle at a later time.

The RETRY* signal can also be used to grant maximum bus bandwidth to a local resource. For example, in a video *frame-grabber* application all VMEbus accesses to a (shared) video buffer can be suspended until it is filled with data. This 'locks-out' VMEbus and increases the bandwidth available to the local processor.

The VME64 specification also requires that a master must relinquish control of the data transfer bus if it receives the RETRY* signal. This is especially useful in bus-to-bus links, where the RETRY* signal is often asserted during deadlock conditions.

The RETRY* signal pin is fully compliant with backplanes designed under pre-VME64 bus specifications. These required that the RESERVED pin be bussed and terminated. Unfortunately, some backplane manufacturers did not bus and terminate this pin, even though they should have. System integrators should verify that pin P2-B3 is bussed and terminated before specifying or purchasing a backplane.

2.2.1.15 Rescinding DTACK* (†)

The VME64 specification allows DTACK* to be operated as a *rescinding* signal. Under earlier versions of the VMEbus specification, DTACK* was strictly an open-collector class signal. That means that the trailing edge of DTACK* is rather slow, as the backplane terminators must overcome backplane capacitance.

As Figure 2-6 shows, the trailing (rising) edge of DTACK* is much faster when it is operated as a rescinding signal, . This speeds up bus cycles by as much as 10 ns., thereby increasing bandwidth. [RETRY* is also a rescinding signal].

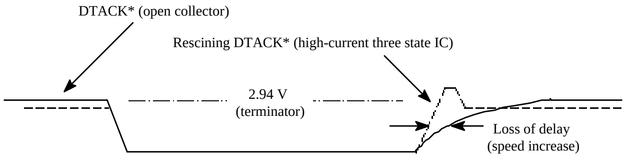
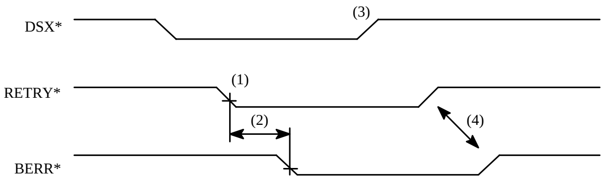


Figure 2-6 Rescinding DTACK*

Originally, DTACK* was classified as an open-collector class signal because it was thought that emulator tools could pace bus cycles by strobing the DTACK* signal. However, this is rarely done because emulation tools of this sort only make sense when CPUs fetch instructions (rather than data) across the VMEbus backplane. Since this is rarely (if ever) done anymore, the VITA Standards Organization considered the speed improvement of the rescinding signal to be more important.

2.2.1.16 RETRY* Interoperability (†)

The RETRY* pin is used in conjunction with the BERR* signal. This permits full backward compatibility with older systems. Figure 2-7 shows how this is done. Whenever a slave asserts RETRY*, it also asserts BERR* a short time later. Under these conditions, newer masters recognize RETRY* and attempt the cycle again at a later time. However, older masters, which don't support the signal, ignore RETRY* and recognize the assertion of BERR*. On most CPU boards the assertion of BERR* causes an on-board exception (software interrupt), and allows the module to handle RETRY* under software control.



(1) Slave asserts RETRY* to request a cycle retry.
(2) 35 ns. setup, BERR* asserted.
(3) Master receives both BERR* and RETRY* low and releases bus.
(4) Slave negates both RETRY* and BERR*.

Figure 2-7 RETRY* termination

The interoperability between older and newer boards is shown in Table 2-8. Any RETRY* cycle that includes a combination of newer and older boards causes the master to acknowledge BERR* and terminate the cycle. When both boards support RETRY*, the BERR* signal is ignored by the master.

Table 2-8 RETRY* interoperability (between new and 'legacy' boards).

If the slave …	And the master …	Then the master …
Doesn't drive RETRY*	Doesn't monitor RETRY*	Only sees BERR* and terminates the cycle with an error routine.
	Monitors RETRY*	
Drives RETRY*	Doesn't monitor RETRY*	Sees both BERR* and RETRY*, terminates cycle and retries later.
	Monitors RETRY*	

On most CPU boards the assertion of BERR* causes an onboard exception (software interrupt), which allows an exception handling routine to handle RETRY*. This is typically quite simple. For example, in 680X0 processors the address of the instruction that was being fetched when the BERR* occurred is passed to the exception handling routine. That instruction can be retried by simply adjusting the stack return address, and performing a return-from-exception instruction (RTE) in the exception processing routine. The instruction that originally caused the BERR* is then re-executed.

It's a good idea to include some kind of random delay in the exception handling routine. For example, the ANSI-'C' random number generator function could be used. This random delay will prevent multiple 'retry collisions' between two or more VMEbus masters.

The VME64 specification does not dictate any procedure for handling BERR* exceptions. Since BERR* exceptions might be generated by other sources (such as a bus timer) the master must somehow determine the source of the BERR* exception. There is no standard way of doing this.

For example, consider two VME64 systems coupled by a bridge that supports the retry cycle. Let's complicate matters by installing processor boards in both systems that don't support the retry cycle. If both of these processors attempt to access the other VME64 system through the bridge at the same time, the bridge will be forced to pick a 'winner' and a 'loser'. It will first issue a RETRY* and a BERR* signal to the loser (to get the loser's VME64 bus). Then it will award the loser's VME64 bus to the winner.

Since the loser doesn't support the retry cycle, it will only see the BERR*. But how does it know whether the BERR* signal was generated by the bridge or a bus timer? The VME64 specification doesn't offer any guidance in this matter.

One solution would be to provide a status bit on the bridge that could be accessed from VMEbus. This would allow the BERR* handling routine to poll the status bit. However, there is a limitation with this approach. Only one VMEbus master in each VME64 chassis would be able to use the bridge.

Suppose that two or more masters in a VMEbus chassis are permitted to use the bridge. Let us further suppose that two events occur almost simultaneously:

- Master A becomes the 'loser' in a bus-to-bus cross-across, causing the bridge to generate a BERR*/RETRY* signal.
- Master B accesses a non-existent memory location, causing the bus timer to issue a BERR* time-out.

Master B might then conduct a poll of the bridge and conclude that it was the loser in a bus-to-bus cross-across.

In situations like this, both masters could simply be configured to assume that all BERR* signals originate with the bridge and re-issue the cycle each time they receive a BERR* signal. Admittedly, this is not a very elegant solution, but it does suffice in many applications.

2.2.1.17 Interlocked Handshaking

VMEbus uses a fully interlocked handshaking mechanism between data strobes DS0* and DS1*, and terminating signals DTACK*, BERR* and RETRY*.

At the beginning of a cycle a master must not assert either data strobe until DTACK*, BERR* and RETRY* have all been negated by the slave. Failing to do so may corrrupt data on the current (or the previous) cycle. At the end of a cycle a slave cannot negate DTACK*, BERR* or RETRY* until the master has negated both data strobes DS0* and DS1*. These two mechanisms form a fully interlocked handshaking mechanism.

Care must be used when evaluating or designing masters that use the 680XX or other microprocessor families. Many microprocessors don't use a fully interlocking bus cycle, and circuitry must be provided to make them totally compliant.

2.2.1.18 Address Pipelining

Some microprocessors or peripherals use *address pipelining* to speed up data transfers. During address pipelining the bus master broadcasts the address of the next bus cycle before the current cycle has completed. As shown in Figures 2-3 and 2-4, immediately after the slave asserts DTACK* the master may negate AS* and place a new address on the bus. With this method, data transfers can be speeded up by overlapping the address broadcast with the previous cycle.

All slaves must function properly in the presence of address pipelining cycles. However, they do not necessarily have to support the speed improvement afforded by them (i.e. they don't have to latch the next address before the end of the current bus cycle).

The removal of a valid address before DTACK*, BERR* or RETRY* are negated is sometimes called *address rot*. Similarly, changing the data lines after DTACK*, BERR* or RETRY* are asserted during a *write cycle* is called *data rot*.

When designing or evaluating slave interfaces, be sure they latch the address and address modifier code before asserting DTACK*, BERR* or RETRY*. Failing to do so may cause the slave to change its data lines before the end of a read cycle (because its on-board address has changed).

Care should be taken when latching the address on the falling edge of AS*. On read cycles, the master can negate and re-assert AS* immediately after a slave asserts DTACK*, BERR* or RETRY*. If the address is latched on the falling edge of AS*, the slave could be presented with a new address before it negates DTACK* or BERR*. This could corrupt the data. On slaves which do not implicitly participate in address pipelining cycles, it is better to latch the addresses on the falling edge of the data strobes. The data strobes can be 'or'ed together and used to latch the addresses.

Address pipelining also reduces the overhead needed to change bus masters (see Chapter 3 - Multiprocessing). During bus arbitration, a new master can assume ownership of the data transfer bus (assert BBSY*) after the current master negates address strobe AS*. Arbitration occurs in parallel to the current master's final data transfer cycle.

When evaluating or designing modules that support block transfer cycles (discussed below), special care should be taken to insure that address pipelining will work. During the block transfer cycle the master may change A01-A31 and LWORD* after the first falling edge of DTACK*, but the address modifier code AM0-AM5 must remain stable until the last falling edge of DTACK*.

Special consideration must also be given to address pipelining after MBLT cycles. For example, during an A64:MBLT cycle the address bus is used to transfer data. The master must keep AS* asserted until DTACK*, BERR* or RETRY* has been negated. This prevents another master from driving the address bus (and the data on it) before the current master has latched the data.

2.2.2 Typical Address (Memory) Map for a CPU

In most VMEbus systems, dynamic address and data sizing can be made relatively transparent to software. This is usually handled through the CPU memory mapping.

With this technique, the various VMEbus address and data port sizes are mapped to individual locations throughout a CPU's memory space. For example, a typical address map for a 32-bit CPU module is shown Figure 2-8.

In this address map there are two address spaces called *CPU* and *logical VMEbus*. The CPU space shows the addresses as seen by CPU software. The logical VMEbus space shows the addresses as seen by the VMEbus address lines.

In this example, the VMEbus A16 and A24 address spaces are relatively small, and are given their own permanent regions in CPU space. A portion of the A32 space is assigned to the middle of the map. Local I/O and memory are delegated to the bottom of the map.

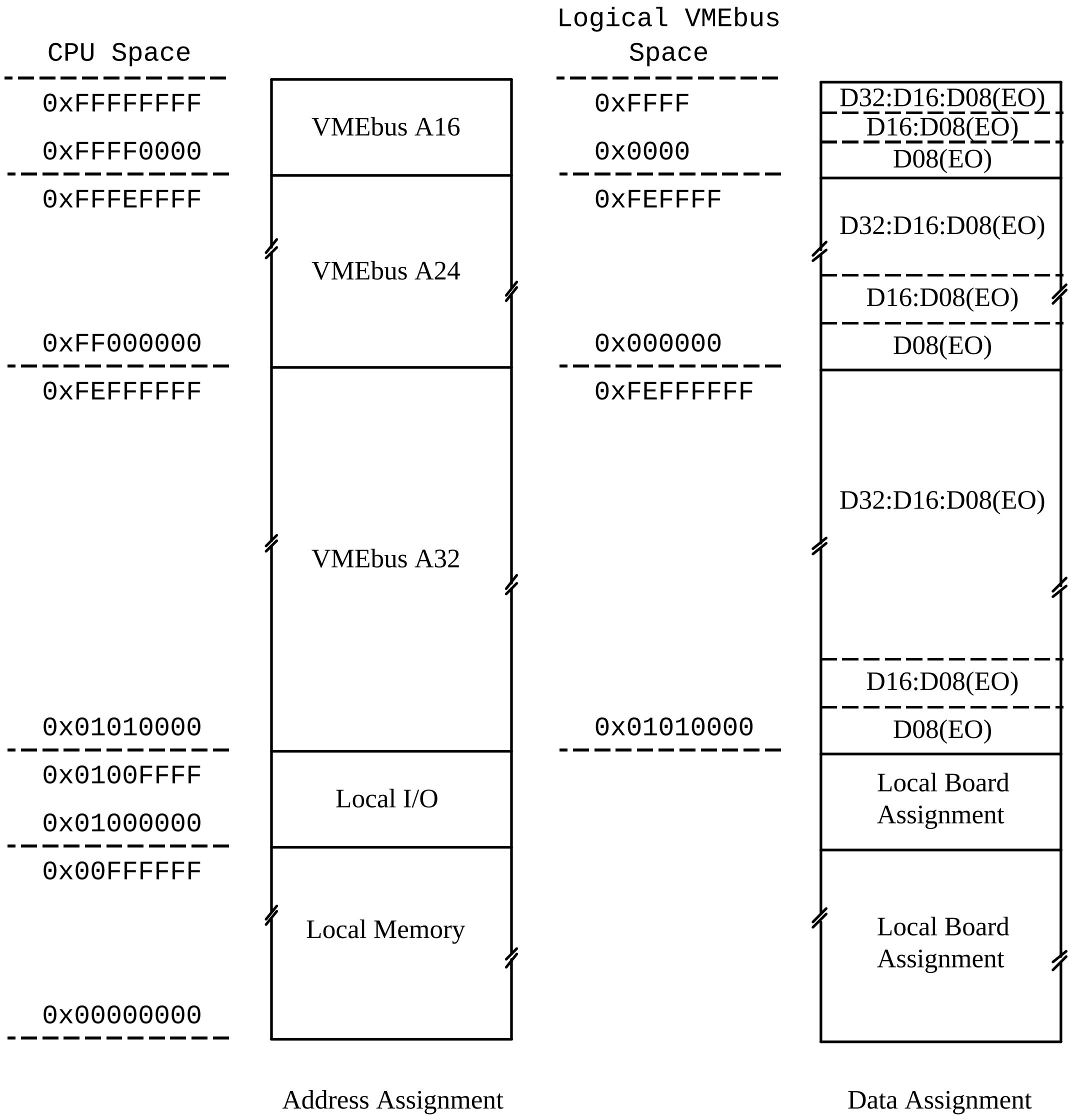


Figure 2-8 Typical address map for a CPU

For example, let's assume that the CPU wants to access an A24 VMEbus module at address 0x000000. To do that, the CPU would actually access its local space at 0xFF000000. CPU hardware would then ignore the upper eight bits of the address, and place 0x000000 onto VMEbus. Since the CPU bus interface hardware is aware of the address map, it would automatically place the correct (A24) address modifier code onto the bus.

In an alternative approach to address decoding (which is not shown in Figure 2-8), the CPU would use a *windowing* technique. This method uses a *mode bit* register so that the CPU can select the upper eight bits (or so) of the VMEbus address. For example, some IBM-PC XT/AT compatible VMEbus modules use this technique. Those modules use 80X86 CPUs, have segmented bus architectures, and map each segment to the mode register.

The advantage of the mode bit register method is simplicity of design. The disadvantage is that system software needs to continuously update the mode register (a very cumbersome and time consuming task).

Dynamic data sizing is usually handled within the address map as well. Figure 2-8 also shows how this is done. The various data port sizes are mapped within each VMEbus address region. For example, the A24 region would contain sub-regions which support D08(EO), D16 and D32 type transfers. Hardware would then automatically route data to the appropriate portion of the data bus.

For example, a CPU module may configure its bus interface as an A24:D16:D08(EO) master during CPU accesses between 0xFF008000 and 0xFF00AFFE, and as an A24:D32D16:D08(EO) master between 0xFF00B000 and 0xFF00FFFC.

Selection of a data port size may also be accomplished with a mode bit register. In this case the CPU sets a bit indicating the type of required access. For example, the bit could be set when the master generates a D32 cycle, and cleared for a D16.

The VMEbus specification does not provide slaves with the ability to acknowledge data port size during a transfer. Some popular microprocessors (like the 68020/30/etc.) require slaves to do so. This requires the use of memory mapping or mode register techniques to sort out the data size.

In actual VMEbus CPUs, the address decoding scheme is much more complicated than that shown in Figure 2-8. Intermediate buses, such as PCIbus, make things even more complex, as they have their own addressing schemes.

2.2.3 Read-Modify-Write Cycle

The read-modify-write cycle is used in multiprocessor and multitasking systems. This special cycle allows many software processes to share common resources by using semaphores. This is commonly done on boards such as disk controllers, serial ports and memory. As the name implies, the read-modify-write cycle reads and writes data to a memory location in a single bus cycle. It prevents the allocation of a common resource to two or more processes. The read-modify-write cycle is sometimes called an *indivisible cycle* or a *test-and-set cycle*.

One possible application for the read-modify-write cycle is an airline ticket reservation system. Consider two people (at different locations) reserving seats on a flight from New York to London. Unless specific care is taken, it is possible for both passengers to be allocated the same seat (if both reservations are made at the same time). The read-modify-write cycle can be used to prevent this situation.

For example, assume the ticket reservation software is set up so that each seat on the flight is represented by a bit in memory. If the bit is zero, the seat is empty. If the bit is one, then the seat is full. To reserve a seat, the ticketing software first reads the bit to see if the seat is occupied. If the bit is zero the software sets it to a one. If it the bit is set (the seat is already occupied), then the software looks for another seat.

The following software is an example of problematic 680XX assembly code that allows two seats to be allocated to the same passenger. This software does not generate a read-modify-write cycle:

```
*
* A0 == LOCATION OF MEMORY BIT REPRESENTING
SEAT
*
          BTST.B #7,(A0)   * IS THE SEAT TAKEN?
          BEQ GETSEAT      * BRANCH IF SO
          •
          •                * LOOK FOR ANOTHER
SEAT
          •
GETSEAT: BSET.B #7,(A0)    * RESERVE THE SEAT
```

The problem occurs between the time the bit is tested (BTST.B) and set (BSET.B). During this interval several instructions are performed (such as BEQ). In a multiprocessing system, a bus arbitration could take place between the time the bit is checked and set. If another processor is running the same code, at the same time, both could get the same seat because they both read the same bit as zero. The outcome would be two passengers booked on the same seat...an embarrassing problem for the airline.

This problem can be solved with a read-modify-write cycle. In the 680XX family, a CPU can generate a read-modify-write cycle using the TAS (test-and-set) instruction. This instruction reads a byte, tests the condition of bit #7, sets it to a one, and writes it back to memory. Rewriting the previous program using the TAS instruction gives the following code:

```
  *
  * A0 == LOCATION OF MEMORY BIT REPRESENTING
SEAT
  *
      TAS (A0)            * TEST BIT AND SET IT
      BEQ GETSEAT         * BRANCH IF AVAILABLE
      •
      •                   * ADDITIONAL CODE
      •
  GETSEAT:                * . . . AND CONTINUE
```

The airline reservation example is simplistic, but it does illustrate the use of the read-modify-write cycle. Many multiprocessing systems must arbitrate for system resources such as disk controllers or network cards.

The functions of the read-modify-write cycle can also be duplicated in software. For example, many real-time operating systems provide resource managment functions that accomplish the same results. In these systems a message is passed to an allocation function, which grants or denies a resource.

Not all VMEbus masters or slaves can participate in read-modify-write cycles. Modules that generate or accept the cycle are said to be *RMW* compatible. All modules must be able to tolerate RMW cycles, however. When evaluating or designing bus modules, look at the software requirements to find out if modules need to be RMW compatible.

The read-modify-write cycle is shown in the timing diagram of Figure 2-9. During the cycle, back-to-back read and write cycles are performed while AS* remains asserted. In the first half of the cycle WRITE* is negated, and data is read from memory. The master modifies the data, asserts WRITE*, and puts it back. Keeping AS* asserted prevents bus arbitration in the middle of the cycle. See Chapter 3 for more information on bus arbitration.

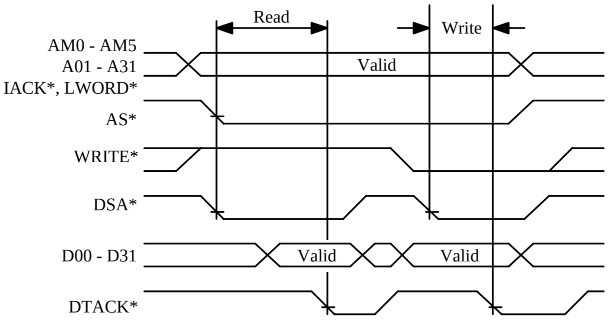


Figure 2-9 The read-modify-write cycle

Note that AS* remains

asserted during the entire cycle.

When evaluating or designing VMEbus masters capable of read-modify-write cycles, make sure they do not change their address lines during the cycle. This can cause problems on processors such as the 68020 which utilizes a special RMC (Read-Modify-Control) pin. Under certain conditions the 68020 can change its address lines during the read-modify-write cycle. One possible solution is to latch the microprocessor address lines before the start of every cycle.

A common design problem on read-modify-write slaves is to use AS* to drive the DTACK* generator. Once the slave has been selected, use data strobes DS0* and DS1* to assert and negate DTACK*. If this is not done, the module could lock up the bus during a RMW cycle.

The address-only cycle allows a master's local memory address decoder to work in parallel with a slave's, and can speed up the bus in some cases. It can also simplify the design of some masters. For example, a 68010 microprocessor with 68451 MMU may terminate a bus cycle after the 68451 asserts AS* (the MMU aborts the external cycle). The bus interface design is simplified if these cycles are allowed to 'leak' onto VMEbus.

2.2.4 Address-Only Cycle

The address-only cycle is used to broadcast an address. Actually, it is a do-nothing cycle because data is not transferred during the cycle. This cycle is differentiated from read/write cycles because masters do not assert either data strobe. Since neither data strobe is asserted, the slave does not terminate the cycle with DTACK*, BERR* or RETRY*. Figure 2-10 shows the address-only cycle.

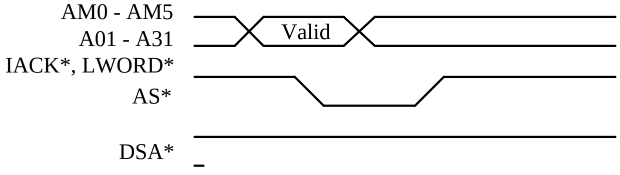


Figure 2-10 Address-only cycle

The ADO mnemonic describes modules that can initiate or tolerate address-only cycles. When evaluating or designing slave modules, make sure they tolerate ADO cycles. The ADO cycle was first permitted under the IEEE-1014-1987 version of the VMEbus specification. Modules designed under earlier specifications are not required to support the cycle.

2.2.5 Block Transfer Cycle

The block transfer cycle moves blocks of data at high speed across the bus. The block transfer cycle is sometimes called a *burst cycle* or a *burst mode*.

When software message passing schemes are used, the messages are often assembled on a CPU board and moved across the bus as a block. The block transfer cycle can streamline this technique. Peripherals, such as disk controllers and network interfaces, also move data in a block mode fashion.

The block transfer cycle is faster than read/write cycles. That's because the master presents an address only once during the cycle. The extra overhead of computing and changing addresses does not take place. The timing diagram for a block transfer cycle is shown in Figure 2-11.

During the cycle both an address and a block transfer address modifier are presented by the master. When a slave is addressed, multiple bytes of data can be read or written (in ascending order) by toggling the data strobes. Block transfer counters on the slave automatically increment their local addresses. This relieves the master from computing and changing the address during every cycle phase.

The slave may not need a local address counter if it contains a FIFO (First In First Out) interface. In this case the first address of the BLT cycle should address the FIFO. All subsequent bus cycles would then move data in or out of the FIFO.

The ability to generate or accept a block transfer cycle is optional. To prevent incompatibility between modules, a mnemonic called *BLT* is used. If a master can generate a block transfer cycle it is called a BLT master. If a slave can accept a block transfer cycle it is called a BLT slave. When integrating VMEbus systems, make sure that compatible modules are used.

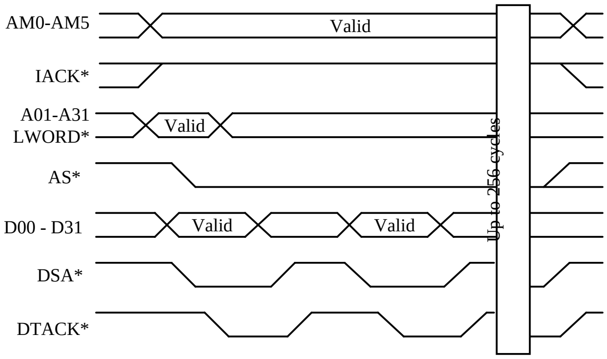


Figure 2-11 Block transfer cycle (write)

To reduce the complexity of block transfer slaves, a rule was introduced in the IEEE 1014-1987 VMEbus specification which forbids block transfers from crossing 256 byte boundaries. This provision solved several problems that existed with the block transfer cycle. It prevented board-to-board crossings during a cycle, it reduced the address counter requirements to eight bits, and allowed the use of commonly available nibble or page mode memories (many standard memories require that block transfers don't cross 256 byte boundaries). If it must cross the boundary the master can stop the cycle, re-submit a new address, and do another block transfer.

The 256 byte rule also prevents a master from hogging the bus with very large block transfers. Since VMEbus arbitration cannot take place with AS* asserted (see Chapter 3) bus ownership cannot change during a block transfer cycle.

2.2.6 MBLT Cycle (†)

The MBLT (Multiplexed Block Transfer) cycle permits 64-bit data transfers. It can only be used with 6U modules. That's because 6U boards have two connectors (P1/J1 and P2/J2). On standard VMEbus modules both connectors are needed to form 32-bit address and data paths. In MBLT cycles theses two paths are combined to form one giant 64-bit path.

3U bus modules only have a single P1/J1 connector, which is insufficient for the wide data paths. Another bus cycle, called the MD32, is customized for 3U boards and will be described shortly.

The MBLT cycle is new in the ANSI/VITA 1-1994 (VME64) bus specification.

The MBLT cycle is shown in Figure 2-12. In that cycle, the first bus transaction (or phase, as some people call it) transfers a valid address across both the address and data lines. The thirty-two bit address and data paths are combined to form one giant sixty-four bit address path.

The VME64 standard permits three addressing options during the first phase of the MBLT cycle. These include twenty-four, thirty-two and sixty-four bit addressing, and are described by the A24, A32 and A64 mnemonics respectively. Each MBLT addressing mode has its own address modifer code.

The second and subsequent phases of the MBLT cycle transfer data 64-bits at a time. Again, the address and data lines are combined to form one giant sixty-four bit data path. Smaller data widths (8, 16, 24 and 32-bits) are not supported by the MBLT cycle.

The slave participating in the MBLT cycle latches the address during the first phase. The addresses of subsequent phases are re-created with counters on the slave itself.

Also note that the LWORD* pin transfers a data bit during each data phase of the cycle. That's because there is no address line A00 on VMEbus. [Stated another way, the LWORD* pin takes the place of bit D32 during the data phase].

Figures 2-13 shows a comparison between the A32:MBLT and the A64:MBLT cycles. Figure 2-14 shows a comparison between the A32:D32BLT and the A32:MBLT cycles.

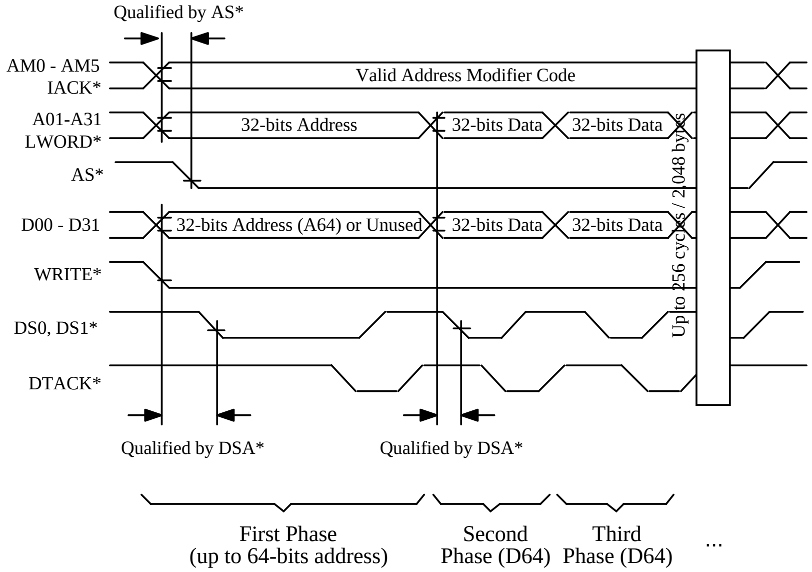


Figure 2-12 MBLT cycle

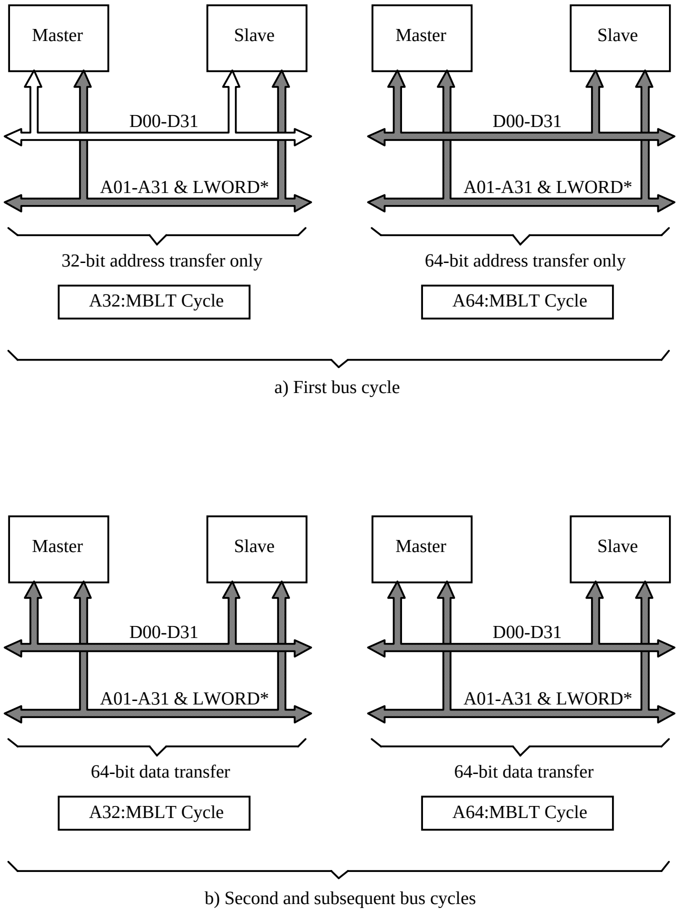
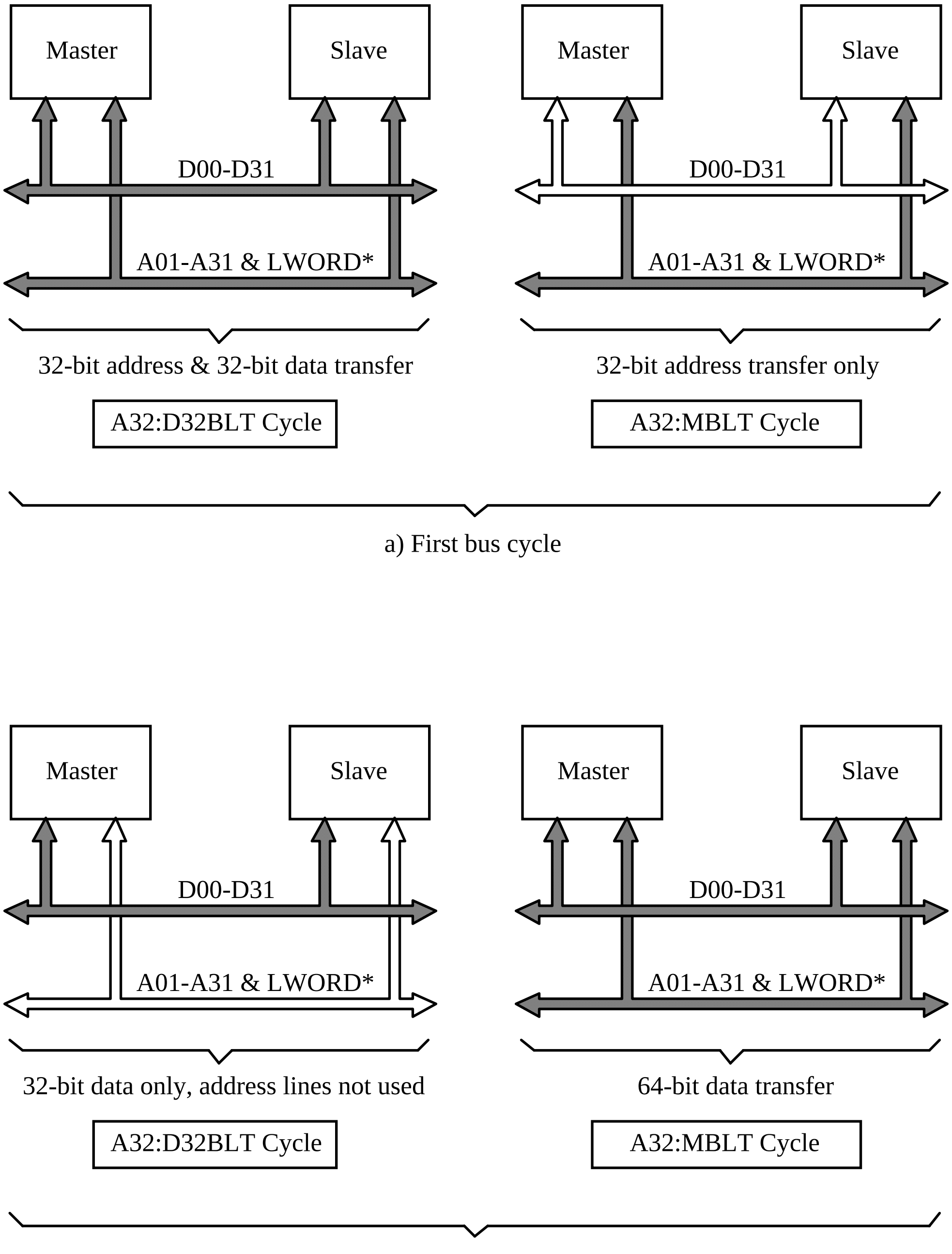


Figure 2-13 Comparison between A32:MBLT and A64:MBLT cycles



a) First bus cycle

b) Second and subsequent bus cycles

Figure 2-14 Comparison between A32:D32BLT and A32:MBLT cycles

The master indicates the end of an MBLT cycle by negating AS*. Any number of phases can be used during the cycle. However, the VME64 standard prohibits data transfers from crossing any 2,048 byte boundary. The reason for this restriction is two-fold:

- Masters are prevented from 'hogging' the bus.
- Address counter designs are simplified.

All MBLT data transfers must take place on even, eight-byte boundaries (address 0x00, 0x08, 0x10 etc.). This simplifies the interface design.

2.2.7 MD32 Cycle (†)

The MD32 (Multiplexed D32) block transfer cycle can be used with either 3U or 6U bus modules. It permits 40-bit address and 32-bit data transfers over the P1/J1 connector. This effectively doubles the bandwidth of 3U boards by doubling the data path from sixteen to thirty-two bits.

The MD32 cycle is new in the ANSI/VITA 1-1994 (VME64) bus specification.

Although there is no specific rule for doing so, it is wise for 6U board designers to support both the MBLT and MD32 cycles. That's because 3U and 6U boards are often mixed in 6U sub-racks. [3U boards can easily be placed in a 6U chassis by using a longer front panel]. This makes the boards more flexible, and allows them to be used in more applications.

The MD32 cycle is shown in Figure 2-15. It is very similar to the MBLT cycle. However, during the MD32 cycle the first phase transfers a valid forty-bit address across the twenty-four address and sixteen data lines. Since these connector pins are all available on the P1/J1 connector, the MD32 cycle can be used with 3U bus modules.

The only address size supported by the MD32 cycle is A40. The A16 and A32 addressing modes are not supported. The MBLT cycle uses address modifier code 0x37.

The second and subsequent phases of the MD32 cycle transfer data thirty-two bits at a time. The address and data lines are combined to form a thirty-two bit data path. Smaller data widths are not supported by the cycle.

The VME64 standard prohibits MD32 data transfers from crossing any 256-byte boundary. Furthermore, all transfers must take place on even, quad byte boundaries (address 0x00, 0x04, 0x08 etc.).

2.2.8 2eVME Cycle (††)

The 2eVME (two-edge VMEbus) cycle was introduced in the VME64x bus specification. It is the fastest protocol on VMEbus. It permits data transfers at rates of up to 160 Mbyte/second, which is twice the speed of the MBLT cycle. This remarkable increase was achieved by using three techniques:

- The transaction time for each data transfer is cut in half.
- Use of multiplexed block transfers (similar to MBLT cycles).
- Higher current ETL bus transceivers.

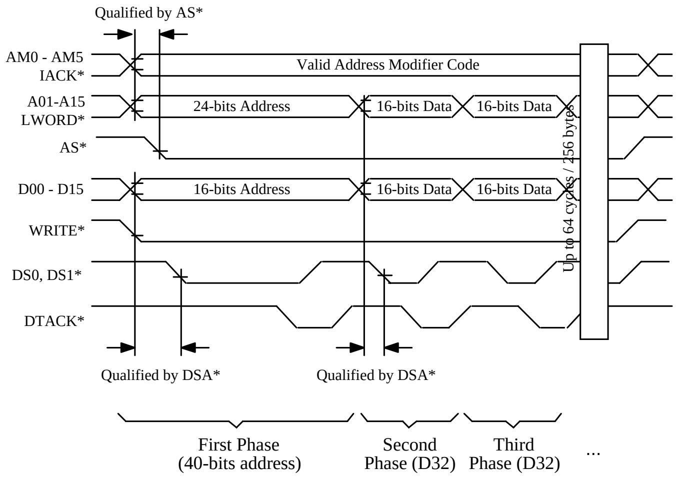


Figure 2-15 MD32 cycle

In all other VMEbus cycles, each data transfer requires a total of four signal edges: two data strobe edges and two DTACK* edges. This requires a minimum bus cycle timing of about 100 nanoseconds.

The 2eVME protocol requires only two signal edges: one data strobe edge and one DTACK* edge. This reduces the minimum bus cycle timing to about 50 nanoseconds.

Figures 2-16 and 2-17 show how the 2eVME cycle works. During the first portion of the cycle the master module broadcasts a valid address modifier code of 0x20 or 0x21. Address modifier code 0x20 is for 6U bus modules, and address modifier 0x21 is for 3U bus modules. The master then broadcasts a valid device address. The slave module at the indicated address then responds by asserting a falling edge on DTACK*.

2eVME cycles multiplex the thirty-two bit address and data busses. Theses are combined to form one giant sixty-four bit path. This is very similar to the method used in MBLT cycles.

A total of three, 64-bit address transfers are made by the master. These are called ‘phases’ in the VME64x specification. As shown in Table 2-9, the first phase (AP1) contains a device address and an extended address modifier code (XAM). The device address is straightforward, and is a simple binary address to be decoded by the slave. However, the extended address modifier is an eight bit code which determines if the device address is 32 or 64 bits wide.

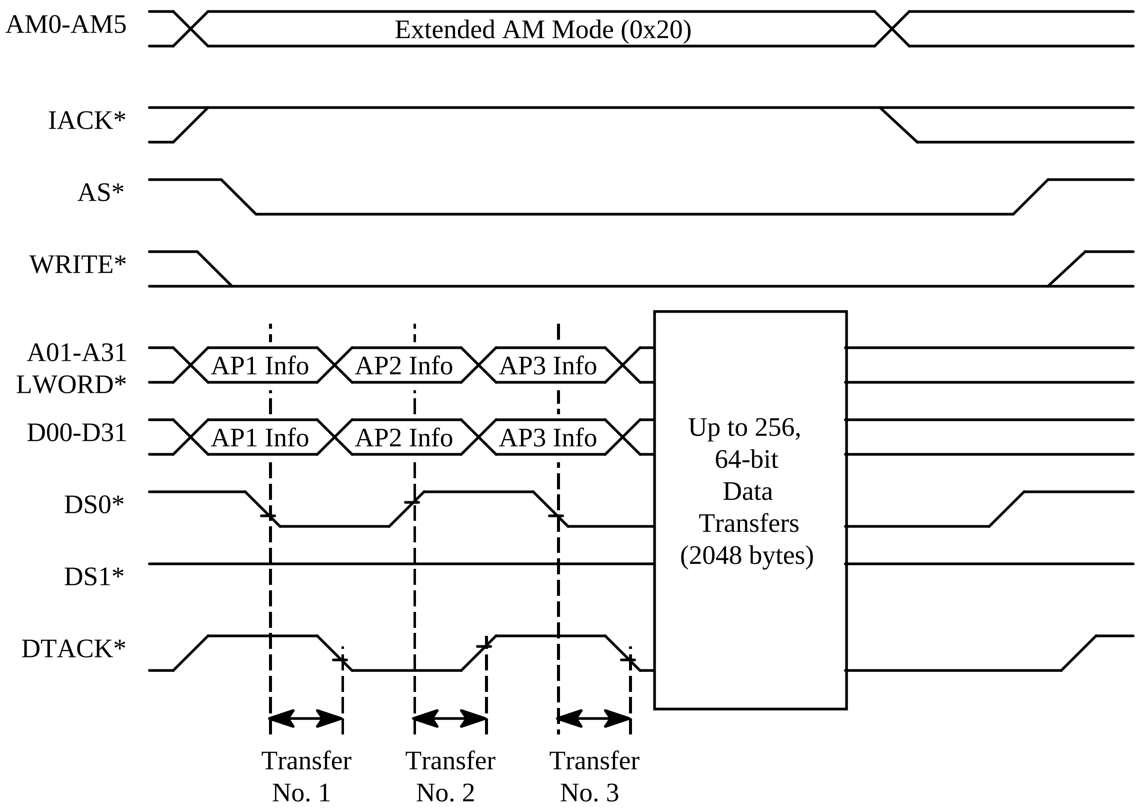


Figure 2-16 2eVME address broadcast phase for a 6U module

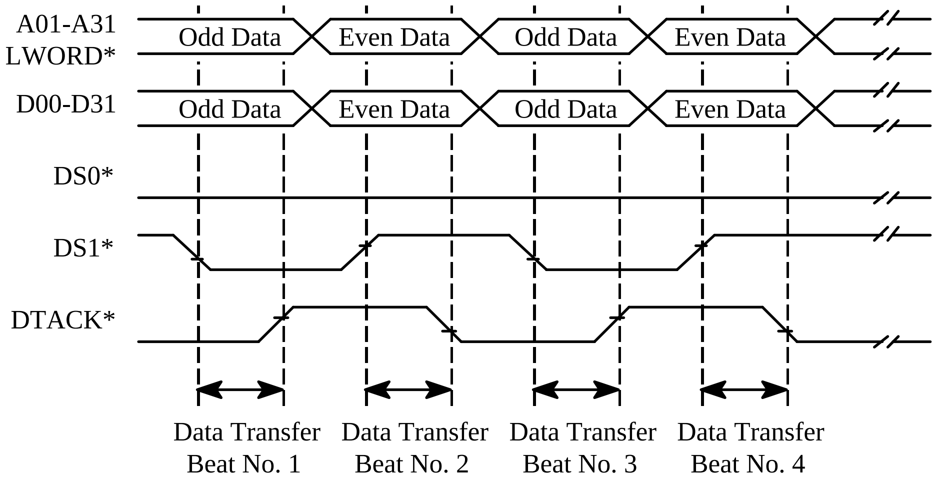


Figure 2-17 2eVME write data cycles for a 6U module

Table 2-9 2eVME address phase encoding for a 6U module

2eVME Address Phase Encoding for 6U Modules					
	D[31..0]	A[31..24]	A[23..16]	A[15:08]	A[07..01] LWORD*
Address Phase 1 (AP1)	Device Address A[63:32] (A64)	Device Address A[32..24]	Device Address A[23..16]	Device Address A[15..8]	XAM Code
Address Phase 2 (AP2)	Reserved	Subunit number	0000GAMA	Beat Count	Device Address A[07..00]
Address Phase 3 (AP3)	Reserved	Reserved	Reserved	Reserved	Reserved
Data Phase	D[31..0]	D[63..56]	D[55..48]	D[47..40]	D[39..32]

The extended address modifier was included in the 2eVME cycle because of a shortage of standard address modifier codes. The following XAM codes are available for 6U modules:

XAM Code	Function
0x00	Reserved
0x01	A32/D64 2eVME Transfer
0x02	A64/D64 2eVME Transfer
0x03 - 0xFF	Reserved

The slave participating in the 2eVME cycle latches the address during the first phase. This is later used by the slave as a base address for an on-board memory access counter. As shown in Table 2-9, the lower eight bits of the device address are not available to the slave at this time.

The second address phase (AP2) contains a subunit number, the geographical address of the master, a maximum beat count and the lower eight bits of the device address. All of this information, with the exception of the device address, is optional.

The third address phase (AP3) is unused and is reserved for future use.

Also note that the address phases are transferred using both edges of data strobe DS0*. The first phase (AP1) is transferred during the first falling edge of DS0*. The slave, which is selected by the address in AP1, responds with a falling edge on DTACK*. When the master monitors that DTACK* has been asserted by the slave, it responds by putting out the next address phase AP2. This procedure is repeated until all three address phases have been transferred.

Once the 2eVME addressing phase is complete, the master can begin transferring data to the slave. This is done with a series of data 'beats'. Beats are always transferred in pairs so that the data strobe and DTACK* are always returned to their logic high levels.

Also note in Figures 2-16 and 2-17 that the address phase is transferred using edges on DS0*, and that the data beats are transferred by edges on DS1*. Also, DS0* is held low during the complete data phase. This means that some 2eVME transfers may take 20 - 30 microseconds to complete. Users should be aware of this fact and set their VME bus timers to at least 32 microseconds (or more).

2eVME transfers can be terminated at any time by the master or the slave. When the master terminates a block transfer, it does so by negating the DS0* data strobe at the end of a data beat, and then terminating the cycle. When the slave suspends the transfer, it does so in one of two ways:

- 6U slaves indicate termination by asserting RETRY* and then asserting BERR*.
- 3U slaves indicate termination by asserting RESP* and then asserting BERR*.

The master, upon seeing one of these two conditions, terminates the transfer. Also note that RESP* is a new signal on the 160 pin P1/J1 connector.

2eVME bus transactions must be made with a new class of TTL bus transceivers. These must drive at least 64 mA of current on all address, data and control lines. Furthermore, the receivers of these signals must conform to a new 1.5 V +/- 0.1 V receiving threshold standard. This speeds up the transactions to their maximum 50 nanosecond cycle times in two ways:

- The 64 mA drivers increase the speed of signal transitions. This also insures monatomic rising and falling bus signals.
- The 1.5 V +/- 0.1 V receiving thresholds increase the high and low level steady-state noise margins.

This new class of TTL bus transceivers are already available on the market. For example, Texas Instruments makes a set of ETL (enhanced transceiver logic) bus transceivers just for VME64x applications. This new logic family is compatible with standard TTL, has a 64 mA output current drive, and reduces the TTL transition region to 1.5 V +/- 0.1 V and achieves incident wave switching. The ETL logic chips also support pin pre-charging, which is required in VME64x hot swappable modules.

2.2.9 ADOH Cycle and the LOCK Command (†)

ADOH cycles must be used to issue LOCK commands. LOCK commands are used to 'lock out' other ports in a multiported resource. For example, issuing a LOCK command to a memory card that has both VMEbus and VSBbus interfaces would cause the card to lock out accesses from the VSB side.

Stated another way, the LOCK command can be used to access a peripheral at its maximum bandwidth, without interference from another port.

The LOCK command itself is embedded in the ADOH (Address Only with Handshake) cycle. Bus masters, wishing exclusive access to a shared resource, generate the LOCK command. The addressed slave then locks out all ports except for VMEbus.

The ADOH cycle and the LOCK commands are new in the ANSI/VITA 1-1994 (VME64) bus specification.

The ADOH cycle has the same protocol as the address phase of the MBLT cycle (discussed earlier). A16:LCK, A24:LCK, A32:LCK, A40:LCK and A64:LCK modes are supported. Each command has its own unique address modifier code.

For example, consider a system where a 'frame-grabber' card transfers video data to a memory card over VMEbus. If the memory card is dual-ported with VSBbus, the LOCK command will cause the memory card to ignore VSBbus cycles. This gives the frame grabber full access to memory at its maximum bandwidth (without interference from VSBbus).

Ports are locked for the duration of bus tenure. Stated another way, when a master releases the bus, the port is 'unlocked'. This limits the amount of time a port may remain locked, and prevents excessive overhead which would happen if, say, an 'UNLOCK' command were required.

Users should take special care when using LOCK commands, as they must be coordinated with VMEbus arbitration. The VMEbus specification has many bus arbitration options, and some may cause conflicts. For example, the RELEASE-ON-REQUEST (ROR) arbitration scheme (see chapter 3) causes a master to release the bus whenever another requests it. If the current master is in the midst of a series of locked bus cycles, the master would release the bus and unlock the slave.

It is best to use other arbitration options (such as RELEASE-WHEN-DONE) with the LOCK command. These give the master (who is doing the locking) full control of bus arbitration during locked cycles. [See Chapter 3 for more information on bus arbitration].

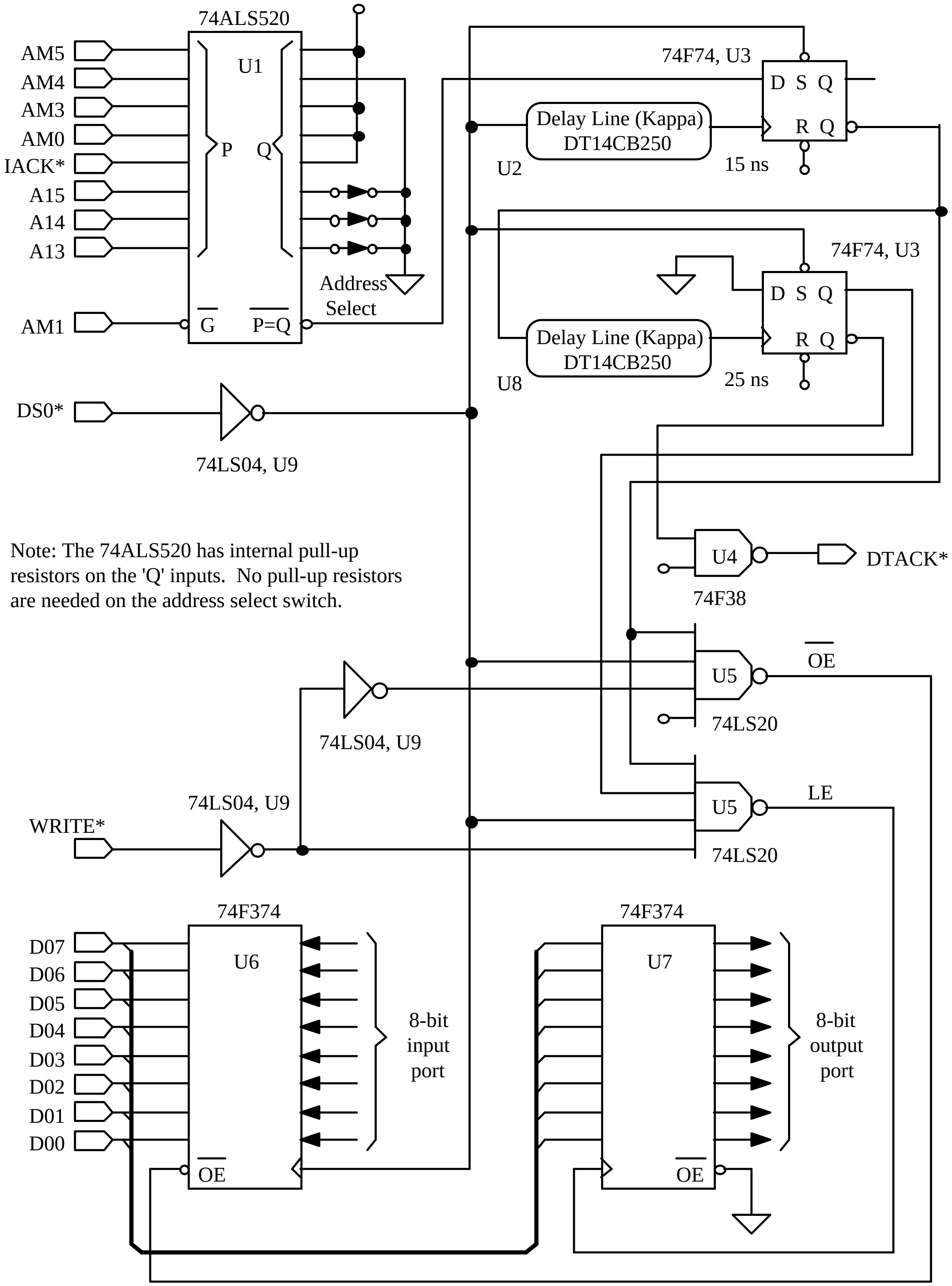
2.3 Circuit Example - Simple 8-bit Parallel I/O Module

While most boards in a system can be purchased through established vendors, the need often arises for at least one custom VMEbus module. These are usually specialized I/O modules that are customized to the application. Often the resources dedicated to these custom projects are larger than those given to the rest of the system. Several circuits are presented here to aide the user in understanding and designing simple VMEbus interface circuits. All *circuit examples* presented in this book have been built and tested. [This is opposed to *circuit ideas*, which have *not* been tested by the author].

Figure 2-18 shows a simple circuit for an 8-bit parallel I/O module with an A16D08(O) interface. This module illustrates some basic slave interfacing concepts including address decoding, bus timing and use of the control signals. It can be used as a building block for real time clocks, A/D converters, D/A converters and other simple I/O functions.

A write cycle to this board causes U7 to latch and output one byte of data. A read cycle returns data present at the input of U6. The address modifiers and address lines are decoded with an 8-bit magnitude comparator U1. Since the board is an A16 slave, address modifiers 0x29 and 0x2D are decoded. Also note that U1 does not monitor AM2 because it's a *don't care* bit:

AM	5	4	3	2	1	0	
0x29	1	0	1	0	0	1	A16 non-privileged access
0x2D	1	0	1	1	0	1	A16 supervisory access



Note: The 74ALS520 has internal pull-up resistors on the 'Q' inputs. No pull-up resistors are needed on the address select switch.

Figure 2-18 Simple A16:D08(O) slave

A16 slaves only need to decode address bits A01-A15, and A16-A31 can be ignored. By decoding only the upper three address bits A13-A15 the circuit can be built using one 8-bit comparator IC. The only drawback to this is the 8 Kbyte address space required by the module. Since the A16 address space is usually reserved for simple I/O devices, large (unused) holes in memory can be tolerated.

The P=Q output of U1 is latched using flip-flop U3. That's because the state of the address lines on VMEbus slaves must be latched at the beginning of every cycle. To understand why, consider the timing diagram of Figure 2-19. The timing rules of the VMEbus specification require that slaves must not assert DTACK* until after the data strobe is asserted. Once the slave asserts DTACK*, however, the master may immediately negate AS* and change the address lines. If the address lines are not latched, the module could address could be changed in the middle of the cycle. If this happens during a read cycle, the module could get confused and return data from a different address at the end of the cycle (and corrupt the data latched by a master).

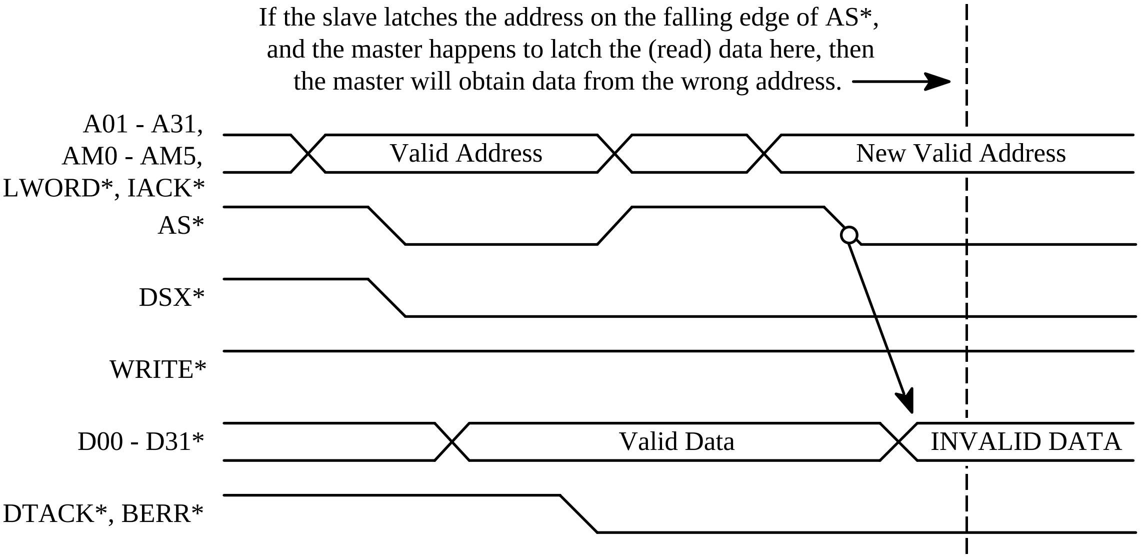

If the slave latches the address on the falling edge of AS*,
and the master happens to latch the (read) data here, then
the master will obtain data from the wrong address.
A01 - A31,
AM0 - AM5,
LWORD*, IACK*
Valid Address
New Valid Address
AS*
DSX*
WRITE*
D00 - D31*
Valid Data
INVALID DATA
DTACK*, BERR*

Figure 2-19 Bus timing showing address rot, and why slaves should latch address lines

The 8-bit I/O module uses DS0* to latch the addresses instead of AS*. If a slave does not support address pipelining (although it must always *tolerate* those cycles), it may latch its address lines using one of the data strobes. In this case using the falling edge of DS0* instead of AS* simplifies the design of the module.

The master may negate and re-assert AS* immediately after DTACK* is asserted by the slave, as Figure 2-19 shows. This is also known as address pipelining or address rot. If the address is latched on the falling edge of AS*, and a second falling edge occurs before the end of the cycle, it could cause the module to return mis-addressed data during a read cycle.

Since the 8-bit I/O module is a D08(O) slave, data is written and read over D00-D07 (see Table 2-6). This means that the module can be accessed one byte at a time at odd addresses, and that only DS0* need be used. By strict interpretation of the VMEbus specification this circuit could actually be classified as an A16:*D32:D16:D08(O)* slave because it does not decode DS1* or LWORD*, and responds to D32 and D16 cycles. If the board is presented with D32 or D16 cycles it will still respond but will ignore the information on all the data lines except D00-D07. During read cycles the backplane termination networks pull all the data lines up, and during write cycles the unused data lines are ignored.

Input data is read from the 74F374 octal flip-flop (U6). This circuit samples input data at every falling edge on DS0*. Any metastable glitches at the output of U6 will damp out before the master latches the data (during the time delay created by U2 and U8). Data is held at the output of the latch until the master negates data strobe DS0*.

A write cycle is similar to the read cycle except that data is latched using 74F374 octal flip-flop (U7). Data is latched before the module asserts DTACK* because the master is permitted to change the data lines immediately after the slave asserts DTACK*.

When evaluating or designing VMEbus modules, the driver and receiver characteristics of the interface ICs must be closely checked. For example, the circuit of Figure 2-18 asserts DTACK* with a 74F38 open-collector driver. The data lines, however, are driven with a 74F374 which has standard 3-state outputs. For more information on interface characteristics see Chapter 6.

In this example the VMEbus reset line [SYSRESET*] is not used. During system reset this module is required to negate DTACK*, and stop driving data lines D00-D07 within 30 microseconds after SYSRESET* is asserted. Since masters are required to negate the data strobes after only 20 microseconds, and this circuit resets itself every time the data strobes are negated, the monitoring of SYSRESET* is useless. For more information on system reset refer to Chapter 5.

Also note that this circuit example (and others in the book) use delay lines. This is a common technique on VMEbus modules when precision time delays are needed. Here, U2 produces an input-to-output delay of 15 nanoseconds, and U8 produces a delay of 25 nanoseconds. There are many companies that make these devices, one is:

Data Delay Devices
3 Mt. Prospect Avenue
Clifton, NJ USA 07013
TEL: (973) 773-2299
www.datadelay.com

2.4 Circuit Example - Simple 16-bit Memory Module

A 16-bit, 16 Kbyte memory module is shown in Figure 2-20. This circuit has an A24:D16:D08(EO) interface. Address decoding is similar to that of the 8-bit parallel I/O module except that it responds to A24 addresses.

The circuit responds to the following four address modifier codes:

AM	5	4	3	2	1	0	Description
0x3E	1	1	1	1	1	0	A24 supervisory program access
0x3D	1	1	1	1	0	1	A24 supervisory data access
0x3A	1	1	1	0	1	0	A24 non-privileged program access
0x39	1	1	1	0	0	1	A24 non-Privileged data access

The four address modifier codes are decoded with a programmable logic device (U6). The logic equations for this device are shown in Figure 2-21. U6 asserts /STAM if IACK* and LWORD* are both negated, and an A24 address modifier is detected. Flip-flop U4 then latches /STAM after the falling edge of either data strobe (signal CYC).

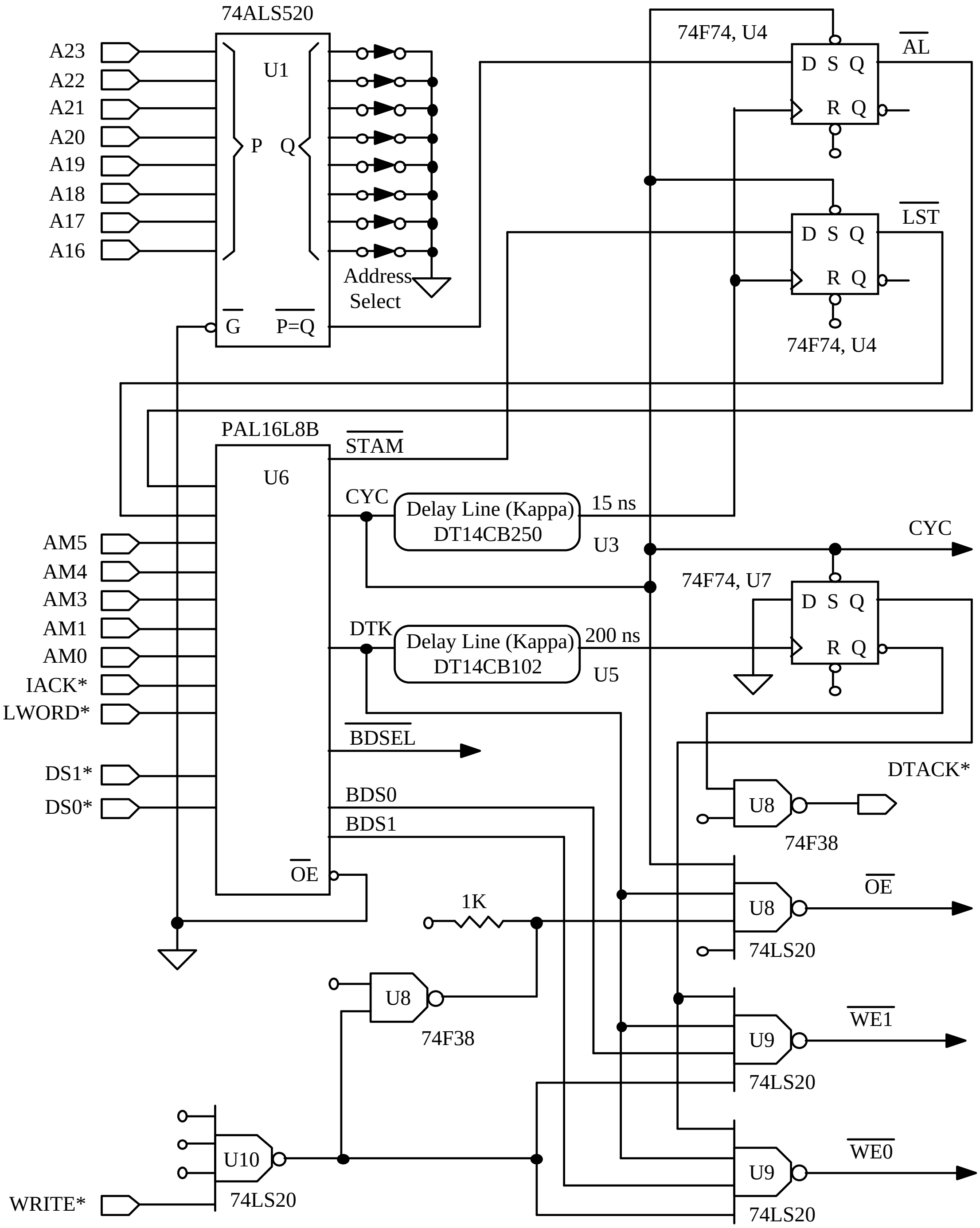
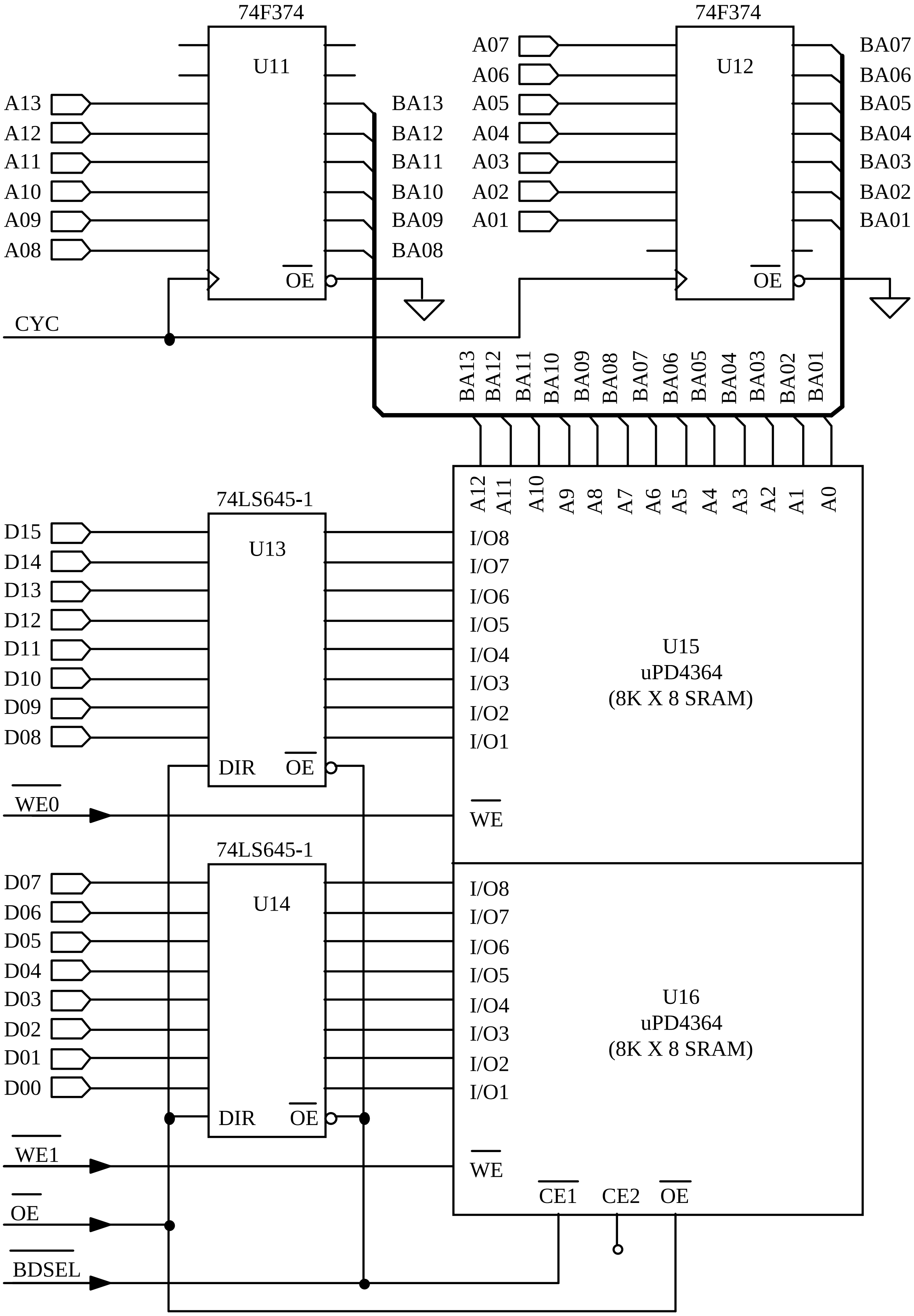


Figure 2-20 16-bit memory module with A24:D16:D08(EO) interface



TITLE ADDRESS DECODER CONTROL PATTERN VH0004.PDS

```
REVISION       A
AUTHOR         WADE PETERSON
COMPANY        (C) 1988 WADE PETERSON. ALL RIGHTS
RESERVED
DATE           JANUARY 29, 1988
CHIP DECODE PAL16L8
AL AM5 LST AM4 AM0 AM3 IACK AM1 LWORD GND
OE BDS1 BDS0 DS0 DS1 BDSEL DTK CYC STAM VCC
EQUATIONS
; VMEBUS CYCLE SELECT
/CYC = DS0 * DS1
; BUFFERED DATA STROBES
/BDS0 = DS0
/BDS1 = DS1
; BOARD SELECT
/BDSEL  = /LST * /AL
/DTK    = BDSEL
; STANDARD (A24) ADDRESS MODIFIER DECODE
/STAM = IACK*LWORD*AM5*AM4*AM3*AM1*/AM0
        + IACK*LWORD*AM5*AM4*AM3*/AM1*AM0
```

Figure 2-21 Logic equations for U6 in Figure 2-20

The lower VMEbus addresses are latched by U11 and U12 on the falling edge of either data strobe. Address bits A16-A23 are compared to the base address select switch using U1, whose output is latched with U4. Since it responds to A24 addresses it does not decode address lines A24-A31.

Those unfamiliar with programmable logic may find the logic equations hard to understand. However, they are actually quite simple. Those given in Figure 2-21 are written in PALASM™. Here the state of each output (on the left side of the equal sign) is asserted or negated depending upon the state of the inputs (on the right side of the equal sign). The Boolean operators of AND (*), OR (+) and NOT (/) can be used in the equations. For example, the equation for the latching signal CYC is:

/CYC = DS0 * DS1

This means that the output (CYC) will be low if inputs DS0 and DS1 are both high. Schematically it would be the same as the circuit of Figure 2-22.

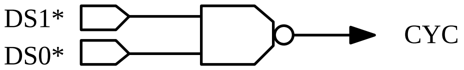


Figure 2-22 Circuit which performs the same function as PLD equation: /CYC = DS0 * DS1

Because the PLD does not assert /STAM if LWORD* is asserted, the module will not respond to D32 or to misaligned cycles. Only single and double byte cycles are accepted. The byte locations that are accessed are controlled by the level of data strobes DS0* and DS1*. These, plus the state of the WRITE* signal, are used to control /OE, WE0, WE1 and DTACK*. The module will accept D16 as well as D08(EO) cycles.

Data is buffered to and from the memory ICs using 74LS645A-1 bus transceivers U13 and U14. These supply the 48 mA drive current required by the VMEbus data lines.

The cycle time of the memory ICs is controlled by delay line U5.

In this example the VMEbus reset line [SYSRESET*] is not used. During system reset this module is required to negate DTACK*, and stop driving data lines D00-D07 within 30 microseconds after SYSRESET* is asserted. Since masters are required to negate the data strobes after only 20 microseconds, and this circuit resets itself every time the data strobes are negated, the monitoring of SYSRESET* is useless. For more information on system reset refer to Chapter 5.

2.5 Circuit Example - Simple 32-bit Memory Module

The circuit for a 32 Kbyte, 32-bit memory module is shown in Figure 2-23. This board has an A32:A24:D32:D16:D08(EO) slave interface and supports misaligned transfers. The 32-bit memory module responds to all of the following address modifier codes:

AM	5	4	3	2	1	0	Description
0x0E	0	0	1	1	1	0	A32 supervisory program access
0x0D	0	0	1	1	0	1	A32 supervisory data access
0x0A	0	0	1	0	1	0	A32 non-privileged program access
0x09	0	0	1	0	0	1	A32 non-privileged data access
0x3E	1	1	1	1	1	0	A24 supervisory program access
0x3D	1	1	1	1	0	1	A24 supervisory data access
0x3A	1	1	1	0	1	0	A24 non-privileged program access
0x39	1	1	1	0	0	1	A24 non-privileged data access

At the falling edge of either data strobe the outputs of comparators U1 and U4 are latched using flip-flop U2. Address decoder PLD U6 decodes the address modifiers and asserts STAM or EXAM depending upon whether the cycle uses an A24 or A32 address. STAM and EXAM are latched shortly after the falling edge of either data strobe using flip-flop U3.

The address width to which the module responds is selected using the STA and EXA inputs of U6. If switch STA is closed the module responds to A24 addresses and if EXA is closed it responds to A32 addresses. If both are closed the module will respond to either one.

When the 32-bit memory module is accessed, U6 will assert the board select signal BDSEL which enables memory ICs U12-U15 and the memory control PLD U16. The logic equations for U6 are given in Figure 2-24, and those for U16 are shown in Figure 2-25.

During read cycles U16 asserts memory output enable /MOE. During write cycles it asserts memory write enables WE0-WE3 depending upon the state of DS0*, DS1*, A01 and LWORD*. Since the module supports misaligned transfers, a multiplexor circuit routes data from the VMEbus data lines to the correct memory IC. This circuit is formed by bus transceivers U17-U24. U16 asserts MUX, NOMUX and WE0-WE3 depending upon how the data gets routed.

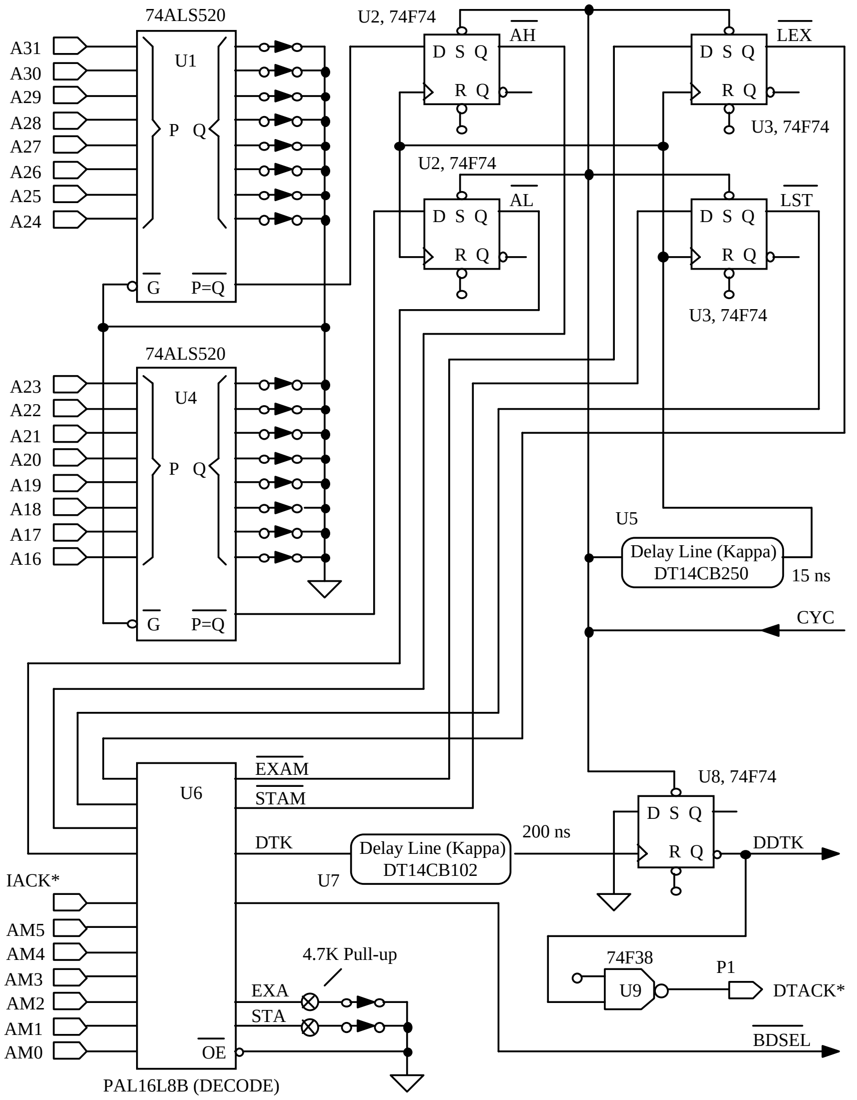
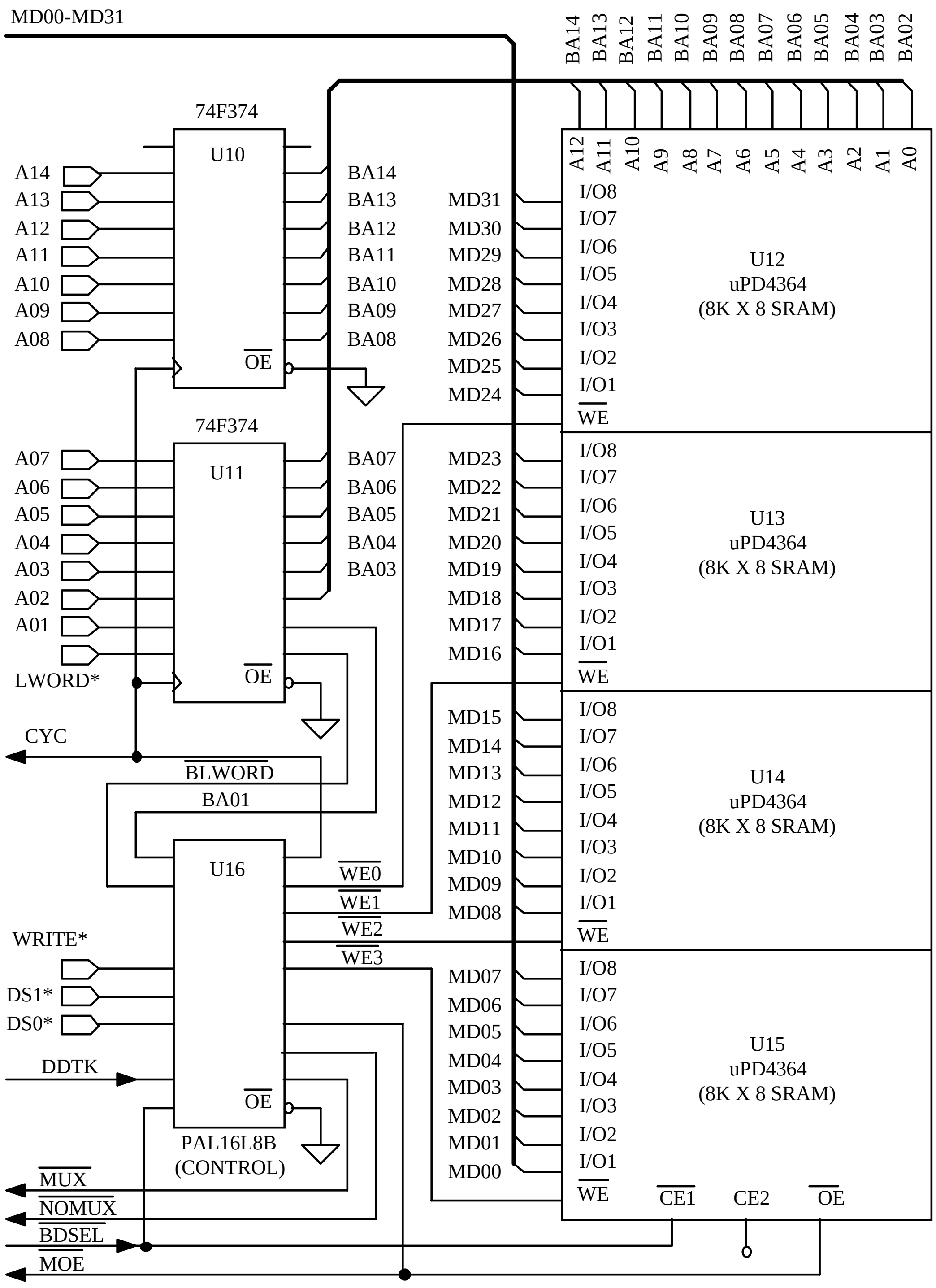
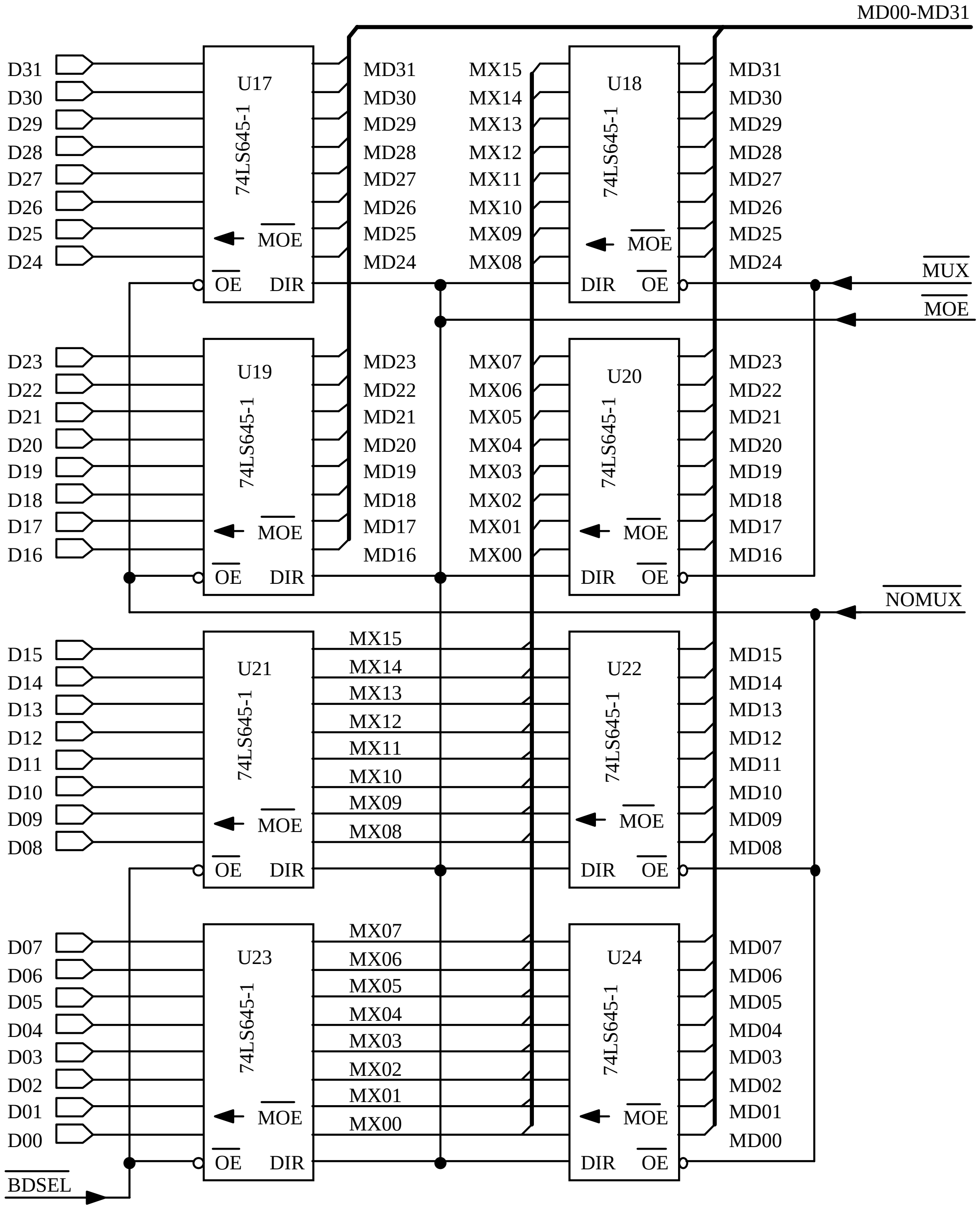


Figure 2-23 32 Kbyte, 32-bit memory module with A32:A24:D32:D16:D08(EO) slave interface


MD00-MD31
BA14 BA13 BA12 BA11 BA10 BA09 BA08 BA07 BA06 BA05 BA04 BA03 BA02
74F374
U10
A14
A13
A12
A11
A10
A09
A08
BA14
BA13
BA12
BA11
BA10
BA09
BA08
OE
74F374
U11
A07
A06
A05
A04
A03
A02
A01
LWORD*
BA07
BA06
BA05
BA04
BA03
OE
CYC
BLWORD
BA01
U16
WE0
WE1
WE2
WE3
WRITE*
DS1*
DS0*
DDTK
OE
PAL16L8B
(CONTROL)
MUX
NOMUX
BDSEL
MOE
MD31
MD30
MD29
MD28
MD27
MD26
MD25
MD24
MD23
MD22
MD21
MD20
MD19
MD18
MD17
MD16
MD15
MD14
MD13
MD12
MD11
MD10
MD09
MD08
MD07
MD06
MD05
MD04
MD03
MD02
MD01
MD00
A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
I/O8
I/O7
I/O6
I/O5
I/O4
I/O3
I/O2
I/O1
WE
U12
uPD4364
(8K X 8 SRAM)
U13
uPD4364
(8K X 8 SRAM)
U14
uPD4364
(8K X 8 SRAM)
U15
uPD4364
(8K X 8 SRAM)
CE1
CE2
OE


MD00-MD31
D31
D30
D29
D28
D27
D26
D25
D24
U17
74LS645-1
MOE
OE
DIR
MD31
MD30
MD29
MD28
MD27
MD26
MD25
MD24
MX15
MX14
MX13
MX12
MX11
MX10
MX09
MX08
U18
74LS645-1
MOE
DIR
OE
MD31
MD30
MD29
MD28
MD27
MD26
MD25
MD24
MUX
MOE
D23
D22
D21
D20
D19
D18
D17
D16
U19
74LS645-1
MOE
OE
DIR
MD23
MD22
MD21
MD20
MD19
MD18
MD17
MD16
MX07
MX06
MX05
MX04
MX03
MX02
MX01
MX00
U20
74LS645-1
MOE
DIR
OE
MD23
MD22
MD21
MD20
MD19
MD18
MD17
MD16
NOMUX
D15
D14
D13
D12
D11
D10
D09
D08
U21
74LS645-1
MOE
OE
DIR
MX15
MX14
MX13
MX12
MX11
MX10
MX09
MX08
U22
74LS645-1
MOE
DIR
OE
MD15
MD14
MD13
MD12
MD11
MD10
MD09
MD08
D07
D06
D05
D04
D03
D02
D01
D00
U23
74LS645-1
MOE
OE
DIR
MX07
MX06
MX05
MX04
MX03
MX02
MX01
MX00
U24
74LS645-1
MOE
DIR
OE
MD07
MD06
MD05
MD04
MD03
MD02
MD01
MD00
BDSEL

```
TITLE           ADDRESS DECODER CONTROL
PATTERN         VH0003.PDS
REVISION        A
AUTHOR          WADE PETERSON
COMPANY         (C) 1988 WADE PETERSON. ALL RIGHTS RESERVED
DATE            FEBRUARY 13, 1988
CHIP DECODE PAL16L8
LEX LST AH AL AM5 AM4 AM3 AM2 AM1 GND
OE STAM AM0 IACK STA EXA BDSEL DTK EXAM VCC
EQUATIONS
; BOARD SELECT
/BDSEL  = /STA * /LST * /AL
        + /EXA * /LEX * /AL * /AH
/DTK    = BDSEL
; STANDARD (A24) ADDRESS MODIFIER DECODE
/STAM   = IACK *  AM5 *  AM4 *  AM3 *  AM2 *  AM1 * /AM0
        + IACK *  AM5 *  AM4 *  AM3 *  AM2 * /AM1 *  AM0
        + IACK *  AM5 *  AM4 *  AM3 * /AM2 *  AM1 * /AM0
        + IACK *  AM5 *  AM4 *  AM3 * /AM2 * /AM1 *  AM0
; EXTENDED (A32) ADDRESS MODIFIER DECODE
/EXAM   = IACK * /AM5 * /AM4 *  AM3 *  AM2 *  AM1 * /AM0
        + IACK * /AM5 * /AM4 *  AM3 *  AM2 * /AM1 *  AM0
        + IACK * /AM5 * /AM4 *  AM3 * /AM2 *  AM1 * /AM0
        + IACK * /AM5 * /AM4 *  AM3 * /AM2 * /AM1 *  AM0
```

Figure 2-24 Address decode logic for U6 in Figure 2-23

```
TITLE           MEMORY CONTROL
PATTERN         VH0002.PDS
REVISION        A
AUTHOR          WADE PETERSON
COMPANY         (C) 1988 WADE PETERSON. ALL RIGHTS RESERVED
DATE            FEBRUARY 13, 1988
CHIP CONTROL PAL16L8
BA01 BLWORD WRITE DS1 DS0 DDTK BDSEL NC NC GND
OE NOMUX MUX MOE WE3 WE2 WE1 WE0 CYC VCC
EQUATIONS
; VMEBUS CYCLE
/CYC = DS0 * DS1
; WRITE ENABLE, MEMORY BANK #3
/WE3 = /BDSEL * /WRITE * /DDTK * /DS0 * /BA01 * /BLWORD
     + /BDSEL * /WRITE * /DDTK * /DS0 *  BA01 *  BLWORD
```

```
; WRITE ENABLE, MEMORY BANK #2
/WE2 = /BDSEL * /WRITE * /DDTK * /DS1 * /BA01 * /BLWORD
     + /BDSEL * /WRITE * /DDTK * /DS1 *  BA01 *  BLWORD
     + /BDSEL * /WRITE * /DDTK * /DS1 * /DS0  * /BLWORD
     + /BDSEL * /WRITE * /DDTK * /DS0 * /BA01 * /BLWORD
; WRITE ENABLE, MEMORY BANK #1
/WE1 = /BDSEL * /WRITE * /DDTK * /DS0 * /BA01
     + /BDSEL * /WRITE * /DDTK * /DS1 * /BA01 * /BLWORD
     + /BDSEL * /WRITE * /DDTK * /DS1 * /DS0  * /BLWORD
; WRITE ENABLE, MEMORY BANK #0
/WE0 = /BDSEL * /WRITE * /DDTK * /DS1 * /BA01
; MEMORY OUTPUT ENABLE
/MOE = /BDSEL * /DS0 * WRITE
     + /BDSEL * /DS1 * WRITE
; DATA MUX CONTROL
/MUX    = /BDSEL * /BA01 * BLWORD
/NOMUX  = /BDSEL * MUX
```

Figure 2-25 Memory control logic for U16 in Figure 2-23

2.6 Circuit Idea - The Tundra Universe

The preceeding circuit examples show 'traditional' VMEbus interfaces. The Universe™ and Universe™ II IC families from Tundra Semiconductor Corporation implement most of the new features and bus cycles allowed in VME64. These devices provide a high performance 64-bit VMEbus to PCI interface in one package. Most of these functions are provided on a single ball-grid-array package as shown in Figure 2-26.

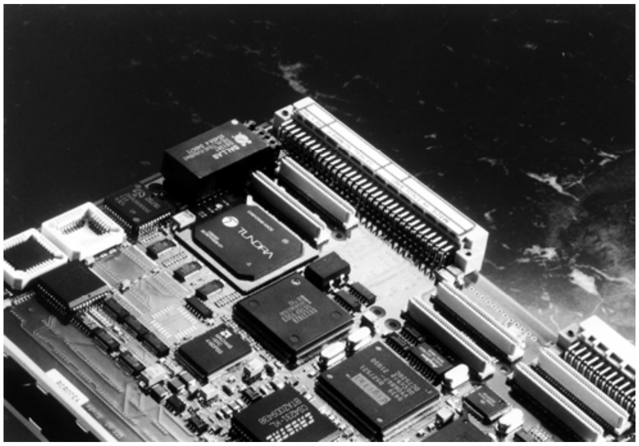


Figure 2-26 The Universe™ bus interface IC

Photo courtesy of Tundra Semiconductor Corporation.

The Universe™ is suited for CPU boards, providing both a master and a slave interface. The IC is particularly useful with implementing a local PCI bus.

Some features of the Tundra Universe™ include:

- 64-bit VMEbus interface.
- 64-bit, 33 MHz PCI local bus interface.
- Integral FIFOs for write posting to maximize bandwidth utilization.
- Programmable DMA controller with linked list support.
- VMEbus transfer rates of 60-70 Mbytes per second.
- Complete suite of VMEbus address and data transfer modes, such as:
    - A32/A24/A16 master and slave
    - MBLT (D64)/D32/D16/D08 master and slave.
    - BLT, ADOH, RMW and LOCK cycles.
- Automatic initialization for slave-only applications.
- Flexible register set, programmable from both the VMEbus and PCI ports.
- Full VMEbus system controller.

A full description of this circuit is very large, and beyond the scope of this book. Users should contact the manufacturer for more details. However, Figure 2-27 shows a functional diagram of the Universe™ to VMEbus interface. That figure shows how the Universe™ provides the core logic necessary to implement the bus interface.

External buffers are needed on most of the VMEbus interface signals. This is typical of many VLSI VMEbus interfaces. That's because external ICs are needed to provide the high current drive (and power dissipation) for the bus interface. Most VLSI solutions simply can't dissipate enough heat to satisfy the backplane requirements.

2.7 Location Monitor

The *location monitor* receives events that are broadcast from masters. First introduced in the Revision IEEE-1014-1987 VMEbus specification, it is intended for distributed intelligence applications. The location monitor is optional and may reside on any number of masters or slaves.

To use the location monitor, a real address space of any size is established. This space must respond to bus cycles with DTACK* or BERR*. When a master accesses this memory space it generates a bus cycle. Location monitors interpret the cycle and perform some on-board task. Intelligent slave modules may interrupt their on-board processors when a specific virtual address is accessed.

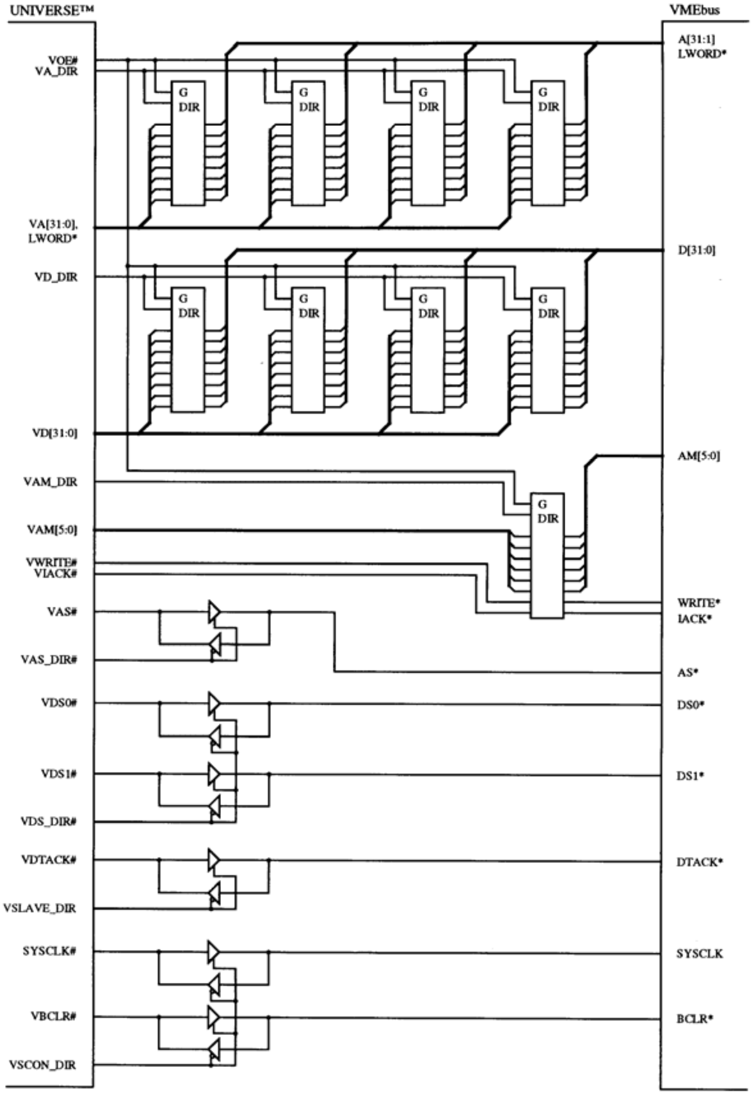
Location monitors themselves do not respond to bus cycles by asserting DTACK* or BERR*. However, the system must be set up so that bus cycles intended for location monitors are acknowledged in some way.

For example, if a location monitor is configured to monitor address 0x003C00, then a module somewhere on the backplane must respond to the cycle (with DTACK* or BERR*). A good way to handle this is to reserve several locations on, say, a memory board. The location monitor initiates activity in response to a read or write cycle, but the memory board acknowledges the cycle by asserting DTACK* or BERR*.

Location monitors may respond to address or data information. For example, a location monitor may respond if 0xFF00 is written to address 0X003C00; but ignore the cycle if 0x4A04 is written to the same address.

Probably the most common use for the location monitor is in backplane analyzers. These devices are available from a variety of manufacturers (e.g. VMETRO). They are used to monitor the backplane signals and provide a diagnostic capability. They are often very similar to standard logic analyzers in capability, but fit on a single VMEbus module. This makes them very easy to use.

Locations monitors can also be used for *broadcast* applications. For example, in digital signal processors a trigger cycle can be broadcast to multiple analog-to-digital (A/D) modules. Location monitors on the A/D modules initiate data acquisition simultaneously, thereby preventing aliasing.


UNIVERSE™
VMEbus
VOE#
VA_DIR
G
DIR
A[31:1]
LWORD*
VA[31:0],
LWORD*
D[31:0]
VD_DIR
VD[31:0]
AM[5:0]
VAM_DIR
VAM[5:0]
VWRITE#
VIACK#
WRITE*
IACK*
VAS#
VAS_DIR#
AS*
VDS0#
DS0*
VDS1#
DS1*
VDS_DIR#
VDTACK#
DTACK*
VSLAVE_DIR
SYSCLK#
SYSCLK
VBCLR#
BCLR*
VSCON_DIR

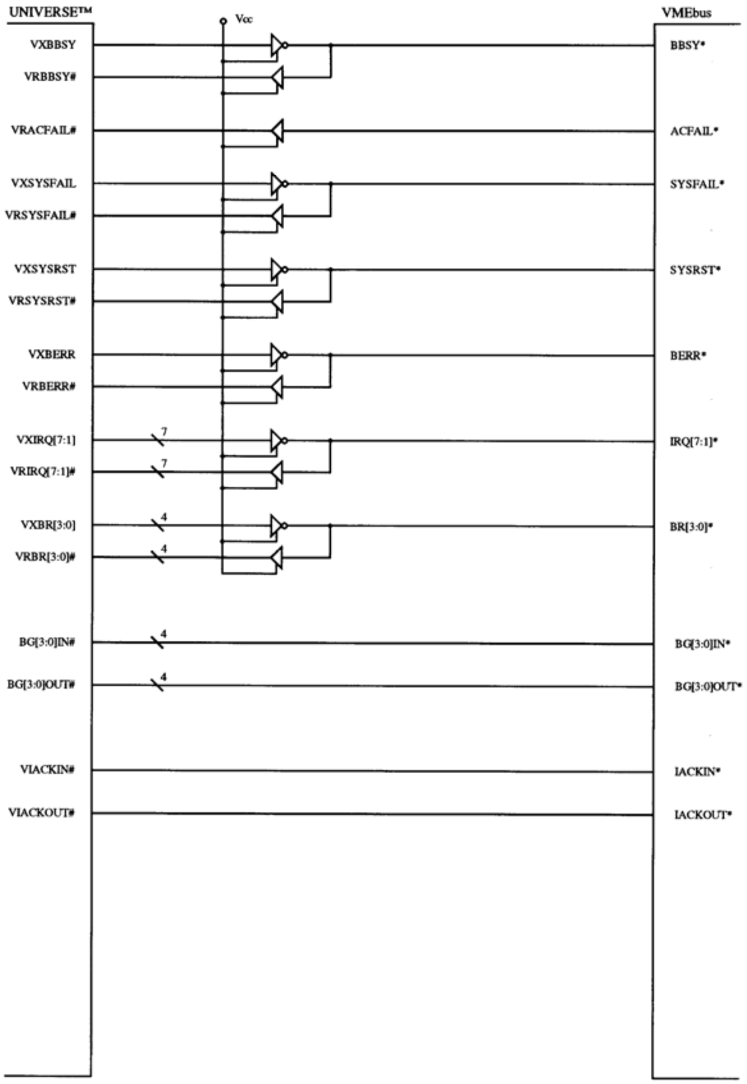


Figure 2-27 Functional Diagram of the Universe™ to VMEbus interface

Standard software interfaces are another use for the location monitor. A long standing goal in the microcomputer bus industry has been to make standard interfaces where firmware from independent vendors works together. This creates a plug-n-play environment and greatly reduces the time and expense of writing application specific software. However, this feature has never been exploited on VMEbus.

To illustrate how this works, consider a system with a CPU and a disk controller as in Figure 2-28. The original vendor of the disk controller is company XYZ and some hardware dependent software was written for this vendor (part of the operating system). If we wanted to change from manufacturer XYZ to manufacturer ABC we could plug the new module into the chassis and it would be electrically compatible, but the software interface would not work. The system would most likely crash. The problem is that company XYZ has a different set of software "hooks" to which the disk controller is ported to the operating system.

With the location monitor a common set of standard "hooks" could be used throughout the industry. If company XYZ and ABC used the same set of hooks they would work with independently developed software. When the CPU module required a disk read it would access a virtual memory location. A location monitor on the disk controller would see this access and notify the disk controller's on-board microprocessor that a disk read is requested. The disk controller would then respond in some previously defined way. The advantage to this protocol is that all firmware for disk controllers could be written to accept the standardized protocol, and disk controllers would be "plug-n-play."

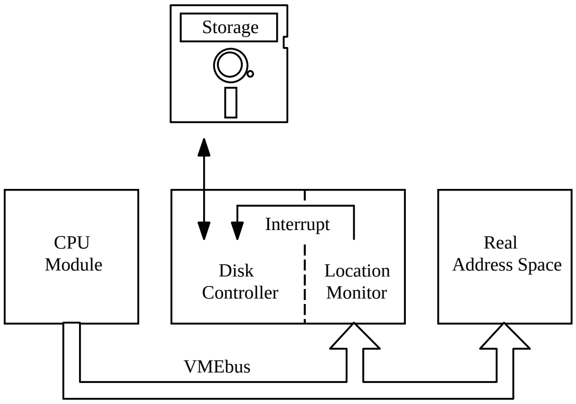


Figure 2-28 Using a location monitor for a disk controller

The same mnemonics for slaves apply to the location monitor. For example, an A24:D16 location monitor describes a module that monitors standard address modifier codes and 16-bit data transfers. The only exception is that D08(O) location monitors are not explicitly permitted.

2.8 Bus Timer

VMEbus handles abnormal bus cycles with the BERR* (bus error) signal. During a normal cycle a master waits until a slave asserts DTACK* before terminating the cycle. BERR* operates like DTACK* (or RETRY*) except that the cycle is terminated with some type of fault.

Examples of these faults include quad byte transfers to single byte slaves, unaligned transfers to non-UAT boards and memory parity errors. Upon receipt of BERR* the master will often generate an exception (interrupt) to a local processor, which causes it to perform an error handling routine. This routine can notify the user, fix the problem, or shut down the system. Also, slaves are not required to generate the BERR* signal.

A functional module called a bus timer can also assert BERR*. These modules assert BERR* in response to a bus lock-up. Lock-ups are caused by software bugs, unauthorized accesses, spurious interrupts, card counting or memory sizing. The bus timer simply monitors data strobes DS0* and DS1*, and asserts BERR* if either has been low for some amount of time.

When the bus timer is used for memory sizing, a master simply checks a variety of addresses on VMEbus, and tests if there is a response. Since normal cycles can only be terminated with DTACK*, the bus will lock up if a non-responding memory location is accessed. A similar situation happens if the software must determine which cards are in the system (card counting). Without the bus timer, the system will crash when performing these operations.

Bus timers are sometimes called *watchdogs* or *bus watchdog timers*. A bus cycle with a participating bus timer is shown in Figure 2-29.

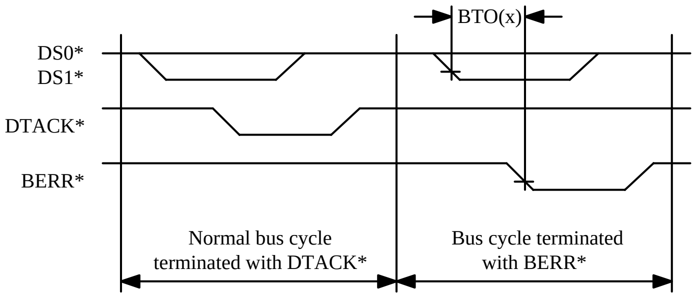

BTO(x)
DS0*
DS1*
DTACK*
BERR*
Normal bus cycle terminated with DTACK*
Bus cycle terminated with BERR*

Figure 2-29 Bus cycle with participating bus timer

When evaluating or designing VMEbus modules, treat DTACK* and BERR* alike. Although they serve different functions, the VMEbus specification usually treats them both the same, at least in terms of bus timing.

For example, a master cannot assert data strobes DS0* or DS1* until DTACK* has been negated by the slave. The same is true for BERR*.

Modules that drive DTACK* and BERR* must never assert them at the same time.

The bus timer can reside anywhere in the VMEbus chassis, but is usually located on the slot 01 system controller. A mnemonic called BTO(x) designates that a bus timer is available, where [x ≤ time out in microseconds ≤ 2x]. Although the time-out can be of any length, it is commonly 64 microseconds. A shorter time-out can cause legal bus cycles to trip the timer, and longer ones can cause long delays during system start up (for memory sizing). The delay can be adjustable.

The VME64x standard also recommends the use of a 2eBTO(x) bus timer. If used, that timer monitors DS0*, DS1* and DTACK*. The BERR* line is driven low when either data strobe has been low for more than 'x' microseconds, and there have been no edges on DS0*, DS1* or DTACK*. This timer is recommended because the 2eVME cycle can hold DS0* low for a long time during the cycle.

2.9 Circuit Example - Bus Timer

A simple bus timer circuit is shown in Figure 2-30. This circuit is suitable for use on slave modules or slot 01 system controllers. It monitors data strobes DS0* and DS1*, and asserts BERR* if either strobe is low for more than 56 microseconds.

This circuit monitors data strobes DS0* and DS1* using a 74LS00 NAND gate (U1). The output of this gate is connected to the master reset inputs of a 74LS393 counter (U3) and 74LS164 shift register (U4). When both data strobes are high the counter and shift register are cleared, and BERR* is negated. When either data strobe is asserted the master reset inputs of the counter and shift register are negated. This enables the counter circuit which divides the 16 MHz SYSCLK down to 62.5 KHz. This causes the Q0-Q7 outputs of U4 to be set at the rising edge of every clock. If a data strobe is asserted for more than 56 microseconds (four positive clock edges) shift register output Q4 is driven high and BERR* asserted. When the master detects BERR* it negates both data strobes, which then causes BERR* to be negated.

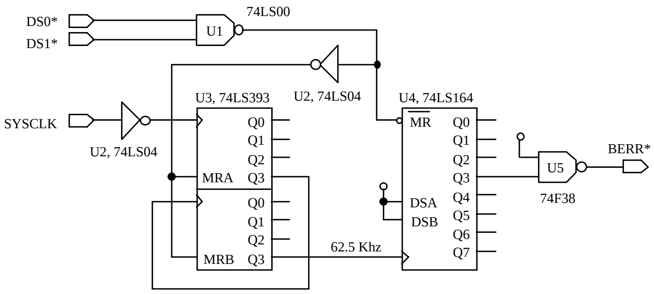


Figure 2-30 Simple bus timer circuit

The mnemonic for this bus timer is BTO(56), which means that it takes at least 56 microseconds to assert BERR*. This time can be easily altered by changing the clock frequency of the shift register, or by tapping a different output of U4. This circuit is also shown as part of the simple system controller circuit in Chapter 3.

2.10 How VMEbus And 680XX Bus Protocols Differ

Although VMEbus was originally considered to be a 68000 bus, it was designed to be non-processor specific. Some of the most common VMEbus interfacing problems happen when designers assume that VMEbus timing is the same as 68000 timing. This is easy to do, as many of the VMEbus signal names are the same as the 68000 microprocessor. However, there are some major differences between the 68000 and VMEbus interfaces.

Unlike 680XX peripherals, VMEbus slave modules are not guaranteed that address and data lines are stable after the assertion of DTACK*. This is especially a problem during write cycles. This difference is sometimes referred to as address rot or data rot.

There are two types of VMEbus cycles: interlocked and broadcast cycles. An interlocked cycle takes place between two specific modules. The receiving module acknowledges the cycle generated by a sender, and only those two participate in the cycle. An example of a VMEbus interlocked cycle is a read/write cycle. 680XX data transfer cycles are not fully interlocked. VMEbus requires that masters do not assert data strobes DS0* and DS1* until DTACK* has been negated by the slave. This is not necessary with 680XX processors. If DTACK* (or DSACK0, 1* on the 68020 or 68030) remains asserted the 680XX assumes that memory is running without *wait states*.

Bus timing parameters for the two protocols vary widely. Be extra careful when evaluating or designing address strobe and data strobe set-up and hold times.

The 68020 and 68030 processors can handle 8, 16 and 32-bit memory (data) port sizes. VMEbus can also do this, but does so differently.

At the start of every bus cycle the 68020 and 68030 microprocessors broadcast the size of the memory transfer they are attempting (8, 16, 24 or 32-bit). They do this using two signals called SIZ0 and SIZ1. 680XX slaves decode these and place data on the appropriate data bus lines. After doing so the slaves assert one or both data transfer acknowledge signals called DSACK0* and DSACK1* depending upon the data port width.

For example, the 68020 MPU always attempts to transfer the largest number of bytes on every transfer. If the processor attempts to write a 32 bit longword into a 16-bit memory, the memory will accept 16 bits of the data and acknowledge to the processor that it only accepted 16-bits using DSACK0* and DSACK1*.

The problem arises when the 68020 is connected to VMEbus, which has a single data transfer acknowledge signal (DTACK*). Circuitry must be added to convert DTACK* to DSACK0* and DSACK1*. In most cases a programmable address decoder is used to combine the VMEbus memory map so that DTACK* is converted to DSACK0* and DSACK1*.

Data for 8-bit and 16-bit transfers are done over different sets of data lines on the 68020 and 68030 microprocessors than for VMEbus. In most cases a data multiplexor is required to route data correctly between the MPU and VMEbus. The 68020 and 68030 processors also handle unaligned transfers differently than VMEbus does.

Under certain circumstances the 68020 and 68030 microprocessors change their address lines during Read-modify-write cycles. The address lines between the MPU and VMEbus may need to be latched. These microprocessors use a RMC (read-modify-control) pin.

2.11 References

Gruender, Jr. et al. Method And System For Automatic Configuration Of Memory Devices US Patent No. 5,463,589 1995

Kidder, Tracy. The Soul Of A New Machine Little Brown 1981

Lau, Nelson D. Slave Controller Utilizing Eight Least/Most Significant Bits For Accessing Sixteen Bit Data Words US Patent No. 5,307,475 1994

Lau, Nelson D. Slave Controller For Effecting A Block Transfer Of Sixteen Bit Words Between A Memory And A Data Transfer Bus US Patent No. 5,319,767 1994

Lawler, Edward P. Digital Data Apparatus For Transferring Data Between A Byte-Wide Digital Data Bus And A Four Byte-Wide Digital Data Bus US Patent No. 5,428,763 1995

Peterson, Wade D. "VME64: Taking The VMEbus Architecture Into The 21st Century" *VMEbus Systems* August 1996

Peterson, Wade D. "VME64x, The VME64 Extensions Standard" *VMEbus Systems* August 1997

Starr, Daryl D. Enhanced VMEbus Protocol Utilizing Pseudosynchronous Handshaking And Block Mode Data Transfer US Patent No. 5,388,231 1995

VMEbus Specification / ANSI/IEEE STD1014-1987 VITA, Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA, Scottsdale, AZ 1995

VME64x Specification / VITA 1.1-1997 (Draft 2.0) VITA, Scottsdale, AZ 1997

Chapter 3
Multiprocessing

One way to increase the speed of a computer system is to increase the number of processing elements. These are sometimes called multiprocessing systems. They are faster than single processor systems because the computer's work load is shared between several parts of the system. When all of the parts run at the same time, the computer goes faster. It's just like building a house: ten people get the job done faster than one.

Multiprocessing systems run faster if *parallelism* is used. Parallelism is the ability of a computer to do several tasks at the same time. Unfortunately, this is often hard to do because most computer software runs sequentially (i.e. one instruction at a time).

Sequential computers can only do one thing at a time, but they do it very fast. This is a result of their basic structure...the so called *Von Neumann architecture*. In this architecture, instructions are fetched from memory, and used by a central processing unit (CPU) to perform operations on data. Unfortunately, this architecture has a major drawback called the *Von Neumann bottleneck*, which is caused by a speed imbalance between the central processor and memory. If the central processor is faster than memory, it gets bogged down fetching program instructions, and doesn't have as much time to get useful work done. Mainframe computer designers have struggled with the problem for years. Microcomputer designers have confronted it since the advent of the 32-bit microprocessor.

Structurally, parallelism can be accomplished by parallel or serial task operations. In parallel operations, tasks are broken up into equal, manageable chunks and allocated to multiple CPUs for processing. Each CPU completes its task, and passes the result back to a master processor. The master processor then coordinates the data from the other CPUs, and produces a result.

A good example of parallelism is a computer running an operating system like UNIX™, where a CPU and an intelligent disk controller work together to store and retrieve data. The CPU runs programs, and the disk controller handles I/O (to and from the disk) at the same time.

In another form of parallel architecture, the CPU executes two or more instructions at the same time. Many Digital Signal Processor (DSP) chips work this way. For example, the TMS320C50 DSP has information encoded into each instruction that prepares the processor for the instruction following it. These dual-purpose instructions create a simple form of parallelism, thereby speeding up the system.

Some systems use *serial parallelism* to break tasks into manageable chunks, and force data to flow between the chunks. These are also known as *pipelining* or *data flow* architectures.

Multiprocessing systems can also use specialized processors for functions like floating point arithmetic or graphics algorithms. In the house building analogy, it's like hiring a plumber to connect the kitchen sink. While anyone could learn to connect the sink, it is much faster to have a specialist do it. The same is true for computers.

3.1 Data Transfer Arbitration Bus

VMEbus allows multiple masters to share the data transfer bus, thereby creating a multiprocessor system. Every master in the system can use the data transfer bus to move data between themselves and slaves. Only one master can use the bus at any time.

A sub-bus, called the *data transfer arbitration bus*, is used to prevent contention. Bus masters use it to acquire ownership of the data transfer bus before they read or write data to slaves. A master does not necessarily have to be a processor (or CPU). Specialized I/O modules with direct memory access (DMA) controllers or interrupt handlers can also be masters.

Figure 3-1 shows how the data transfer arbitration bus uses two functional modules called the *requester* and the *arbiter*. In order to acquire the bus, the master first notifies its on-board requester that it needs the bus. The requester then performs handshaking with the arbiter, and notifies the master when the bus is available.

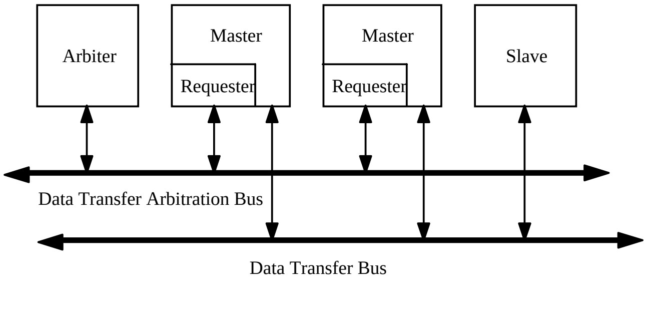


Figure 3-1 Masters acquire ownership of the data transfer bus using the data transfer arbitration bus

Besides preventing bus contention, the data transfer arbitration bus also efficiently schedules bus mastership. Optional scheduling algorithms grant the bus to masters on an equal or priority basis.

3.1.1 Acquiring The Bus

To obtain the bus, the requester asserts one of four bus request signals called BR0*, BR1*, BR2* and BR3*. Each of these signals are formed from open-collector logic, which allows any number of masters to request the bus. For example, ten processors could use level BR2*, and five processors could use BR1*.

The arbiter, always located in the slot 01 backplane position, monitors the request lines until one of them is asserted. The next time the bus is available, the arbiter grants it to the waiting master.

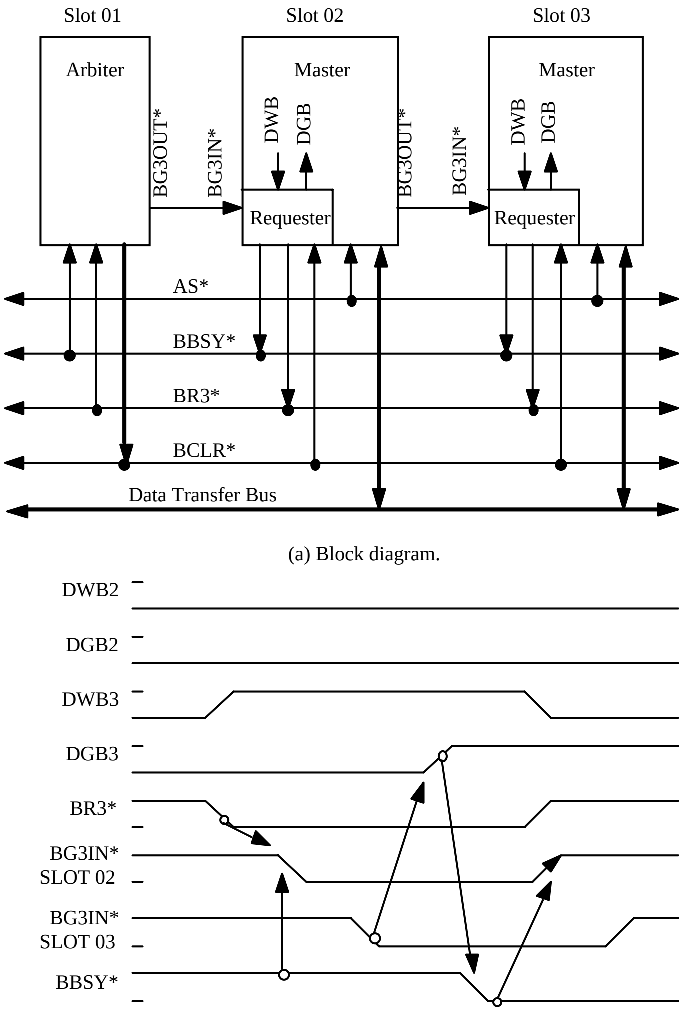
The waiting master is granted the bus by way of the *bus grant daisy-chains*. There are four daisy-chains called BG0IN*/BG0OUT*, BG1IN*/BG1OUT*, BG2IN*/BG2OUT* and BG3IN*/BG3OUT*. As shown in Table 3-1, each of these chains correspond to a bus request level BR0*, BR1*, BR2* and BR3*. Bus grants generated by the arbiter travel from module to module on one of the daisy-chains. Each requester in the system monitors these daisy-chains, and uses them to determine when they have been granted the bus.

Table 3-1 Bus request levels and their associated daisy-chains

Bus Request Level	Associated Daisy Chain
BR3*	BG3IN* / BG3OUT*
BR2*	BG2IN* / BG2OUT*
BR1*	BG1IN* / BG1OUT*
BR0*	BG0IN* / BG0OUT*

Each requester in the system monitors its designated bus grant input BG0IN*-BG3IN*, and drives the associated bus grant output BG0OUT*-BG3OUT*. When a requester receives a bus grant it either (a) accepts bus ownership or (b) passes the daisy-chain onto the next module. If it initially requested the bus, and the daisy-chain of the same level becomes active, then it acquires the bus by asserting the bus busy (BBSY*) signal. This notifies the arbiter that (a) the bus is assigned to a master and (b) that it should negate the bus grant.

To illustrate how the arbitration handshaking works, consider the block diagram of Figure 3-2(a). The associated timing waveform for the arbitration handshaking is shown in Figure 3-2(b). These diagrams show a bus arbiter in slot 01, and two masters in slots 02 and 03, both assigned to request level BR3*. The other bus request levels (BR0*-BR2*) are not shown in these diagrams for simplicity.



(a) Block diagram.

(b) Timing Waveform.

Figure 3-2 Bus arbitration handshaking

If the master in slot 03 requires the bus, its requester asserts BR3*. The arbiter monitors BR3*, and asserts BG3OUT* when the bus is inactive (BBSY* negated). BG3OUT* of slot 01 is directly connected to BG3IN* at slot 02 on the backplane. The master located in slot 02 monitors that BG3IN* is asserted, and passes the daisy-chain to BG3OUT* (i.e. it did not initially request the bus). When the requester in slot 03 monitors that BG3IN* is asserted it acquires the bus by asserting BBSY*, and notifies its master that it has the bus. The master waits until AS* is negated and then drives the data transfer bus.

When the arbiter monitors that BBSY* is asserted, it negates BG3OUT*. The daisy-chain then propagates back to the module that obtained the bus. This forms a completely interlocked bus arbitration cycle.

Once a master has obtained the data transfer bus, it can hold it for any length of time. However, the arbiter does have a method for removing the master from the bus using the bus clear (BCLR*) signal. If, for example, a master of higher priority requests the data transfer bus, the arbiter can assert BCLR* to force the master to relinquish the data transfer bus.

The master itself can also relinquish control of the data transfer bus. Typically, this is done either (a) at the end of every cycle, (b) at the end of a DMA or other block transfer or (c) when another master requests it. Most commercial CPU boards, for example, allow several methods for releasing the bus. These are generally selected with a DIP-switch or a software bit.

A single level bus grant daisy-chain can theoretically handle an infinite number of masters. Daisy-chains, however, take time to propagate since each master must decide to accept or pass them. If there are a large number of masters, arbitration may be slowed down too much. In addition the bus masters closest to the slot 01 arbiter can 'hog' the bus. To alleviate these problems the VMEbus architects created the four levels of arbitration, each with its own daisy-chain.

It should also be noted that the bus grant daisy-chains are broken if a VMEbus module is pulled out of the backplane. There are several methods for overcoming this problem, and are discussed in more detail in Chapter 7. For example, jumpers can be installed on the backplane wherever bus modules are removed. Many manufacturers place jumpers on the front and rear of the backplane to make them more accessible to the user. Some connector manufacturers also incorporate these jumpers into the backplane or connector.

When evaluating or designing bus modules, make sure that all of the daisy-chain signals are passed through the board. Boards with requesters (masters and interrupt handlers) always do this, but slave boards (which do not participate in bus arbitration) should connect their BGXIN* and BGXOUT* pins together. This eliminates the need to install daisy-chain jumpers in occupied slots. Some manufacturers forget to do this.

It is not necessary to place jumpers after the last module in the system (farthest from the arbiter) since the daisy-chains never propagate beyond that point. Also, if single level arbitration is used, only the BG3IN*-BG3OUT* daisy-chain need be jumpered. The other levels are not used.

3.1.2 Bus Arbiter

A central bus arbiter is required in all VMEbus systems. This functional module prevents more than one master from using the data transfer bus at any time. It also schedules bus ownership in multi-master systems. The arbiter must reside in the slot 01 position, and is sometimes called the system controller.

In its minimum configuration the system controller includes a bus arbiter, system clock driver, system reset circuit and IACK* daisy-chain driver. Although it is not required by the VMEbus specification, most system controller circuits also include a power monitor (to generate SYSRESET*) and a bus timer. Many vendors include system controllers on their CPU modules. This is not required, but has become popular since all VMEbus systems have at least one CPU module.

Arbiters are classified by the scheduling algorithm they use. Three popular types are the single level, priority, and round robin arbiters, but others are allowed.

3.1.2.1 Single Level Arbiter

The single level is the simplest type of arbiter. As the name implies it monitors only one bus request level, and grants the bus using one daisy-chain. By convention this arbiter monitors only bus request level BR3*.

The term 'single level arbiter' is something of a misnomer because it doesn't really arbitrate at all. Instead it uses the bus grant daisy-chain to do that job. It merely asserts BG3OUT* when BR3* is asserted and BBSY* is negated. Since each requester decides to accept or pass the daisy-chain, no central arbitration function is needed. It is quite popular since it is fast and can be built with very few parts. Its main disadvantage is that it prioritizes bus masters so that those located nearest to slot 01 will get the bus most often. However, this disadvantage can be overcome by using the FAIR requester (which is discussed below).

Single level arbiters may (optionally) assert BCLR* when a bus request is pending, and BBSY* is asserted.

The SGL mnemonic is used to describe the single level arbiter.

3.1.2.2 Circuit Example - Simple VMEbus System Controller

A simple VMEbus system controller is shown in Figure 3-3. This circuit includes an SGL arbiter, system reset generator, system clock driver, IACK* daisy-chain driver and a bus timer.

System reset is generated with RC network R1 & C1, and Schmidt trigger U1. When power is first turned on capacitor C1 is discharged and SYSRESET* is asserted using open collector NAND gate U2. The capacitor then charges through the 1.6 megohm resistor until the threshold voltage of U1 is reached (about 1.6 volts).

SYSRESET* is then negated, approximately 300-400 milliseconds after power is turned on.

When reset switch S1 is pushed, capacitor C1 discharges through R2. This asserts SYSRESET* for at least 200 milliseconds after the switch is released. Diode CR1 discharges C1 if the system power is quickly turned off, and then on again. The 5.6 ohm resistor R2 limits the current through the switch and diode to about 1 Amp (peak). For more information about system reset see Chapter 5.

The 16 MHz system clock is generated with TTL oscillator U3. This oscillator maintains the required frequency and duty cycle of SYSCLK. SYSCLK is driven using an IC with high current totem-pole outputs (U4). See Chapter 5 for more information about the system clock.

Two NAND gates (U5) form the SGL arbiter. When BBSY* is negated and BR3* is asserted, BG3OUT* is asserted with U4.

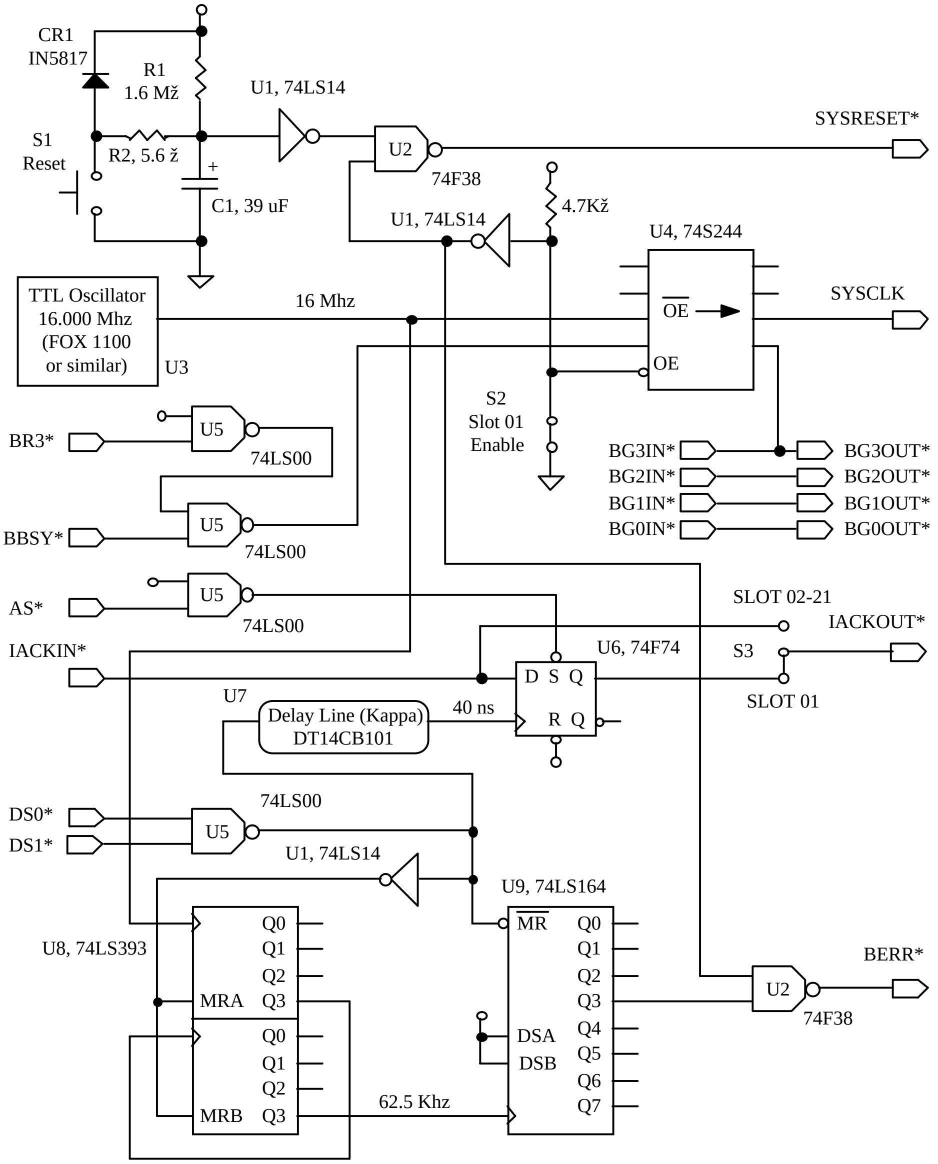


Figure 3-3 Schematic diagram for a simple VMEbus system controller

U5, U6 and U7 form the IACK* daisy-chain driver. This functional module negates the IACK* daisy-chain during back-to-back interrupt acknowledge cycles. Chapter 4 describes the IACK* daisy-chain driver in further detail.

A BTO(56) bus timer asserts BERR* during system lock-ups. When either data strobe (DS0* or DS1*) is asserted, the reset inputs of U8 and U9 are negated. Dual 4-bit counter U8 reduces 16 MHz down to 62.5 KHz for use by shift register U9. This clocks the outputs of shift register U9 high until 56 microseconds has passed. If the strobes are still asserted, then BERR* is asserted using U2. For more information about the bus timer see Chapter 2.

This system controller circuit can be located on any module in the system. Switches S2 and S3 must be configured according to the card slot in which the module resides (i.e. slot 1 or slots 2 - 21).Priority Arbiter

The priority arbiter assigns a priority to each bus request level (and master). By convention, BR3* is the highest priority and BR0* is the lowest. When two or more requests are pending the arbiter determines which has the highest priority, and asserts the appropriate bus grant daisy-chain.

The priority arbiter must assert BCLR* when a master of higher priority (than the one in control of the bus) initiates a request. For this reason, priority arbiters latch the state of the bus grant lines during an arbitration. When BBSY* is asserted, and a request is pending, the arbiter asserts BCLR* if the pending request is higher than that latched during the last arbitration. For example, if a master on level BR1* is granted the bus, BCLR* is asserted if requests occur on levels BR2* or BR3*. Requests on levels BR0* and BR1* are ignored by the arbiter. BCLR* is only asserted while the bus is busy (BBSY* asserted). Table 3-2 shows when the arbiter asserts BCLR*.

Table 3-2 Conditions when a priority arbiter asserts BCLR*

BCLR* State		Pending bus request level			
		3	2	1	0
Level of current master	3	1	1	1	1
	2	0	1	1	1
	1	0	0	1	1
	0	0	0	0	1
0 = Asserted, 1 = Negated					

A typical application using a priority arbiter is a system with a CPU module and an intelligent disk controller. The CPU module could be assigned to level BR2* and the disk controller to BR3*. Disk controllers often require a higher priority than CPU modules since they must off load data with minimum bus latency. Here the arbiter always grants the bus to the disk controller before the CPU. The CPU could be removed from the bus using a BCLR* release mechanism.

The PRI mnemonic describes the priority arbiter.

3.1.2.4 Round-Robin Arbiter

Round-robin arbiters give equal priority to all the bus request levels (and masters). These arbiters grant the bus on a rotating basis much like the four position rotary switch shown in Figure 3-4. When a master relinquishes the bus the switch is turned to the next position, and the bus is given to a master on that level. When a request is not pending on a certain level the arbiter skips over it and continues on to the next one. In this way all of the masters are granted equal access to the bus.

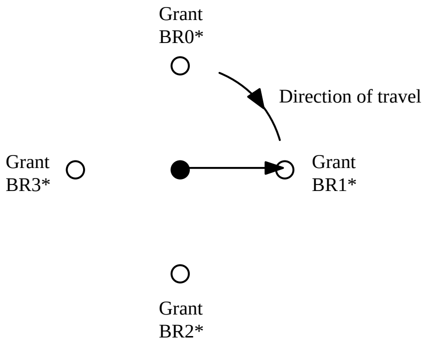


Figure 3-4 Round-robin arbiters grant the bus on a rotating basis much like a rotary switch

Under the Revision B VMEbus specification round-robin arbiters could not assert BCLR*. However, Revision C and later specifications permit it. If a round-robin arbiter does assert BCLR*, it does so 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. This prevents the arbiter from removing the current master from the bus prematurely. The conditions under which RRS arbiters assert BCLR* are shown in Table 3-3.

The conditions described here for the use of BCLR* on round-robin arbiters are *suggested* by the VMEbus specification. Some arbiters *may* use a different scheme. If BCLR* is used with an RRS arbiter, consult the board manufacturer for more details.

Round-robin arbiters are popular in data acquisition systems with intelligent peripherals, where data is collected and placed in shared memory. Often these peripherals must off load data to memory on an equal basis. *Priority arbiters* do not work well in these applications because some peripherals would receive more bus bandwidth than others, causing data 'gridlock'.

The RRS mnemonic describes the round-robin arbiter.

Table 3-3 Conditions when a round-robin arbiter asserts BCLR*.

BCLR* State		Pending bus request level			
		3	2	1	0
Level of current master	3	1	0	0	0
	2	0	1	0	0
	1	0	0	1	0
	0	0	0	0	1

0 = Asserted, 1 = Negated

3.1.2.5 Arbitration Time-out

If a requester does not assert BBSY* after a bus grant, the arbiter may withdraw the grant. This is called arbitration time-out, and prevents system crashes if a requester fails. Under normal conditions the arbiter does not negate a bus grant until a master asserts BBSY*.

The time to arbitrate must exceed the longest time for the bus grant daisy-chain to propagate the length of the bus, plus the length of time the slowest requester takes to assert BBSY*.

If an arbiter withdraws a bus grant it must be negated long enough for the daisy-chain to negate itself along the entire length of the bus.

Many vendors of VMEbus products do not specify how long it takes for a module to pass the bus grant daisy-chain, how long it takes to assert BBSY* or how long it takes to negate the daisy-chain. Without these numbers it is impossible to determine how long the arbitration time-out period should be. When evaluating or designing VMEbus modules for systems using the arbitration time-out feature, insist that these times are specified. Most modules take less than 1 microsecond to pass or negate the daisy-chain. A good rule of thumb would be a time-out exceeding 16 microseconds.

3.1.3 Requester

The requester is a functional module used by masters and interrupt handlers to obtain ownership of the data transfer bus. Figure 3-5 shows how a requester is used with a master. Here the master asserts device wants bus (DWB) to the requester when it needs the bus. The requester then participates in handshaking with the arbiter over the data transfer arbitration bus. When it obtains bus ownership it asserts device granted bus (DGB) to the master. When AS* has been negated (from the previous bus cycle) the master can then use the data transfer bus.

The master waits until AS* is negated because of the VMEbus *early bus release* mechanism (which is associated with address pipelining). This permits bus arbitration during a data transfer or interrupt acknowledge cycle, thereby speeding up bus cycles. Once a master begins a bus cycle (AS* asserted), the requester may negate BBSY* so that arbitration handshaking takes place during the cycle.

If the requester does not have a bus request pending it passes the BGXIN*-BGXOUT* daisy-chain.

Early bus release timing is shown in Figure 3-6. Here a master notifies its requester that it needs the bus by asserting DWB (A) which causes the requester to assert BR3* (B). When the master currently in control of the bus negates BBSY* (C) the arbiter asserts a grant to the requester (D). The requester then asserts BBSY*, asserts device granted bus to its on-board master and removes its bus request (E, F, H). The arbiter, upon monitoring BBSY* asserted, removes the bus grant (G). When AS* is negated by the previous master (I) the master takes control of the data transfer bus and begins doing bus cycles (J).

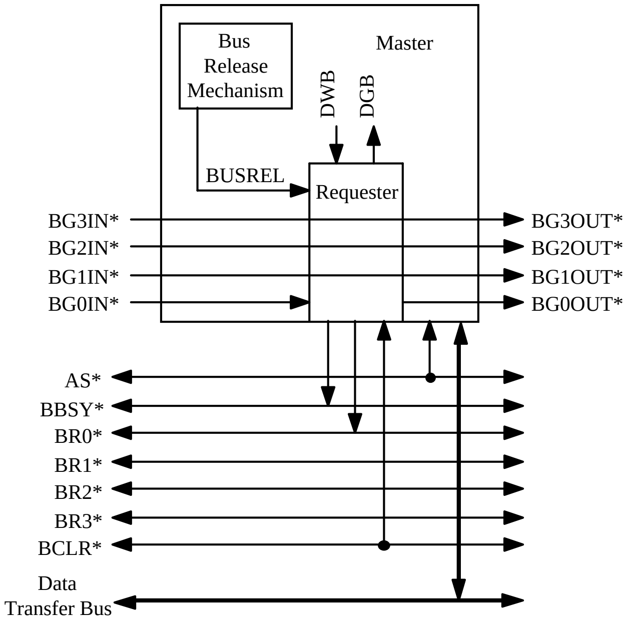


Figure 3-5 Bus requester functional module

Not all requesters implement the early bus release feature. The VMEbus specification allows requesters to wait until AS* has been negated by the last master before asserting BBSY*.

The requirement that the requester not drive the data transfer bus until AS* is negated also permits read-modify-write and block transfer cycles. AS* remains asserted during these cycles (or at least until the final DTACK*) which prohibits bus arbitration during the cycles.

VMEbus masters and interrupt handlers are allowed to have more than one requester. This allows bus requests on two or more priority levels. However, this is rarely done on commercial VMEbus boards.

There are three types of bus requester options on VMEbus: the *release-when-done*, *release-on-request* and the *FAIR* requester.

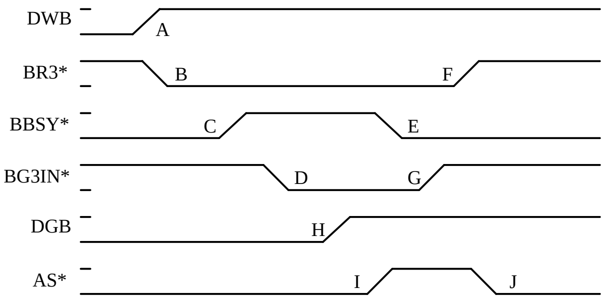


Figure 3-6 Early bus release timing

3.1.3.1 Release When Done

Requesters using the release-when-done option release the bus when they have completed their transfers. How it is established that the master or interrupt handler is done with the bus depends upon the board design. DMA modules may release the bus when they are done with a block transfer; CPU modules can give it up at the end of every bus cycle or under software control. The choice depends on the application. Most manufacturer's data sheets will describe how and when the bus is released.

Applications where the bus is released at the end of a block transfer cycle are common on peripherals with DMA (direct memory access) such as disk controllers. Popular DMA ICs, such as the MC68450, have a DONE* pin which signals the requester that it has completed its transfers.

Release-when-done requesters can also give up the bus at the end of every cycle. While quite simple to implement, this type of mechanism causes a large arbitration overhead to occur and should be avoided in applications where high bus bandwidth is required. A good application is on interrupt handler modules. Interrupt handlers must obtain the bus to fetch a STATUS/ID word (interrupt vector). When they are done with the fetch they can give up the bus to process the interrupt.

CPU modules with release-when-done requesters can give up the bus under software control. Often they release the bus when some local address location is accessed. Sometimes the BCLR* signal is connected to a local interrupt request line. The BCLR* interrupt handler routine then generates a bus cycle to the release address. This method has the disadvantage that it requires software overhead to release the bus. Its main advantage, however, is that it allows a large degree of flexibility in releasing the bus. In situations where a CPU must acquire and hold the bus for multiple transfers it may be the only solution.

For example, a non-intelligent, non-DMA disk controller slave may require its local CPU to perform many consecutive transfers during a disk read operation. This could be a problem in situations where many VMEbus masters are present. Other masters might acquire and hold the bus during the disk read cycles. With a software release mechanism, the CPU could hold the bus until it has completed the transfers.

The software release mechanism may speed up bus accesses in certain situations. Usually, VMEbus requesters perform some degree of *local arbitration* during every bus access, regardless of whether or not it currently owns the bus. The software release mechanism eliminates the need for this overhead. Since the on-board bus release cycle never takes place at the same time as an off board request, there is no need for arbitration.

The software release mechanism is desirable in single processor systems. Since bus arbitration never takes place, there is no need for any extra arbitration overhead.

To understand why most requesters (other than those with software release) must have a local (internal) arbiter, consider the block diagram of Figure 3-7. Here a requester (for a CPU) has an external release mechanism. This requester monitors two inputs: WANTBUS and RELEASEBUS. Whenever the CPU needs to perform a VMEbus cycle it asserts WANTBUS, and the requester acknowledges with GRANTBUS. At any time the CPU may lose bus ownership if the release mechanism asserts RELEASEBUS. If the requester has ownership of the bus, and WANTBUS and RELEASEBUS are asserted simultaneously, the requester must arbitrate between the two and either assert GRANTBUS or negate DGB. This arbitration will add at least 50 nanoseconds of overhead to every bus cycle. Clever requester designs remove the arbiter and use a software release method. For example, a CPU module may access an on-board register to release the bus. When this happens, an off-board access cannot take place since the microprocessor is busy accessing the bus clear register. Bus requests and bus releases never occur at the same time, and therefore arbitration in the requesters isn't required.

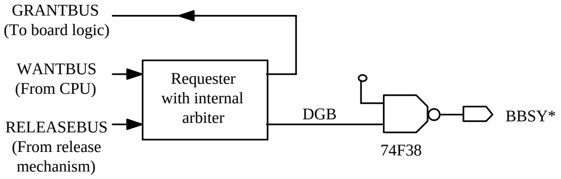


Figure 3-7 Some requesters have an internal arbiter which adds overhead to every bus cycle

The RWD mnemonic describes the release-when-done requester option.

3.1.3.2 Release On Request

The release-on-request requester releases the data transfer bus whenever another module requires it. Figure 3-8 shows a simple circuit for the release mechanism. If one or more of the bus request lines BR0* - BR3* are asserted by another module, the release (BUSREL) is triggered. This method is popular since it works well for most applications, is reasonably fast, and is transparent to the user. Its main disadvantage is that arbitration overhead becomes great when many masters need the bus at the same time. [For an explanation of arbitration overhead, refer to the previous section entitled Release When Done Requester].

The ROR mnemonic denotes the release-on-request option.

Other release mechanisms can be used in conjunction with RWD and ROR requesters. For example, some boards release the bus if ACFAIL*, SYSFAIL* or BCLR* is asserted. This clears the bus for other urgent activity.

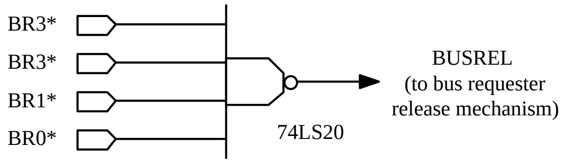


Figure 3-8 Simple release mechanism for release-on-request.

The ANSI/VITA 1-1994 (VME64) VMEbus specification requires that masters, which respond to the RETRY* signal, must release the data transfer bus whenever RETRY* is asserted. In terms of the requester, the hardware for a RETRY* bus release can be handled in much the same way as other masters during release-on-request. The main difference is that the master must terminate the current cycle on a RETRY* operation, and repeat it at a later time.

3.1.3.3 FAIR Requester

The round-robin arbiter assures that all requesters gain equal access to the bus when up to four masters are present. When five or more are needed, at least two masters must reside on a single bus grant daisy-chain. This lowers the bus bandwidth available to processors farthest down the daisy-chain. Those closest to the arbiter have an inherently higher bus priority. A requester which provides equal priority to all masters was introduced in the IEEE 1014-1987 bus specification, and is called the FAIR requester. These requesters can be used when many masters need fixed bandwidth.

The FAIR requester provides 'fairness' by preventing its master from gaining control of the data transfer bus until all of the other masters (on the same arbitration level) have obtained bus ownership at least once. It accomplishes this by holding off VMEbus requests until its designated VMEbus request signal (BR0* - BR3*) is negated. [Remember, the bus request signals use open-collector logic, and are negated only when there are no pending requests]. This insures that each master is given an equal share of bus bandwidth.

To illustrate the benefits of the FAIR requester, consider the system block diagram of Figure 3-9. Here six GPIB modules (with DMA) transfer data to a global memory module. Each GPIB module has a release-when-done requester. This is typical in data acquisition systems where multiple channels must off load data to global memory. All of the modules reside on bus request level BR3*, with an SGL arbiter. If all six boards request the bus at the same time, the module in slot two would acquire the bus first (it would be the first to receive the bus grant daisy-chain). This module would perform its data transfers and then release the bus. During the next arbitration cycle the module in slot three would obtain the bus. If the module in slot two, however, were to again request the bus, it would be the next bus master because of its proximity to slot 01. The modules in slots four, five, six and seven might not be able to transfer data. In effect the modules in slots two and three would use all of the available bus bandwidth.

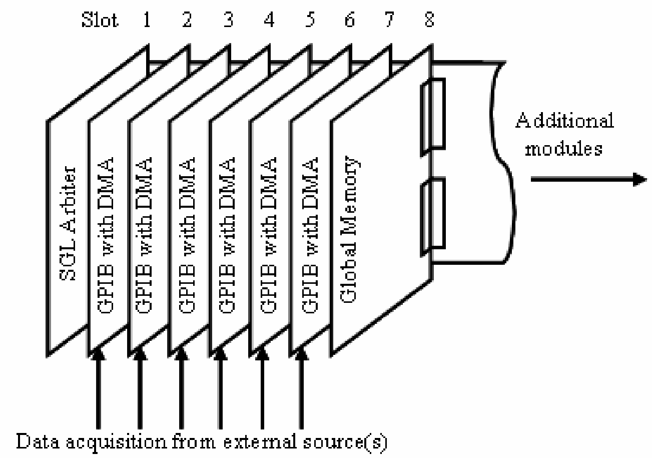


Figure 3-9 System example where a FAIR requester should be used

A good solution to this problem is to use the FAIR requester. As Figure 3-10 shows, these requesters do not assert their bus request signal until all of the other modules have completed their transfers. Each requester has a mechanism which prohibits it from asserting a bus request until the bus request line is negated (remember, BR0*-BR3* are open collector signals). When this happens all modules waiting for bus ownership assert their bus request output. In this way all the modules wait until no more requests are pending before issuing a new bus request. For this reason some manufacturers refer to it as a *request-on-no-request requester* (quite a mouthful).

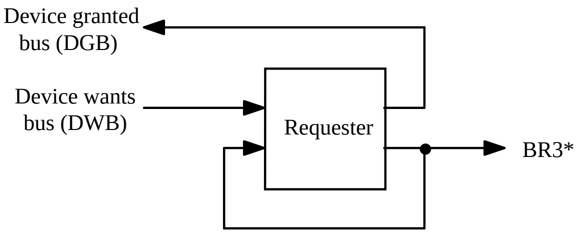


Figure 3-10 FAIR requesters

Monitors the VMEbus request lines and will not initiate a bus request until that level has been negated. This gives all masters equal bandwidth.

The FAIR requester can also speed system integration by reducing the time it takes to 'tune' the arbitration bus. Tuning is sometimes necessary to insure that all masters get sufficient bus bandwidth. It is accomplished by adjusting the arbitration method, the type of requester that each module uses, the ordering of the modules in the backplane and the bandwidth loading of each master. This can be a simple or laborious task, depending on the application.

Some software systems provide static and dynamic loading mechanisms to help tune the bus. These move application programs between processors so that all of them are given equal work. Systems with static loading are configured before run time, and those with dynamic loading do so at run time.

For the FAIR requester to work properly, all bus grant levels with multiple masters or interrupt handlers should have FAIR requesters. If one or more modules don't have one, it should be verified that the bandwidth used by the non-FAIR module is less than that used by the FAIR requester(s).Early Withdrawal of Bus Requests

Under certain circumstances bus requests can be withdrawn by a requester. For example, consider the situation where an intelligent slave (such as a disk controller) attempts to transfer data from a local buffer to VMEbus memory. For efficiency, many boards are designed so that they off-load data when a buffer becomes partially full, often using a microprocessor to do the transfers. When this happens the module's requester asserts its bus request output. Once it has asserted the request the module's microprocessor may need to service some local task (like an interrupt) while it is waiting for the VMEbus arbiter to grant the bus. When this happens many systems may wish to withdraw the request, and attempt the access again later. This is allowed (under certain conditions) by the VMEbus specification.

If a requester has a bus request pending, and if it sees some other requester assert BBSY*, then it may negate its request within 50 nanoseconds after BBSY* has been asserted (by the other requester). The stipulation that it must remove the request within 50 nanoseconds must be maintained so that the bus arbiter does not erroneously assert a bus grant, which would cause a system crash. Figure 3-11 shows the timing window when a requester may negate its bus request.

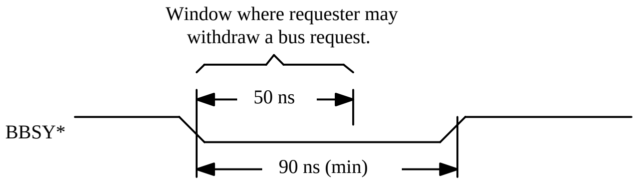


Figure 3-11 Timing window where a bus request may be withdrawn

Early withdrawal of bus requests should not be confused with the VMEbus early release mechanism (which is associated with address pipelining).

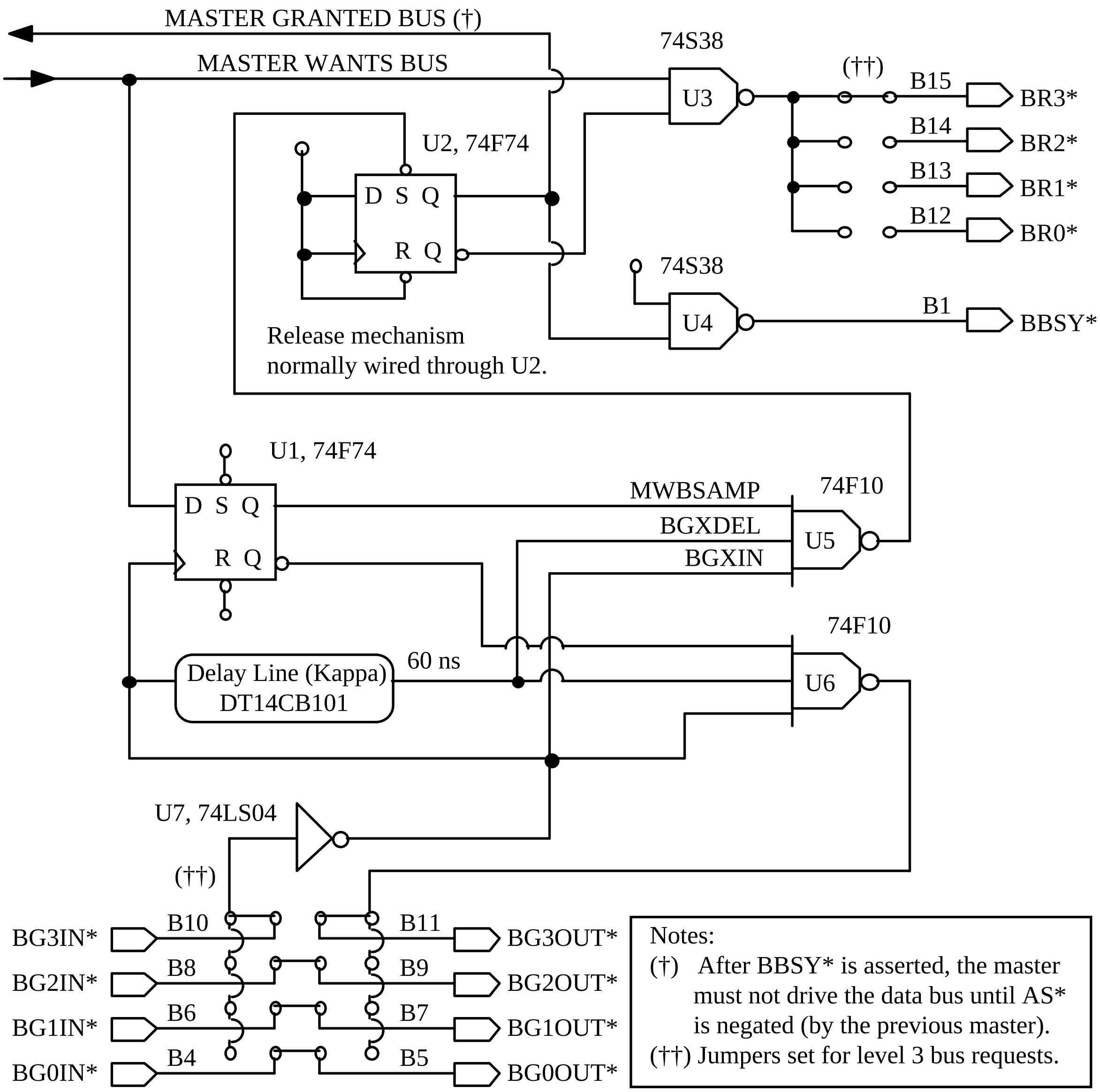
3.1.3.5 Circuit Idea - Asynchronous Bus Requester

Figure 3-12 shows a circuit idea for an asynchronous bus requester. Address decoding logic associated with the on-board master develops the active high signal MASTER WANTS BUS. When this signal is high, the requester asserts BRX*.

The system arbiter drives the bus grant daisy-chain in response to every bus request. When BGXIN* is asserted at the input of the requester, it determines whether to assume ownership of the bus or to pass the BGXOUT* daisy-chain. This is done with arbitration circuit U1, U5 and U6.

If MASTER WANTS BUS is high when BGXIN* is asserted, then the Q output of flip-flop U1 is set. After a 60 ns. delay, the requester negates BRX* and asserts BBSY* and MASTER GRANTED BUS using flip-flop U2. If MASTER WANTS BUS is low when BGXIN* is asserted, then the Q output of flip-flop U1 will be cleared, and U6 will assert the bus grant daisy-chain.

The master must not drive the data bus until the previous master has negated AS*. This prevents bus contention with other masters that utilize the early release feature of VMEbus. A circuit for accomplishing this task is not shown.



Notes:
(†) After BBSY* is asserted, the master must not drive the data bus until AS* is negated (by the previous master).
(††) Jumpers set for level 3 bus requests.

Figure 3-12 Circuit idea for an asynchronous bus requester.

The bus release mechanism is not shown (see text).

This circuit does not provide for a bus release mechanism. While all requesters must have a way of releasing the bus, it was omitted here because each of the release mechanisms (RWD, ROR and FAIR) would release the bus in a different way. The circuit would also change depending on whether the master implements the early bus release feature. The release would normally be wired into flip-flop U2 (which is disabled for the purposes of this example).

When designing bus release mechanisms, keep in mind that BBSY* *cannot* be negated until BGXIN* has been negated. There can be considerable delay between the time BRX* is negated by the requester, and when BGXIN* is negated at the module. This delay varies depending on the speed of the arbiter and the position of the board in the system. If BBSY* were negated before BGXIN* was negated, this requester could generate spurious outputs on BGXOUT*.

3.2 Arbitration Reliability Considerations

Noise in electronic systems is unavoidable and can never be eliminated. TTL buses in general are somewhat noisy and VMEbus is no exception. VMEbus has some specific problems which are relevant to many areas of interface design, especially the data transfer arbitration bus. These include ground shift, cross talk, metastability and wire-or glitches. VMEbus system integrators and designers should be aware of these problems.

3.2.1 BBSY* Noise

The bus arbiter is especially susceptible to noise on its BBSY* input. This noise mostly comes from inductive cross talk in the DIN41612 connectors, and occurs when data lines D00-D15 change state.

In single master systems these glitches are rarely a problem because bus arbitration takes place only once (when the master is first granted the bus). In multiprocessor systems this noise can seriously degrade the reliability of the system unless precautions are taken.

Before discussing the root causes and remedies of BBSY* glitches, it is useful to understand how bus arbitration is affected by them. There are two common scenarios of BBSY* noise which could affect the arbiter. These are shown as 'A' and 'B' in the timing diagram of Figure 3-13.

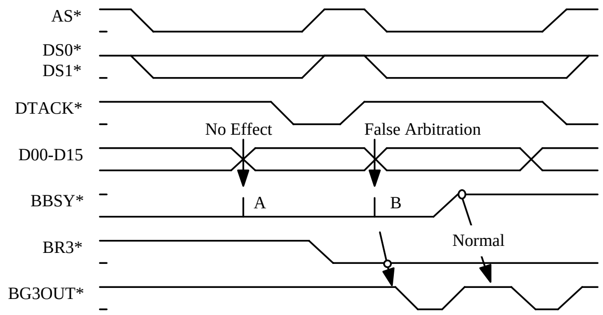


Figure 3-13 Three scenarios of BBSY* glitches

Scenario 'A' occurs when BBSY* is asserted, the bus request lines (BR3*) are negated, and the data lines switch thereby causing a positive BBSY* glitch. In most arbiters this glitch does not cause a problem since there are no pending bus requests. The arbiter assumes that the current master has simply relinquished the bus.

Scenario 'B', however, can cause serious system crashes. In this case a positive glitch occurs while a bus request is pending. This glitch could be (and often is) misinterpreted by the arbiter as a rising edge on BBSY*. The arbiter assumes that the present master has relinquished the bus, and asserts a bus grant (BG3*) to another requester. As a result, two masters acquire the data transfer bus, and crash the system.

Unfortunately, system crashes due to BBSY* glitches can go unnoticed or uncorrected until the final stages of system integration. It is usually first noticed as a random crash, and can be affected by application software. Furthermore, the problem often disappears when oscilloscope probes are attached to the data lines or BBSY* signal.

Glitches on BBSY* are caused by backplane cross talk and ground shift. The biggest contributor is ground shift, which is the variation of ground voltage from its ideal state of zero volts. It is caused by the inductance of power supply traces, connector pins and IC leads. While the term *ground shift* refers to ground (or power return), the concept applies equally well to signal and power supply traces. The phenomenon is certainly not limited to VMEbus. Manufacturers of integrated circuits and large digital systems have been familiar with the problem for years, especially as logic speeds have increased.

The term *ground shift* is sometimes referred to as *ground bounce*.

The intensity of BBSY* glitches will vary depending upon the inductance of the power return and the BBSY* signal trace inductance. Experimental data shows that these glitches can reach 2-3 volts and last more than 10 nanoseconds. This is sufficient to trigger 74ASXX logic ICs which need a glitch of only 1.4 volts and two nanoseconds in length to cause a false logic condition.

3.2.2 Reducing BBSY* Glitches

The effects of BBSY* glitches can be reduced by either a) removing the cause of the glitches or b) filtering the BBSY* signal wherever it is used. Practical experience shows that there is no single solution to this or any other noise problem. A total program of noise reduction techniques should be used.

3.2.2.1 Removing The Cause Of BBSY* Glitches

The best way to reduce BBSY* glitches is to eliminate the source of the problem. As described above, BBSY* glitches are caused by inductive coupling between signal traces and the adjacent BBSY* input line. For the most part, this coupling occurs within the DIN41612 connectors. Therefore, any way of reducing the inductive coupling will result in a lower excitation.

There are two effective ways of reducing inductive coupling: a) reducing the switching speed of logic ICs or b) reducing the inductance of signal traces.

Inductive coupling is proportional to the switching speeds of logic ICs. The faster an IC switches, the larger the current rate of change. [Those familiar with inductive circuits will recognize that the voltage across an inductance is V_l = L di/dt]. Therefore, BBSY* excitation will be lowered with slower rise-times on the data transceivers.

A simple circuit to investigate this effect is shown in Figure 3-14. The circuit allows BBSY* glitches to be measured using various 74XX245 or 74XX645 bus transceivers.

The BBSY* evaluation circuit uses a 74F74 flip-flop (U2) to divide a 16 MHz signal source (U1) to 8 MHz. This increases the fan-out and rise time of the 16 MHz TTL oscillator. Sample devices are installed in U3, and BBSY* glitches are measured on the VMEbus backplane.

Switch S1 allows the evaluation of BBSY* in both logic states. BBSY* can be asserted to investigate ground shift, and negated to investigate cross talk.

Figure 3-15 shows the relative BBSY* glitch intensities of several IC logic families. This data was obtained using the circuit of Figure 3-14. Clearly, BBSY* glitches can be reduced by the careful selection of logic families used on bus interface circuits. A general rule of thumb is that ground shift becomes more serious as logic edge speeds increase. The logic families on the right of Figure 3-15 have faster edge speeds than those on the left. The maximum numbers given above the bars are less important than the relative intensities of the various logic families. These numbers will vary depending upon the layout and connectors used in the circuit shown in Figure 3-14.

Another way to reduce inductive coupling between the data lines and the BBSY* pin is to use the 160 pin connectors. These connectors add sixteen additional ground pins on the 'z' row as shown in Figure 3-16. The 160 pin connectors were first allowed under the ANSI/VITA 1-1994 (VME64) specification.

The 160 pin connector reduces BBSY* noise by cutting the size of inductive loops created by the shortage of ground pins on the top of the P1/J1 connector. By adding additional ground pins on the 'z' row, the ground return loop area on the data and BBSY* signal pins are reduced, thereby cutting crosstalk.

One unfortunate problem with eliminating the *source* of BBSY* glitches is that they must be performed on every module in the system. This is not practical for most system integrators that buy modules. It is sometimes more practical to eliminate the effects of BBSY* glitches, especially the arbiter BBSY* input.

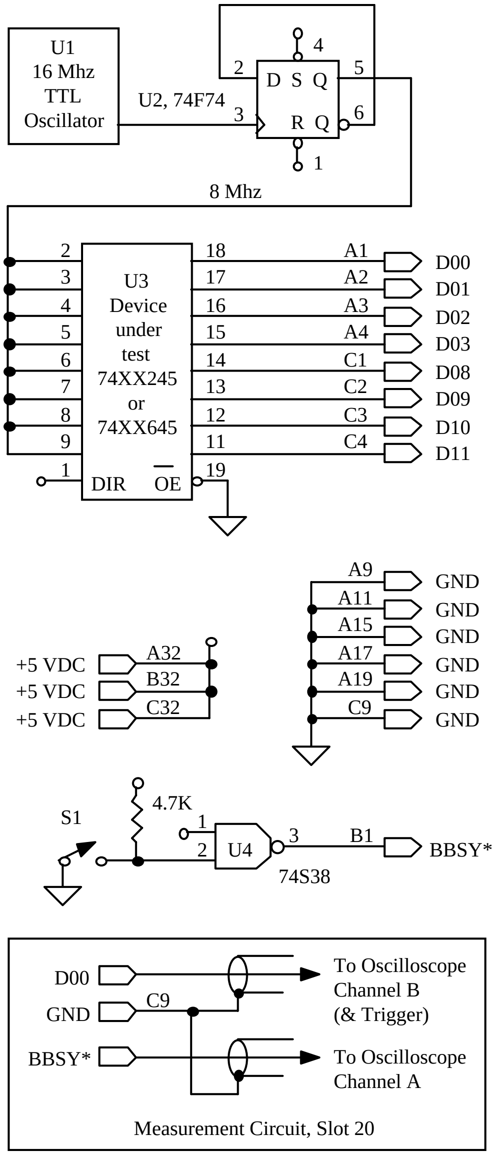


Figure 3-14 Simple circuit for evaluating BBSY* glitches and cross talk on VMEbus backplanes

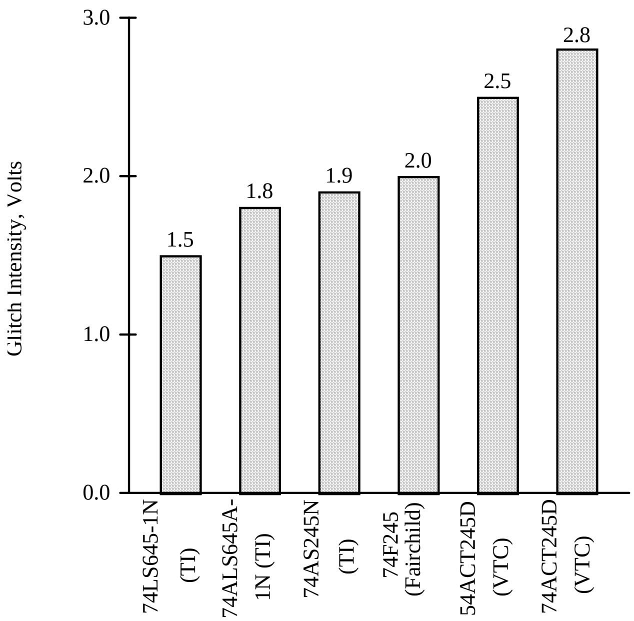


Figure 3-15 BBSY* glitch intensities of various logic families

The maximum numbers given above the bars are less important than the relative intensities of the various logic families. These numbers will vary depending upon the layout and connectors used in circuit of Figure 3-14. The relative intensities shouldremain about the same, however.

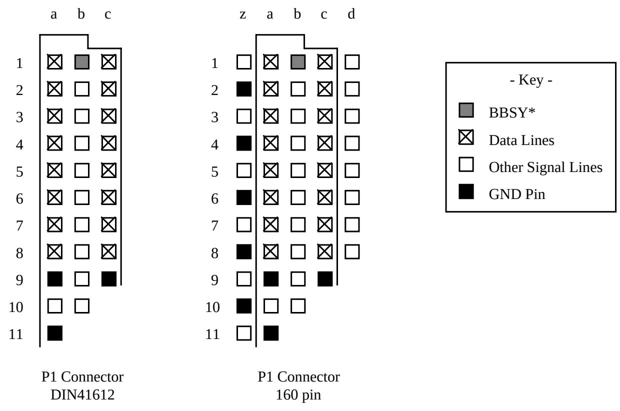


Figure 3-16 BBSY* glitches are caused mostly by trace inductance in the DIN41612 connectors

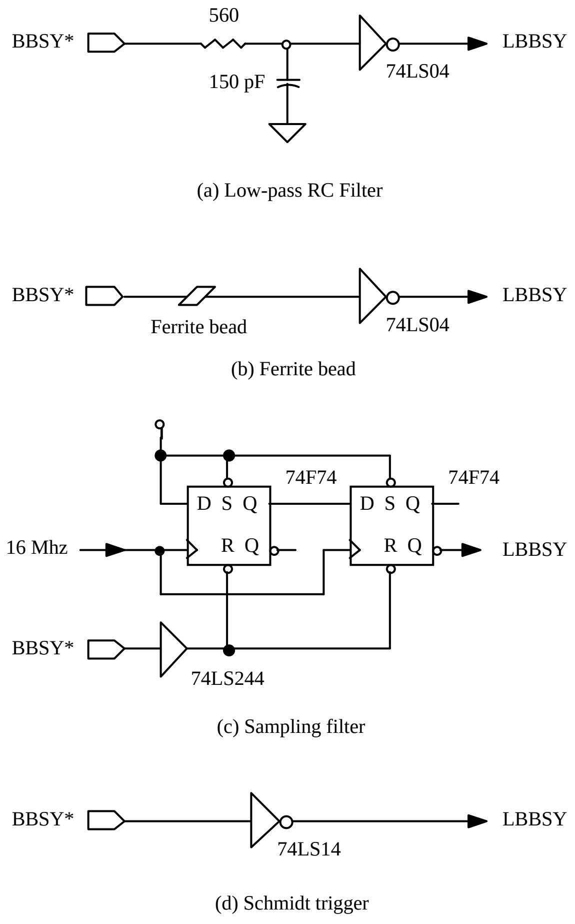
One way to minimize trace inductance in the connectors is to use a 160 pin connector, which has extra ground pins on the 'z' row.

3.2.2.2 Removing The Effects Of BBSY* Glitches

Many techniques are in use for reducing the effects of BBSY* glitches. These include adding inductors to signal pins and placing filters at the bus arbiter BBSY* input.

A clever solution to ground induced logic shifts was proposed by Mr. Richard DeBock of Matrix Corporation. His fix involves the addition of an inductor to steady state signal pins on the VMEbus module to divide the BBSY* glitch intensity down. An article which fully explains his technique is given in the *References* section at the end of this chapter.

Bus arbiters can also be designed to reject BBSY* glitches. Since the amplitude and duration of these glitches are somewhat predictable, simple filtering circuits can be used on the BBSY* input. Four popular ways to do this are shown in Figure 3-17. These include a low-pass RC filter, a ferrite bead, a digital sampling filter and a Schmidt trigger.



(a) Low-pass RC Filter

(b) Ferrite bead

(c) Sampling filter

(d) Schmidt trigger

Figure 3-17 Four ways to filter BBSY* glitches at arbiter inputs

The low-pass filter of Figure 3-17(a) rejects high frequency BBSY* glitches using an RC network. Its major advantage is its simplicity and ability to be field installed.

A more popular technique using a ferrite bead is shown in Figure 3-17(b). Like the RC network this solution works as a low pass filter, is rather simple and can be installed in the field. The ferrite bead can be placed over pin B1 of the P1/J1 connector as shown in Figure 3-18. Ferrite beads of this type can be obtained from:

FAIR-RITE Products Corporation
Wallkill, NY USA 12589
TEL: 914.895.2055

FAIR-RITE sells a *Bead, Balun & Broadband Kit* with a variety of samples (part number 0199000001). Four beads that fit onto pin B1 of the VMEbus P1/J1 connector are included with the kit. The only difference between them is their magnetic material (i.e. permeability). Part numbers for these beads are:

FAIR-RITE No.	Material	Maximum Permeability
2643000101	'43 Material	3000
2664000101	'64 Material	(try first) 375
2673000101	'73 Material	4000
2675000101	'75 Material	8000

Since BBSY* glitches don't exceed 10-15 nanoseconds, a digital sampling filter like the one shown in Figure 3-17(c) can be used. This circuit is especially useful because it can also be implemented on programmable logic parts. BBSY* is sampled on two consecutive 16 MHz clock edges using two 74F74 flip-flops. When BBSY* is asserted both flip-flops are reset and LBBSY is asserted (active high). When a rising edge occurs on BBSY* the flip-flops will negate LBBSY after two positive clock edges. This means the circuit will sample BBSY* every 62.5 nanoseconds, which means that it will not pass positive going glitches of shorter duration. The advantage of this circuit is its reliability and predictability. Its disadvantages are complexity and slow speed.

Another way to cut BBSY* noise is to place a Schmidt trigger receiver IC on the input as shown in Figure 3-17(d). The Schmidt trigger increases both the positive and negative going noise margins of the receiver. A 74LS04 inverter, for example, typically triggers on a positive going glitch at about 1.2 volts as shown in Figure 3-18. The 74LS14 Schmidt trigger will improve this to about 1.6 volts.

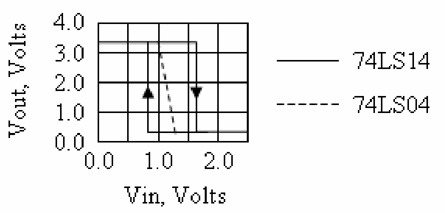


Figure 3-18 Vin vs Vout characteristics for 74LS04 and 74LS14 ICs shows that a 74LS14 has higher noise immunity

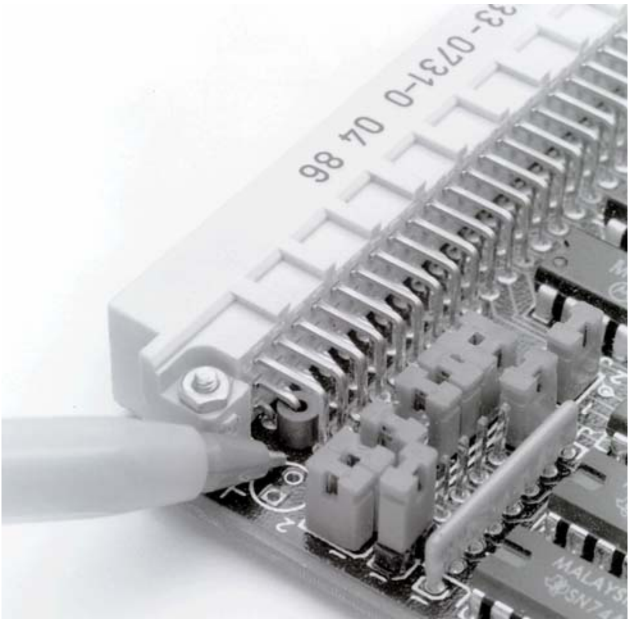


Figure 3-19 A ferrite bead can be placed over pin B1 of the P1 connector to reduce the effect of BBSY* glitches. Photo by Rob Bigelow

3.2.3 Metastability

Metastable glitches occur when the input set-up times of flip-flops are violated. Figure 3-20 shows how glitches can be generated at the output of a flip-flop when the clock and D inputs both change at the same time. The length and intensity of these glitches vary depending upon the logic family, ambient temperature, power supply voltages and other factors. Most manufacturers of integrated circuits will not specify the metastable behavior of their devices. Several references listed at the end of this chapter do give circuit design guidelines for various logic families, however.

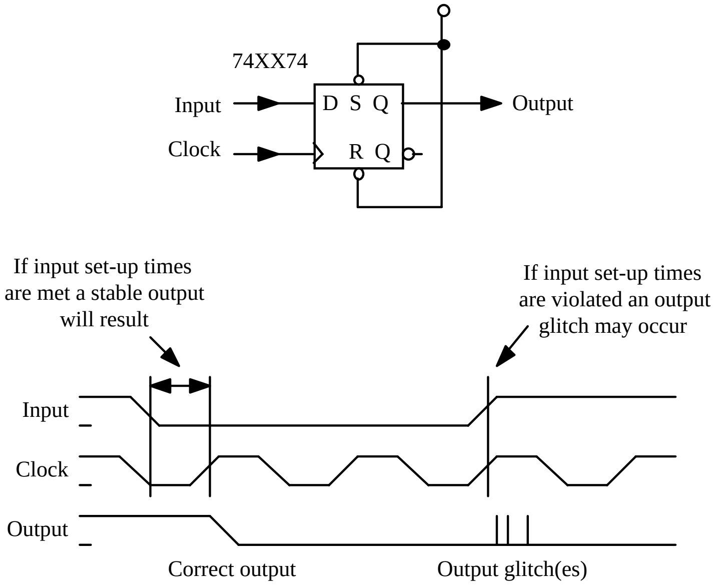

74XX74
Input
Clock
D S Q
R Q
Output
If input set-up times are met a stable output will result
If input set-up times are violated an output glitch may occur
Input
Clock
Output
Correct output
Output glitch(es)

Figure 3-20 Metastable state of flip-flops occurs when input set-up times are not met

In most microcomputer buses the metastability problem cannot be avoided. Failure to deal with it can cause unreliable system operation. The problem is made worse because system crashes due to metastability occur at random, and can be affected by software changes or the positioning of bus modules in the backplane. On VMEbus, metastability problems usually happen in circuits that have some type of arbiter. These include bus arbiters, requesters, interrupters and shared memory circuits.

For example, a common mistake on requester circuits is to latch a bus request using a bus grant input (BGXIN*). An example of this is shown in Figure 3-21. Here BG3IN* clocks the local device wants bus (DWB) signal. On the low to high transition of BG3IN* it clocks the state of DWB, and asserts BG3OUT* if the module did not initiate the arbitration cycle. Unfortunately BG3IN* is completely asynchronous to DWB and metastable glitches will occasionally occur on BG3OUT*, causing system crashes. This happens when a bus grant happens to be passing the board (from another bus requester farther down the daisy-chain) just as DWB is asserted.

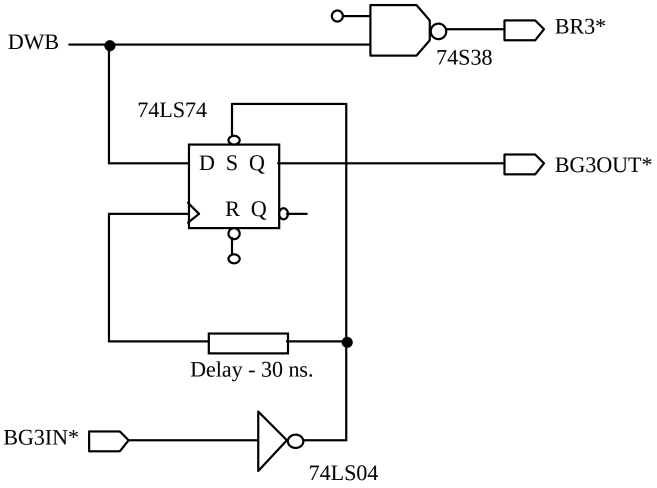


Figure 3-21 An example of a poorly designed requester circuit that generates glitches on BGXOUT*

The metastability problem can be avoided by altering the circuit as shown in Figure 3-22. In that circuit the bus grant is double clocked by a second flip-flop. A 60 ns. interval is added to allow any metastable glitches to dampen out.

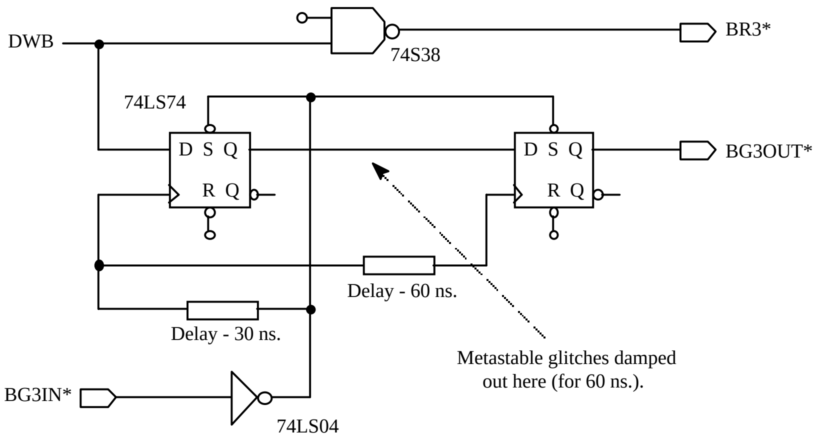


Figure 3-22 The circuit of Figure 3-21 re-designed to prevent metastable glitches on the BG3OUT* line

It should also be noted that the circuits of Figure 3-21 and 3-22 do not include any kind of local device grant signal. This was eliminated from the circuits for clarity. As an exercise, the reader may wish to design that part of the circuit.

Bus arbiters must also be designed to handle metastable glitches. The VMEbus specification requires that all bus request lines BR0*-BR3* must be latched before the arbiter can assert a bus grant. Since the bus requests are asynchronous to each other, these latches will occasionally go metastable. The arbiter may pass metastable glitches to the bus grant daisy-chains unless precautions are taken. The result of this, of course, is a random system crash.

Bus arbiters are often *synchronous* circuits because *asynchronous* arbiters can be slow or unreliable. Unfortunately, synchronous arbiters are notorious for glitchy operation because input synchronizers (flip-flops) become metastable. Clock speeds should be selected accordingly.

Bus requesters and interrupters usually have some sort of arbiter circuit because each must decide to accept or pass a daisy-chain. Since the daisy-chains are asynchronous to bus or interrupt requests, any flip-flops they use could become metastable. If this happens, the symptoms of the problem can sometimes be altered by changing the order of the modules in the system. If this helps or makes the problem worse, then check for glitches on the bus grant or interrupt acknowledge daisy-chains.

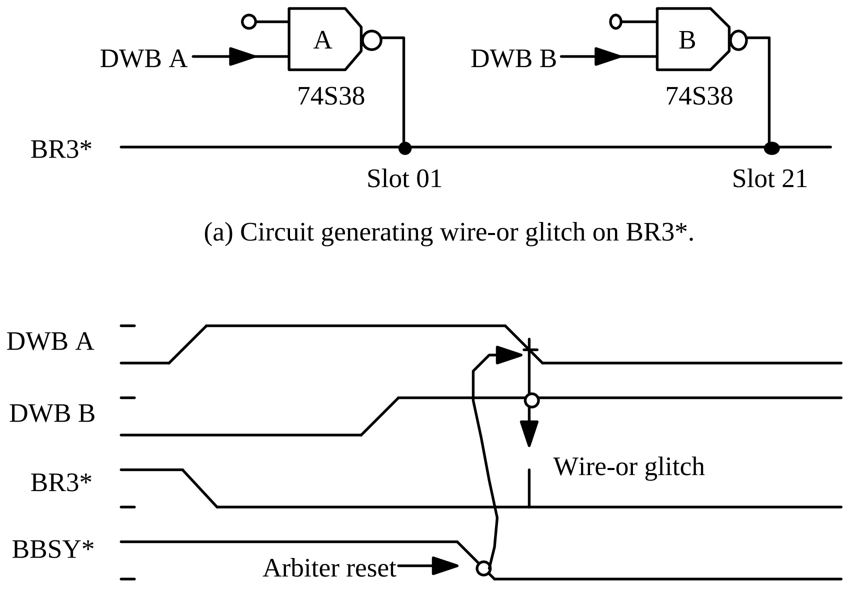
Bus modules with dual port RAM (shared memory) often have an arbiter to prevent concurrent accesses from VMEbus and a local processor. Since these accesses are completely asynchronous, they can cause metastable glitches. These can sometimes cause dual port ram logic to fail.

When evaluating or designing VMEbus interface circuits be sure to check for metastability. If the input of a flip-flop can change asynchronously with respect to its clock, the device will generate glitches. Check for metastability in bus arbiters, requesters, interrupters and shared memory arbiters.

3.2.4 Wire-or Glitches

Some VMEbus signals use wire-or logic. This logic, also known as open-collector logic, can sometimes cause glitches. Under the right circumstances these can show up on the bus request (BR0* - BR3*), interrupt request (IRQ1* - IRQ7*), ACFAIL*, SYSFAIL* and SYSRESET* signals.

Figure 3-23 shows how wire-or glitches are formed. That figure shows two wire-ored ICs, 'A' and 'B', driving a single backplane connection (BR3*). Assume, for example, that driver A asserts BR3* and a short time later driver B asserts BR3*. If driver A then negates its output a short wire-or glitch will be introduced on BR3*.



(a) Circuit generating wire-or glitch on BR3*.

(b) Wire-or glitch is generated when BR3* is negated.

Figure 3-23 Wire-or circuit and timing shows how glitch is generated

The problem comes from switching delays between wire-or drivers. In our example, driver A first turns on and sinks about 30 mA of current. When driver B turns on, however, its output will not have to sink much current (driver A is already doing that). When driver A turns off the load shifts to driver B. After driver A shuts off, however, it takes time for the load current to switch from A to B due to the propagation delay between the two ICs on the backplane. During this time a 2-3 Volt glitch lasting up to 5 nanoseconds may be generated.

Not all of the VMEbus open-collector signals will exhibit wire-or glitches. Some of these signals will never have two ICs driving a signal trace at the same time. For example, DTACK* can be an open-collector signal that never has more than one IC driving it at any time. The bus request, interrupt request, ACFAIL*, SYSFAIL* and SYSRESET* signals may, however. When evaluating or designing circuits that monitor these it is prudent to verify that they will tolerate the short glitches.

For example, bus arbiters can be designed to ignore wire-or glitches. VMEbus was designed so that bus requests (which are wire-or signals) can only be removed while BBSY* is low. This means that arbiters can be designed so they reset and ignore the bus request inputs when BBSY* is low. This eliminates the possibility of any glitches passing through to the bus grant outputs.

FAIR requesters are susceptible to wire-or glitches because they sample the bus request lines before generating a bus request. Without protective circuitry the wire-or glitch may cause these requesters to behave in an erratic manner. When evaluating or designing systems with FAIR requesters, make sure that wire-or glitches won't cause problems. One of the BBSY* deglitch circuits of Figure 3-17 could be used on the bus request sampling logic of FAIR requesters.

The wire-or glitch can also occur on the interrupt request lines IRQ1* - IRQ7*. Many popular microprocessors such as the 680XX series reject them by double clocking their interrupt inputs.

3.3 References

Chaney, Thomas J. "Measured Flip-Flop Responses To Marginal Triggering" *IEEE Transactions on Computers* December 1983

DeBock, Richard M. "Simple Solution Cures Glitches On High-Speed Buses" *EDN Magazine*. April 30, 1987

Johnson, Howard W. and Martin Graham. High-Speed Digital Design: A Handbook Of Black Magic Prentice Hall, Englewood Cliffs, NJ 1993

Kleeman, Lindsay and Antonio Cantoni. "Metastable Behavior In Digital Systems" *IEEE Design & Test of Computers* December 1987

Ott, Henry W. Noise Reduction Techniques In Electronic Systems Wiley, New York 1976

Peterson, Wade D. "How To Design A Single-Chip VME System Controller" *VMEbus Systems*. Mar-Apr 1987

Peterson, Wade D. "VME System Controller Reduces Parts Count" *EDN Magazine*. Mar 18, 1987

Taub, D.M. "Overcoming The Effects Of Spurious Pulses On Wired-Or Lines In Computer Systems" *Electronics Letters* April 28, 1983

Van Doren, Thomas P. Grounding And Shielding Electronic Instrumentation University of Missouri - Rolla 1988

VMEbus Specification / Revision C.1 PRINTEX. Phoenix, AZ 1985

VMEbus Specification / ANSI/IEEE STD1014-1987. VITA Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA Scottsdale, AZ 1995

Wendling et al. System For Arbitrating For Access On VME Bus Structures US Patent No. 5,321,818 1994

Chapter 4
Interrupts

In microcomputer buses, interrupts are used to gain the attention of a processor. They are used for task switching, message passing and obtaining CPU time. For example, a peripheral (like a disk controller) can interrupt a processor when it needs attention. After the peripheral generates an interrupt, the processor saves its current state and jumps to a program called an interrupt service routine. There it services the peripheral and then returns to what it was doing just before the interrupt.

Interrupts are analogous to reading the newspaper at home on your day off. Several things can happen that could cause you to interrupt your reading. If the telephone rings you make a mental note of where you are reading, and then answer the phone. When you are done with the call, you return to reading the newspaper. In this case a peripheral (the telephone) needs attention, causing you to stop what you are doing (reading the newspaper). You then perform a service (answer the phone), and then return to what you were doing.

Multiple interrupt levels are available on VMEbus. These can be prioritized so that certain events get serviced before others.

For example, in our newspaper analogy, assume that the telephone and the doorbell ring at the same time. You may wish to answer the door before the telephone. In this case the doorbell has a higher priority than the telephone.

Microcomputers prioritize interrupts the same way. This also gives the ability to ignore (mask off) some interrupt sources, an important feature in some applications.

4.1 VMEbus Interrupts

VMEbus has a seven level prioritized interrupt architecture. The seven levels are called IRQ1* - IRQ7*, with IRQ7* having the highest priority. Modules that generate interrupts, such as serial I/O boards, use a functional module called an interrupter. Modules that service interrupts, such as CPUs, do so with a functional module called an interrupt handler.

Figure 4-1 shows an interrupt in a sample system. In that example the interrupter asserts one of the seven interrupt request lines (to initiate the interrupt). Each of these is wire 'or'ed together (using open collector logic) and may be driven by any number of modules. Interrupt handlers monitor these signals and generate an interrupt acknowledge cycle in response to the requests. Each interrupt request line can be monitored by only one interrupt handler.

The interrupt acknowledge cycle performs two important functions. These include interrupt arbitration and a STATUS/ID (interrupt vector) read cycle. To generate an interrupt acknowledge cycle the handler first acquires the data transfer bus. This is necessary because it must obtain a STATUS/ID byte from the interrupter. All interrupt handlers must have a bus requester to acquire the data transfer bus. All modules that participate in the interrupt acknowledge cycle must support the passage of the STATUS/ID byte.

Once the bus has been acquired, the handler asserts address lines A01-A03, IACK* and AS*. The assertion of IACK* notifies all modules that the current cycle is an interrupt acknowledge cycle.

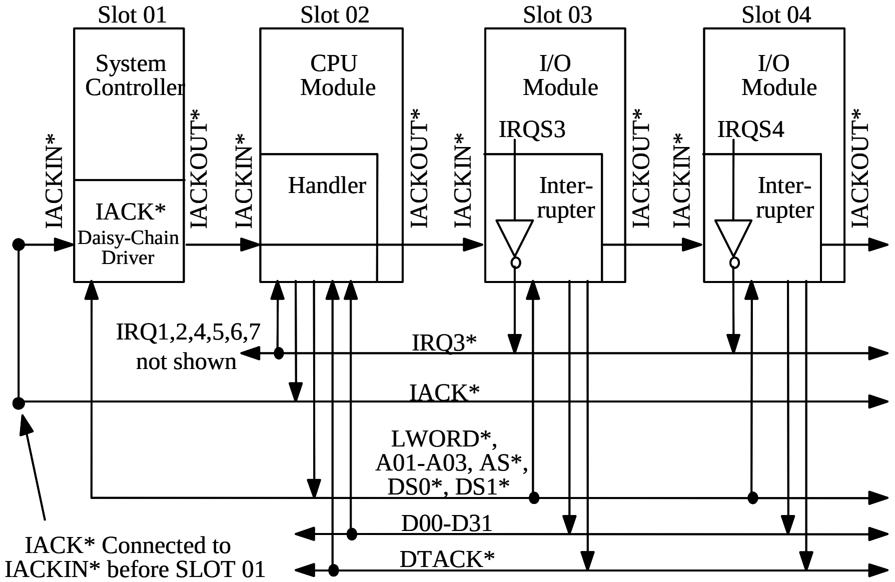


Figure 4-1 Interrupt example

The handler places a three bit address code on address lines A01-A03. This notifies all interrupters of the priority level being acknowledged. Table 4-1 shows how they are driven. The address modifier code AM0-AM5 is not used during the interrupt acknowledge cycle.

Table 4-1 Three bit code (A01-A03) used during interrupt acknowledge cycle

Interrupt line being acknowledged	3 bit code		
	A03	A02	A01
IRQ7*	1	1	1
IRQ6*	1	1	0
IRQ5*	1	0	1
IRQ4*	1	0	0
IRQ3*	0	1	1
IRQ2*	0	1	0
IRQ1*	0	0	1

When AS* and DS0* (and/or DS1*) are asserted, the IACK* daisy-chain is driven by the slot 01 IACK* daisy-chain driver. The daisy-chain propagates from module to module until it reaches the interrupter that initially requested the interrupt. That module then places an 8, 16 or 32-bit STATUS/ID (vector) onto data lines D00-D31, and terminates the cycle with DTACK*. The STATUS/ID byte can be used by the interrupt handler to determine which interrupter initiated the interrupt. This speeds up interrupts because the handler does not need to poll the interrupt sources to determine which one required service.

In the majority of systems, a CPU module functions as the interrupt handler. Typically, these cause an on-board microprocessor to jump to an interrupt service routine in response to the interrupt request. The VMEbus specification does not define what happens during the interrupt service routine. The interrupt service routine may or may not require use of the VMEbus backplane.

To illustrate an interrupt acknowledge cycle, consider again the block diagram of Figure 4-1. If the interrupter located in slot 04 generates an interrupt, it initiates an interrupt acknowledge cycle like the one shown in the timing diagram of Figure 4-2. Here the interrupt handler, located in slot 02, monitors and responds to interrupt requests on level IRQ3*. When it sees that IRQ3* is asserted, it arbitrates for the data transfer bus. Once it has obtained the bus it asserts IACK*, and places the level of interrupt it is acknowledging on A01-A03. The handler then asserts AS* and at least one data strobe.

Figure 4-2 shows a D08(0) interrupt acknowledge cycle. The levels of DS0* and DS1* indicate the width of STATUS/ID that the handler expects to receive from the requester. 8, 16 or 32-bit STATUS/ID words are allowed. Mnemonics describing the interrupter and handler types, and the selection of DS0* and DS1*, are shown in Table 4-2.

Virtually all commercial VMEbus modules pass an 8-bit STATUS/ID byte. Modules that pass 16 and 32-bit words during the interrupt acknowledge cycle are rare.

The VMEbus Revision B specification allowed only the D08(O) STATUS/ID. For this reason, interrupters may provide fewer bytes than requested by the handler. Undriven data lines are pulled up to a logic '1' by the backplane termination networks.

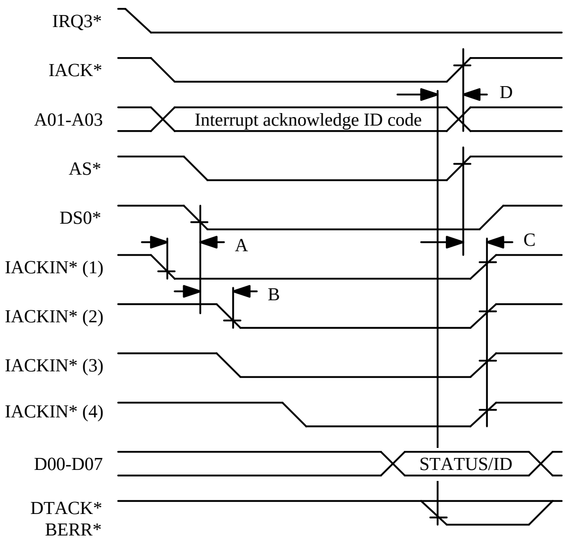


Figure 4-2 D08(O) interrupt acknowledge cycle

4.1.1 The IACK* Daisy-chain

The assertion of IACK* and one or both of the data strobes starts the IACKIN*/IACKOUT* daisy-chain. The daisy-chain is driven by the slot 01 IACK* daisy-chain driver. As shown in the timing diagram of Figure 4-2 (points A & B) it delays IACKOUT* until a data strobe has been asserted for some minimum time. Its purpose is to allow back-to-back interrupt acknowledge cycles, and to guarantee IACKIN* set-up times for interrupters.

The IACKIN*/IACKOUT* daisy-chain propagates from module to module until it reaches the interrupter that generated the request. When the responding interrupter receives IACKIN*, it places the STATUS/ID onto the data bus, and terminates the cycle with DTACK*.

As Figure 4-2 shows (point C), IACKIN*/IACKOUT* from each VMEbus module must be negated within some specified time after AS* is negated, regardless of the state of DS0* or DS1*. This provision is made so that back-to-back interrupt acknowledges may be performed. If not met, this timing parameter (40 ns.) could cause IACKIN* to remain asserted until the start of another interrupt acknowledge cycle, and cause a system crash. When evaluating or designing VMEbus modules, make sure that IACKOUT* is negated within the minimum specified time after AS* is negated.

All commercial VMEbus backplanes provide IACK* daisy-chain jumpers or some other method of closing the daisy-chain (which are discussed in Chapter 7). These are similar to those used for the bus grant daisy-chains. These jumpers allow propagation of the daisy-chain if one or more slots in the backplane are empty. For example, consider a 21 slot VMEbus system with only two modules: a slot 15 interrupter and a slot 01 handler. During the interrupt acknowledge cycle there would be no way for the daisy-chain to propagate between slots 01 and 15 if the intermediate slots were empty, and the system would crash. To prevent this, IACKIN* and IACKOUT* at each empty slot should be shorted together with a backplane jumper. This allows the daisy-chain to propagate in a normal fashion.

Table 4-2 STATUS/ID options for interrupt acknowledge cycles

Mnemonic	When applied to an	Means that
D08(O)	Interrupter	Responds to 8, 16 and 32 bit interrupt acknowledge cycles. These modules place an 8 bit STATUS/ID onto data lines D00-D07. They must monitor DS0* but not DS1* or LWORD*. They must not drive D08-D31.
	Interrupt handler	Generates 8 bit interrupt acknowledge cycles and reads an 8 bit STATUS/ID on D00-D07.
D16	Interrupter	Responds to 16 and 32 bit interrupt acknowledge cycles. These modules place a 16 bit STATUS/ID onto data lines D00-D15. They must monitor DS1* but do not drive data lines D16-D31.
	Interrupt handler	Generates a 16 bit interrupt acknowledge cycle and reads a 16 bit STATUS/ID on D00-D15.
D32	Interrupter	Responds to 32 bit interrupt acknowledge cycles with a 32 bit STAUTS/ID on D00-D31. These modules must monitor DS0*, DS1* and LWORD*.
	Interrupt handler	Generates a 32 bit interrupt acknowledge cycle and reads a 32 bit STATUS/ID on data lines D00-D31.

It is not necessary to place jumpers after the last module in the system. In this example, jumpers do not have to be placed in slots 16-21.

Most vendors of VMEbus modules include a printed circuit trace between IACKIN* and IACKOUT* on all non-interrupter modules. This eliminates the need to install jumpers in the backplane when modules are installed.

The ANSI/VITA 1-1994 VMEbus specification requires a shorting trace between IACKIN* and IACKOUT* on boards that do not generate interrupts. However, some versions of the specification did not require this. If you are unsure if a module has this trace refer to the manufacturer's data sheet or use an ohmmeter to determine if IACKIN* and IACKOUT* are connected.

4.1.2 Interrupt Acknowledge Cycle Timing

In general, the interrupt acknowledge cycle will follow the same timing rules as the read cycle. This includes address rot as Figure 4-2 shows (point D). When DTACK* is asserted by the interrupter, the handler will wait for an indefinite period before negating AS*, IACK* and A01-A03. The interrupter, however, must continue to drive the valid STATUS/ID onto the data bus until the handler negates both data strobes.

4.2 Interrupter

The interrupter functional module generates interrupts. Examples of interrupters include serial I/O boards and disk controllers. Board manufacturers use the I(x) and I(x-y) mnemonics to describe which interrupt levels can be used.

There are two classes of interrupters: release-on-acknowledge and release-on-register-access. The mnemonics ROAK and RORA are used to describe them. The ROAK interrupter negates its interrupt request line in response to an interrupt acknowledge cycle. The interrupt release is shown in the timing diagram of Figure 4-3. The ROAK mechanism works with all handlers.

The RORA interrupter releases its request when the handler accesses an on-board register during the interrupt service routine. As shown in Figure 4-4, the handler performs the acknowledge cycle, but the interrupter does not immediately negate its request. Sometime during the service routine the handler writes to a register on the interrupter. This causes it to negate the request.

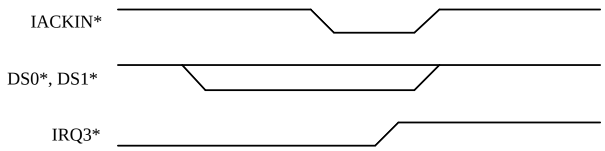


Figure 4-3 Timing for ROAK release mechanism

The interrupt request, in this case IRQ3*, is negated in response to the interrupt acknowledge cycle.

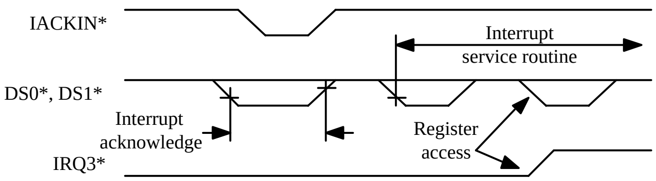


Figure 4-4 Timing for RORA release mechanism

The interrupt request, in this case IRQ3*, is negated when a register on the interrupter module is accessed by the handler.

The RORA release mechanism works with most CPU handlers because the CPU can access the interrupter's internal registers during the service routine. Handlers, however, are not required to have the capability to obtain the data transfer bus and do read/write cycles (they are only required to obtain the bus to read a STATUS/ID byte). If the handler cannot perform read/write cycles it is impossible to force the RORA interrupter to release its interrupt. In these cases an ROAK interrupter would have to be used.

The Revision B VMEbus specification allowed only the ROAK mechanism. This caused problems with some VLSI chips (such as serial and parallel I/O ports) and the RORA mechanism was added in Revision C. For example, the MC68681 serial port IC generates an interrupt from several sources. If two or more of these were to happen at the same time it would assert and hold its interrupt request until all were serviced. This type of request is called a level sensitive request because it is intended that a host CPU keep servicing interrupts as long as a valid request is present. Before the RORA interrupter, the logic necessary to interface this device to the VMEbus was cumbersome (and often impossible) to build. A good rule of thumb is that the ROAK release mechanism should be used only with edge sensitive interrupt requests.

The mnemonics used to describe the interrupt release mechanisms are summarized in Table 4-3.

Table 4-3 Mnemonics that describe interrupt release capabilities

Mnemonic	When applied to an	Means that
RORA	Interrupter	Releases its interrupt request line when some master accesses an on-board status or control register.
ROAK	Interrupter	Releases its interrupt request line when its STATUS/ID is read during the interrupt acknowledge cycle.

4.2.1 Circuit Example - Interrupter

Figure 4-5 shows a circuit for a simple ROAK interrupter. It is a clock generator which produces an interrupt 100 times each second. It can be used for a real time clock source, a task switching (heartbeat) timer or for any other timing purpose. The circuit can be easily adapted to handle other applications where edge-triggered interrupts are needed.

This circuit illustrates some of the key concepts of VMEbus interrupt generation. It shows how to initiate an interrupt, why the local interrupt must be latched during the interrupt acknowledge cycle, how to compare the acknowledge level, how to control timing and prevent race conditions, how to place a STATUS/ID byte onto the bus and when to negate DTACK* and IACKOUT*. The circuit also illustrates how the ROAK release mechanism works.

The 100 Hz time base is generated with a MM5369 timer (U1) and a 3.57 MHz crystal (which is a standard television color burst frequency in the US and Canada). The square wave output of U1 is latched by flip-flop U3 when switch K1 is open. U3 drives any of the seven interrupt lines IRQ1* - IRQ7* depending upon the state of jumper block K3. Figure 4-5 is set for level IRQ5*. The interrupt lines are driven by an open-collector driver IC, U9. This is a 74F38 device that can sink up to 48 mA of current as required by the VMEbus specification.

After the interrupter generates the interrupt, the handler responds with an interrupt acknowledge cycle. The interrupt handler initiates this cycle by placing the acknowledging interrupt level on A01 - A03, and asserts IACK*, AS* and the data strobe(s) DS0*/DS1*. The IACKIN*/IACKOUT* daisy-chain then propagates from module to module until it reaches the interrupter. When IACKIN* is asserted, the interrupter decides whether to acknowledge the interrupt or pass the daisy-chain. The decision flow for this process is shown in Figure 4-6.

The circuit of Figure 4-5 uses a 74ALS520 comparator (U8) and a 74LS08 AND gate (U2) to determine when a valid interrupt acknowledge is in progress. At the falling edge of DS0* (on every bus cycle) the state of the local interrupt request line IRQX is sampled using flip-flop U3. When IACKIN* and DS0* are both asserted, a rising edge propagates through delay line U7.

If this interrupter has an interrupt request pending, and the handler is issuing an interrupt acknowledge cycle on the same interrupt level as was requested, the output of U2 is asserted (active high). Flip-flops U4 and U5 latch the state of U2 at the rising edges of the 80 and 100 nanosecond taps of U7. If the interrupter participates in the cycle, it places a STATUS/ID byte onto the data bus and asserts DTACK*. If it does not participate in the cycle, then IACKOUT* is asserted.

A metastable state can appear on the output of U3 because it latches the state of the interrupt request asynchronously to the interrupt acknowledge cycle (i.e. the interrupt request could happen just as IACKIN* is asserted in response to some other interrupter). The 100 nanosecond delay provided by U7 allows this metastable state to settle out before the module asserts IACKOUT* or DTACK*.

DTACK* and IACKOUT* are negated under different circumstances. At the end of the interrupt acknowledge cycle, DTACK* is negated after the handler negates DS0*. IACKOUT*, on the other hand, must be negated within 30 nanoseconds after AS* is negated. This permits back-to-back interrupt acknowledge cycles.

This circuit does not determine if it should participate in the interrupt acknowledge cycle until IACKIN* has been asserted. Faster versions could latch the interrupt request and determine whether the module participates in the cycle by monitoring IACK*. All metastable states could settle out before the IACKIN* daisy-chain reaches the module.

This module responds to all requested widths of STATUS/ID, whereas some modules return only 16 or 32-bit STATUS/IDs. If it were designed to return a D16 or D32 STATUS/ID, and the interrupt acknowledge cycle required a D08(O) STATUS/ID, then the module would not respond to the cycle and would pass the IACK* daisy-chain. The flowchart of Figure 4-6 shows the decision tree used by this interrupter.

When evaluating or designing modules with interrupters, verify that race conditions do not occur on the IACK* daisy-chain. Just as with requesters, interrupters must have an internal arbiter. In the case of the interrupter, the arbiter must determine whether to return a STATUS/ID or assert IACKOUT*. In Figure 4-5 this arbitration is provided by the delay line and flip-flops U4 and U5.

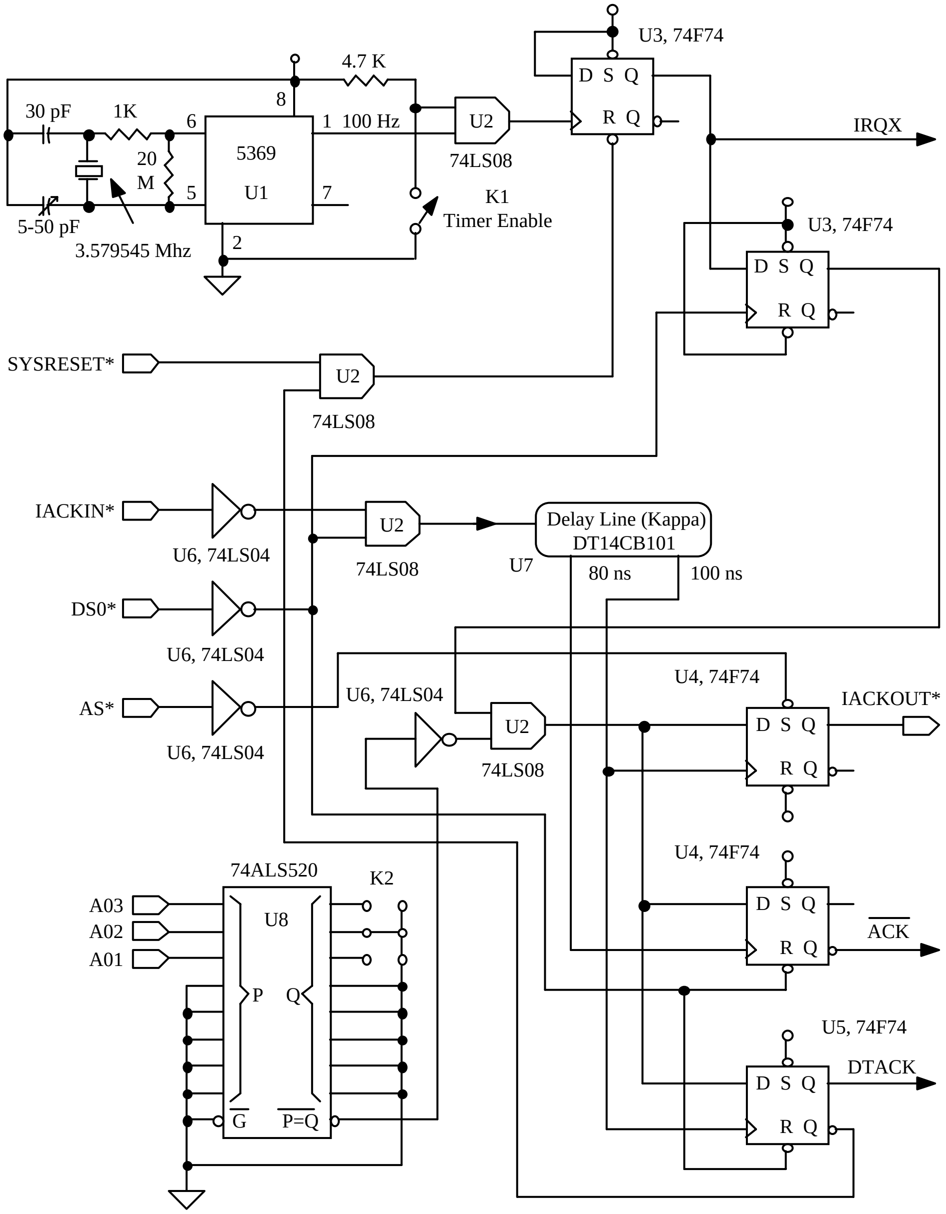
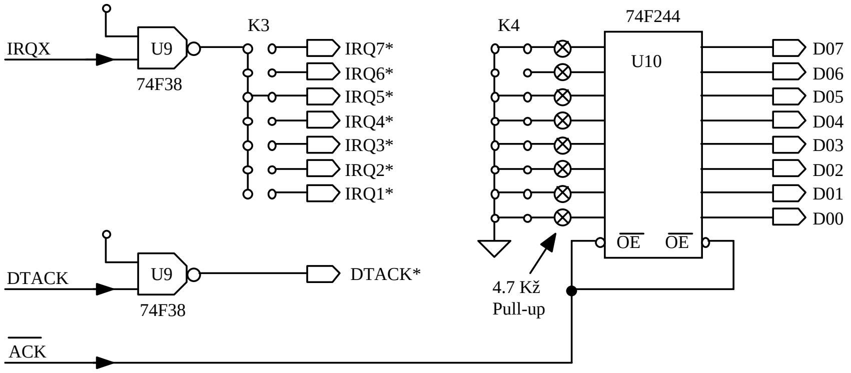


Figure 4-5 Simple ROAK interrupter circuit



4.2.2 Circuit Idea - The 68153 Bus Interrupter Module

When a sophisticated interrupter is needed, the 68153 Bus Interrupter Module (BIM) IC from Motorola (or from Modular Semiconductors) can be used. This device is a four channel interrupter for VMEbus. The block diagram of Figure 4-7 shows it configured as a slave in the short I/O address space. VMEbus masters can program the device using the registers shown in Figure 4-8. This programmability eliminates the need for jumpers on the module.

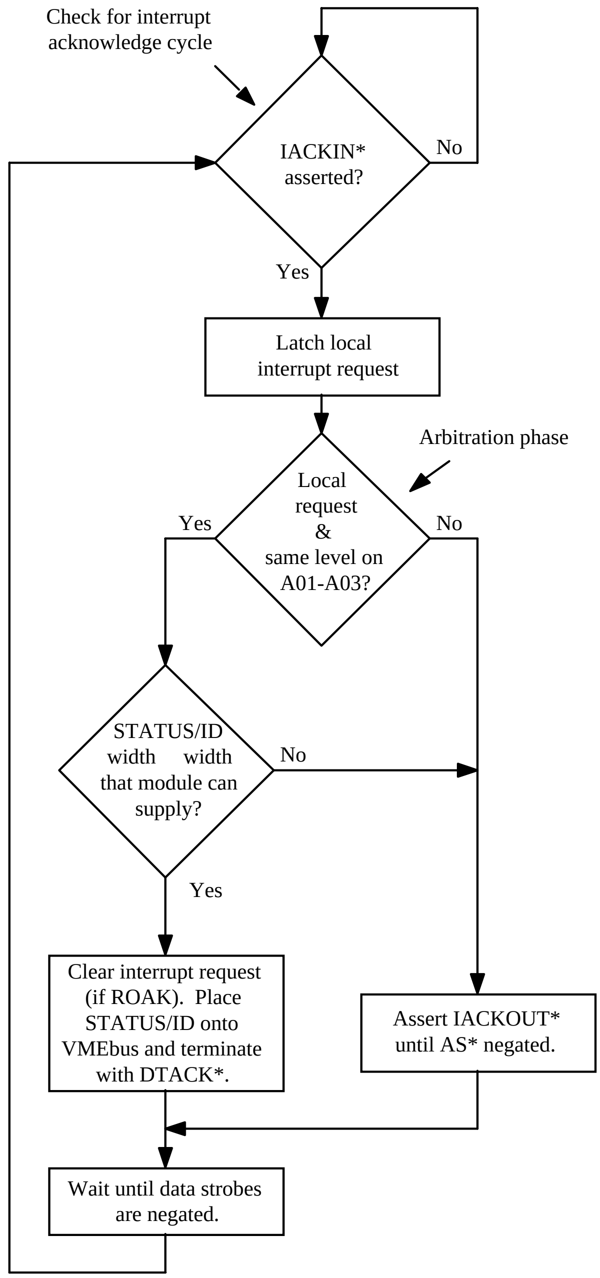

Check for interrupt acknowledge cycle
IACKIN* asserted?
No
Yes
Latch local interrupt request
Arbitration phase
Local request & same level on A01-A03?
Yes
No
STATUS/ID width width that module can supply?
No
Yes
Clear interrupt request (if ROAK). Place STATUS/ID onto VMEbus and terminate with DTACK*.
Assert IACKOUT* until AS* negated.
Wait until data strobes are negated.

Figure 4-6 Decision flow diagram for an interrupter

Interrupts are generated by asserting a local request on INT0*-INT3*. The 68153 BIM monitors these and asserts a VMEbus interrupt request on IRQ1* - IRQ7*. The level at which this interrupt is generated depends upon how the control register (bits D0-D2) for that channel are set. Each interrupt may be disabled by clearing bit D4 of the associated control register.

When the 68153 monitors an interrupt acknowledge cycle (IACK* and IACKIN* asserted) it responds by providing an 8-bit STATUS/ID or by passing the IACKOUT* daisy-chain to the next module. If it participates in the cycle it will respond in one of two ways. How it responds depends upon the state of the X/IN bit (D5) in the control register. If it is programmed for an external STATUS/ID it will place the device number (0-3) onto INTAL0-INTAL1 and assert INTAE*. This forces the interrupting peripheral to supply its own STATUS/ID byte. If programmed for an internal STATUS/ID it will place the byte in the vector register onto the data bus.

The 68153 automatically clears its interrupt request during the interrupt acknowledge cycle. It is classified as an ROAK interrupter.

The external decoder/control logic shown in Figure 4-7 must guarantee some bus timing for the 68153. For example, since the 68153 has no AS* input, IACK* must be qualified in the external logic. This external logic must guarantee that IACKIN*-IACKOUT* timing is correct. If the device is to reside in the slot 01 backplane position an external IACK* daisy-chain driver must also be added.

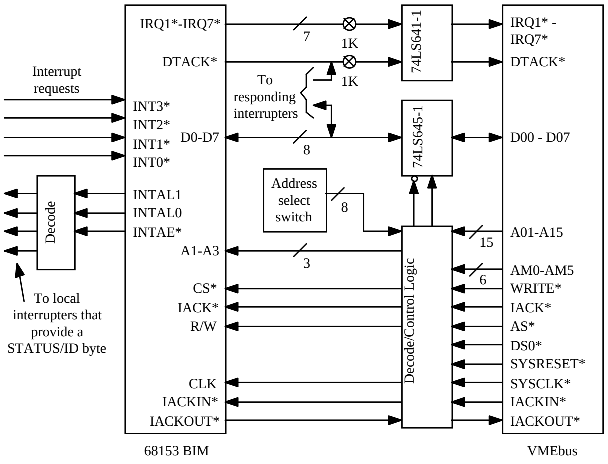


Figure 4-7 Block diagram of an interrupter using the 68153 Bus Interrupter Module

The device is shown as a slave in the short I/O address space.

68153 Register Set			
Address			Description
A3	A2	A1	
0	0	0	Channel 0 Control
0	0	1	Channel 1 Control
0	1	0	Channel 2 Control
0	1	1	Channel 3 Control
1	0	0	Channel 0 Vector
1	0	1	Channel 1 Vector
1	1	0	Channel 2 Vector
1	1	1	Channel 3 Vector

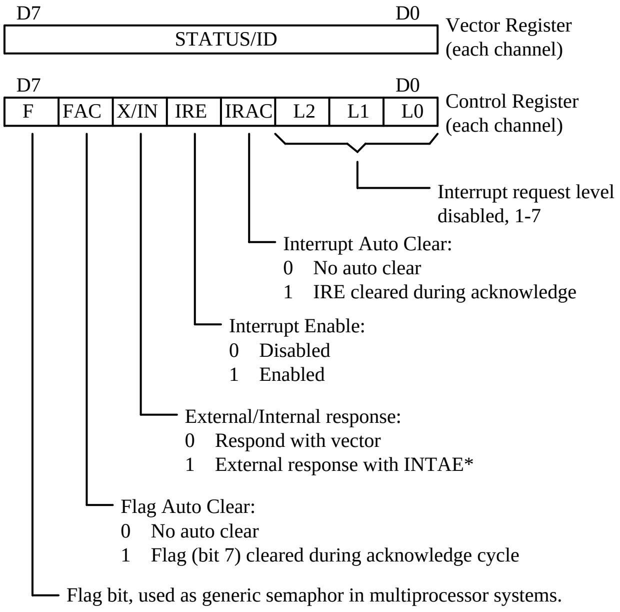


Figure 4-8 68153 Programmer's model

4.3 IACK* Daisy-chain Driver

The IACK* daisy-chain driver was first introduced in the Revision C VMEbus specification and permits back-to-back interrupt acknowledge cycles. It is located in slot 01 and is part of the system controller. Without the IACK* daisy-chain driver, the interrupt acknowledge daisy-chain might not get negated between back-to-back cycles. The net result could be a crashed system. This timing is shown in Figure 4-9.

The IACK* daisy-chain driver must be located in the Slot 01 backplane position. When a VMEbus board maker claims to have a system controller on a module, it should have an IACK* daisy-chain driver circuit.

The IACK* daisy-chain driver also guarantees that IACKIN*/IACKOUT* is negated for a minimum time between cycles, and that IACKIN*/IACKOUT* will not be asserted until after the data strobes have been asserted. This simplifies the design of interrupters.

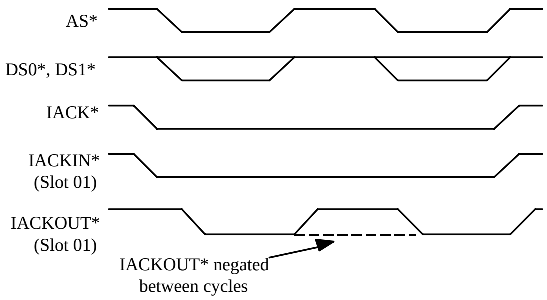


Figure 4-9 Timing diagram showing back-to-back interrupt acknowledge cycle

Without an IACK* daisy-chain driver IACKIN*/IACKOUT* would not be negated between cycles and could crash the system.

Most interrupt handlers, including MC680XX based CPU modules, negate IACK* between two interrupt acknowledge cycles, and therefore do not require the IACK* daisy-chain driver. It is necessary, however, that it be used in all systems since most manufacturers do not specify whether or not their modules produce back-to-back acknowledge cycles.

4.4 Circuit Example - IACK* Daisy-chain Driver

A simple circuit for an IACK* daisy-chain driver is shown in Figure 4-10. 40 nanoseconds after either data strobe (DS0* and/or DS1*) is asserted, flip-flop U2 latches the state of IACKIN*. If the cycle is an interrupt acknowledge cycle then IACKOUT* is asserted until AS* is negated. If it is not an interrupt acknowledge cycle, then IACKOUT* is not asserted.

If the circuit resides on a module located in slot 01, then S1 is configured as shown in the figure. If in any other slot, it should be configured to pass the IACKIN*/IACKOUT* daisy-chain.

The circuit of Figure 4-10 is part of the simple system controller circuit shown in Chapter 3.

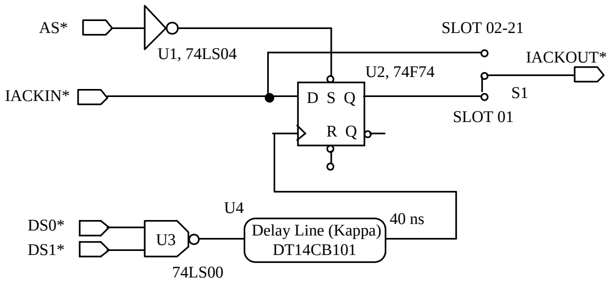


Figure 4-10 Simple circuit for an IACK* daisy-chain driver

4.5 Handler

Interrupt handlers can be designed to accept interrupts on levels one to seven. While any number of interrupters may reside on any level, only one interrupt handler may monitor each level.

Table 4-4 shows the mnemonics describing the handler options. Interrupt handlers with the mnemonic IH(x) can monitor a single level of interrupt. For example, a module that monitors level 5 would be called an IH(5) handler. When more than one level is handled the IH(x-y) mnemonic is used, where x and y are a range of interrupts. For example, a handler which monitors levels 2, 3, 4 and 5 is called an IH(2-5) handler.

Table 4-4 Interrupt handler options and their mnemonics

Mnemonic	Description
IH(x)	Generates interrupt acknowledge cycles in response to interrupt requests on level IRQx.
IH(x-y)	Generates interrupt acknowledge cycles in response to interrupt requests on level IRQx to IRQy.

By strict interpretation of the VMEbus specification, a IH(x-y) handler can only monitor consecutive interrupt request lines. For example, it may monitor levels 2, 3, 4 and 5 but may not monitor levels 2, 3, 5 and 6. In most cases this can be done, however. The only reason for this requirement is so the IH(x-y) mnemonic will apply in all situations.

4.5.1 Circuit Example - Handler For An MC680XX CPU Module

An example of a VMEbus interrupt handler is shown in the block diagram of Figure 4-11. Interrupts IRQ1* - IRQ7* are encoded to a three bit code (IPL0* - IPL2*) for use by the MC680XX microprocessor. When the MC680XX processor monitors a pending interrupt on these inputs, it initiates an interrupt acknowledge cycle. During the interrupt acknowledge cycle VIACK* is asserted by a decoder. After VIACK* is asserted, the local MC680XX processor expects to fetch a STATUS/ID byte from VMEbus. Before the byte can be fetched, however, VMEbus mastership must first be obtained using the bus requester. In most cases VIACK* is 'or'ed with off-board memory access requests. For simplicity this logic is not shown.

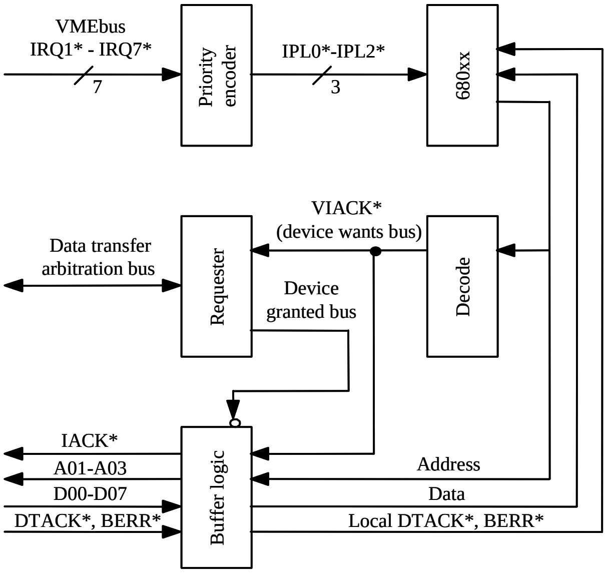


Figure 4-11 Block diagram of an interrupt handler on an MC680XX based CPU module. Portions have been omitted for clarity.

Once VMEbus ownership is obtained, an interrupt acknowledge cycle is generated. During the interrupt acknowledge cycle the local master asserts IACK*, sets A01-A03 to the level that is being acknowledged, and asserts AS* and DS0*. The interrupter then places the STATUS/ID byte onto the bus and asserts DTACK*. The local processor reads the byte and terminates the cycle.

When evaluating or designing interrupt handlers be careful with the 7 to 3 priority encoder between IRQ1* - IRQ7* and IPL0* - IPL2*. A proven solution is the circuit of Figure 4-12. Here a PAL16L8B is programmed as shown in the truth table of Table 4-5 and the equations of Figure 4-13. This circuit has a jumper block which allows programming of the interrupt levels to be handled. The example is set to accept interrupts on levels IRQ1*-IRQ5*, and to ignore levels IRQ6* and IRQ7*. A common mistake is to design the jumper block with a pull-up resistor as shown in the inset of Figure 4-12. The pull-up resistor violates the maximum input low current allowed for IRQ1* - IRQ7*. Chapter 6 describes the electrical characteristics of the bus interface in more detail.

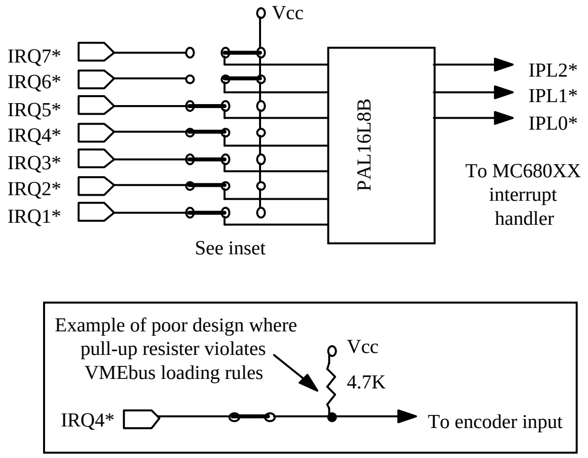

Vcc
IRQ7*
IRQ6*
IRQ5*
IRQ4*
IRQ3*
IRQ2*
IRQ1*
PAL16L8B
IPL2*
IPL1*
IPL0*
To MC680XX interrupt handler
See inset
Example of poor design where pull-up resister violates VMEbus loading rules
Vcc
4.7K
IRQ4*
To encoder input

Figure 4-12 Simple VMEbus priority encoder used with MC680XX handlers

Table 4-5- Truth table for PLD shown in Figure 4-12

Truth Table									
IRQ7*	IRQ6*	IRQ5*	IRQ4*	IRQ3*	IRQ2*	IRQ1*	IPL2*	IPL1*	IPL0*
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	1	1	0
1	1	1	1	1	0	X	1	0	1
1	1	1	1	0	X	X	1	0	0
1	1	1	0	X	X	X	0	1	1
1	1	0	X	X	X	X	0	1	0
1	0	X	X	X	X	X	0	0	1
0	X	X	X	X	X	X	0	0	0

A second problem has become more prevalent as faster microprocessors come onto the market. This problem is a result of the sampling mechanism used by the MC680XX interrupt inputs. Consider the timing waveforms for the MC68020 IPL0* - IPL2* inputs shown in Figure 4-14(a). Since the IPL0* - IPL2* inputs are completely asynchronous to the microprocessor clock, they must be synchronized before its internal logic can use them. The MC68020 does this by clocking the inputs on two consecutive negative clock edges. In addition to synchronizing the inputs, the microprocessor also verifies that the same interrupt level is pending on two consecutive clocks. This is because priority encoders of the type shown in Figure 4-12 take time for their outputs to settle. This settling time is shown in the timing waveform of Figure 4-14(a). There, two interrupt requests occur on levels IRQ2* and IRQ5*. The first negative clock latches the IRQ2* input, and the second IRQ5*. Since they were not the same, the MC68020 rejects the request until the next negative clock edge. In general this clocking mechanism works quite well. It can cause problems, however, if a slow priority encoder is used.

```
TITLE             INTERRUPT ENCODER
PATTERN    VH0005.PDS
REVISION   A
AUTHOR            WADE PETERSON
COMPANY    (C) 1988 WADE PETERSON, ALL
RIGHTS RESERVED
DATE       MARCH 27, 1988
CHIP ENCODE PAL16L8

IRQ7 IRQ6 IRQ5 IRQ4 IRQ3 IRQ2 IRQ1 NC NC
GND
OE IPL2 IPL1 IPL0 NC NC NC NC NC VCC
EQUATIONS
/IPL2 = /IRQ7
     +  IRQ7 */IRQ6
     +  IRQ7 * IRQ6 */IRQ5
     +  IRQ7 * IRQ6 * IRQ5 */IRQ4
/IPL1      = /IRQ7
     +  IRQ7 */IRQ6
     +  IRQ7 * IRQ6 * IRQ5 * IRQ4 */IRQ3
     +  IRQ7 * IRQ6 * IRQ5 * IRQ4 * IRQ3
*/IRQ2
/IPL0      = /IRQ7
     +  IRQ7 * IRQ6 */IRQ5
     +  IRQ7 * IRQ6 * IRQ5 * IRQ4 */IRQ3
     +  IRQ7 * IRQ6 * IRQ5 * IRQ4 * IRQ3
* IRQ2 */IRQ1
```

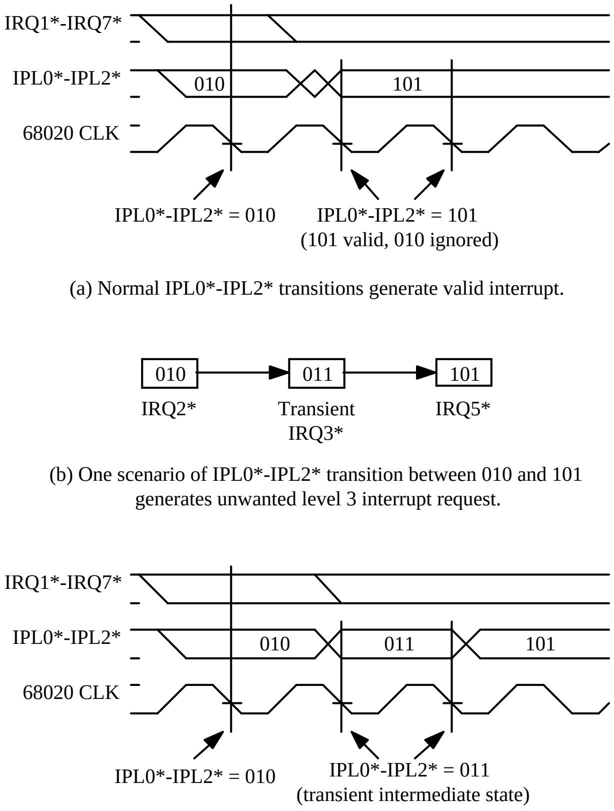
Figure 4-13 Logic equations for the PLD shown in Figure 4-12.

When an encoder, like the one shown in Figure 4-12, changes states it can go through intermediate transient states. In our example, where IRQ5* is asserted just after IRQ2*, IPL0* - IPL2* can occupy the IRQ3* state shown in Figure 4-14(b). If the encoder were slow it could occupy this intermediate state for some time and trigger a level three interrupt as shown in Figure 4-14(c). The result would be a random system crash (probably rare) when a level three interrupt acknowledge cycle occurs.

A fast priority encoder fixes this problem. If we were to use a 25 MHz MC68020 this means that we would need an encoder that would change states within 40 nanoseconds (1/25 MHz = 40 ns). If this requirement is met we should have no problems with our circuit.

While the MC68020 clocks the IPL0* - IPL2* inputs on every negative edge, the MC68000 does so on its positive and negative edges. Since this represents twice the sampling frequency of the MC68020, a 12.5 MHz MC68000 would also need a 40 nanosecond encoder. The situation is exacerbated if the microprocessor clock duty cycle degrades.

When evaluating or designing VMEbus interrupt handlers, be sure to check them for the wire-or glitch problem. The wire-or glitch was described in Chapter 3, and comes from using open-collector (wire-or) logic. MC680XX microprocessors guard against this by double clocking their inputs. When using other microprocessors on VMEbus they should be evaluated for compatibility with wire-or interrupt logic.



(a) Normal IPL0*-IPL2* transitions generate valid interrupt.

(b) One scenario of IPL0*-IPL2* transition between 010 and 101 generates unwanted level 3 interrupt request.

(c) A slow encoder with a fast MC68020 may cause a system crash due to transient interrupt request.

Figure 4-14 Poor choice of priority encoder may cause problems on a MC680XX µP

4.6 References

MC68153 Bus Interrupter Module Technical Data Sheet Motorola Inc.

VMEbus Specification / Revision C.1. PRINTEX Phoenix, AZ 1985

VMEbus Specification / ANSI/IEEE STD1014-1987 VITA, Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA, Scottsdale, AZ 1995

Chapter 5
Utility Functions

The Utility Bus is used for system initialization, periodic timing and power failure. It's really not a bus at all, but the VMEbus specification calls it one.

5.1 System Clock

SYSCLK is a general purpose 16 MHz clock signal. It has no relationship to other bus timing, and can be used for any purpose. Typical applications for SYSCLK include DTACK* generators, bus timers, memory refresh circuits, serial I/O time bases and synchronous state machines.

SYSCLK is generated by the slot 01 system controller, and can be used by any module. It is never turned off, even during system reset.

5.1.1 Timing

Figure 5-1 shows SYSCLK timing. At best it has a 50% duty cycle, but this can vary between 40% and 60%. Because SYSCLK is driven over the backplane, its wave shape may not always be ideal. Capacitance and inductance of the backplane, impedance mismatches and bus loading will cause distortion of the waveform as Figure 5-2 shows. SYSCLK is allowed to vary as much as 1.6% from its ideal frequency of 16 MHz (over voltage and temperature).

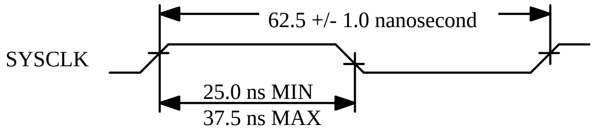


Figure 5-1 SYSCLK timing

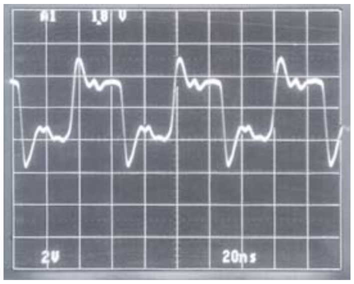


Figure 5-2 Typical SYSCLK wave shape

Photo by Wade Peterson.

The duty cycle of SYSCLK can vary between 40% and 60%, and care should be taken on circuits using both the rising and falling edges. When evaluating or designing VMEbus modules that use SYSCLK, make sure they can tolerate the entire duty cycle range.

5.1.2 Circuit Example - SYSCLK Driver

A simple SYSCLK driver circuit is shown in Figure 5-3. A TTL crystal oscillator (U1) generates the 16 MHz clock, and guarantees the frequency and duty cycle of the signal. The oscillator cannot directly drive VMEbus. A 74S244 (U2), which can sink up to 64 mA of current, buffers the signal. SYSCLK can be turned off by opening S1. This allows the circuit to reside in any slot of the system. When it's in slot 01, then S1 should be closed. This circuit is also part of the simple VMEbus system controller circuit shown in Chapter 3.

5.2 SYSRESET* And SYSFAIL* Signals

Microcomputer buses must provide a way to orderly start up and shut down the computer system. This is accomplished using the SYSRESET*, ACFAIL* and SYSFAIL* signals.

5.2.1 System Initialization

The SYSRESET* signal is used to reset the VMEbus system. During power-up, the VMEbus specification requires that SYSRESET* remain asserted for at least 200 milliseconds after the +5 VDC power supply has stabilized. However, it is recommended by the author that this be extended to at least 500 milliseconds. This gives enough time for power to stabilize in heavily loaded systems.

Usually, at least one reset switch is also provided on a system. This allows a restart without powering down the entire system.

When SYSRESET* is asserted, bus modules initialize themselves, and masters suspend all bus cycles. All signals except SYSCLK*, ACFAIL* and SYSFAIL* must be negated or three-stated within the times specified by Table 5-1.

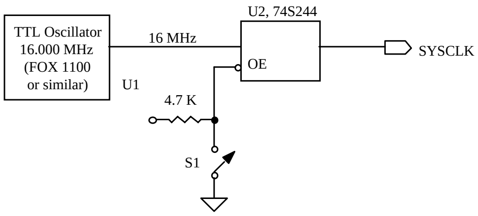


Figure 5-3 Simple SYSCLK driver circuit

The timing rules given in Table 5-1 can be exploited when evaluating or designing VMEbus interface circuits. For example, an interrupter must negate IACKOUT* within 30 microseconds after a system reset. Some designers of interrupter circuits negate IACKOUT* when SYSRESET* is asserted. Some will rely on the fact that masters and interrupt handlers must negate AS* within 20 microseconds after SYSRESET* is asserted. In these cases IACKOUT* can be negated after AS* is negated. This makes the bus interface circuit simpler and more reliable.

If any VMEbus module needs more than the minimum 200 millisecond reset time, it may keep SYSRESET* asserted after the normal power-up delay. This is possible since SYSRESET* is an open collector signal and any number of VMEbus modules may drive it.

Table 5-1 Module drive during start-up and shut-down sequences

Module Type	Shall not	After this time, following SYSRESET*
Masters Interrupt Handlers	Assert AS*, DS0* or DS1*	5 µs
	Drive IACK*, LWORD*, AS*,DS0*, DS1*, AM0-AM5,A01-A31, WRITE*, or D00-D31	20 µs
Slaves Interrupters	Drive D00-D31, DTACK*, BERR* or RETRY*	30 µs
Interrupters IACK* DCD	Drive IRQ1*-IRQ7*, Assert IACKOUT*	30 µs
Bus Timer	Drive BERR*	30 µs
Arbiter	Assert BG0IN*-BG3IN*	5 µs
	Drive BG0IN*-BG3IN*	30 µs
Requesters	Drive BBSY* Assert BGXOUT*	30 µs

5.2.2 Circuit Example - SYSRESET* Circuit

A simple circuit to generate SYSRESET* is shown in Figure 5-4. This circuit provides both the power-up and manual reset functions.

On power-up, SYSRESET* is asserted for at least 200 milliseconds after the +5 VDC power supply stabilizes. Capacitor C1 charges through resistor R1 until the threshold voltage of U1 is reached (about 1.6 volts). Once it has, SYSRESET* is negated.

SYSRESET* may be asserted manually by using switch S1. When the switch is closed capacitor C1 is discharged through R2. The value of R2 is selected to keep the discharge current through the switch at less than one ampere. When the switch is opened C1 is again allowed to charge through R1. This causes SYSRESET* to be driven for at least 200 milliseconds on a manual reset, with switch debouncing inherent to the circuit.

A 74LS14 Schmitt trigger is used for U1. The input voltage to this device changes slowly because SYSRESET* glitches may result if standard TTL IC inputs are used.

If the power is turned off and on again, C1 discharges through diode CR1. The 1N5817 is a Schottky diode with a forward voltage drop of about 0.3 Volts. When Vcc drops to zero, CR1 becomes forward biased and conducts. Without this diode, SYSRESET* would not be re-asserted unless power is kept off for an extended length of time. Almost any Schottky diode may be substituted for CR1.

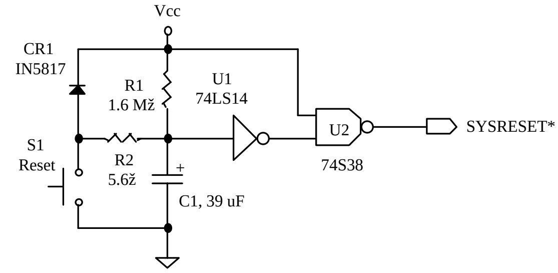


Figure 5-4 Simple SYSRESET* circuit

Resister R1 is abnormally large (1.6 Megohm) for an RC time constant of 200 milliseconds. This is because all the charging current to C1 does not come through R1. Since U1 is a TTL device, current is sourced by its input when it is low. Immediately after power-up U1 sources much of the current to C1. As the input voltage rises, U1 contributes less and less current. CR1 also has some reverse (off) current which also charges C1. Using components with standard tolerances will likely cause the SYSRESET* delay to exceed 200 milliseconds. 300 - 400 milliseconds delays are not uncommon with this circuit. This should not cause problems since the delay is confined to reset cycles.

It is recommended that intelligent VMEbus modules have a local reset capability. This permits a single module to be restarted, without resetting the entire system. This greatly simplifies software debugging in real-time systems.

5.2.3 Diagnostic Capability

VMEbus has a diagnostic and system failure capability. During start-up each bus module can be tested to determine if it is functioning. Once the module has been successfully tested the system 'boots' normally. If a module fails the test, the system can stop and display error messages. A utility signal called SYSFAIL* can be used to aide in system diagnostics.

VMEbus modules are not required to drive SYSFAIL*. It is highly recommended that they do, however.

SYSFAIL* is an open-collector class signal. During system reset, SYSFAIL* is asserted by any of the bus modules. If the module passes inspection its SYSFAIL* output is negated. If it fails it remains asserted. Most modules that drive SYSFAIL* also have a front panel status indicator, usually a RED/GREEN LED. During system reset all modules turn their LEDs red. If a module passes inspection its LED turns GREEN (when its SYSFAIL* driver is negated). This provides a quick and easy way to debug system problems. Modules that have SYSFAIL* capability often have a software status indicator in addition to the front panel display. This allows more sophisticated troubleshooting techniques.

Intelligent VMEbus modules often have on-board firmware diagnostics. After system reset these modules check themselves and negate SYSFAIL* if they are functional. VMEbus modules without a microprocessor can't do this. For example a memory card can't check its own memory, but another VMEbus master can. Non-intelligent slaves must provide a status byte to be accessed by the external master. On memory modules it is often located in the short I/O address range.

The SYSFAIL* signal can be driven anytime during normal operation. This is useful for early detection of catastrophic problems. For example, industrial systems can connect SYSFAIL* to an external emergency stop circuit, preventing erratic movements of a machine. Robotic machines are a classic case where a failed computer may cause a property or life threatening situation. If any part of the computer detects a failure, the system is immediately shut down using SYSFAIL*.

5.2.4 Circuit Example - SYSFAIL* Driver

SYSFAIL* timing during system reset is shown in Figure 5-5. A circuit which provides this timing is shown in Figure 5-6. This circuit asserts SYSFAIL* when SYSRESET* is asserted, and turns the LED red. After initialization the 74F74 flip-flop is cleared either by a local or a VMEbus processor. When it is cleared, SYSFAIL* is negated and the LED turns GREEN.

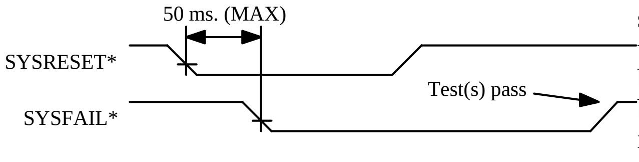


Figure 5-5 SYSFAIL* timing during system reset

SYSFAIL* is asserted when SYSRESET* is asserted, and negated when the module's diagnostic test passes.

The circuit of Figure 5-6 allows SYSFAIL* to be asserted anytime by writing to the 74F74 flip-flop. This could be done in response to a catastrophic failure.

5.2.5 SYSFAIL* Application Example

Multiprocessor systems provide special challenges in real-time control systems, especially when it comes to fault detection and orderly shut-down of the system. The multiprocessing feature of VMEbus makes normal hardware watchdogs cumbersome to use, and the absence of parity checking on the bus makes real-time fault detection difficult. When used correctly, however, the SYSFAIL* signal enables the clever system integrator to overcome these shortcomings.

For example, consider a real-time, multiprocessing, closed loop VMEbus control for an industrial robot. This application can be tricky during control failures, especially when hydraulics are used.

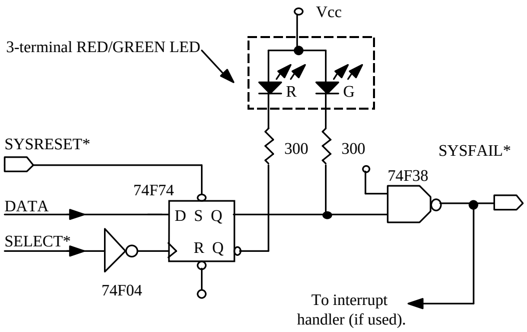


Figure 5-6 Simple SYSFAIL* driver circuit

Controllers for a hydraulic pick-and-place robot often start moving a manipulator arm after an 'EXTEND' command. The arm moves (and keeps moving) until the controller sends a 'STOP' command. If the controller fails between the 'EXTEND' and 'STOP' commands, then the robot arm continues to move. This failure could cause severe damage to the robot, whatever the robot was handling or to any people standing nearby.

In order to prevent this sort of catastrophe it is necessary to detect when a failure occurs, and shut down the system. It is also nice to have some way of identifying the cause of the failure. This means leaving 'a trail of bread crumbs' so that the root cause of the problem can be found. This not only helps in field failures, but also in debugging application software.

Most system failures fall into one of the following categories:

- Undefined response to a system reset
- Power failure
- Hardware component failures
- Detected software errors

- Undetected software errors

The biggest trick is to detect a failure. Figure 5-7 shows one way that a system monitor can be built to detect all five types of failures in a multiprocessing environment. It also provides the hardware hooks for a 'trail of bread crumbs' diagnostic capability.

During power-up, all masters and slaves assert SYSFAIL*. In an ideal system each module would have a RED/GREEN front panel FAIL indicator displaying the status of the module's SYSFAIL* driver circuit. RED for FAIL, GREEN for GO. During power-up initialization each module undergoes a diagnostic check, after which the module negates it's SYSFAIL* driver.

Front panel LED indicators also help service personnel to diagnose system problems. When the system powers up, all LEDs turn RED. As each module completes its diagnostic test, the LED turns GREEN, indicating that the module has passed. Boards with RED indicators are then swapped out until the problem is fixed.

During power-up, the monitor board itself also asserts SYSFAIL*. This insures that the emergency stop relay will open in response to a system reset. This is essential because machine controls are sometimes reset by the operator while the machine is running.

Each of the system processors has an associated watchdog timer on the system failure monitor module. Each watchdog is triggered at some predetermined time interval. If a processor fails (or its software is corrupted), the associated watchdog is tripped, and the emergency stop interlock chain is opened. Since the watchdogs are located on the system failure monitor module (as opposed to the processor modules), the VMEbus interface of every processor is also verified.

The system monitor module has three watchdog timers. This is important for two reasons: a single watchdog timer can be misused in multiprocessor environments (i.e. if all three processors trigger the same watchdog timer, a failure of a single processor would not be detected), and single watchdogs do not facilitate post-mortem failure analysis of the system.

The positioning of the watchdog update software routines is very important. Watchdogs should not be tripped in response to an on-board interrupt timer, as this would not catch many software faults. If a real-time software kernel is used, it is a good idea to create a watchdog task. The kernel adds this routine to a round-robin task queue, which insures that the kernel is operating as well.

Once all modules have passed the power-up diagnostic procedure, one processor 'arms' the system fail monitor module by writing to the CLEAR SYSFAIL register. This negates SYSFAIL*, and closes the emergency stop interlock relay.

Once the board is armed, any falling edge on SYFAIL*, SYSRESET*, ACFAIL* or the watchdog timers will again open the emergency stop interlock relay. The relay remains open until the system is again brought up in an orderly manner.

Emergency stop relay chains are common in industrial systems. The chain often runs between emergency stop palm buttons located near the hazardous parts of the machine. The palm buttons are normally closed switches. When the operator sees a possible hazardous condition, the palm button is pushed and the emergency stop chain is opened. This opens master power relays, and de-energizes the system.

The system failure monitor module is the VMEbus computer's private emergency stop button. If some problem is detected in the computer, the emergency stop chain is opened and the system is de-energized.

The architecture of the system failure monitor module also allows a post-mortem diagnostic capability. This is especially important in industrial systems, where down time can be very expensive. Post-mortem diagnosis is also very helpful for fixing application software bugs during system integration.

The system failure monitor module supports post-mortem diagnostics in two ways: by latching the state of all of the emergency stop sources at the time of the emergency stop, and by facilitating the use of a 'trail of bread crumbs' feature on the processor modules.

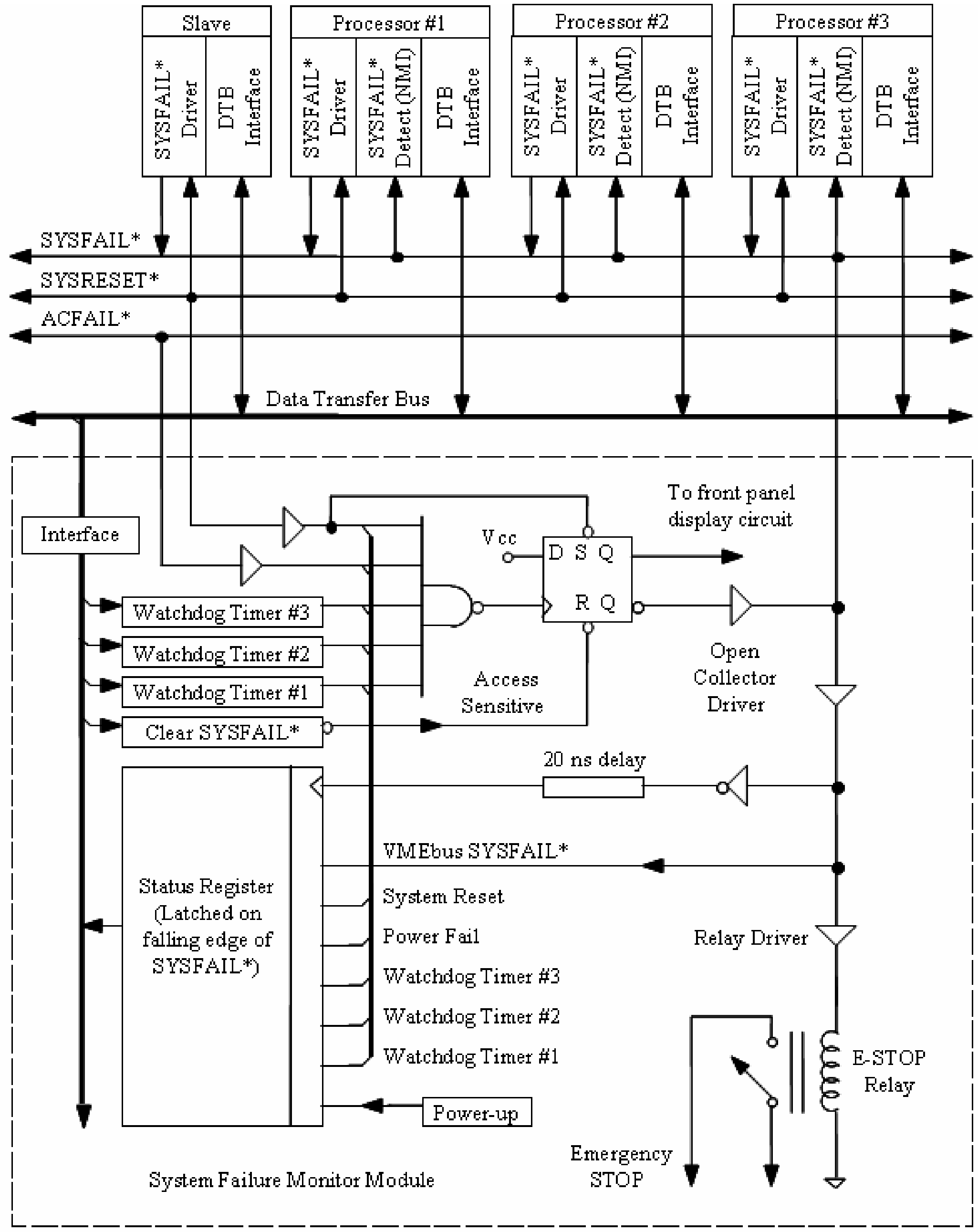


Figure 5-7 Block diagram of the failure monitor board

This module detects when a VMEbus machine control has failed. In response to a failure, the monitor module opens an emergency stop relay. The module also provides a 'trail of bread crumbs' for diagnostic purposes.

In real-time machine control applications, the source of a machine shut down can be hard to determine. If the VMEbus control is the source of the shut down, the monitor board can be queried with a system debug monitor to determine what caused the interlock relay to open. [The state of the emergency stop source is latched by the board].

Another powerful analysis tool is the 'trail of bread crumbs' non-maskable interrupt as shown in Figure 5-8. Most VMEbus processors that support the SYSFAIL* option can be configured so that a falling edge on SYSFAIL* generates a non-maskable interrupt. The software on each processor board can be set up so that a register dump is generated in response to the SYSFAIL* non-maskable interrupt. This saves the state of internal registers (including the program counter) at the time the emergency stop relay opened, and gives the operator the ability to track down the state of each processor at the time of the failure.

Table 5-2 summarizes how the system failure monitor module responds to computer failures. The provision for a power-up status bit is recommended on monitor board designs. This allows debugging software the ability to determine if the emergency stop interlock relay was opened in response to power cycling.

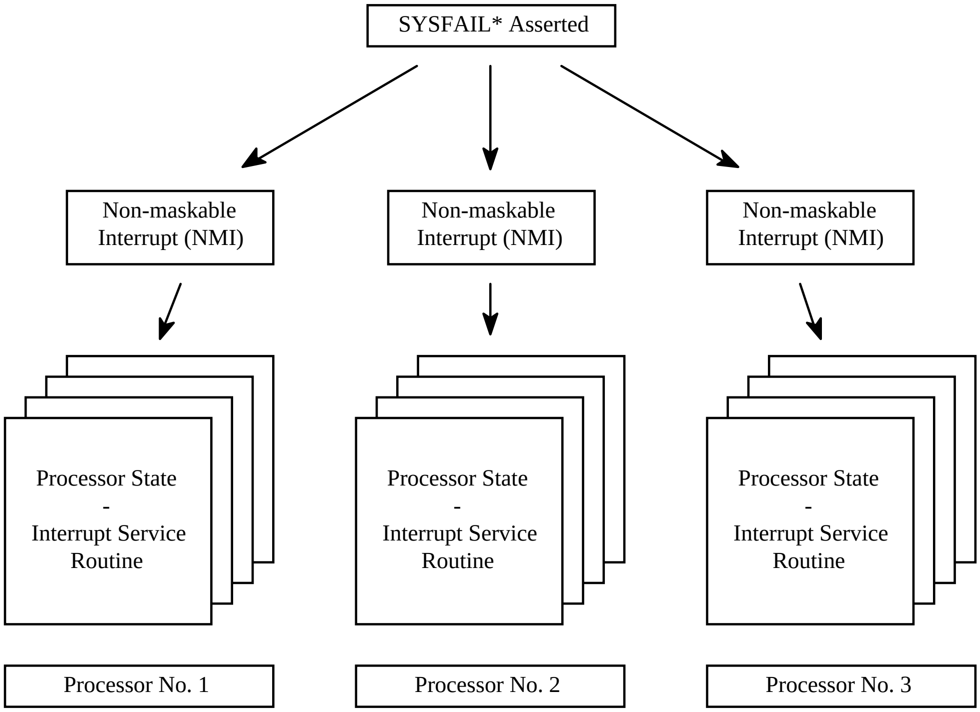

SYSFAIL* Asserted
Non-maskable Interrupt (NMI)
Non-maskable Interrupt (NMI)
Non-maskable Interrupt (NMI)
Processor State - Interrupt Service Routine
Processor State - Interrupt Service Routine
Processor State - Interrupt Service Routine
Processor No. 1
Processor No. 2
Processor No. 3

Figure 5-8 The state of each processor is saved at the time of the SYSFAIL* nonmaskable interrupt

Types of information that can be saved include the program counter and internal registers which can aid in post-mortem debugging.

In many cases the types of control failures presented in this example are not 100% preventable. At best they can only be minimized. In all industrial systems the potential for damage to life or limb must be avoided by means other than the control system. The ideas for system faults given here are for guidance only and must be reviewed on a system-by-system basis. The author and publisher of The VMEbus Handbook, Fourth Edition assumes no responsibility for the use or misuse of the ideas presented.

5.3 AC Power Failure

Some computer systems must shut down in an orderly manner to prevent data loss, property damage or human injury. The power monitor functional module facilitates this by notifying all VMEbus modules when a power failure is imminent. A block diagram is shown in Figure 5-9. In this example the power monitor asserts the ACFAIL* signal on the VMEbus backplane when the AC line voltage stops. This notifies all of the bus modules that power is about to quit. The power supply must then continue to provide DC voltage to the VMEbus backplane for at least four milliseconds, and then quit. Figure 5-9 also shows an optional standby power source that may be used by bus modules to retain memory or real-time clock sources.

Some disk controller modules can corrupt data if they happen to be writing to the disk when the power shuts off. One way to prevent this is to notify the controller that power is about to be turned off. Often this is done with a shut down command given by the operator before the power is turned off. Unfortunately there is no time to issue such a command after the power fails. In this case the ACFAIL* signal could be used to notify the disk controller that a shut-down is about to occur.

Table 5-2 System failure handling chart

Failure	System response	Diagnostic tool
System reset	Falling edge on SYSRESET* opens emergency stop relay.	System reset bit set in system failure monitor status register.
Power failure	Falling edge on ACFAIL* opens emergency stop relay.	Power fail bit set in system failure monitor module status register.
Bus error (BERR*)	Software definable (BERR* is often used for memory sizing). If BERR* capability is needed, it is handled by software or by watchdog time-outs.	None.
Hardware failures	Processors: watchdog timers. Slaves: assert SYSFAIL*	Associated watchdog timer bits are set in the monitor module status register. Trail of bread crumbs register dump. Slaves can be polled to determine SYSFAIL* driver status.
Detected software failures	Processor modules assert SYSFAIL*.	Trail of bread crumbs register dump. Failure task queue.
Undetected software failures	Processors: watchdog timers Slaves: assert SYSFAIL*..	Associated watchdog timer bits are set in the monitor module status register. Trail of bread crumbs register dump. Slaves can be polled to determine SYSFAIL* driver status.

Power fail monitors can also smoothly shut down industrial equipment. One such example is a robot arm with a hydraulic actuator (i.e. a piston). Without the power fail monitor the robot's arm may make wild or erratic movements after a power loss...endangering property or personal safety. The power fail monitor allows software time to shut down hydraulic valves, reducing the chance of erratic movements.

Some systems use the power fail monitor to save important data in non-volatile memory before the power quits. Saving the time of a power failure (from a real-time clock) can sometimes help track down the source of the power failure.

The timing diagram of Figure 5-10 shows what the power monitor does during power-up and power-down situations. During power-up, ACFAIL* is negated at least 200 milliseconds before SYSRESET* is negated. During power-down ACFAIL* is asserted at least two milliseconds before SYSRESET*, but at least four milliseconds before +5 VDC quits.

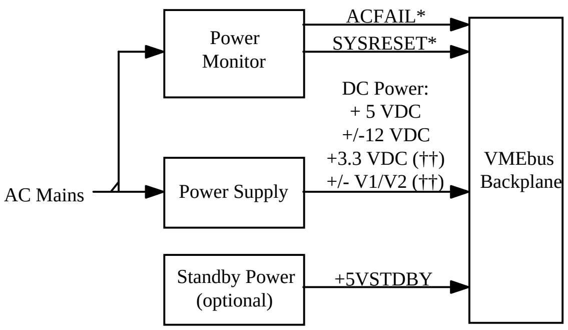
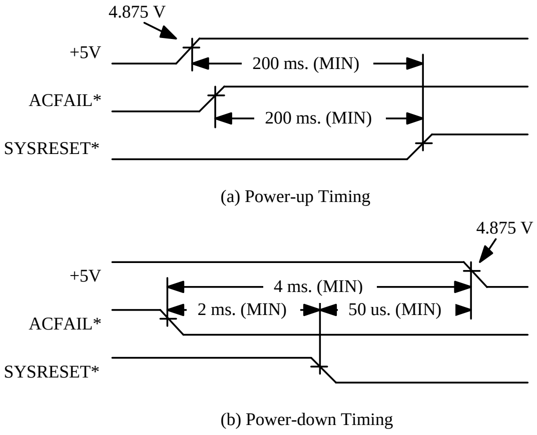


Figure 5-9 Power monitor functional module

In most systems the +5 VDC power is derived from an AC (alternating current) line source, which is converted to DC (direct current) using a power supply. Most supplies can provide DC power for some amount of time after the AC line quits. This short reserve is obtained from energy stored in its output filter capacitors. In this example we will assume that the power supply will continue to provide +5 VDC for at least four milliseconds after the AC line power stops. [This is also the minimum amount of time required by the VMEbus specification].

The power fail monitor asserts ACFAIL* when it has detected that AC power has stopped. Before +5 VDC goes out of tolerance, however, it must assert SYSRESET*. This prevents erratic behavior of the bus modules as the voltage drops below 4.875 volts. Most power fail monitors do not directly measure +5 VDC before asserting SYSRESET*. Usually it is driven some preset time after the AC line voltage stops.



(a) Power-up Timing

(b) Power-down Timing

Figure 5-10 Power monitor timing

In actual operation, the timing of ACFAIL* will probably vary with the loading of the power supply. ACFAIL* is usually generated with a timer that measures a delay after the AC power fails. If more cards are added to the backplane, the delay between ACFAIL* and SYSRESET* will get shorter because the output capacitors of the power supply drain faster.

Some power supplies which are designed for VMEbus (like the one shown in Figure 5-11) include a power fail monitor. Some open-frame supplies have a power-fail output but do not conform to the ACFAIL* and SYSRESET* timing required by the VMEbus specification. If a power supply does not have a power fail monitor, an external version like that shown in Figure 5-12 can be used.

5.3.1 Circuit Example - Power Fail Monitor

A simple power fail monitor circuit is shown in Figure 5-13. This circuit provides both power-up and power-down timing for ACFAIL* and SYSRESET*.

Power-up reset is generated using RC network R7/C3, and Schmitt trigger U5. During power-up reset the flip-flops (U3) are cleared. This negates ACFAIL* and asserts SYSRESET*. Sometime after 200 milliseconds SYSRESET* is negated. The same thing happens during a manual reset when switch S1 is pushed.

AC power is detected using the HCPL-3700 (U1). This device is an opto-coupler which provides both AC line isolation and signal detection. It has an internal full wave bridge which rectifies the AC input. The rectified signal is then used to drive an internal LED, which turns on once every 1/2 AC line cycle. For example, at 60 Hz it will turn on and off every 8.33 milliseconds (8.33 ms. = 1/120). The LED then energizes an output transistor, negating ACDET. The wave shape on ACDET is also shown in Figure 5-13.

The HCPL-3700 can also provide threshold detection of the AC input voltage. This helps detect brown-out conditions. Capability for DC input voltage detection is also possible. For more information the reader is directed to the HCPL-3700 data sheet available from Hewlett Packard.

The output of opto-coupler U1 is sent to the 74LS123 one-shot (U2). After SYSRESET* has been negated, and AC power is functioning correctly, the one-shot will be continuously triggered by ACDET. If the AC line voltage were to stop, the one-shot would quit triggering and would cause flip-flop U3 to be set and ACFAIL* asserted.

When ACFAIL* is asserted a second one-shot (in U2) is triggered. At the end of the time constant set by R5 and C2 flip-flop U3 is triggered, asserting SYSRESET*.

The AC line frequency is adjusted using R4 and C1. Since the HCPL-3700 has an internal full-wave rectifier its output will be twice that of the AC line frequency (the circuit is set for 50 or 60 Hz). The time delay between ACFAIL* and SYSRESET* asserted is adjusted using R5 and C2. The circuit is set for about 2.5 milliseconds. The ideal length of this delay will depend upon the power supply and the current draw by the VMEbus system.

5.3.2 ACFAIL* Detection

The VMEbus specification does not say what is done in response to a power failure. Intelligent modules will often use ACFAIL* to generate a non-maskable interrupt to their local processors. The interrupt service routine then performs the emergency shut down procedure(s).

The response can be more complex in multiprocessor systems. There is always a limited amount of time for the emergency shut down procedures to occur. If many bus masters require the bus simultaneously after ACFAIL*, a log-jam can take place. This problem can be avoided by using requesters with ACFAIL* release mechanisms, or using an arbiter that changes its priority during ACFAIL*.

5.4 Geographical Addressing (††)

The VME64x standard defines six pins on the 160 pin connector for geographical addressing. Geographical addressing is a feature that allows each bus module to determine its card slot position in the backplane. This feature has been lacking in VMEbus for some time. There are four areas where geographical addressing can be useful to the system integrator:

- Making sure that cards are located in the correct backplane position. This is especially important when the P0/J0 or P2/J2 user defined pins are used.
- Loading the correct software on a CPU board.
- Determining if the bus module is connected to a VME64x backplane.
- Enabling the slot 01 system controller function.

As shown in Table 5-3, a five bit geographical address is encoded into each card slot on the VME64x backplane. A sixth pin is used as a parity bit. Each pin on the backplane is either tied to ground or floating. Pull-up resistors *on the bus module* convert these into logic high or logic low signals. On-board logic then invert the signals and read them as a binary code.

A pull-up resistor scheme is used because it allows each bus module to determine if it is plugged into a VME64x backplane or a legacy backplane. If the module detects an indicated card position of zero (i.e. all of the pins are floating), then it is plugged into a legacy backplane. If an indicated slot position of 01 - 21 is found, then it is plugged into a VME64x backplane.

In some cases it is important that a bus module know if it's plugged into a legacy backplane, as the card would then ignore it's 'z' and 'd' row pin connections.

Standard VMEbus or VME64 does not have any sort of geographical addressing mechanism. There is no way for a card to know which slot it's in. This has caused some integration problems in the past.

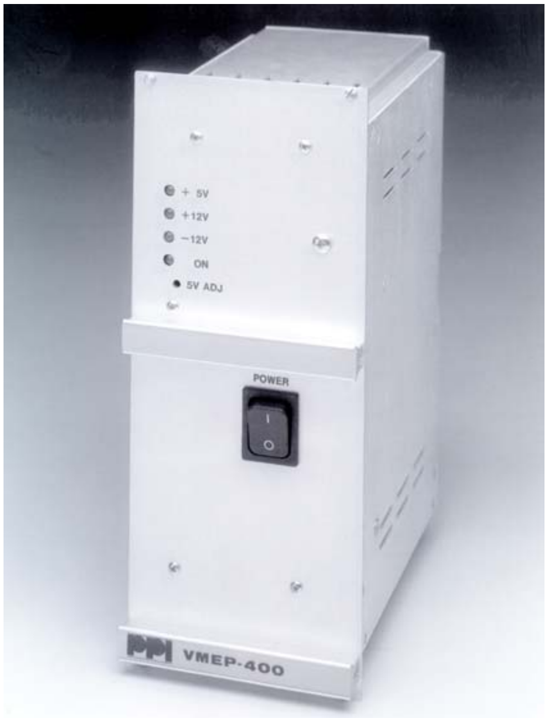


Figure 5-11 Power supplies, like this one designed specifically for VMEbus, sometimes include a power fail monitor

Photo by Rob Bigelow.

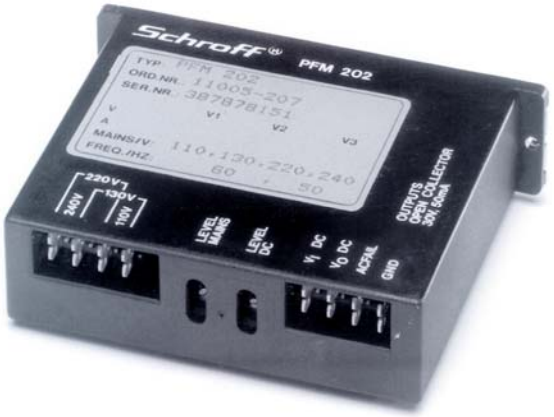


Figure 5-12 Commercial power fail monitor

Photo courtesy of Schroff Inc.

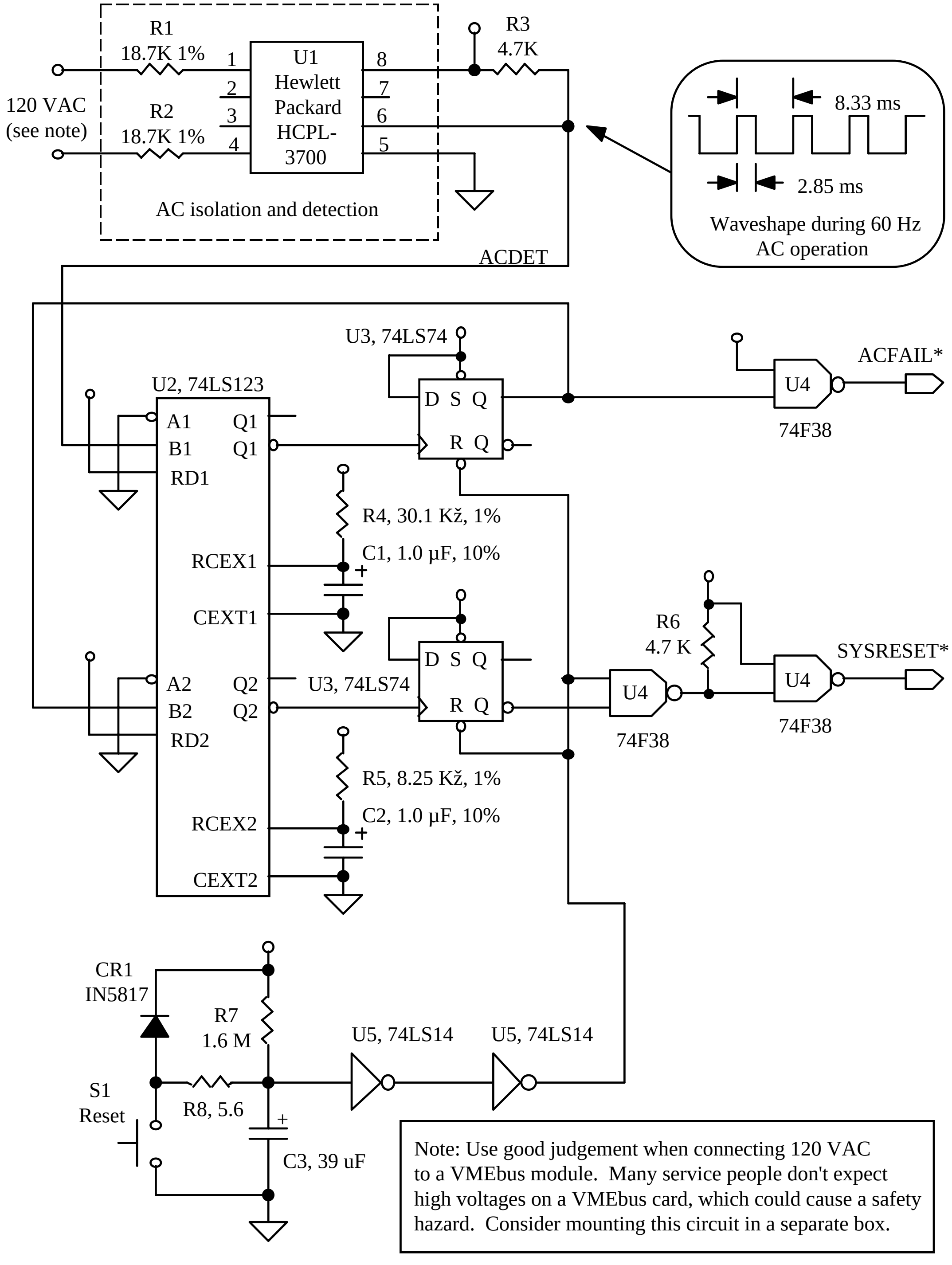
For example, software can use geographical addressing to double check that each card is located in the correct slot. For example, a CPU card may use its P2/J2 user defined pins for a network connection. Another board may use the same set of pins for digital I/O. Geographical addressing allows the system software to determine if the boards have been located in the correct slot.

Geographical addressing also allows multiple (identical) CPUs to have identical EPROM code. Very often each CPU in a multiprocessing system must have a unique set of EPROMs located on it. That's because each CPU might require slightly different software. With geographical addressing, each CPU could learn its backplane position at boot-up time, and run the appropriate software. This eliminates multiple sets of EPROM source code (and simplifies system installation and maintenance).

Another use for geographical addressing is to turn on the VMEbus system controller. All VMEbus computers require a system controller in the first backplane slot. The system controller is a set of utility functions such as an arbiter, system clock driver, power monitor and IACK* daisy-chain driver. Geographical addressing simplifies the task of configuring a board for this function.

In commercial VMEbus products there may be system controller hardware on several boards. For example, CPU modules usually have a system controller. This means that the system controller must be enabled or disabled on each board, depending on whether or not it's in the first slot. This is usually done by flipping a DIP-switch. While not particularly difficult, the task diminishes the plug-and-play ability of VMEbus. The same job can be done automatically with geographical addressing.

The VME64 specification also allows the use of a *first-slot detector* to solve this problem. The first-slot detector monitors the bus-grant daisy-chains during power-up. If the module is in the first slot, the daisy-chains remain floating (because there are no upstream cards). If a module resides in any other slot, it senses that there is at least one card to its left (because the daisy chains are driven high). This method may be preferable, as it works equally well in VME64x or legacy backplanes.



Note: Use good judgement when connecting 120 VAC
to a VMEbus module. Many service people don't expect
high voltages on a VMEbus card, which could cause a safety
hazard. Consider mounting this circuit in a separate box.

Figure 5-13 Simple power fail monitor circuit

Table 5-3 Geographical address pin encoding

Geographical Address Pin Encoding (As Read by a VMEbus Module						
Slot No.	GAP*	GA4*	GA3*	GA2*	GA1*	GA0*
(§)	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN
01	OPEN	OPEN	OPEN	OPEN	OPEN	GND
02	OPEN	OPEN	OPEN	OPEN	GND	OPEN
03	GND	OPEN	OPEN	OPEN	GND	GND
04	OPEN	OPEN	OPEN	GND	OPEN	OPEN
05	GND	OPEN	OPEN	GND	OPEN	GND
06	GND	OPEN	OPEN	GND	GND	OPEN
07	OPEN	OPEN	OPEN	GND	GND	GND
08	OPEN	OPEN	GND	OPEN	OPEN	OPEN
09	GND	OPEN	GND	OPEN	OPEN	GND
10	GND	OPEN	GND	OPEN	GND	OPEN
11	OPEN	OPEN	GND	OPEN	GND	GND
12	GND	OPEN	GND	GND	OPEN	OPEN
13	OPEN	OPEN	GND	GND	OPEN	GND
14	OPEN	OPEN	GND	GND	GND	OPEN
15	GND	OPEN	GND	GND	GND	GND
16	OPEN	GND	OPEN	OPEN	OPEN	OPEN
17	GND	GND	OPEN	OPEN	OPEN	GND
18	GND	GND	OPEN	OPEN	GND	OPEN
19	OPEN	GND	OPEN	OPEN	GND	GND
20	GND	GND	OPEN	GND	OPEN	OPEN
21	OPEN	GND	OPEN	GND	OPEN	GND
(§) Note: There is no Slot 00 position on a VMEbus backplane. However, this condition indicates that the board is plugged into a legacy backplane.						

5.5 First Slot Detector (†)

The first slot detector was first defined in the VME64 specification. This device simplifies the task of configuring the system controller when a board is placed into a backplane.

All VMEbus systems require several utility functions in the first backplane slot. These include an arbiter, system clock driver and IACK* daisy-chain driver. Other functions such as a power monitor (for system reset) and a bus-error timer are generally included with the arbiter. These are collectively known as the 'system controller'.

In commercial VMEbus products, the system controller may reside on more than one board. This means that it must be enabled or disabled, depending on whether the board is placed in the first slot. This is usually done by flipping a DIP-switch or setting a soft-bit. While not particularly difficult, the task diminishes the 'plug-and-play' ability of VMEbus.

The first slot detector solves this problem by monitoring the bus-grant daisy-chains during power-up. If the module is in the first slot, then there are no up-stream bus modules and the daisy-chains remain floating. If a module resides in any other slot, it senses that there is at least one card to its left (because the daisy chains are driven high).

Older VMEbus specifications did not have any sort of geographical addressing mechanism. Stated another way, there was no way for a card to 'know' which slot it's in. This has caused some integration problems in the past. However, a new geographic addressing scheme is available in VME64x. That standard defines several pins on the new 160 pin connector, so that each card can learn its backplane position.

Figure 5-14 shows a circuit idea for a first slot detector. During system power-up, a voltage monitor detects when the +5 VDC power supply exceeds 4.75 VDC. At that time the PWR_INSPEC signal is asserted. After a 40 ms delay, the state of the bus grant signal pins are latched with D-type flip-flops.

If the module is located in the first slot, the BG3IN* pin is floating and the input is pulled down with a 10 kΩ resistor. This condition is latched by the flip-flop at the end of the 40 ms time delay, and SLOT_ONE is asserted.

If the module is located in any other slot, the BG3IN* daisy chain driver from the previous slot is driven high during system power-up, and SLOT_ONE is negated. The state of this flip-flop (SLOT_ONE) indicates whether the system controller should be enabled.

The first slot detector requires that BG3IN* be monitored with a low input current IC. If a standard totem-pole IC (such as a 74LS04) is used, the pull-down resistor may not work very good.

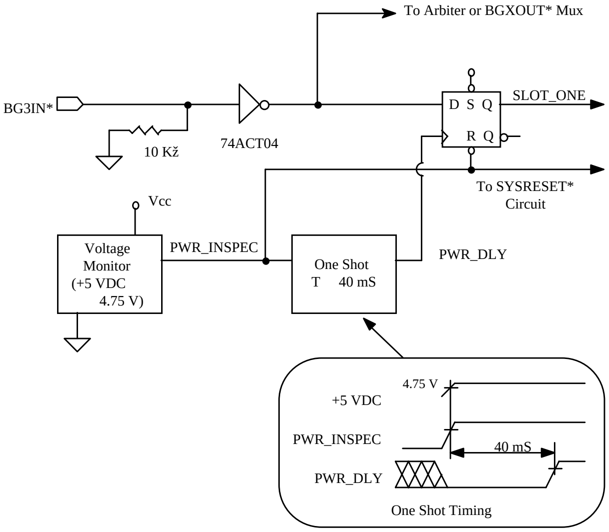


Figure 5-14 Circuit idea for the first slot detector

The first slot detector can also be used to configure other system controller options such as arbitration type or bus timer interval. Although it is not shown in Figure 5-14, a three-bit binary code can be obtained on the BG0IN* - BG2IN* signal pins, and monitored at power-up. This code can be generated from jumpers (or hard-wired logic) located on the backplane.

5.6 CR/CSR Registers (†, ††)

Control and status registers were added in the VME64 standard. These are also known as CR and CSR registers. They are implemented as a unique memory space, with each region of the space containing data specific to an individual board. This is called CR/CSR space, and is accessed with a special A24 address modifier code. As shown in Figure 5-15, each space is divided into thirty-two 512 Kbyte regions, one for each board in the system.

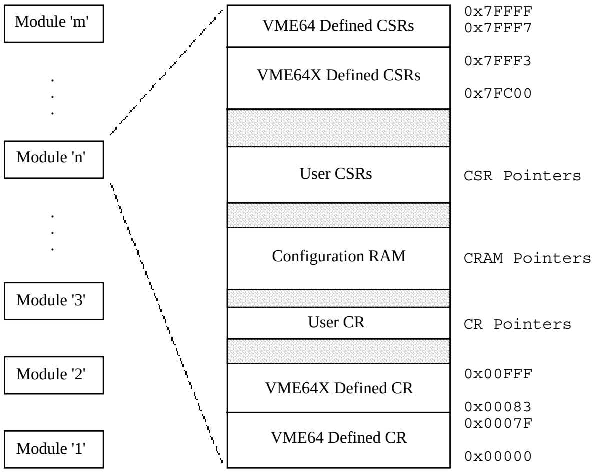


Figure 5-15 CR/CSR space

The five most significant address bits of CR/CSR space are unique to each bus module. In the absence of geographical addressing, the VME64 specification gave several options for selecting this space. Usually, they are set by a DIP-switch or by the Auto Slot ID configuration functions. [The Auto Slot ID allows a base address to be assigned to each board under software control]. However, under the VME64x standard the base address of each board can also be selected by the geographic addressing pins.

The lower addresses of each 512-Kbyte CR/CSR region has a configuration ROM, which is accessed as a D08(O) slave. The VME64 ROM address mapping is shown in Table 5-4, and contains information such as manufacturer ID, board ID, board revision and so on. The minimum configuration ROM data width must be D08(O), with access to every fourth byte starting at address 0x03.

Table 5-4 Configuration ROM (CR) space

VME64 & VME64X Configuration ROM (CR) Register Definitions	
CR/CSR Address Offset	Description
- Configuration ROM (CR) Area Defined under VME64 -	
0x00003 - 0x00013	CR checksum, length and width.
0x00017	CSR data access width
0x0001B	CSR space specification ID
0x0001F	Valid carriage return character
0x00023	Valid line-feed character
0x00027 - 0x0002F	Manufacturer ID (controlled by the IEEE)
0x00033 - 0x0003F	Manufacturer board ID (controlled by board MFG)
0x00043 - 0x0004F	Revision ID (controlled by board MFG)
0x00053 - 0x0005B	Pointer to ASCII string
0x0005F - 0x0007B	Unused / reserved
0x0007F	Program ID
- Configuration ROM (CR) Enhancements Under VME64X -	
0x00083 - 0x00097	Pointers to user control register (CR) area
0x0009B - 0x000AF	Pointers to configuration ram (CRAM) area
0x000B3 - 0x000C7	Pointers to user control/status register (CSR) area
0x000CB - 0x000DF	Board serial number
0x000E3 - 0x000E7	Slave characteristics
0x000EB - 0x000EF	Master characteristics
0x000F3	Interrupt handler capabilities
0x000F7	Interrupter capabilities
0x000FB	Reserved
0x000FF	Configuration ram (CRAM) access width.
0x00103 - 0x0011F	Function 'N' data access width
0x00123 - 0x0021F	Function 'N' AM code mask(s)
0x00223 - 0x0061F	Function 'N' XAM code mask(s)
0x00623 - 0x0069F	Function 'N' address decoder mask(s) (ADEM)
0x006A3 - 0x006AB	Reserved
0x006AF	Master data access width capability.
0x006B3 - 0x006CF	Master AM capability
0x006D3 - 0x0074F	Master XAM capability
0x00753 - 0x00FFF	Reserved

VME64x expands the CR/CSR (Configuration ROM / Control Status Register) register set that was originally defined in the VME64 specification. This allows boards to be more easily identified and configured by software. The VME64x standard has enhanced the VME64 register set by adding new information such as:

- Configuration RAM area.
- Address space relocation (i.e. base address of board).
- Pointers to user control, configuration RAM and CSR areas.
- Board serial number.
- Supported master & slave interfaces (RMW cycles, RETRY pin, etc.).
- Interrupter and interrupt handler characteristics.

Byte offset 0x0001B in the CR registers contains the revision number of the CR/CSR space. If the CR/CSR registers conform to the VME64 standard, then this location contains 0x01. If the registers conform to the VME64x standard, then this location contains 0x02.

The higher addresses of each CR/CSR space are shown in Table 5-5, and contain the control and status registers. Under VME64 these just controlled the local reset, SYSFAIL* and base address of each board. However, the VME64x added several new functions to this space. For example, an owner of the configuration RAM area can be defined.

Table 5-5 Control/status register (CSR) space

VME4 & VME64X Control and Status Register (CSR) Definitions	
CR/CSR Address Offset	Description
- Control & Status Register (CSR) Area: Defined under VME64 -	
0x7FFFF	CR/CSR Base Address Register (BAR)
0x7FFFB	Bit set register (local reset, SYSFAIL*, etc.)
0x7FFF7	Bit clear register (local reset, SYSFAIL*, etc.)
- Control & Status Register (CSR) Enhancements under VME64X -	
0x7FFF3	CRAM_OWNER Register
0x7FFEF	User defined bit set register
0x7FFEB	User defined bit clear register
0x7FFE3 – 0x7FFE7	Reserved
0x7FF63 – 0x7FFE3	Function Table 'N' ADER Registers
0x7FC00 – 0x7FF5F	Reserved

5.7 AUTO ID (†)

The Auto Slot ID function was introduced in the VME64 specification, and allows automatic configuration of bus modules as part of a power-up initialization. This function:

- Allows the base address of each board to be configured under software control.
- Determines the relative position of each board in the backplane.

The Auto Slot ID sequence is shown in Figure 5-16. After reset, each board generates an interrupt on level IRQ2*. A level two interrupt handler module, called the *monarch*, performs interrupt acknowledge cycles in response to each interrupt request. [The monarch waits until all boards have negated SYSFAIL* before performing the Auto Slot ID tasks]. During the interrupt service routine, each slave responds to the monarch at an initial CR/CSR space of zero (i.e. CR/CSR base address of zero). The monarch then configures the base address of each board using the registers.

The monarch board can be a generic CPU module with a level 2 interrupt handler. The monarch is distinguished from other CPU modules by its Auto Slot ID software.

The base address of each board is configured in its BAR (Base Address Register - CR/CSR address 0x7FFFF). The base address is a number between one and thirty-one, and contains the five most significant bits of the CR/CSR address. Once the BAR is configured, the board will then respond at its new base address.

Since the interrupt acknowledge daisy-chain always propagates away from the first backplane position, the *relative* location of each board can be found by the monarch. However, it cannot find the *absolute* location of each board.

Knowing the relative position of a board is of limited use in real systems. However, as the performance of arbitration and interrupt functions in VMEbus are slot dependent, the relative position of boards can be double-checked by software.

5.8 Live Insertion / Hot Swap (††)

There are three VMEbus standards that define features for live insertion modules:

- Board Level Live Insertion for VMEbus: ANSI/VITA 3-1995.
- High Availability VMEbus Specification: VITA 1.2-199x (draft).
- Live Insertion for VME64x: VITA 1.4-199x (draft).

The ANSI/VITA 3-1995 describes a system for inserting standard VMEbus modules into a hot-swap rack. The advantage to this method is that any VMEbus card can be live insertable. However, a fair amount of interface hardware is required on the backplane or on a transition module. This logic itself is not necessarily hot swappable, and the method has seen limited use in the industry.

The VME64x specification sets the stage for second generation live insertion standards called the VITA 1.2 and 1.4. At the time of writing of this book these standards were not yet complete. However, they are expected to define a new hot-swappable VMEbus module that requires very little support logic.

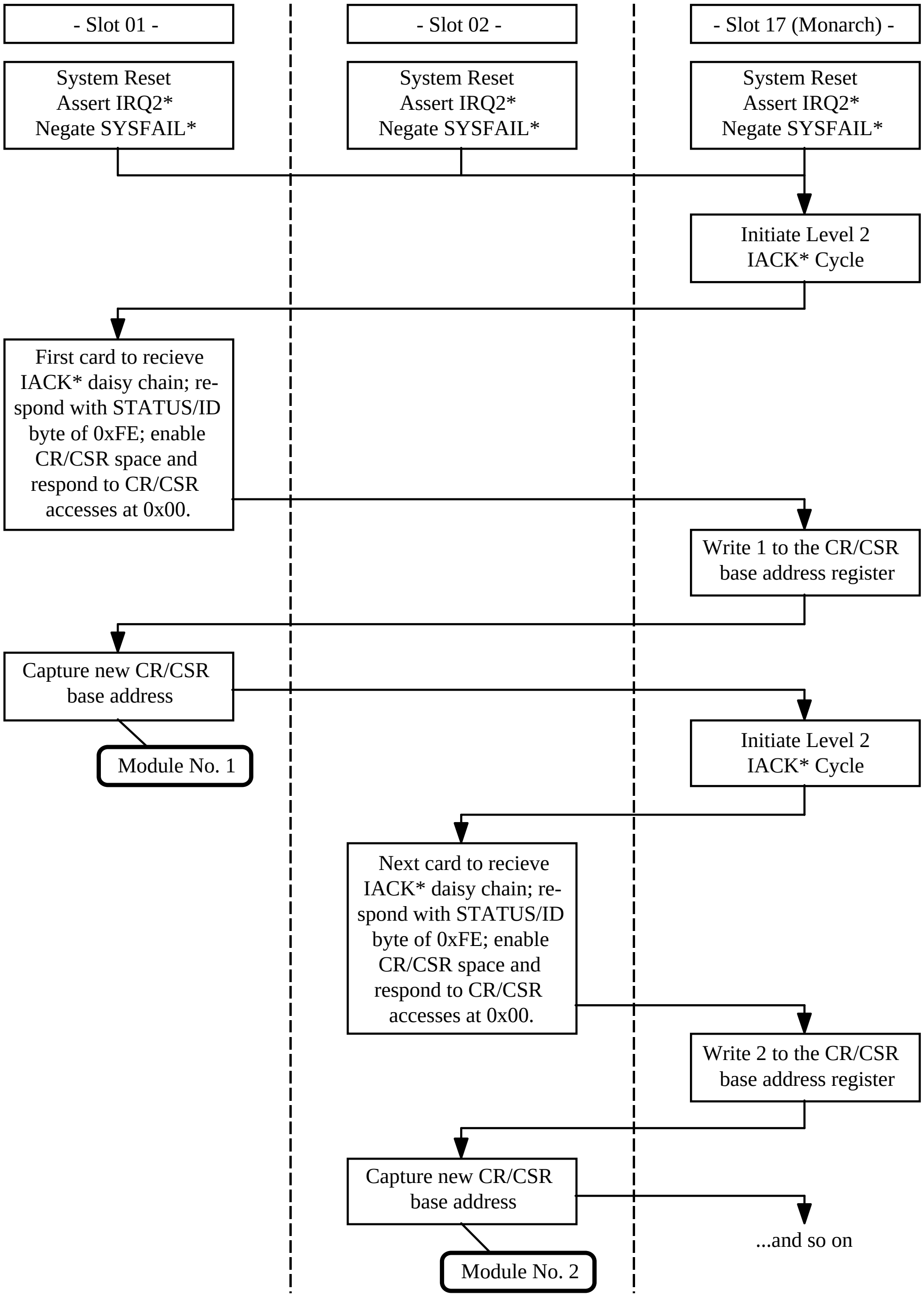


Figure 5-16 Auto ID configuration example

The VME64x specification lays the groundwork for the new live insertion standards. The standard itself does not detail the exact methodology for live insertion, but does add many of the hardware features that will be needed. Some of these include:

- A 160 pin connector with voltage pre-charge pins.
- Live insertion logic pins.
- Test and maintenance bus.

For the most part, the VMEbus hot swap requirements are driven by the telecom industry. That industry has very difficult up-time system requirements. For example, a telephone switch cannot be down for more than four minutes per year. That means that the switch can never be shut off, and that active components must be replaceable in a live system.

Although the VME64x standard does not specify how a live board is replaced in a system, it must make some assumptions in order to provide the right hardware 'hooks'. The VME64x standard assumes that a board is replaced using the following steps:

1) A failed board is detected by system software, and alerts a technician as to which board is bad. Failed boards can be detected by hardware, run-time software or by the VME64x test and maintenance bus. In some cases the failed board is detected by 'easter-egging'. That's where boards are swapped out until the failure condition goes away.

2) The failed board is logically disconnected from the system. This may be done automatically, or by the technician. By 'logically disconnected', it is meant that the system software no longer recognizes the board about to be replaced.

3) The technician connects a grounded wrist-strap to the VMEbus chassis.

4) The technician pulls the failed board out of the chassis.

5) A replacement board is inserted into the card cage. While it is being pushed into place, any static charge on the PCB is dissipated through the VME64 ESD strips (located on the card guides).

6) Before the VME64x module is seated in the backplane, the alignment pin on the injector/extractor front panel precisely aligns the module with the backplane connectors.

   During this time the front panel keying mechanism determines whether or not the board belongs in the slot. If it doesn't, then it cannot be pushed into the backplane connectors.

7) When the VME64x module mates with the backplane, the extended ground and voltage pre-charge pins on the 160 pin connectors make contact first. These pre-charge the ETL bus transceivers. ETL bus transceivers have a pre-charge pin which virtually eliminates capacitive glitches on the backplane. Without the pre-charge function, these signal glitches would disrupt the system.

   The location of the elongated pre-charge pins allows the board to be inserted at a slight angle to the backplane. In 6U bus modules the pre-charge pins are located at the top of the P1/J1 connector, and at the bottom of the P2/J2 connector. This forces the pre-charge pins to make first contact, regardless of the angle of the module. In 3U bus modules the pre-charge pins are located at the top and bottom of the P1/J1 connector.

   The pre-charge happens very fast, and the bus transceivers are fully charged by the time the rest of the connector pins make contact.

8) Once the board is seated, it is powered up by locking the VME64x injector front panel handles. These handles have a micro switch embedded in them, which energizes the board when closed.

9) Live insertion logic on the 160 pin connector (pins LI/I* and LI/O*) notify the board and the system that the bus module is installed. This logic also enables the VMEbus IACK* and bus grant daisy chains through the card.

10) Before bringing the board on line, the system can verify that it's working with the test and maintenance bus.

11) If necessary, software is uploaded to the board.

12) The board is logically reconnected by system software, and brought on-line.

Several pins are committed, but currently remain undefined in the VME64x standard. These include the live insertion logic (LI/I* and LI/O*) and the test & maintenance bus (MCLK, MCTL, MMD, MPR and MSD). Detailed use of these pins will be described in the VME Board Level Live Insertion and VME64 High Availability standards. However, the test & maintenance bus follows the IEEE 1149.5 test bus. This works at the board level, and is similar in operation to the popular JTAG bus.

5.9 References

Application Note 1004: Threshold Sensing For Industrial Control Systems With The HCPL-3700 Interface Opto-Coupler Hewlett Packard, Palo Alto, CA 1979

ETL Specification / VITA P2/Draft 0.4a VITA, Scottsdale, AZ 1993

High Availability VME / VITA 1.x-199x (Draft 0.2) VITA, Scottsdale, AZ 1995

Peterson, Wade D. "VME System Controller Reduces Parts Count" *EDN Magazine* Mar 18, 1987

VMEbus Specification / Revision C.1. PRINTEX Phoenix, AZ 1985

VMEbus Specification / ANSI/IEEE STD1014-1987 VITA, Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA, Scottsdale, AZ 1995

VME64x Specification / VITA 1.1-1997 (Draft 2.0) VITA, Scottsdale, AZ 1997

VME64x Live Insertion System Requirements / VITA 1.x-199x (Draft 0.2) VITA, Scottsdale, AZ 1997

Chapter 6
Electrical Characteristics

The VMEbus specification describes the power distribution characteristics of the boards and backplanes. Voltages such as +3.3 VDC, +5 VDC and +/- 12 VDC are defined, along with the maximum current-carrying capacities of the connectors. This insures the reliable operation of the system, and also guarantees that each bus module does not generate too much heat.

The bus specification also describes the electrical characteristics of the interface ICs. This is important to maximize the bandwidth of the system, as the maximum speed of a VMEbus module is usually limited by the speed of its interface ICs. This is in contrast to synchronous buses, where speed is usually limited by a transfer clock. This also means that asynchronous buses, like VMEbus, must place special emphasis on backplane impedance, signal reflections, device technology (such as TTL, ETL or CMOS) and noise.

6.1 Power Distribution

Power is distributed to bus modules using both the J1/P1 and J2/P2 connectors. Under VME64 the available voltages are +5 VDC, +/- 12 VDC and +5 VDC standby. All of these share a common ground which is used both as a power return and a signal reference. Table 6-1 shows the specifications for VMEbus power distribution. The VME64x standard adds additional power resources to the backplane. These include:

- Ten +3.3 VDC power pins.
- Three +5 VDC power pins (using VPC).
- Two 48 VDC power supplies (+/- V1/V2).

Table 6-1 VMEbus power supplies

VMEbus Power Supplies							
Parameter (Ta ≤ 70°C)		Power Supply (Note 3)					
		+5V	+12V	-12V	+5V STDBY	+3.3V	+/- V1/V2
Volts, maximum		5.250	12.60	-11.64	5.250	3.45	75.0
Volts, nominal		5.000	12.00	-12.00	5.000	3.30	48.0
Volts, minimum		4.875	11.64	-12.60	4.875	3.25	38.0
Noise, mVp-p (< 10 MHz)		50	50	50	50	(Note 1)	N/A
VMEbus IEEE1014	6U # Pins	6	1	1	1		
	6U Current, max, amp	7.2	1.5	1.5	1.5		
	3U # Pins	3	1	1	1		
	3U Current, max, amp	3.6	1.5	1.5	1.5		
VME64 VITA 1	6U # Pins	6	1	1	1		
	6U Current, max, amp	7.2	1.5	1.5	1.5		
	3U # Pins	3	1	1	1		
	3U Current, max, amp	3.6	1.5	1.5	1.5		
VME64x	6U # Pins (Note 2)	9	1	1	1	10	2
	6U Current, max, amp	10.8	1.5	1.5	1.5	12.0	1.4
	3U # Pins (Note 2)	5	1	1	1	10	2
	3U Current, max, amp	6.0	1.5	1.5	1.5	12.0	1.4
VME64x (Hot-swap)	6U # Pins	6	1	1	1	10	2
	6U Current, max, amp	7.2	1.5	1.5	1.5	12.0	1.4
	3U # Pins	3	1	1	1	10	2
	3U Current, max, amp	3.6	1.5	1.5	1.5	12.0	1.4
Notes : (1) Inclusive of supply voltage, ≤ 20 MHz (2) Number of pins available when VPC pins are used for +5V (3) Major change between VME64 and VME64x: shaded area							

6.1.1 +5 VDC

Most logic families, like TTL and CMOS, use + 5 VDC as their main power source. This is also true for microprocessors, memories and interface ICs.

Under VME64x, the three voltage precharge (VPC) pins can also be used as additional +5 VDC power pins (if the board is not used in a hot-swap application). The VME64x 160 pin J1 and J2 backplane connectors route these pins to the +5 VDC power supply. While they are intended for precharging the onboard bus transceivers in hot-swap applications, they can also be used to provide additional +5 VDC power to boards with high current requirements. The extended length of these pins isn't a problem because, in this case, the boards will only be inserted or extracted when the system power is off.

6.1.2 +/- 12 VDC

The +/- 12 VDC supplies can be used for many purposes. One common application is for RS-232C drivers and receivers. Other uses include op-amps, D/A and A/D converters. Some manufacturers convert - 12 VDC to a -5.2 VDC ECL logic supply. Unusual voltages are converted from +/- 12 VDC or +5 VDC using a regulator or a DC-DC converter.

Generally, the +/- 12 VDC supplies are somewhat noisy for low-level analog circuits. That's because these supplies share the ground return with +5 VDC (i.e. they are not isolated), and because other boards may introduce digital switching noise into them.

6.1.3 +5 VDC STDBY

+5 VDC STDBY is intended for non-interruptible power sources. It is often used to power real-time clocks and static memory while +5 VDC is off. +5 VDC STDBY power supplies are optional, and are not commonly used.

A common +5 VDC STDBY voltage also eliminates the need for individual batteries on each VMEbus module. These batteries can be quite annoying, especially to maintenance and field service people. Very often a system can use five or ten different battery styles, and field service people must find and install replacements every few years.

It is also a common practice to ignore the tight voltage specifications on +5 VDC STDBY, and use a standard battery voltage. Popular batteries, such as nickel-cadmium (Ni-cad) cells, supply power only in multiples of 1.2 volts. + 4.8 VDC is commonly used. Also, this voltage drifts with battery condition.

Another option is to supply +5 VDC STDBY with a small wall-mount AC/DC power supply. These supplies don't work if AC power is lost, but they do support many applications where standby power is used as a convenience. For example, time-of-day clocks can be maintained when the system is off, but AC power is still available.

6.1.4 + 3.3 VDC (††)

The VME64x specification includes ten +3.3 VDC pins on the 160 pin connector. These can be used to supply up to twelve amps of power to each bus module. These were added to VME64x because of the many low voltage logic ICs. Until VME64x, +3.3 VDC had to be generated by on-board voltage regulators. All of these pins are located on the P1/J1 connector, so the additional power is available on both 3U and 6U boards.

It is also possible to power the backplane termination networks with +3.3 VDC. Theoretically, a totally +3.3 VDC VMEbus system could be built. A further advantage of lower voltage terminators is power consumption. Standard, +5 VDC, 6U VMEbus backplanes with passive terminators draw about 1.3 amps of current (dissipating 6.5 Watts). Lower voltage +3.3 VDC backplanes can reduce this to about 0.3 amps of current (dissipating about 1.1 Watts).

6.1.5 +/- V1/V2 Auxiliary (††)

Two auxiliary power supplies (+/- V1/V2) are defined on the 160 pin P1/J1 connector in VME64x. These pins are intended to be wired in parallel, and to provide a 48 VDC battery bus for telecom users. However, the VME64x standard allows them to operate at any voltage between 38 and 75 VDC.

Low-level analog designers will appreciate these supplies as well, as they can be split into dual supplies, and be totally isolated from the main power grounds. For example, they can be used as dual analog supplies such as + 48 VDC and - 48 VDC.

The VITA Standards Organization made the nominal voltage on these pins 48 VDC to satisfy the needs of telecom users. However, there's no technical reason why industrial users couldn't use these pins to provide lower voltages, such as +/- 15 VDC or +/- 24 VDC.

It should also be noted that UL standards consider it unsafe for humans to contact voltage differentials greater than 60 VDC. Users wishing to do so must label their systems accordingly.

In general, the VME64x standard allows considerable leeway in the actual voltages present on +/- V1/V2 power pins. This may cause some compatibility problems in the field. However, in actual practice these voltages are only used by specialty I/O boards. For example, a telecom user might use it for a 48 VDC battery bus, but an industrial user might use it as a low-level analog supply. This should not be a big problem, as telecom and industrial boards rarely coexist in the same rack.

Most system vendors will probably include a power supply or battery for these pins as a special order item. That will remind the user that this is a special function that needs design attention.

This situation is very similar to the old +5V STDBY pin. Most system makers don’t include a standby power supply, and when they do they often ignore the tolerances specified in the standard. For example, a standard NI-CAD battery voltage of 3.6 or 4.8 VDC is often placed on +5V STDBY, which has a very tight tolerance of 4.875 - 5.25 VDC.

6.1.6 Ground

In the VMEbus backplane, a single ground plane is used both as a power return and a signal ground. Since there is only one ground on the backplane, special consideration must be made with respect to multiple grounding systems. Some system integrators take special grounding precautions if the system is large, the environment is noisy or if there are special safety problems.

RS-232C communication links are one example where a single ground path may lead to problems. RS-232C is a popular serial interface used in microcomputer systems. It has two grounds: a signal ground and a safety ground.

Signal grounds are used by RS-232C data transceivers, and provide an electrical reference to other signals. The safety ground, also known as a protective or frame ground, carries current during short circuits and electrostatic discharges. Because of the large voltage and current surges that safety grounds carry, they are usually separated from signal grounds.

If 120 VAC is shorted to a cabinet in an external system (such as a data entry terminal), surge currents could be sent through the RS-232C safety ground. If the safety ground were connected to the VMEbus ground plane, then the system could be disrupted, the components could be damaged or a safety hazard could be generated.

How then, should the RS-232C safety ground be connected to the VMEbus system? The most common solution is to connect the safety ground to the VMEbus chassis through the front panel.

This can be a problem on older VMEbus mechanical hardware. On those systems a solid grounding path between the front panel and the chassis was not assured. The only way to guarantee a solid ground was to (a) buy special 'positive grounding' front panels or (b) add a flying lead between the front panel and the chassis.

Front panels were sometimes not grounded because (a) the aluminum parts were anodized (which creates an insulating junction), (b) there was not a reliable mating surface at the front panel / chassis connection point or (c) the front panel screw was held in place with a plastic insulator.

The VME64x specification added a new contact region between the front panel and the chassis. This region is located on a part called the injector/ejector front panel alignment pin, and it was designed as an electrical contact. For more information on the new front panel, see Chapter 7.

Another solution on older systems is to connect the front panel to the chassis through the user defined I/O pins on the P2/J2 connector. This is not always a good solution, as the isolation voltage of the connector pins is rated at only 100 Volts DC (pin to pin) which is usually too low for a safety ground. Safety grounds should be able to handle the maximum system voltage (usually 120 or 240 VAC).

6.2 Current Carrying Capacity of VMEbus Modules

The maximum current that can be drawn by any VMEbus module is limited by the capacity of the connector pins. The VMEbus specification defines this with the graph shown in Figure 6-1. The maximum current is temperature dependent, with capacity dropping as temperature increases. This is due to the self-heating of the connector pins, which can be damaged if they get too hot. The self-heating of the connector pin is caused by resistance.

Resistance can cause unequal current flow through two or more connector pins tied in parallel. This derates the maximum current-carrying capability of the connector pins as shown in the lower curve of Figure 6-1. A model that explains this is given in Figure 6-2. The model shows two parallel connected pins, with each pin having slightly different resistances (due to manufacturing and mating contact inconsistencies). Since direct current always takes the path of least resistance, the contact with the lowest resistance will carry the bulk of the DC current.

For example, if we assume that R1 is about 10 milliohms and R2 is about 12 milliohms (20% more than R1), and the total current drawn by the VMEbus module is 3.0 amps, then the current through the pin of lower resistance would be 20% greater (1.64 amps) than the other (1.36 amps). This forces the maximum current to be derated if the pins are tied together.

Also note that the current-carrying capacity is rated as an average current, and not surge current. That's because connector heating (which is a relatively slow process) is the limiting factor on maximum current-carrying capacity.

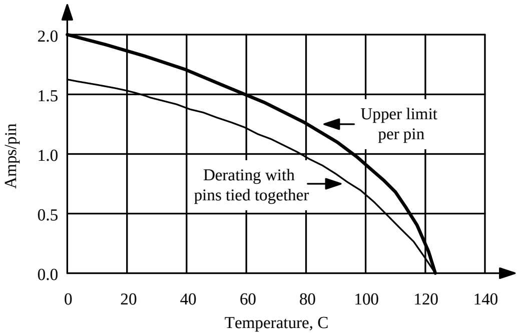


Figure 6-1 Maximum rated current-carrying capacity of VMEbus DIN 41612 connectors

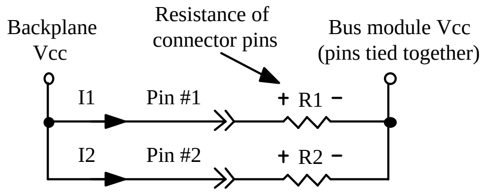

Backplane Vcc
Resistance of connector pins
Bus module Vcc (pins tied together)
I1
Pin #1
+ R1 –
I2
Pin #2
+ R2 –

Figure 6-2 DIN 41612 connector pin resistance can cause unequal current flow

+/-12 VDC and +5 VDC STDBY do not need to be derated because they have only one connector pin. The +5 VDC and +3.3 VDC supplies, however, have multiple pins and must be derated.

Wide temperature boards (e.g. some military modules should not use the standard VMEbus connector. Those boards should use wide temperature connectors.

6.3 Caution - Shorting +/- 12 VDC To Signal Traces

Shorting +/-12 VDC to signal traces can destroy bus interface ICs. This can easily happen when measuring the power supply voltage levels on the rear of the backplane. As shown in Figure 6-3, the +/-12 VDC power pins are next to address lines A01 and A08. If these address lines are shorted to the +/- 12 VDC power pins, then the bus interface ICs on every board can be destroyed. Voltmeter leads, oscilloscope probes and wedding rings have been known to cause the problem.

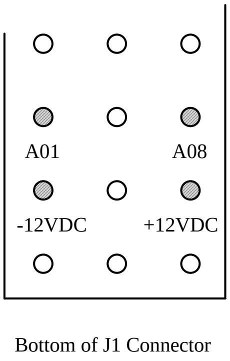


Figure 6-3 Relative location of address lines A01 and A08 in relation to +/- 12 VDC

The problem is aggravated when the VMEbus chassis is located in a dimly lit area. This makes it hard for service personnel to see the pins. The problem can be avoided by taking special care when working near the +/- 12 VDC pins on the backplane. Clipping or insulating the +/- 12 VDC pins on the rear of the backplane, measuring the voltages on the front of the backplane or measuring them at the backplane power connections all minimize the risk of shorting these pins.

6.4 Characteristics Of Bus Interface ICs

Signals are sent between VMEbus modules using TTL voltage levels. TTL, or Transistor-Transistor-Logic (sometimes called T^2L or T-squared-L), was selected for VMEbus because of its high drive current, high speed, high reliability and low cost.

6.4.1 Electrical Interface Checklist

The following checklist may be helpful to board designers when selecting VMEbus interface ICs:

1) ***Device type.*** Determine the type of IC that is being specified. For example, an IC that is connected to data line D00 is called a *standard three-state device*. Once you find the device type, look it up in Chapter 6 of the VMEbus specification.

2) ***Noise margins (Vol, Voh, Vil, Vih, Iol, Ioh).*** The noise margins are one way of specifying the voltage levels (at the rated current) that ICs use to communicate with each other. Verify that the high and low voltage levels of each device meet or exceed the VMEbus specifications. On receivers and transceivers, check the input voltages. On drivers and transceivers check the output voltages. Also remember

that TTL output voltage levels are a function of current. For example, on a standard three-state driver IC you must verify that Voh is greater than or equal to 2.4 V at the rated current (Ioh) of 3 mA.

3) ***Load current (Iil, Iih, Iozl, Iozh, Ioz, Ii).*** The load current is the amount of current consumed by a device's input. Verify that transceiver and receiver ICs do not exceed the load current requirements of the device type. For example, the low-level input load current on a standard three-state device cannot exceed 0.7 mA.

   Refer to the section of this chapter on load currents for more information about interpreting Iil, Iih, Iozl, Iozh, Ioz and Ii.

4) ***Output short circuit current (Ios)***. The output short circuit current is a simple way of specifying the output impedance (at DC) of a driver IC. For example, a standard three-state device must have an output short circuit current of: 50 mA ≤ Ios ≤ 225 mA.

5) ***Input clamp voltage (Vclamp)***. Bipolar TTL receiver ICs usually have input clamping diodes to prevent excessive, negative voltage excursions. These excursions are caused by a combination of (a) backplane inductance and (b) fast edge speeds. All VMEbus boards must provide clamping on each VMEbus signal line that they monitor to prevent excursions below -1.5 V. Also note that some CMOS devices (such as the 74ACT logic family) might not provide these clamping diodes.

6) ***Capacitive loading (C)***. The VMEbus specification requires that capacitive loading on any signal trace be kept below 20 pF. This includes IC input capacitance, IC output capacitance and trace capacitance. Since trace capacitance is usually 2 - 3 pF, it is best to keep the IC capacitive load less than 17 - 18 pF.

6.4.2 Noise Margins

Electronic noise is present in all VMEbus systems and can never be eliminated. At low levels it should not cause problems, but at high levels it can render a system unusable. The amount of noise will directly correspond to the reliability of the system, and it is therefore prudent to reduce it as much as possible. Practical system integrators understand that noise cannot be eliminated with one or two solutions. A program of noise reduction is called for, employing a variety of techniques.

In digital systems the level of noise that can be tolerated depends upon the logic noise margins. For example, the simple circuit in Figure 6-4 shows two gates called A and B. Noise is added to the interconnection between A and B by some external source. If this noise is large enough, it could cause B to misread what A is sending. Simply stated, the noise margin is the level of noise that B can tolerate before it reads a false input.

In a logic circuit, binary data is represented by a logic low or a logic high. In an ideal TTL system we might expect logic low to be 0 volts, and logic high 5 volts. However, in practical TTL systems the logic low and high voltages are somewhere between 0 and 5 volts. The noise margin is also different for high and low-level signals. We refer to these as the low-level and high-level DC noise margins. The low-level margin is:

$$V_{nml} = V_{il} - V_{ol}$$

and the high-level margin is:

$$V_{nmh} = V_{oh} - V_{ih}$$

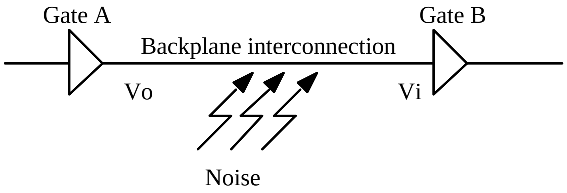


Figure 6-4 Noise margins are defined as the amount of noise that can be accepted by a circuit before it fails

For example, consider again the example in Figure 6-4. If the output low voltage (V_{ol}) of A is 0.6 volts, and the input low voltage (V_{il}) of gate B is 0.8 volts, then the noise margin of the circuit is $V_{nml} = V_{il} - V_{ol} = 0.8$ V - 0.6 V or 0.2 V. Up to 0.2 V (200 millivolts) of noise could be added to the signal before B receives a false input.

A similar situation happens for the high-level. If the output high voltage (V_{oh}) of A is 2.4 Volts, and the input high voltage (V_{ih}) of gate B is 2.0 Volts, then $V_{nmh} = V_{oh} - V_{ih} = 2.4 - 2.0 = 0.4$ V. By the way, the values given for V_{il}, V_{ol}, V_{oh} and V_{ih} in this example are defined as worst case by the VMEbus specification.

The output voltage of TTL devices is strongly dependent on how much current the device is sourcing or sinking. High and low-level output voltages are usually specified at the rated current of the IC.

Figure 6-5 shows the noise margins for several common bus interface ICs. Note that some families have better noise margins than others. The wider the margin the better.

In Figure 6-5 the area between V_{il} and V_{ih} is called the transition region. The actual voltage levels where switching occurs vary between logic families, manufacturers and individual ICs. For example, a transition from low to high may cause a 74LS645A-1 receiver to switch at about 1.0 volts, but a 74F245 may switch at 1.5 volts. This is true even though both of their V_{ih} voltages are rated at 2.0 volts. 2.0 volts would be the worst case V_{ih} over voltage and temperature.

Also note that most of the logic families have similar transition regions, except for the ETL logic. Among other things, VMEbus uses ETL logic for hot-swap boards. When hot-swap boards are inserted into a rack their ETL drivers are pre-charged to about 1.5 Volts. This, along with superior noise margins, prevents insertion noise (cause by capacitance on the board interconnections) from corrupting backplane signals.

Figure 6-5 also shows the backplane termination network voltage. This is the voltage of any un-driven signal, such as with an open-collector or three-state device. Here V_{oh} = 2.94 volts, resulting in a high-level noise margin of: V_{nmh} = 0.9 volts. This is an added benefit of the termination networks, and can be utilized when designing bus interface circuits.

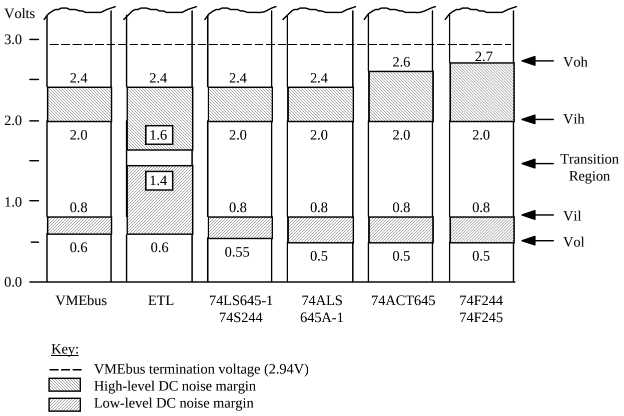


Figure 6-5 Noise margins for VMEbus and some popular logic families

6.4.3 Relation Between Drive Current, Load Current And Noise Margins

The drive and load characteristics of VMEbus interface ICs are carefully defined to insure compatibility between modules. One important reason for this is to prevent ICs from burning out due to overcurrent stresses. Another is to guarantee the minimum noise margins. TTL noise margins decrease as output currents increase. Because of this, the VMEbus specification limits the load that can be applied to any signal.

TTL outputs have some resistance (and impedance) associated with them. An output of a TTL gate is shown in Figure 6-6. When its output is high, transistor Q1 is turned on and Q2 is off. This allows current to flow from Vcc, through resistor R and transistor Q1 to the output. Because of the resistance of R and Q1, the output voltage decreases as the device tries to source more current. Therefore as the output high current goes up, the output voltage goes down, and the upper noise margin decreases.

A similar scenario happens when the low-level sink current goes up. When the TTL output is in its low-level, Q1 is turned off, and Q2 is turned on. Because of the resistance of Q2, the output voltage level increases as the low-level sink current increases. This decreases the lower noise margin.

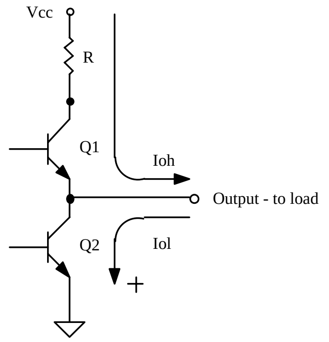
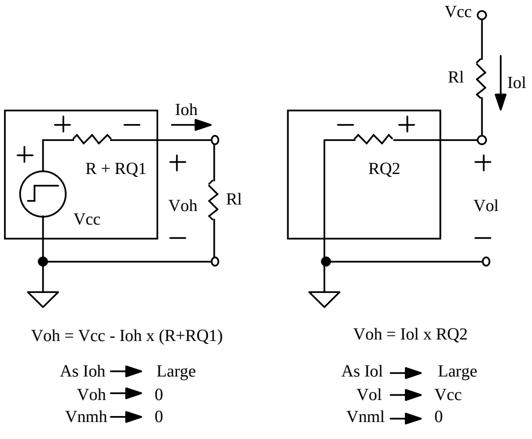


Figure 6-6 TTL output



(a) Output high: Q1 on, Q2 off. (b) Output low: Q1 off, Q2 on

Figure 6-7 Thevenin equivalents of a TTL output stage.

These diagrams are useful for explaining why noise margins diminish as output current increases.

To better understand why noise margins degrade with current, consider the Thevenin equivalent of a TTL output shown in Figure 6-7(a). Here Voh goes down as the IC output is loaded with more gates (I_{oh} gets large). When this happens, the voltage drop across the internal resistance becomes large and Voh decreases. This degrades the upper noise margin.

A similar situation happens for the lower noise margin. The equivalent circuit of Figure 6-7(b) shows how V_{ol} increases as the IC output becomes heavily loaded (I_{ol} increases). This causes V_{ol} to increase, degrading the lower noise margin.

The Thevenin equivalent models of Figure 6-7 are somewhat simplistic. Q1 and Q2 don't act like perfect resistors as the model suggests. These models are, however, useful for understanding the behavior of TTL outputs as they become heavily loaded. As more VMEbus modules are added to the system, the DC noise margins degrade.**Load Current Defined**

Some users find that the VMEbus IC drive and load currents are hard to understand. This section defines how to interpret the bus specifications.

There are two things that load a VMEbus signal at DC: the backplane terminators and the ICs of other modules. The backplane termination networks are fixed on all systems. The load current from the receiver ICs vary with the number of modules in the system and their individual loading.

At higher frequencies the resistive loading of other bus modules becomes negligible, and the backplane impedance becomes the predominate loading factor.

VMEbus modules must limit the loading of each signal connection. The calculation of this current is straightforward using IC manufacturers' data sheets. If a signal is loaded with a receiver IC, the value of I_{il} and I_{ih} is used. If the signal is loaded with a three-state device in the off state (such as a data line), the three-state off current I_{ozh} or I_{ozl} should be used.

Transceiver and receiver ICs are sometimes used in combination on a signal line. For example a CPU module with interrupter may have to drive address lines A01-A03 during read/write cycles, and monitor them during interrupt acknowledge cycles. In these cases, the transceiver (or buffer) three-state off current is added to the input high or low current. This is shown in Figure 6-8 along with the correct current direction (polarity).

6.4.5 Output Short Circuit Current

The output short circuit current (I_{os}) is an indicator of the output impedance of a buffer. It is the amount of current which flows through the output of an IC when it is grounded.

To understand I_{os}, consider the Thevenin equivalent circuit of Figure 6-9. This models the TTL output of Figure 6-6 when its output is shorted to ground (through an amp meter). The saturated collector-emitter voltage of the NPN transistor is about 0.2 volts. If the current sourced by the output is measured, then the internal value of R can be calculated with Ohms law.

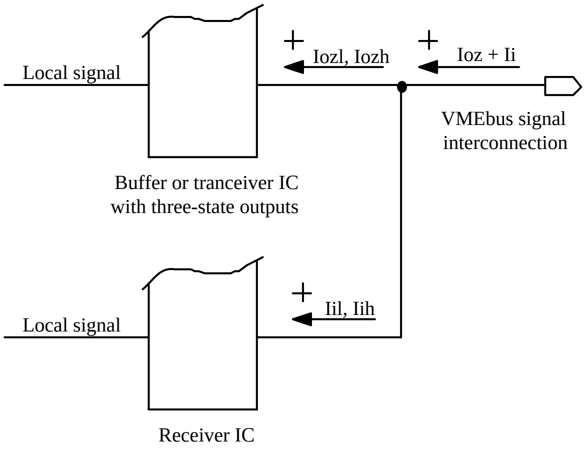

Local signal
+
Iozl, Iozh
+
Ioz + Ii
VMEbus signal interconnection
Buffer or tranceiver IC with three-state outputs
Local signal
+
Iil, Iih
Receiver IC

Figure 6-8 Block diagram showing how VMEbus load currents are defined

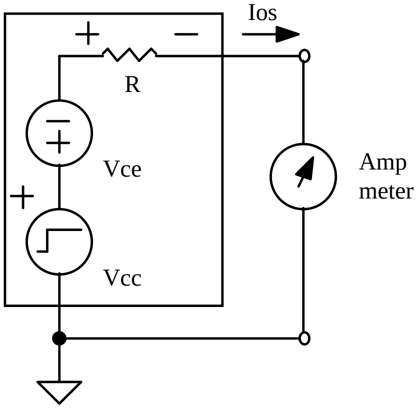


Figure 6-9 Thevenin equivalent circuit for a TTL output shows how I_{os} is measured

For example, if I_{os} of a IC is 100 mA, we find that:

$R = V_{cc} - V_{ce} / I_{os}$

R = 5.0 V - 0.2 V / 0.1 A

R = 48 ohms

Here R is roughly the output impedance of the IC in the high state. The VMEbus specification specifies I_{os} in order to control the output impedance of some of driver ICs. The output impedance of an IC is important to minimize impedance mismatches (which show up as noise on the backplane), and to insure sufficient rise and fall times during switching.

6.4.6 Input Clamp Voltage

In an ideal TTL system, signal voltages outside of the 0 to 5 volt power supply range should never be expected. However, in real systems excursions can be found outside of this range on signal traces. These are caused mostly by signal trace inductance and noise. Modern TTL devices can tolerate these transients, but their input characteristics should be double checked before using them on a bus interface.

Some logic families can handle positive going transients to 5.5 volts, and most to 7.0 volts. The VMEbus specification does not specify a maximum logic voltage, and it is rarely a problem. Negative excursions, however, must be clamped below -1.5 volts by every receiver. Most logic manufacturers' data sheets will specify the reverse clamp voltage of input buffers.

Exercise caution when using low power Schottky (LS) ICs with PNP inputs. Some LS ICs have pap inputs, and some don't. Negative voltage transients on these inputs can cause slow recovery times, sometimes in excess of 150 nanoseconds. In some cases, this recovery time can violate VMEbus timing specifications.

6.4.7 Capacitive Loading

Even though TTL is capable of driving highly capacitive loads, the total capacitance on any signal trace must be limited. This is because propagation delays through bus interface ICs can be lengthened when outputs are heavily loaded with capacitance.

The primary sources of capacitance are connector pins, IC pins and signal traces. The VMEbus specification requires that capacitive loading be kept below 20 pF on modules. Most IC manufacturers do not rate the worst case capacitive load they present, and signal traces are hard to control. A good rule of thumb when designing bus modules is to limit each bus interface signal to two IC pins, with less than two inches of PC traces between the connector pin and the IC pins (including branches)

If exact capacitance values are not available, the VMEbus specification recommends the values given in Table 6-2. It further suggests that if a driver propagation delay is available for a 300 pF load, use that timing parameter. If the only propagation delay values are for a 30 pF load, add 10 nanoseconds to the propagation delay and 15 nanoseconds to the turn-on delay.

Table 6-2 Recommended capacitance values

Item	Typical Capacitance
Receiver IC pin	3 – 5 pf
Driver IC pin	10 -12 pf
Transceiver IC	15 – 18 pf
pin	
2 inch PC trace	2 – 3 pf

6.5 Classes Of Bus Interface ICs

There are six classes of bus interface ICs used on VMEbus. These include standard totem-pole, high current totem-pole, standard three-state, high current three-state, open-collector and ETL devices. The signal names and their associated receiver and driver types are shown in Table 6-3.

Table 6-3 Summary of signal classifications (using non-ETL protocols).

Signal Name	Type
AS*	High current, three-state
ACFAIL*	Open collector
AM0 - AM5	Standard, three-state
A01-A31	Standard, three-state
BBSY*	Open collector
BCLR*	High current totem-pole
BERR*	Open collector
BG0IN* - BG3IN*	Standard totem-pole
BG0OUT* - BG3OUT*	Standard totem-pole
BR0* - BR3*	Open collector
DS0*, DS1*	High current, three-state
D00 - D31	Standard, three-state
DTACK*	Open col. / High current, 3-state (†)
IACK*	Open collector or standard three-state
IACKIN*	Standard totem-pole
IACKOUT*	Standard totem-pole
IRQ1* - IRQ7*	Open collector
LWORD*	Standard, three-state
RESP* (††)	ETL (used in 2eVME cycles only)
RETRY* (†)	High current, three-state
SERCLK* (†)	High current totem-pole
SERDAT* (†)	Open collector
SYSCLK	High current totem-pole
SYSFAIL*	Open collector
SYSRESET*	Open collector
WRITE*	Standard, three-state

(†) Parameter change under VME64
(††) Parameter change under VME64x

6.5.1 Standard Three-state

The standard three-state drivers and receivers are used for most of the VMEbus signals. As Table 6-4 shows, they must be able to source 3 mA and sink 48 mA. Their outputs must be capable of three-state operation (also known as Tri-state™). Note that IACK* can be a standard three-state or an open-collector device.

Table 6-4 Characteristics of standard three-state drivers and receivers

Standard three-state

Where used:

AM0-AM5	A01-A31	D00-D31
IACK*	LWORD*	WRITE*

Parameter	Conditions	MIN	MAX	UNIT
Vclamp		-1.5	0	V
Vih			2.0	V
Vil		0.8		V
Voh	Io = -3 mA	2.4		V
Vol	Io = 48 mA		0.6	V
Ioh		-3		mA
Iol		48		mA
Ioz1 + IIL	Vol = 0.6 V	-700		µA
Iozh + IIH	Voh = 2.4 V		150	µA
Ios	Vo = 0 V	50	225	mA
Ct			20	pF

Suggested devices:

- Drivers: 74LS645-1, 74ALS645A-1, 74F244, 74AS573, 74AS580, 74F543, 74F652
- Receivers: 74LS240, 74LS241, 74LS244, 74F543, 74F652, PAL16L8
- Transceivers: 74LS645-1, 74ALS245-1, 74ALS646-1, 74ALS648-1, 74F543, 74F652
- Avoid 74ASXX, 74FXX ad 74ACTXXX devices for D00-D15 because of fast edge speeds (they generate a lot of ground bounce).

6.5.2 High Current Three-state

The high current three-state drivers and receivers are similar to standard three-state except they are capable of higher drive currents. They are used on signals with high edge speeds. Timing for almost all address and data transfers are measured with respect to high current outputs such as address and data strobes (AS*, DS0*, DS1*). The higher edge speeds permit tighter timing which speed up bus transfers. A summary of high current three-state drivers and receivers is shown in Table 6-5.

Under the VME64 standard, DTACK* could be driven by either an open-collector or a high current three-state driver. This allows it to be operated as a rescinding signal.

Table 6-5 Characteristics of high current three-state drivers and receivers

High current three-state

Where used:

AS*	DS0*	DS1*
DTACK* (†)	RETRY* (†)	

Parameter	Conditions	MIN	MAX	UNIT
Vclamp		-1.5	0	V
Vih			2.0	V
Vil		0.8		V
Voh	Io = -3 mA	2.4		V
Vol	Io = 64 mA		0.6	V
Ioh		-3		mA
Iol		64		mA
Ioz1 + IIL	Vol = 0.6 V	-450		µA
Iozh + IIH	Voh = 2.4 V		100	µA
Ios	Vo = 0 V	50	225	mA
Ct			20	pF

Suggested devices:

- Drivers: 74S241, 74S244, 74F241, 74F244
- Receivers: 74LS240, 74LS241, 74LS244, PAL16L8

(†) Changes under VME64

6.5.3 Standard Totem-Pole

The standard totem-pole drivers and receivers are used for the bus arbitration and interrupt acknowledge daisy-chains. Most bipolar families of the 7400 series logic will meet the standard totem-pole requirements. A summary of their characteristics is shown in Table 6-6.

Table 6-6 Characteristics of standard totem-pole drivers and receivers

Standard totem-pole

Where used:

BG0IN*	BG0OUT*	IACKIN*
BG1IN*	BG1OUT*	IACKOUT*
BG2IN*	BG2OUT*	
BG3IN*	BG3OUT*	

Parameter	Conditions	MIN	MAX	UNIT
Vclamp		-1.5	0	V
Vih			2.0	V
Vil		0.8		V
Voh	Io = -0.4 mA	2.4		V
Vol	Io = 8 mA		0.6	V
Ioh		-0.4		mA
Iol		8		mA
Ioz1 + IIL	Vol = 0.6 V	-600		µA
Iozh + IIH	Voh = 2.4 V		50	µA
Ct			20	pF

Suggested devices:

- Drivers: Most LS, S, F, AS, ALS, and PLD outputs
- Receivers: Most LS, F, AS, ALS, and PLD inputs

Avoid driving daisy chain using flip-flops with unbuffered outputs (see text)

When selecting daisy-chain driver ICs, avoid using flip-flops with unbuffered outputs. These flip-flops can produce short glitches which are caused by a feed-back from the device's output. Programmable logic devices usually have flip-flops with buffered outputs, and should be acceptable.

6.5.4 High Current Totem-pole

High current totem-pole outputs are similar to high current three-state outputs except they are never turned off. In most cases a high current totem-pole driver can be substituted by a high current three-state driver (with outputs permanently enabled). Characteristics of the high current totem-pole drivers and receivers are shown in Table 6-7.

Table 6-7 Characteristics of high current totem-pole drivers and receivers

<table>
<tr><th colspan="5">High current totem-pole</th></tr>
<tr><td colspan="5">Where used:
BCLR* SYSCLK SERCLK</td></tr>
<tr><td>Parameter</td><td>Conditions</td><td>MIN</td><td>MAX</td><td>UNIT</td></tr>
<tr><td>Vclamp</td><td></td><td>-1.5</td><td>0</td><td>V</td></tr>
<tr><td>Vih</td><td></td><td></td><td>2.0</td><td>V</td></tr>
<tr><td>Vil</td><td></td><td>0.8</td><td></td><td>V</td></tr>
<tr><td>Voh</td><td>Io = -3 mA</td><td>2.4</td><td></td><td>V</td></tr>
<tr><td>Vol</td><td>Io = 64 mA</td><td></td><td>0.6</td><td>V</td></tr>
<tr><td>Ioh</td><td></td><td>-3</td><td></td><td>mA</td></tr>
<tr><td>Iol</td><td></td><td>64</td><td></td><td>mA</td></tr>
<tr><td>Ioz1 + IIL</td><td>Vol = 0.6 V</td><td>-600</td><td></td><td>µA</td></tr>
<tr><td>Iozh + IIH</td><td>Voh = 2.4 V</td><td></td><td>50</td><td>µA</td></tr>
<tr><td>Ios</td><td>Vo = 0 V</td><td>50</td><td>225</td><td>mA</td></tr>
<tr><td>Ct</td><td>System controllers w/drivers</td><td></td><td>20</td><td>pF</td></tr>
<tr><td>Ct</td><td>Modules with no drivers</td><td></td><td>12</td><td>pF</td></tr>
<tr><td colspan="5">Suggested devices:
• Drivers: 74S241, 74F241, 74F244
• Receivers: 74LS240, 74LS241, 74LS244, PAL16L8</td></tr>
</table>

6.5.5 Open-Collector

Open-collector logic is used whenever signals need to be wire-or'ed. Wire-or signals are used when more than one bus module must drive a signal at one time. Characteristics of the drivers and receivers are shown in Table 6-8. Note that IACK* can be an open-collector or a standard three-state device.

Under the VME64 standard, DTACK* could be driven by either an open-collector or a high current three-state driver. This allows it to be operated as a rescinding signal.

Table 6-8 Characteristics of open-collector drivers and receivers

<table>
<tr><th colspan="5">Open collector</th></tr>
<tr><td colspan="5">Where used:</td></tr>
<tr><td>ACFAIL*</td><td>DTACK* (†)</td><td colspan="2">IRQ1*</td><td>SERDAT*</td></tr>
<tr><td>BBSY*</td><td>IACK*</td><td colspan="2">IRQ2*</td><td>SYSFAIL*</td></tr>
<tr><td>BERR*</td><td></td><td colspan="2">IRQ3*</td><td>SYSREST*</td></tr>
<tr><td>BR0*</td><td></td><td colspan="2">IRQ4*</td><td></td></tr>
<tr><td>BR1*</td><td></td><td colspan="2">IRQ5*</td><td></td></tr>
<tr><td>BR2*</td><td></td><td colspan="2">IRQ6*</td><td></td></tr>
<tr><td>BR3*</td><td></td><td colspan="2">IRQ7*</td><td></td></tr>
<tr><td>Parameter</td><td>Conditions</td><td>MIN</td><td>MAX</td><td>UNIT</td></tr>
<tr><td>Vclamp</td><td></td><td>-1.5</td><td>0</td><td>V</td></tr>
<tr><td>Vih</td><td></td><td></td><td>2.0</td><td>V</td></tr>
<tr><td>Vil</td><td></td><td>0.8</td><td></td><td>V</td></tr>
<tr><td>Vol</td><td>Io = 48 mA</td><td></td><td>0.6</td><td>V</td></tr>
<tr><td>Iol</td><td></td><td>64</td><td></td><td>mA</td></tr>
<tr><td>Ioz1 + IIL</td><td>Vol = 0.6 V, DTACK*, BERR*</td><td>-400</td><td></td><td>µA</td></tr>
<tr><td>Ioz1 + IIL</td><td>Vol = 0.6 V, all signals other than DTACK* and BERR*</td><td>-600</td><td></td><td></td></tr>
<tr><td>Iozh + IIH</td><td>Voh = 2.4 V</td><td></td><td>50</td><td>µA</td></tr>
<tr><td>Ios</td><td>Vo = 0 V</td><td>50</td><td>225</td><td>µA</td></tr>
<tr><td>Ct</td><td></td><td></td><td>20</td><td>pF</td></tr>
<tr><td colspan="5">Suggested devices:
• Drivers: 74S38, 74F38, 74LS641-1, 74LS642-1, 74F641, 74F642
• Receivers: 74LS14, 74LS240, 74LS241, 74LS244, PAL16L8
(†) Changes under VME64</td></tr>
</table>

6.5.6 ETL (Enhanced TTL Transceiver Logic

ETL logic was first specified in the VME64x standard for 2eVME bus cycles. However, at the time this book was printed it was also being specified in the VMEbus hot-swap standard(s). ETL is specified in the VITA 2-199x draft standard.

ETL is used on VMEbus because it is (or has):

- Compatible with TTL logic.
- Wider noise margins.
- High drive current (at least 64 mA).
- A pre-charge pin (for hot swap applications).
- Controlled rise and fall times.

Characteristics of ETL drivers and receivers are shown in Table 6-9.

Table 6-9 Characteristics of ETL drivers and receivers

ETL – Enhanced TTL Transceiver Logic (†††)				
Where used: All signals participating in 2eVME cycles (††)				
Parameter	Conditions	MIN	MAX	UNIT
Vclamp		-1.5	0	V
Vih			1.6	V
Vil		1.4		V
Voh	Io = - 64 mA	2.4		V
Vol	Io = 64 mA		0.6	V
Ioh		- 64		mA
Iol		-64		mA
Ioz1 + IIL	Vol = 0.6 V		10	µA
Iozh + IIH	Voh = 2.4 V		10	µA
Ct			20	pF
Suggested devices: SN74ABTE16245m 246 (††) Changes under VME64x (†††) Proposed enhancement under VITA 2-199X (draft) standard				

6.6 Backplane Impedance Control And Noise

Maintaining constant characteristic impedance on bus modules and backplanes is important for reliability. That's because impedance mismatches can cause signal reflections which show up as noise, sometimes causing system crashes. Controlling the characteristic impedance of bus modules and backplanes improves system reliability.

Those unfamiliar with characteristic impedance may find it hard to understand at first. The concepts of characteristic impedance and reflection is analogous to a length of rope tied to a wall as in Figure 6-10. If the free end of the rope is rapidly moved up and down, it would cause a wave to travel along its length (A). When the wave hits the wall it can't pass through, and its energy is reflected back toward the source of the wave (B). The wave traveling along the rope is analogous to VMEbus signal propagation, and the wall is analogous to an impedance mismatch at the end of the backplane.

The reflected wave is unwanted, and appears as noise. To minimize this noise, a termination network is placed at both ends of the VMEbus backplane and has the effect of damping out the reflected waves. If the characteristic impedance of the termination network exactly matches that of the signal trace, then there is no reflected wave.

Wave reflections can also happen in the middle of the backplane. This is analogous to a length of rope attached to a steel cable as in Figure 6-11. The point where the rope is tied to the cable is called a *discontinuity*. In other words, it is the point where the mass of the propagation medium changes from low to high. When a wave hits the discontinuity two waves are generated. The first wave, called the *reflected wave*, is sent back to the source just as in the wall example of Figure 6-10. Another wave, called the *transmitted wave*, propagates across the discontinuity. Both the reflected wave and the transmitted wave are smaller than the incident wave.

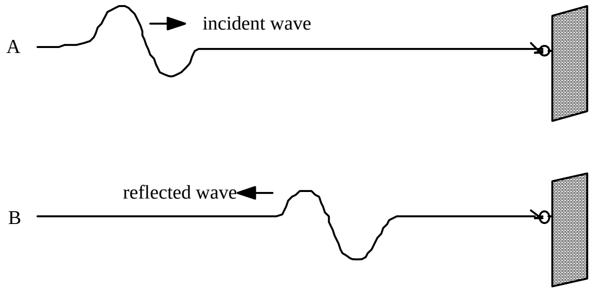


Figure 6-10 A mismatch of characteristic impedance occurs at the ends of the VMEbus backplane

This can be demonstrated using a length of rope tied to a wall.

A similar situation happens with electronic signals. Instead of traveling through rope and cable, they travel through an electric conductor. The discontinuities are caused by impedance changes, not mass changes. When an electronic signal hits the impedance discontinuity, some of the signal is reflected and some is passed over the discontinuity. Discontinuities occur in the backplane when signal traces pass over connectors, attach to VMEbus modules or run through PC board feed-through holes. Noise due to impedance mismatches is minimized by maintaining a constant impedance throughout the backplane.

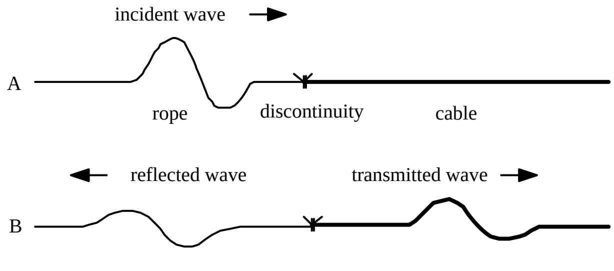


Figure 6-11 Impedance mismatches, also called discontinuities, occur at many points on the VMEbus backplane

They are analogous to a length of rope tied to a cable.

The characteristic impedance of a signal trace can be mathematically derived using the model shown in Figure 6-12. Here a VMEbus signal trace, also called a *transmission line,* is modeled using an infinite number of inductors and capacitors. Each of the incremental inductors (L_O) and capacitors (C_O) are identical. Characteristic impedance is the ratio of the incremental inductance to the incremental capacitance:

$$Z_O = L_O/C_O$$

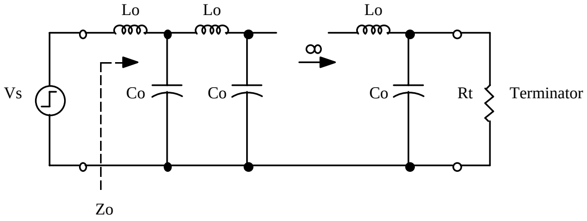


Figure 6-12 Model used to derive characteristic impedance

The derivation of this equation from Figure 6-12 is beyond the scope of this book. However, the books listed in the Bibliography, or any good physics text, will describe the concept of characteristic impedance in more detail.

Note that the impedance Z_O is constant regardless of the length of the transmission line. Since L_O and C_O are incremental values, it does not matter if the length of the trace is one inch or one foot. The impedance will be the same for any length of transmission line.

Now consider the block diagram of Figure 6-13. Here we have a transmission line of impedance Z_O connected to one of impedance Z_1, such as the case when a signal trace meets a bus module. When a signal traveling through Z_O encounters a transmission line of impedance Z_1 it goes through the discontinuity. When a signal does so, some of the wave propagates through it, and some is reflected backwards. In ideal systems there is no discontinuity and all of the wave propagates through the system with no losses.

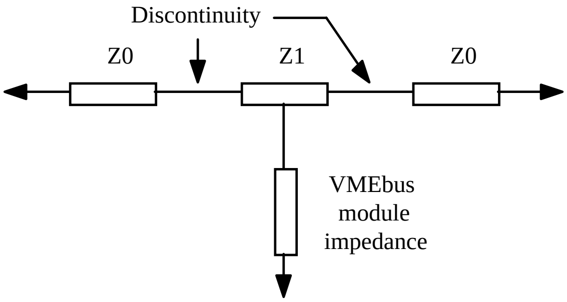


Figure 6-13 Impedance discontinuities on VMEbus backplanes

The reflections due to impedance discontinuities show up as noise on signal traces. Small discontinuities occur at many points on a VMEbus backplane. Every plated hole and connector pin introduces discontinuities. The result is small noise glitches which individually are of little consequence, but collectively can add up. Whenever possible, a constant impedance should be maintained on signal traces.

Special instruments can be purchased (or rented) for evaluating impedance mis-matches and propagation delays in VMEbus backplanes. Perhaps the most popular is the *time domain reflectometer* or TDR. These instruments inject a signal into a trace, and monitor the reflected waves from discontinuities.

The VMEbus specification requires that signal traces running to and from each connector pin be less than two inches in length. Unfortunately this rule is often overlooked in the final layout of VMEbus modules. When evaluating or designing modules, verify that this requirement is met (if possible).

The VMEbus specification suggests that signal trace dimensions be selected to keep impedance close to 100 ohms. When plated through holes, connectors and modules are added, however, the impedance can drop to around 20-50 ohms. Controlling the impedance helps insure that VMEbus modules from many vendors work together.

At each end of the backplane a discontinuity occurs at the ends of signal traces. At that point the impedance of the signal trace does not match that of free space. To reduce the amplitude of reflections off the ends of the backplane, termination resistors are required.

A termination network uses two resistors as shown in Figures 6-14(a) and 6-14(b). When combined they give the Thevenin equivalent resistance of about 194 ohms as shown in Figure 6-14(c). While this is not a perfect match to the backplane impedance of 20 - 100 ohms, it is close. A perfect match would require smaller termination resistors. *Smaller networks are not used because bus transceiver ICs would have to source or sink more current at DC. This means that the VMEbus* *backplane will always operate in an un-matched condition.*

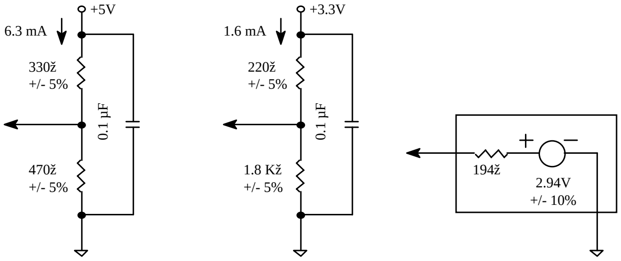
Most bus driver ICs actually use the backplane reflection to force their outputs from low to high. These reflections are actually helpful in this respect, but they should not be so excessive as to cause ringing. This return current is often called *standing current* and helps TTL output driver transistors to switch off faster, thereby increasing rise time.

Some logic families (such as ETL) use a technique called *incident wave switching* to reduce the reflections from the ends of the backplane. This technique controls the rise and fall time of the signal edges, which has the effect of reducing reflections. Incident wave switching also cuts crosstalk on the backplane and in the connectors.

The VME64x standard also allows +3.3 VDC terminators as shown in Figure 6-14(b). These require less power than +5 VDC terminators. Standard, +5 VDC, 6U VMEbus backplanes with passive terminators draw about 1.3 amps of current (dissipating 6.5 Watts). Lower voltage +3.3 VDC backplanes can reduce this to about 0.3 amps of current (dissipating about 1.1 Watts).

Active termination networks are used by some manufacturers to reduce power consumption even further. These are usually composed of an op-amp (with low output impedance) that is biased at 2.94 VDC. The output of the op-amp drives the signal interconnection through a 194 Ω series resistor.

Use a 0.01 or 0.1 µF bypass capacitor with all backplane terminators. This reduces power supply noise.



a) Standared termination network.

b) Low voltage network allowed under VME64x.

c) Thevenin equivalent for both terminators.

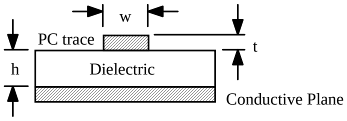
Figure 6-14 VMEbus termination network

6.6.1 Microstrip Lines

Two common types of transmission lines are used on VMEbus modules and backplanes. They are called the microstrip line and the strip line. The microstrip line is a printed circuit trace located above a power or ground plane. Its cross section is shown in Figure 6-15 along with the equations for its characteristic impedance, inductance and propagation delay.

William R. Blood, in his MECL System Design Handbook, reports that the equation shown for Z_O will predict the characteristic impedance to within +/- 5%. Z_O will vary, however, if the dimensions of the board are not kept constant or when plated through holes are used.

Note that the propagation delay of the trace is dependent on the dielectric constant of the printed circuit. A typical propagation delay for the microstrip (using G-10 epoxy board) is about 1.7 ns/ft.



Characteristic impedance $$Zo = \frac{87}{\sqrt{E_r + 1.41}} \ln\left(\frac{5.98h}{0.8w + t}\right) \text{ ž}$$

Inductance/Foot $$Lo = Zo^2 Co$$

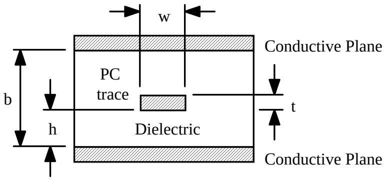
Propagation Delay $$tpd = 1.017 \sqrt{0.475 E_r + 0.67} \text{ nanosecond/ft}$$

Where Zo = characteristic impedance, Er = dielectric constant (~ 5 for G-10 epoxy board), hwt = dimensions given in figure (t~ 0.0015" for 1 oz copper), Lo = inductance/ft, Co = capacitance/ft, tpd = propagation delay.

Figure 6-15 Microstrip line

6.6.2 Strip Lines

Traces sandwiched between two conductive planes are called strip lines. The cross section of the strip line is shown in Figure 6-16. The strip line is often used where many layers of traces are needed. Sandwiching them between power and ground planes also limits capacitive coupling between signal layers.



Characteristic impedance $$Zo = \frac{60}{\sqrt{E_r}} \ln\left(\frac{4b}{0.67\text{š}w\,(0.8 + \frac{t}{w})}\right) \text{ ž}$$

Propagation delay $$tpd = 1.017 \sqrt{E_r} \text{ nanosecond/ft}$$

Where Zo = characteristic impedance, Er = dielectric constant (~ 5 for G-10 epoxy board), hwtb = dimensions given in figure (t~ 0.0015" for 1 oz copper), tpd = propagation delay.

Figure 6-16 Strip line

W.R. Blood reports that the characteristic impedance shown in Figure 6-15 is accurate when w/(b-t) < 0.35 and t/b < 0.25.

The typical propagation delay for the strip line is about 2.3 ns/ft (using G-10 epoxy board).

6.7 References

ATM Cells Bus (ACB) on VME / VITA 22-199x (Draft 0.1) VITA, Scottsdale, AZ 1997

Blood, William R., MECL System Design Handbook, Motorola Semiconductor Products Inc. 1983

ETL Standard / VITA 2-199x (Draft 0.5). VITA, Scottsdale, AZ 1997

Extensions to ANSI/VITA 6-1994 Signal Computing System Architecture / ANSI/VITA 6.1-199x (Draft 5.1). VITA, Scottsdale, AZ 1996

FAST Data Book. Fairchild, South Portland, ME 1985

F100K ECL User's Handbook Fairchild, Mountain View, CA 1982

Ott, Henry W. Noise Reduction Techniques In Electronic Systems Wiley, New York, NY 1976

TTL Data Manual 1986 Signetics Corporation, Sunnyvale, CA 1986

Sokal, Nathan O. "Use Of Check List Prevents Problems In TTL Systems" *EDN Magazine* November 13, 1986

ALS/AS Logic Data Book Texas Instruments, Dallas, TX 1986.

VMEbus Specification / ANSI/IEEE STD1014-1987 VITA, Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA, Scottsdale, AZ 1995

VME64x Specification / VITA 1.1-1997 (Draft 2.0) VITA, Scottsdale, AZ 1997

Chapter 7
Mechanical Hardware

The Eurocard packaging system was chosen as the mechanical standard for VMEbus. The term *Eurocard* loosely describes a family of products based around a group of rack standards including the DIN 41494 and IEC 297-3, and the connector families in accordance with DIN 41612 and IEC 603-2. More recently the IEEE-1101/.1/.2/.10/.11 standards have also been used.

In many ways it was the mechanical standard that made VMEbus a popular bus architecture. Thousands of components are available from scores of manufacturers. This competitive environment helps keep costs down and product quality up.

7.1 VMEbus Connectors

Each VMEbus module has up to three connectors on the rear edge of the card. These are called the P0, P1 and P2 connectors. This naming convention follows the standard nomenclature where the plug connector (P0, P1 and P2) is located on the bus module, and the jack connector (J0, J1 and J2) is located on the backplane.

The P1/J1 connector is used on both the single height (3U) and double height (6U) bus modules. This connector contains most of the control signals, as well as 16 data lines and 23 address lines.

The P2/J2 connector is used only on double height (6U) bus modules. This connector has 16 data lines, 8 address lines and 64 user-defined I/O pins.

The P0/J0 connector was added in the VME64x standard. It has 95 pins, and is located between the P1/J1 and P2/J2 connectors. All of the pins on the P0/J0 connector are user defined.

Originally, VMEbus used the standard 96 pin DIN 41612 for the P1/J1 and P2/J2 connectors. However, there are two other versions of this connector that are important. These are the:

- Auto grant connector (not part of the VMEbus standard).
- 160 pin connector (added in the VME64 specification).

7.1.1 DIN 41612 Connector

The DIN 41612 connector was developed in Europe, and was later embraced by engineers elsewhere. As shown in Figure 7-1, each connector has three rows of 32 pins, thereby providing a total of 96 pins.

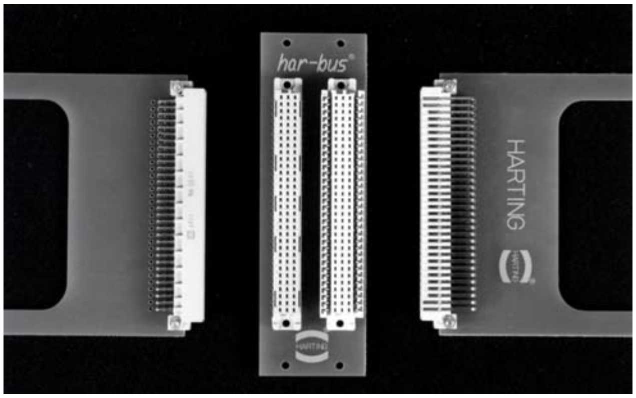


Figure 7-1 VMEbus connectors

From left to right: DIN 41612 plug connector,

DIN 41612 jack connector, 160 pin jack connector and 160 pin plug connector.

Samples courtesy of Harting Electronik, Inc. Photo by Wade Peterson.

DIN 41612 connectors are commonly available in four IEC quality classes: from one to four with class one being the highest. VMEbus requires at least a class two connector, which provides more than 400 cycles of mechanical endurance over the commercial temperature range (0 - 70 C).

Wide temperature military users must specify connectors that are better than the IEC class II products. That's because some of those systems must operate at temperatures higher than 70 C.

The DIN 41612 connectors are widely accepted as more reliable than printed circuit board edge connectors. That's because they provide a gas tight barrier in the contact region.

The pinout for the VMEbus DIN 41612 connectors is given in Chapter 1.

7.1.2 160 Pin Connector (†, ††)

The 160 pin connector was first introduced in the VME64 standard as a replacement for the DIN 41612. It adds 64 pins to each of the P1/J1 and P2/J2 connectors.

However, only 32 of these pins are defined in the VME64 standard. The rest were classified as ground or reserved pins. The reserved pins were later specified in the VME64x standard. A photo of the 160 pin connector is shown in Figure 7-1.

The 160 pin connector was a radical change to the VMEbus standard. It has thirty-two additional pins located on both sides of the original DIN 41612 connector. These are designated as the 'z' and 'd' rows. The new connector was added for two reasons:

- Additional signal connections allowed more backplane functions.
- Additional ground pins suppressed ground-bounce and cross-talk.

Users should note that early drafts of the VITA 1-1994 VME64 specification, before it received ANSI recognition, contained references to an enhanced DIN connector that contained an outer row of interconnected pins used to provide shielding. This connector was sometimes referred to as the *enhanced DIN connector*, and is manufactured by AMP, Inc. It was originally included as a method of providing better grounding for high speed VME64 signals.

Later, the VME64 standards committee abandoned this connector for one with individual outer rows of pins. Every other pin on the 'z' row of pins is grounded. This connector became known as the *160 pin connector*, and is now specified in the ANSI/VITA 1-1994, VME64 specification. At the time of printing of this handbook, this connector is being standardized within the 1076-4 family of IEC connectors.

The 160 pin connector is made by HARTING Inc. of North America (Elgin, IL, USA 847-741-1500). Several variations of the VME64x plug and jack connectors can be found in the HARTING *har-bus 64* connector family. Besides the right-angle plug and press-fit jack versions, there are also right-angle jack and wire-wrap tail types. These are used for VME64x extender cards and rear I/O connection points.

The two end pins on row 'd' of the 160 pin connector are elongated. These are defined by the VME64x standard as voltage pre-charge and ground pins. They are required for the new board level live insertion and high availability VME64x standards, and are used for a hot-swap capability. The elongated pins form a make-first, break-last contact, and allow ETL bus transceivers to be pre-charged during board insertion.

The new ground pins cut noise produced by ground-bounce and cross-talk. Earlier VMEbus systems were susceptible to these problems on several signal lines.

One especially problematic area is the notorious BBSY* glitch problem. This is caused by inductive coupling between the lower sixteen VMEbus data lines and the BBSY* signal. This coupling takes place (for the most part) within the connector itself. The BBSY* signal is located next to the lower sixteen data lines D00-D15. However, it is a considerable distance from the nearest ground pin. Ground loops are formed between the data lines and BBSY* pin, thereby forming inductive loops. The inductive loops act like the primary and secondary windings in a transformer. Signal transitions on the data lines are coupled into the BBSY* signal, where significant glitches are created. See Chapter 3 for more information about BBSY* glitches.

The 160 pin connector adds additional ground pins on the 'z' row, thereby cutting the distance between any signal pin and ground. This reduces the area (and inductance) of each pin, thereby lowering the induced noise on BBSY*. The new pinout reduces inductively-coupled noise by 50% - 80%. This is especially helpful, as this noise is exacerbated when using faster bus driver IC's such as 74ACT family devices. These chips have edge speeds that are two or three times faster than, say, 74LS parts. The faster edge speeds are necessary to boost bandwidth on VMEbus.

The 160 pin connector is 100% forward and backward compatible with the 96 pin DIN 41612 connector. For the most part, VME64x boards are able to plug into legacy backplanes. However, in this case the functions of the new connector pins will not be available because they are not connected to the backplane.

In general this should not be a problem, as the enhanced capabilities just won't be available on the board. However, there are still some important compatibility issues. For example, if a VME64x board requires a +3.3 VDC power supply, then it won't function in a legacy backplane (because the +3.3 VDC pins are on the row 'd' pins).

The VME64x standard allows manufacturers to design boards that *require* a VME64x backplane. When purchasing a VME64x board for use in a legacy backplane, the user is urged to check with the manufacturer to make sure it will work.

The pinout for the 160 pin connector is given in Chapter 1.

7.1.3 Auto Grant Connector

The auto grant connector is shown in Figure 7-2. This connector is compatible with standard DIN 41612 jack connectors, and provides an automatic switching mechanism for the bus grant and interrupt acknowledge daisy-chains. This eliminates the need for backplane jumpers.

The auto grant connector has a unique mechanical switch inside the insulator cavity of a standard DIN 41612 connector. It is mounted to the backplane in the same manner as the 96 pin DIN. When a VMEbus module is installed, the switches on the bus grant and interrupt acknowledge daisy-chains open, thereby routing the daisy-chain signals through the card. When the module is removed, the switches close and pass the daisy-chain signals on the backplane.

This connector eliminates the need to reconfigure backplane jumpers when VMEbus modules are installed or removed.

Two other ways of handling the daisy-chains are discussed in Chapter 8. These include auto grant backplanes and bypass boards.

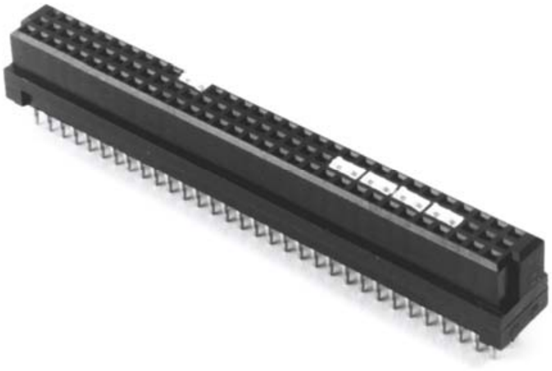


Figure 7-2 Auto grant connector

This connector eliminates the need for backplane jumpers with internal switches on the bus grant and interrupt acknowledge daisy-chains. The white areas on the connector face indicate the locations of the internal switches.

Photo courtesy of Augat Incorporated.

7.1.4 P0/J0 Connector (††)

A P0/J0 connector was added in the VME64x standard. This connector, which is shown in Figure 7-3, is placed between the P1/J1 and P2/J2 connectors.

The P0/J0 connector was added because of the higher I/O demands now placed on VMEbus systems. This is especially true in military, aerospace and telecom industries. For example, in many applications the use of front panel cables are frowned upon because:

- There is not much room for connectors on conduction cooled boards.
- Front panel cables are often susceptible to shock and vibration problems.
- Bus modules are harder to replace when cables are attached to the front panel.

The P0/J0 connector conforms to the IEC 1076-4-101 standard. These connectors are available from AMP Inc., and ERNI components. Unlike the DIN 41612 connector family, they include specifications for mating impedance and maximum capacitance. This makes them an excellent choice for high speed signals.

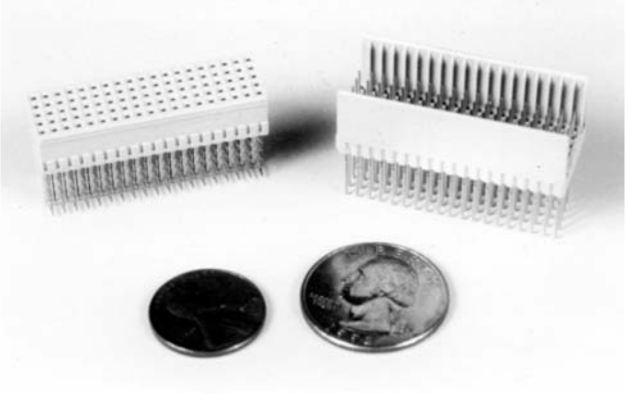


Figure 7-3 P0/J0 connector

Photo by Wade Peterson.

All 95 pins on the standard connector are user defined. A variant of the IEC 1076-4-101 connector can also be used. That connector has outer rows of shielded ground pins. The VME64x specification allows either type of connector to be used.

The VME64x standard also allows the use of custom connectors in the region between the P1/J1 and P2/J2 connectors. For example, a coaxial cable or fiber-optic connector could be used. This practice is not recommended, as the board will not be compliant with VME64x backplanes. However, if a custom connector must be used, then the VME64x standard recommends that the front panel keying mechanism be installed to prevent insertion of an incompatible module.

I/O from the P0/J0 connector can be:

- Routed out the rear of the backplane using ribbon cable connectors.
- Routed onto a rear transition module (plug-in unit).
- Bussed across the backplane.

Since the P0/J0 connector signals are user defined, they can be used as needed. However, users should be aware that some other standards have already assigned pinouts for this connector. For example, this is done in the ATM Cells Bus Standard (VITA 22-199x). That standard is intended for the telecom industry, and puts an ATM (Asynchronous Transfer Mode) communications port on the P0/J0 connector.

Rear I/0 for the P0/J0 user defined pins are handled in much the same way as it is on the P2/J2 user defined pins. For example, in the case of the ATM Cells Bus, a mezzanine backplane can be placed on the rear of the VMEbus backplane. This is also done on sub-buses such as VSB and SCSA. In these cases the mezzanine backplane has all necessary bussed and terminated signals. Similarly, users have the option of buying custom backplanes with these signals already bussed and terminated, thereby saving the cost of the mezzanine backplane.

Other standards are also mapped to the P0/J0 user defined pins. For example, pinouts already exist for user I/O from IP and CMC/PMC mezzanine modules. These are sanctioned by the VSO under documents which are separate from the VME64x standard.

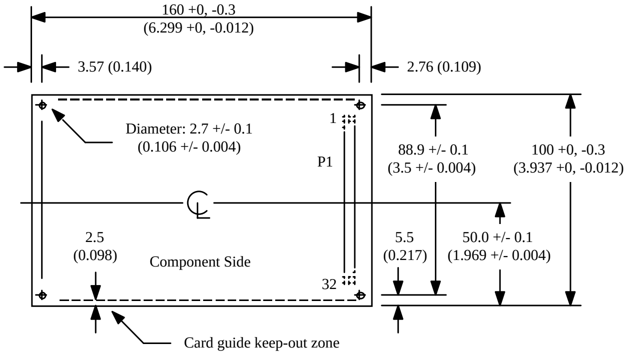
At first glance it would appear that all of these options on the P0/J0 connector could lead to incompatibilities. However, this problem should be no worse than that on the P2/J2 user defined pins. In fact, module keying (which we'll discuss shortly) can also be used to insure that VMEbus modules are placed into compatible backplane slots.

7.2 Card Sizes

VMEbus modules come in two sizes: single height (3U) and double height (6U). The single height module uses one connector, the double height uses two or three. The PWB dimensions for these two modules are shown in Figures 7-4 and 7-5. Both are 160 mm (6.299") deep...only their heights are different.

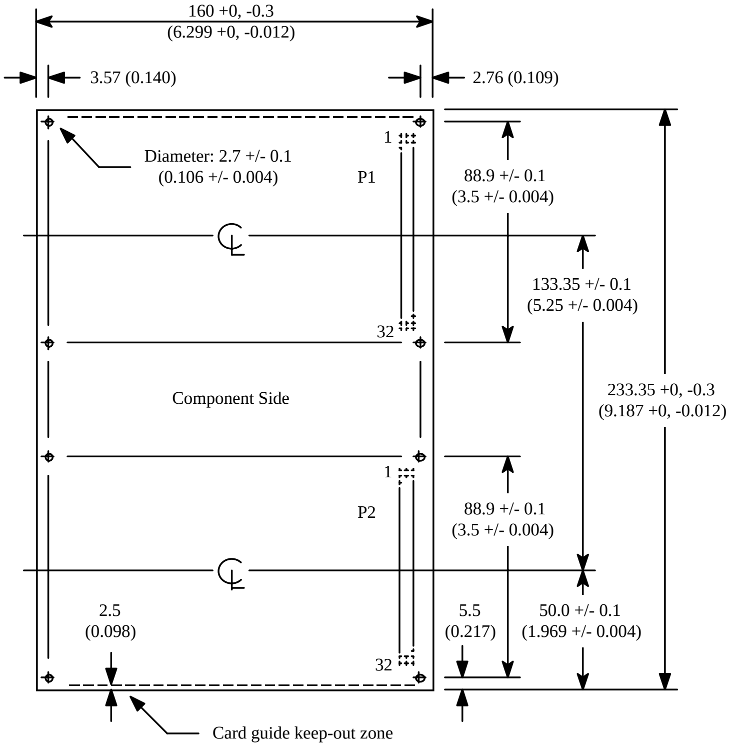
The VMEbus specifications only describe 3U and 6U bus modules. However, *ANSI/VITA 1.3, VME64x 9U x 400 mm Format* describes a larger card size.

In general, the larger the board the more susceptible it is to shock and vibration problems. That's because larger boards have a higher mass, and lower (mechanical) resonant frequency. Because of this, 3U modules are usually the most robust.



Notes:
1) All dimensions in millimeters. Inch dimensions in parentheses.
2) PC Boards must be 1.6 +/- 0.2 mm (0.063 +/- 0.008 inch) thick in the card guide keep-out zone.

Figure 7-4 Single height (3U) PWB dimensions



Notes:
1) All dimensions in millimeters. Inch dimensions in parentheses.
2) PC Boards must be 1.6 +/- 0.2 mm (0.063 +/- 0.008 inch) thick in the card guide keep-out zone.

Figure 7-5 Double height (6U) PWB dimensions

The distance between the card guides on the subracks determines the maximum height of components on the bus modules. The minimum VMEbus slot width is 0.8". That gives a maximum component height of 0.55" when using 0.063" thick circuit boards. The maximum lead length off the back of the module is 1.53 mm (0.06").

7.3 Front Panels

VMEbus modules generally use a front panel. The front panel is optional, but is useful for mounting LED lamps, switches and I/O connectors. It also prevents airflow from leaking out the front of the chassis, it can attenuate electromagnetic emissions and it increases board rigidity. Newer, VME64x front panels also support a board keying mechanism which insures that bus modules are installed in their correct backplane slots. If front panels are not used, a card ejector can be placed at the top and bottom of the module.

There are quite a few options when it comes to front panels. These include:

- Type of front panel (standard or EMC compatible).
- Handle style (fixed, ejector or injector / ejector).

- Handle location.
- Alignment pin.
- Keying capability.

7.3.1 Standard Front Panels

The standard front panel is usually formed from 0.1" thick aluminum or steel sheet stock. They are the most inexpensive type of front panel, but do not incorporate any of the EMC features (that we'll discuss shortly).

Standard front panels can be made from scratch or from pre-punched blanks. Custom front panels are necessary for several reasons: sometimes a board vendor doesn't supply a front panel, sometimes a logo needs to be applied, or sometimes a custom board is used.

Also, if modules from several vendors are used, the system may not look very good. Surface finish, materials, screw types, handle shapes and colors usually differ on purchased boards. These details are not standardized. The discriminating user may wish to hide the front of the VMEbus subrack, or create custom front panels for all of the boards. Figure 7-6 shows the steps necessary to make a semi-custom front panel. These include (from left to right in the photograph):

1) Pre-punched front panel blanks. These can be purchased from a reliable Eurocard vendor. Generally, a variety of handles, surface finishes and attachment hardware is available. Standard or EMC compatible blanks are available. The front panel can also be punched from raw sheet stock. This is probably less expensive in larger quantities.
2) Holes for switches, connectors and indicator lamps are drilled or punched.
3) Markings are silk-screen stenciled onto the punched front panel blank. Logos or other markings can be added to the small aluminum plates on the handles. However, front panels with injector/ejector handles usually don't have these plates.
4) Mounting hardware is added to the front panel in preparation for PC board attachment.

Single and double height VMEbus modules are electrically compatible, and may reside together in the same chassis. Figure 7-7 shows a 3U bus module with a 6U front panel.

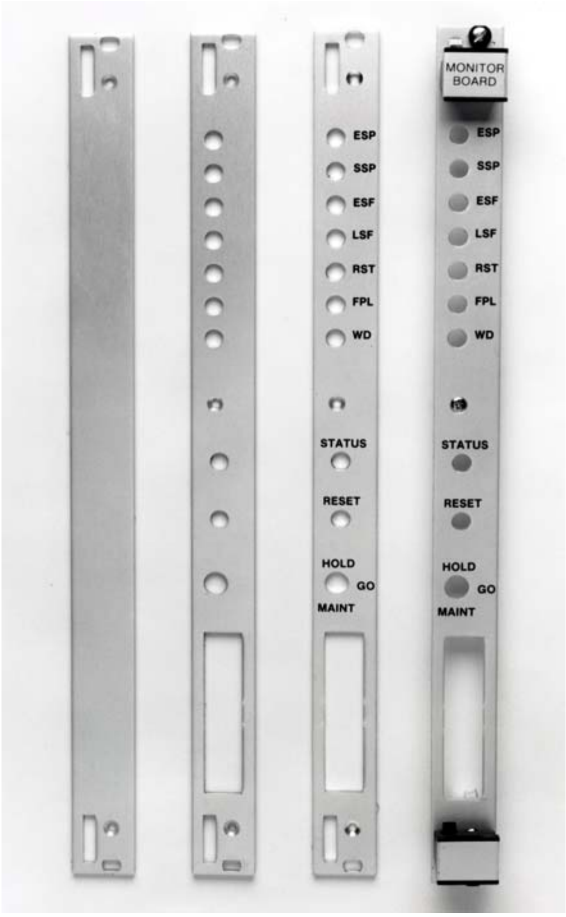


Figure 7-6 Steps necessary to make a semi-custom front panel

Photo by Rob Bigelow.

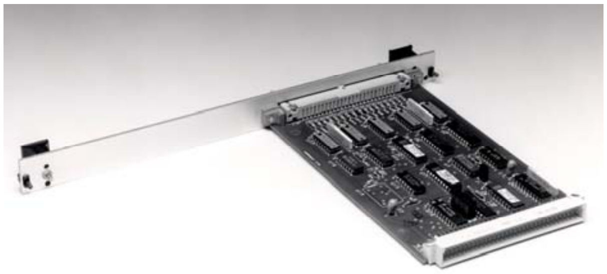


Figure 7-7 Front panel used to mount a single height board in a double height rack

Photo by Rob Bigelow.

7.3.2 EMC Front Panels (††)

An EMC (ElectroMagnetic Conformance) front panel was adopted in the VME64x standard. These help prevent electromagnetic waves from entering or leaving the VMEbus chassis through the front panel area. They are detailed in the IEEE 1101.10 specification, and have two major new components:

- Front panel EMI gasket.
- Reliable grounding through an alignment pin.

EMC performance has become a major issue for VMEbus system integrators. Since 1996, government laws in Europe have required that products entering the European Economic Community must conform to the 'CE' regulations. These include maximum EMI (ElectroMagnetic Interference) emissions and susceptibility requirements.

Figure 7-8 shows an EMC front panel. On the right side of the panel is an EMI gasket. This gasket makes an electrical contact with the front panel in the next card slot. This forms a nearly seamless ground plane along the front of the chassis, and prevents electromagnetic waves from penetrating through the narrow gap between the front panels.

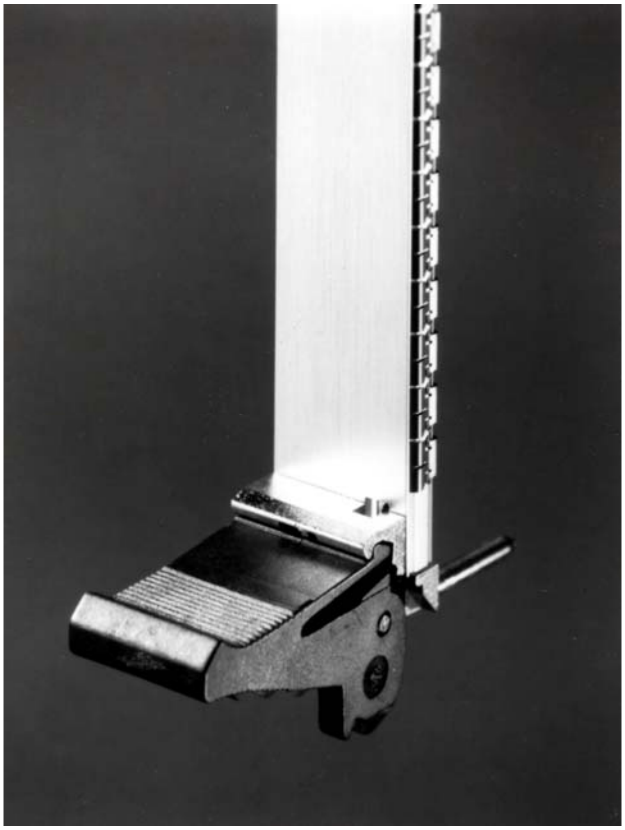


Figure 7-8 EMC front panel and injector/ejector handle

Note the gasket on the right side of the panel. Photo courtesy of Rittal Corporation.

An alignment pin provides a positive grounding path between the front panel and the subrack. As shown in Figure 7-9, the newer style card guide can accept the alignment pin through a round hole. Inside the hole is a clip which contacts the alignment pin, and provides a reliable ground to the subrack.

On bus modules, the front panel should always be grounded to the subrack, and never to the ground plane on the printed circuit board. This eliminates ground loops and reduces ESD (ElectroStatic Discharge) hazards.

Older style VMEbus front panels do not necessarily make a good, positive ground contact to the subrack. That's because:

- If the front panel screws aren't tight, there is sometimes an insulating gap between the front panel and the subrack.
- There was no requirement in the standard for a good electrical contact between the front panel and the subrack.
- Some VMEbus board manufacturers anodize their aluminum front panels. Anodizing creates an insulating oxide layer on the surface of the aluminum, thereby increasing the resistance between the front panel and the subrack.

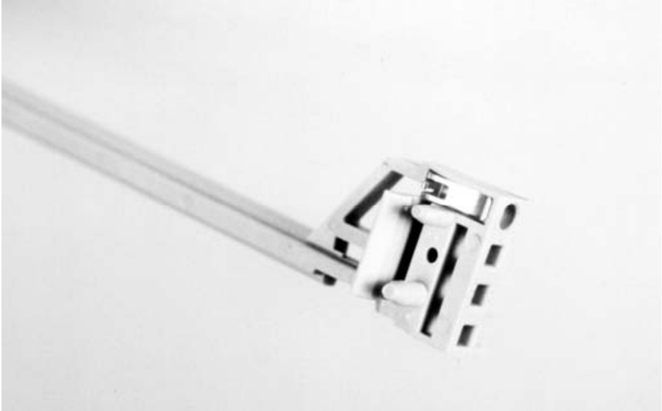


Figure 7-9 Card guide with alignment pin contact

Photo by Wade Peterson.

The author has had the opportunity to bring several VMEbus chassis through the CE conformance process. However, in most cases the new EMC front panels were not needed to meet the regulations. The bus modules did not emit sufficient electromagnetic radiation through the front panel (in the frequency ranges specified by the CE standard). However, it is comforting to know that the new front panels are available, if needed.

The VME64x standard also allows covers to be placed over the solder side of the circuit board. These prevent the EMC gasket from rubbing against components and leads on the rear side of the circuit board. Theoretically, this can happen when installing or removing a VMEbus module because the gasket extends past the interboard separation plane (i.e. it may interfere slightly with the next board).

In the author's experience, however, the rear covers are probably a poor idea, as they create as many problems as they eliminate. In industrial systems, for example, boards tend to be thrown around and handled in a rough manner. The aluminum rear covers often get dented and bent after a few years of service. This can cause problems, especially if the cover is damaged over a component lead. This can short out the lead, and cause the board to fail.

7.3.3 Handle Styles

There are generally three styles of VMEbus handles: fixed, ejector and injector/ejector. When designing or purchasing VMEbus modules, the user is urged to evaluate the handles on each bus module.

7.3.3.1 Fixed Handle

An example of a fixed handle is shown in Figure 7-10. These are the simplest and least expensive handles available on VMEbus. Generally, they are sufficient for 3U bus modules. However, they may not provide a comfortable extraction force on 6U bus modules, or modules that use the 160 pin or the P0/J0 connectors.

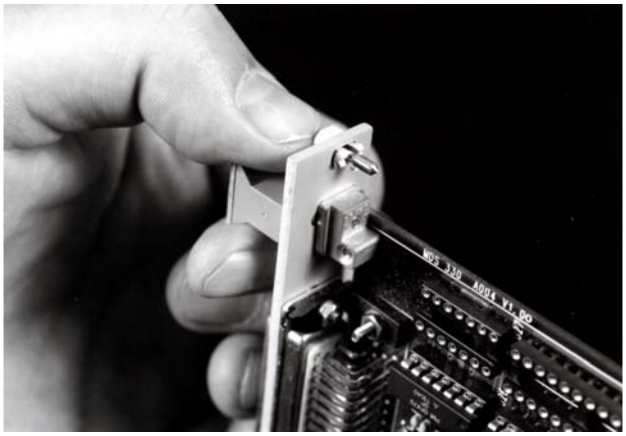


Figure 7-10 Fixed handle

Photo by Wade Peterson.

7.3.3.2 Ejector Handle

An example of an ejector handle is shown in Figure 7-11. The ejector handle is most popular on 6U bus modules. This handle provides a lever action to overcome extraction forces.

The fixed and ejector handles generally have the same shape, and insure that bus modules from different vendors look the same. However, the handles do come in different colors (usually black or gray) which detracts from the overall appearance of the system.

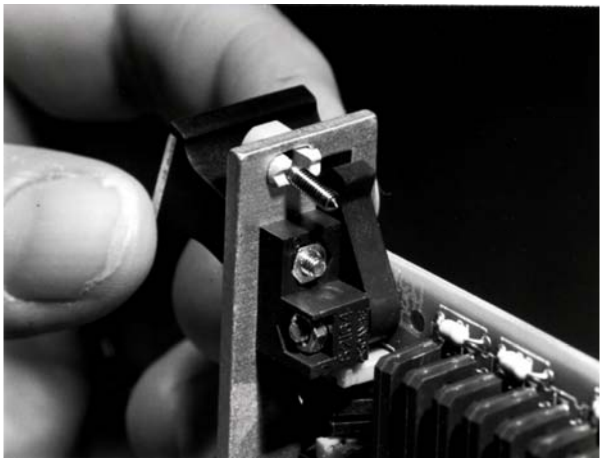


Figure 7-11 Ejector handle

Photo courtesy of Motorola Computer Group.

7.3.3.3 Injector/Ejector Handle (††)

The VME64x specification allows the use of injector/ejector handles. These are detailed in the IEEE 1101.10 specification. They have both an *injector* function, which pushes the module into the rack, and an *extractor* function, which pulls the board out of the rack.

Figure 7-8 shows an injector/extractor locking handle, and Figure 7-12 shows the new subrack rails that are required to make them work. The main benefit of the injector feature is to overcome the increased insertion forces when using the 160 pin P1/J1 and P2/J2, and 95 pin P0/J0 connectors.

The new horizontal rails are required at the top and bottom of the subrack. They allow the handles to grab the subrack, and pull the board in. The new rail has rectangular holes, which are shown at the lower left hand corner of Figure 7-12.

Although the new front panels will fit into an older VMEbus chassis, the injector feature will not work. To make it work, new horizontal top and bottom rails must be installed into the old subrack. These can generally be purchased at low cost, and installed in just a few minutes.

The locking feature of the handle also eliminates the need for screws at the top and bottom of the front panel. Traditionally, these screws prevent the bus module from vibrating out of the rack. The screws are very annoying because they must be jiggled when inserting a board into the subrack.

The injector/ejector handles usually have a different shape than the fixed or ejector handles. This detracts from the overall appearance of the system.

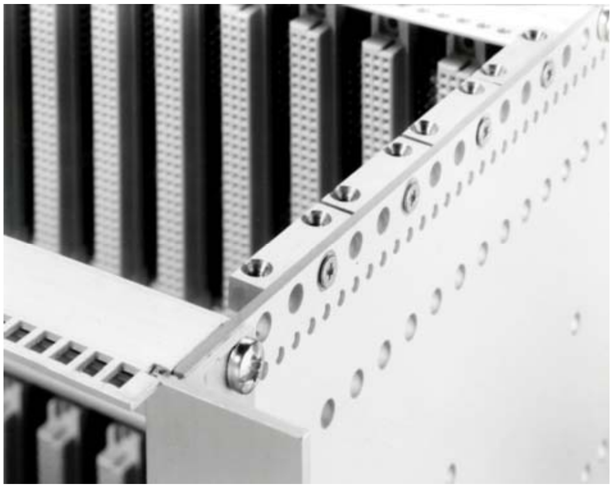


Figure 7-12 Subrack horizontal rail

(shown at the lower left hand corner) for the injector/extractor front panel must have rectangular holes. The rails are located at the top and bottom of the chassis.

Photo courtesy of Rittal Corporation.

7.3.4 Handle Location

The VMEbus specification allows handles to be placed in the locations shown in Figure 7-13. Most manufacturers of single height modules place one handle at the bottom as in Figure 7-13(b). Double height modules usually use two handles as shown in Figure 7-13(i).

7.3.5 Board Keying (††)

When the P2/J2 or P0/J0 user defined pins are used, it is often necessary to match a particular card to a particular slot in the card cage. That's because I/O cables and rear transition boards are often connected to a particular slot. The VME64x standard offers a card keying scheme that insures that every board is plugged into its correct slot. The details of this scheme are outlined in the IEEE 1101.10 standard.

For example, the tenth slot in a system may be a CPU card with its P2/J2 user defined I/O pins connected to a network cable. The eleventh card slot could be a motion control card, with its I/O pins connected to a power supply and a motor. Obviously, if these two boards were swapped, the system won’t work right. Even worse, if the motor controller has some high voltage power pins, it could destroy the CPU board.

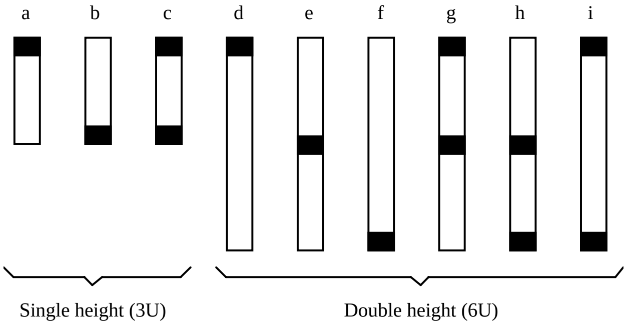


Figure 7-13 Front panel handle locations

A variety of placement options are available, but type (b) is most popular for single height modules, and type (i) for double height.

VME64x offers a card keying mechanism to prevents boards from being placed into the wrong slots. As shown in Figure 7-14, each card guide in the subrack has three square keying holes and one round hole. The square holes accept ‘L’ shaped keying pins. The round hole accepts the alignment pin from the front panel. As shown in Figure 7-15, each ‘L’ shaped keying pin can be inserted into each square key hole in one of four directions.

The keying pins are matched up with identical pins located on the VME64x front panel. The keys are designed so that a board cannot be engaged into the backplane unless the correct key combination is present.

When a VME64x keyed module is purchased, a set of loose keys is delivered with the board. These keys can then be configured on the bus module and subrack by the user.

The number of unique keying combinations in this scheme is 4 x 4 x 4 x 4 x 4 x 4 = 4,096. In applications where both the top and bottom have identical keys, then 4 x 4 x 4 = 64 unique keying combinations are possible. Installing identical key combinations in both the top and the bottom positions is the more robust method, as the keys are sometimes broken during rough board handling.

A *no key* hole allows a family of boards to be plugged into specific slots. For example, a system may reserve four slots for use with identical CPU boards.

The front panel alignment pin insures that each key lines up perfectly during board insertion.

The same card keying scheme can also be used on rear I/O transition modules. This verifies that each transition module is plugged into the correct slot.

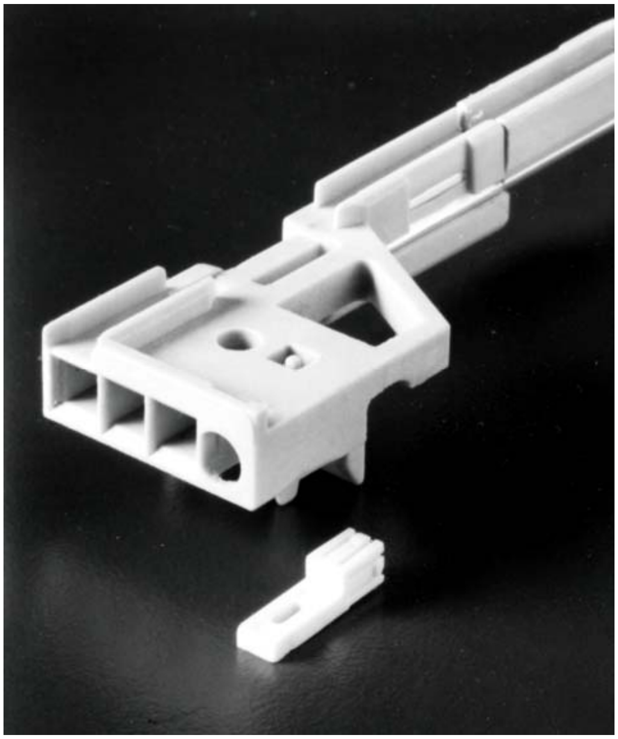


Figure 7-14 Card guide with key

Photo courtesy of Rittal Corporation.

7.4 ESD Strip (††)

There are two features in the VME64x specification that help with electrostatic discharge (ESD) problems. These include:

- Better grounding of the front panels.
- ESD strip.

Electrostatic discharge is familiar to most people. For example, in dry climates you often get a small spark that jumps from your finger when you touch a metal object (such as a door knob). This is known as electrostatic discharge, and it can be surprisingly powerful. For example, if you can see a spark jump from your finger it has a potential of at least 2,000 volts. ESD discharges from the human body can easily reach 10,000 volts, and have been measured as high as 40,000 volts.

Generally, ESD discharges off the human body are less severe than those produced from a metallic object. That's because the rise time of the discharge is slower from the human body, and is less destructive.

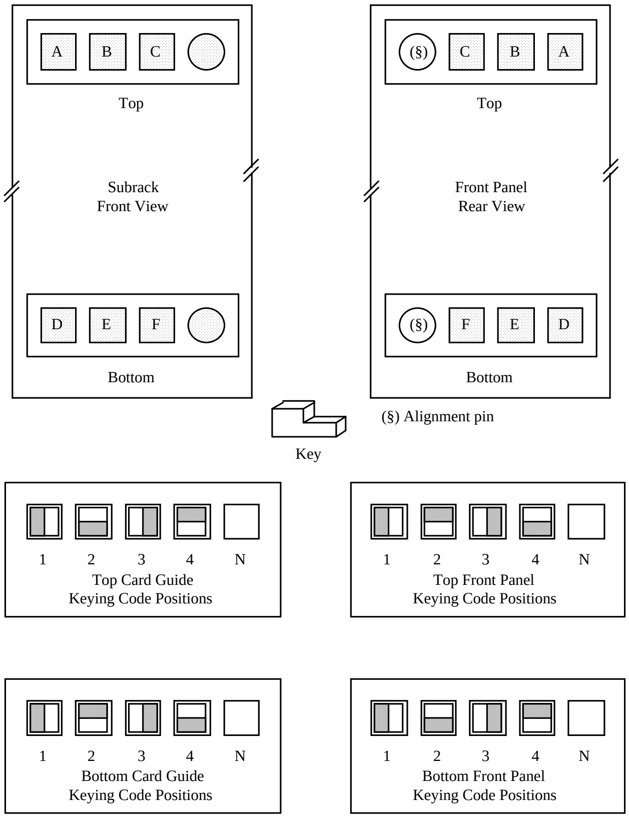


Figure 7-15 Keying methodology

If this high voltage energy reaches a sensitive component on the PC board it can disrupt or destroy the system. This has been a problem in the past, as the front panels have not been grounded well. For example, if a VMEbus front panel is not grounded to the subrack, then touching the front panel with your finger could cause the ESD spark to jump from the front panel to the circuit board, thereby disrupting or destroying the board.

The VME64x front panels prevent this problem by providing a reliable, low resistance ground path to the subrack.

In some applications it is desirable to bleed any static charge from the board while it is inserted into the subrack. One application for this is in the telecom industry, where live-insertion techniques are used. The VME64x specification supports a method for discharging the board as it is inserted into the subrack. This is done with a discharge strip and card guide contacts, which are located at the top and/or bottom edges of the board as shown in Figure 7-16.

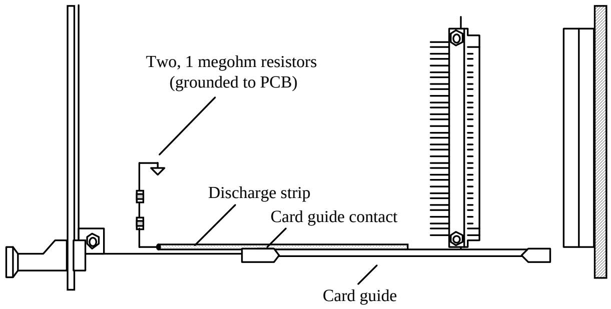


Figure 7-16 ESD discharge strip

The discharge strip is nothing more than a plated area on the component side of the printed circuit board. It is connected to the PCB ground layer through two, 1 megohm resistors (in series). These resistors limit the rate at which the board can discharge. When inserting a VMEbus module, the discharge strip is connected to the subrack ground through a card guide contact. The card guide contact, which is shown in Figure 7-17, is connected to the subrack ground.

Two single, 1 megohm resistors are used to prevent arcing of the discharge resistors. The contact areas of small surface mount resistors are very close, and can arc over if the voltages get too high. Using two resistors reduces this problem.

The electrostatic discharge happens very fast...usually in less than a few hundred milliseconds. That means that the card is fully discharged by the time the connectors engage the backplane.

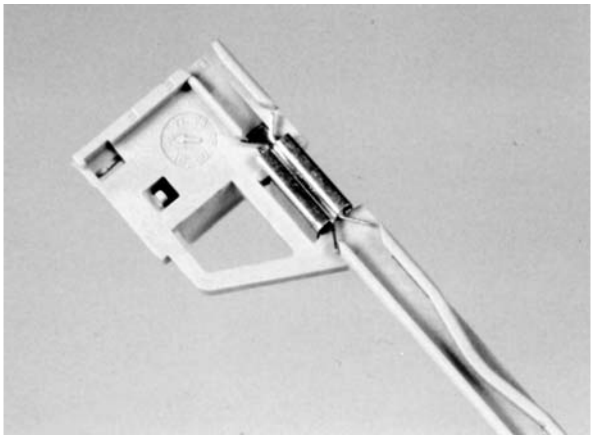


Figure 7-17 Card guide contact

Photo by Wade Peterson.

Once the bus module is fully inserted into the rack, the card guide contact disconnects from the discharge strip. This eliminates any potential ground loops that might form between the logic and frame grounds.

7.5 Subrack

VMEbus modules are generally placed into subracks like those shown in Figures 7-18 and 7-19. These are available in a wide variety of configurations and options. Subracks can then be placed into enclosures like the ones shown in Figures 7-20, 7-21 and 7-22. This modular approach makes the design, production and service of the system much easier. Most of the packaging components are available as off-the-shelf parts. This is usually cheaper than building a custom rack.

7.6 Conduction Cooled Eurocards

Most VMEbus modules require some airflow for cooling. They are dependent upon either convection or forced air currents. However, a special class of bus modules called *conduction cooled Eurocards* dissipate heat by thermal conduction.

The mechanical standard for conduction cooled Eurocards is the IEEE 1101.2-1992. The standard only describes 6U bus modules. The boards themselves are called *conduction cooled assemblies* or *CCAs*, and are used almost exclusively in military and aerospace applications. However, there are some commercial applications such as railway control systems. Ideal application environments include:

- Very tight chassis space (e.g. Humvee or Bradley fighting vehicle).
- Operation in a near or complete vacuum (e.g. aerospace).
- Zero gravity (e.g. space flight).

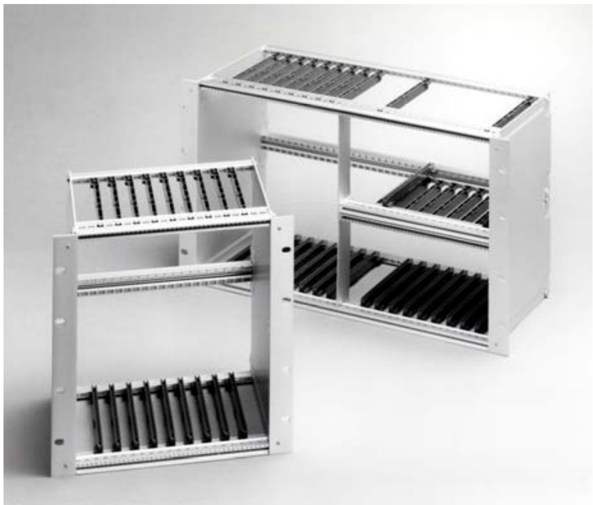


Figure 7-18 Eurocard subracks

Photo courtesy of VERO Electronics, Inc.

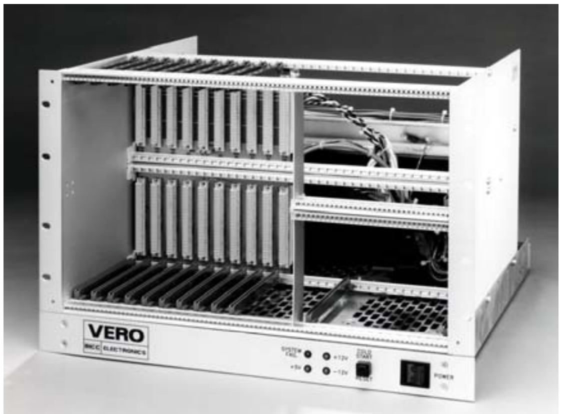


Figure 7-19 10-slot, 6U subrack with J1/J2 backplane, lower fan tray and power supply

Dual 3U racks are available for mounting disk drives, tape drives, power supplies, connector panels and other options. Photo courtesy of VERO Electronics, Inc.

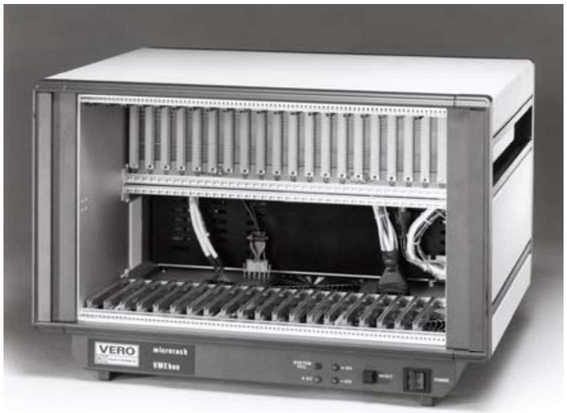


Figure 7-20 Desk top enclosure with 20-slot / 6U subrack, J1 backplane and power supply

Photo courtesy of VERO Electronics, Inc.

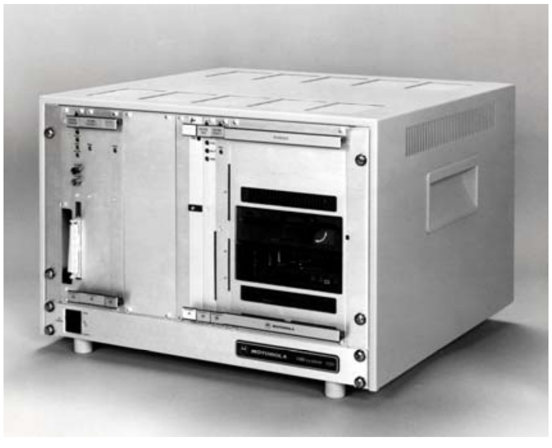


Figure 7-21 Desk top enclosure with 6U subrack, bus modules and disk drives

Photo courtesy Motorola Computer Group.

Figure 7-22 Free standing floor enclosure with 19" subracks

Mounting VMEbus subracks in free standing enclosures allows for modular systems. This makes designing, production and servicing of systems much simpler. Photo courtesy of Schroff Inc.

The IEEE 1101.2-1992 standard is a collection of general guidelines, and has very few rules. One rule is that conduction cooled assemblies must be capable of operating in convection cooled racks. However, the opposite is not necessarily true, and convection cooled cards may not fit well into conduction cooled racks. [They will fit into the chassis, but the boards may 'flop' around].

The IEEE 1101.2-1992 standard has guidelines for the following environmental factors:

- Thermal management
- Humidity
- Size
- Weight
- Shock & vibration

Figure 7-23 shows the key features of a conduction cooled assembly, and Figure 7-24 shows a conduction cooled CPU module.

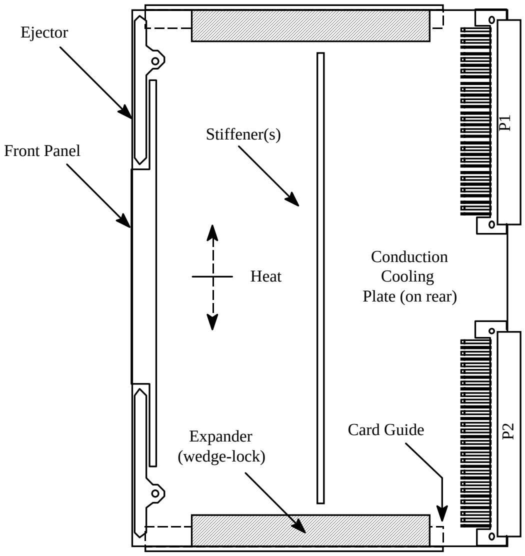


Figure 7-23 Key features of conduction cooled assemblies

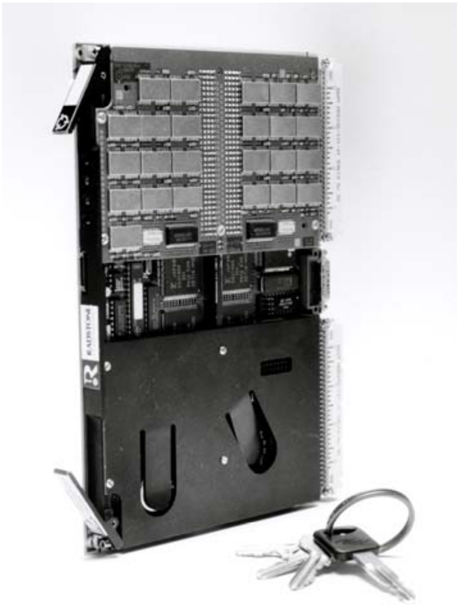


Figure 7-24 Conduction cooled CPU module

Photo by Wade Peterson.

The conduction cooled modules must be placed into a conduction cooled enclosure. Figures 7-25 and 7-26 show ATR and vetronics racks. These have special card guides that thermally bond the VMEbus module to the outer wall of the enclosure. The heat can be dissipated through the outer wall of the enclosure to an underlying chill plate. Alternatively, heat can be carried away from the enclosure by a liquid coolant circulating within the chassis wall.

Heat can be conducted away from printed circuit board components with a conduction cooling plate. This can be mounted on either side of the board. The plate is often thermally bonded to higher powered components.

Conduction cooling plates add significant cost to the VMEbus module. However, they are optional, and are not necessary if sufficient heat can be carried away by the printed circuit board itself, or by board stiffeners.

Card guide expanders (wedge-locks) carry heat from the VMEbus module to the chassis wall. As shown in Figures 7-27 and 7-28, the expander resides in the card guide region of the bus module. After the board is installed into the conduction cooled rack, a small hex screw is tightened, which forces the expander against the chassis wall. This provides a good thermal bond between the board and the chassis.

Figure 7-29 shows a cut away view of a honeycomb enclosure wall. The honeycomb wall allows liquid coolant to be circulated through the chassis, thereby carrying the heat away from the chassis.

As shown in Figure 7-30, the card guide expander is designed so that the edge of the printed circuit board engages the card guides in both the convection and conduction cooled chassis. This is an important feature of conduction cooled modules, as it allows them to be calibrated, tested and debugged using a low cost convection cooled chassis.

Up to three stiffeners can be used on conduction cooled modules. These increase the rigidity of the board. The board's front panel also acts like a stiffener.

The front panel of the conduction cooled module is much narrower than the convection cooled board. That's because conduction cooled cards can be located on either 0.65" or 0.8" centers.

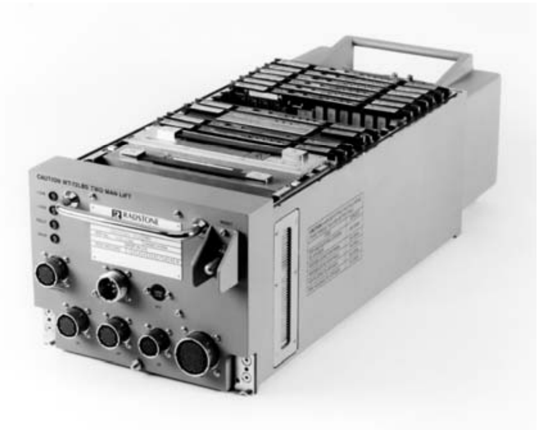


Figure 7-25 Conduction cooled ATR (Air Transport Rack) with boards

At the top rear of this enclosure you can see two nipples which are used for liquid coolant connections.

Photo courtesy of Radstone Technology PLC.

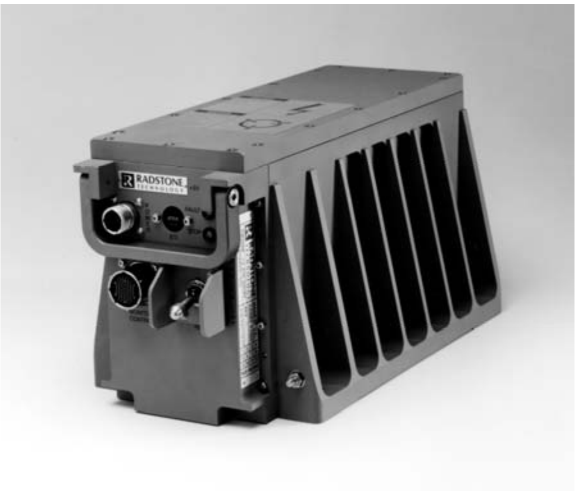


Figure 7-26 Conduction cooled vetronics (VEhicle elecTRONICS) rack

Photo courtesy of Radstone Technology PLC.

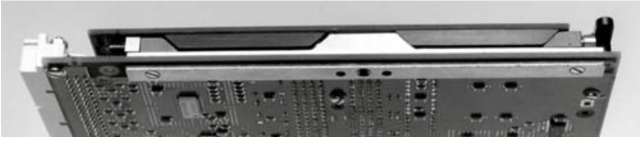


Figure 7-27 Card guide expander (wedge-lock) on a conduction cooled VMEbus module

A hex screw at the right side of this photograph is used to expand and retract the expander.

Photo by Wade Peterson.

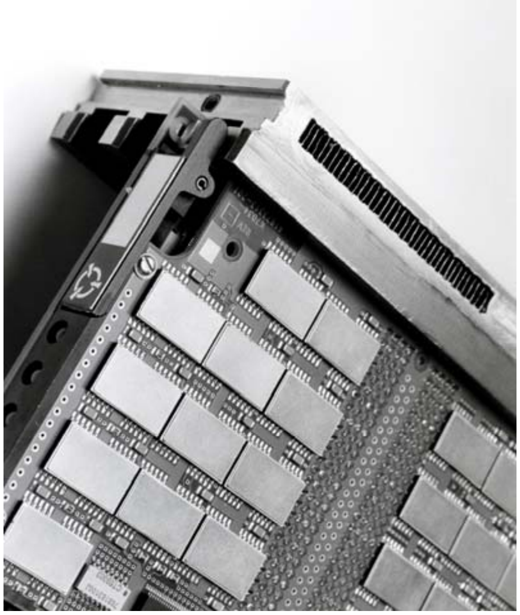


Figure 7-28 Card guide expander (wedge-lock) engaged in a cut-away section of a conduction cooled rack

Photo by Wade Peterson.

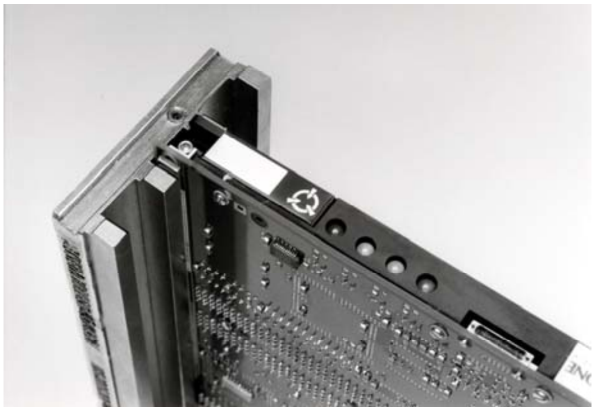


Figure 7-29 Cut away view of a honeycomb chassis wall

Liquid coolant is circulated through the wall, thereby cooling the bus modules.

Photo by Wade Peterson.

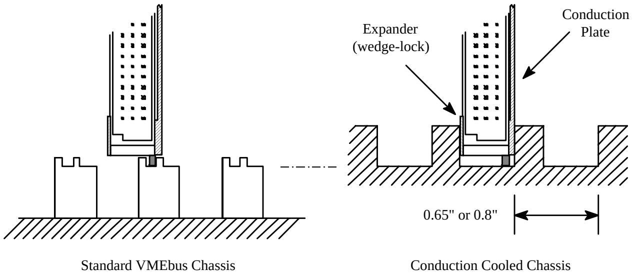


Figure 7-30 Conduction cooled modules can be installed in either convection or conduction cooled racks

7.7 References

Conduction Cooled PCI Mezzanine Card (CCPMC) / VITA 20-199x (Draft 1.3) VITA, Scottsdale, AZ 1997

Czeschka, Franz. Shielding Device for Electric Plug Connectors US Patent No. 4,959,024 1990

IEEE Standard for Mechanical Core Specifications for Microcomputers Using IEC 603-2 Connectors / IEEE Std 1101.1-1991 IEEE, New York, NY 1992.

IEEE Standard for Mechanical Core Specifications for Conduction-Cooled Eurocards / IEEE Std 1101.2-1992 IEEE, New York, NY 1992

IEEE Standard for Additional Mechanical Specifications for Microcomputers Using the IEEE Std 1101.1-1991

Equipment Practice / IEEE Std 1101.10-1996 IEEE, New York, NY 1997

IEEE Standard for Mechanical Rear Plug-in Units Specifications for Microcomputers Using the IEEE 1101.1 and the IEEE 1101.10 Equipment Practice / IEEE Std P1101.11-1996 (draft 3.1) IEEE, Piscataway, NJ 1997

Johnson, Lennart B. et al. Backplane Connector US Patent No. 4,655,518 1987

Johnson, Lennart B. et al. Backplane Connector US Patent No. 4,869,677 1989

Kielstra, Peter J. et al. Electronic Shelf Keying and Alignment Combination US Patent No. 5,402,320 1995

Korn, Heidi. "Military VME: A Design Study" *Proceedings of the Buscon/88 East Conference* CMC Norwalk, CT October 1988

Licht, Harold J. et al. Field Wirable VME Compatible Edge Card Connector US Patent No. 5,163,854 1992

Mistry, Balwantrai. VMEbus Compatible Backplane and Shelf Arrangement. US Patent No. 5,488,541 1996

Northey, William A. et al. Connector Assembly. US Patent No. 5,403,196 1995

Ott, Henry W. Noise Reduction Techniques In Electronic Systems Wiley, New York, NY 1976

Peterson, Wade D. "VME64: Taking The VMEbus Architecture Into The 21st Century" *VMEbus Systems* August 1996

Peterson, Wade D. "VME64x: The VME64 Extensions Standard", *VMEbus Systems* August 1997

Patterson, Doug. "Board and System Issues In Militarized and Mil-spec Applications" *Proceedings of the Buscon/88 East Conference* CMC Norwalk, CT October 1988, pp 353-357.

Storrow, Alan J. et al. Rack Mounted Circuit Board US Patent No. 4,879,634 1989

Storrow, Alan J. et al. Rack Mounted Circuit Board US Patent No. 5,010,444 1991

Van Doren, Thomas P. Grounding and Shielding Electronic Instrumentation University of Missouri - Rolla 1988

VMEbus Pin Assignments For Military Format-E Form Factor Boards and Backplanes / VITA 18-199x (Draft 1.2). VITA, Scottsdale, AZ 1997

VMEbus Specification / ANSI/IEEE STD1014-1987. VITA, Scottsdale, AZ 1987

VME64 Specification / ANSI/VITA 1-1994 VITA, Scottsdale, AZ 1995

VME64 9U x 400mm Format Draft Standard / VITA 1.3-199x (Draft 0.3) VITA, Scottsdale, AZ 1995

VME64x Specification / VITA 1.1-1997 (Draft 2.0) VITA, Scottsdale, AZ 1997

Waltz, Eike. "On the Mechanical Reliability Of VME Electronics Packaging" *VMEbus Systems* Summer 1985

Chapter 8
System Integration

Integration of a VMEbus system involves many design choices. These include selecting bus modules, choosing a backplane, determining the size, features and specifications of the power supply, deciding what chassis components to use, estimating cooling capacity and determining where and how to route I/O. Each of these factors is driven by the application, operating environment, software and a multitude of other issues. Since no two applications are alike, each integration is unique. The flexibility of the Eurocard packaging system, together with the large numbers of available components, can make integration a painful or painless experience. The purpose of this chapter is to present the user with system integration options and techniques.

8.1 Backplanes

There are three basic styles of VMEbus backplanes. These are called the J1, the J2 and the combination J1/J2 backplanes. Single height systems need only a J1 backplane. Double height systems can use a J1 and a J2 backplane, or a combination J1/J2 backplane. All three backplanes are available with between one and twenty one slots. Examples of J1, J2 and combination J1/J2 backplanes are shown in Figures 8-1, 8-2 and 8-3.

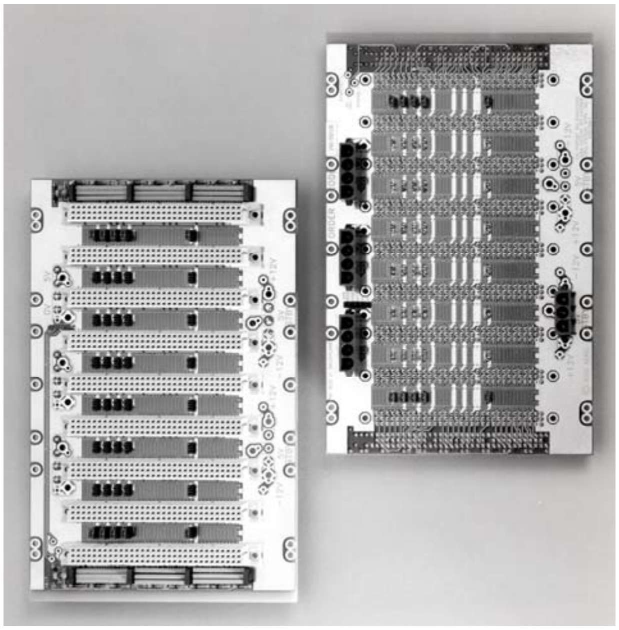


Figure 8-1 Front and rear of a J1 backplane with on-board passive terminators and power connectors

Photo courtesy of VERO Electronics, LTD.

Single slot backplanes may seem superfluous because they do not accept more than one bus module. However, modern VMEbus CPU modules can be very powerful, and the system may require only one card slot.

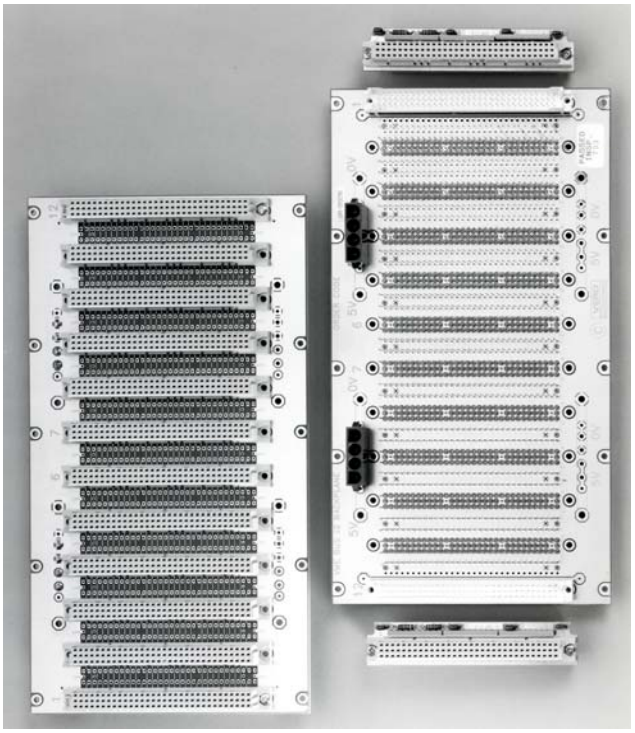


Figure 8-2 Front and rear of a J2 high speed backplane with off-board passive terminators and power connectors

The term *high speed* refers to the two rows of ground pins that straddle rows a and c of the connector. Photo courtesy of VERO Electronics, LTD.

Most VMEbus backplanes use *press-fit* connectors. These are generally considered to be more reliable than solder-tail connectors, as they allow more flexure in the backplane. Solder connections can crack more easily when the backplane is under shock and vibration, thereby causing the system to fail.

Also note that the J1 and J2 backplane connectors are not keyed, and a single height module can accidentally be inserted into a J2 backplane. This can ruin a single height module if damaging voltages are present on the user defined I/O pins. For example, if +12 VDC is present on the P2 I/O pins and a single height card were installed, it could blow out some ICs on the module. It's a good idea to either (a) label single height modules and J2 backplanes to inform users of this potential problem or (b) incorporate the VME64x keying mechanism.

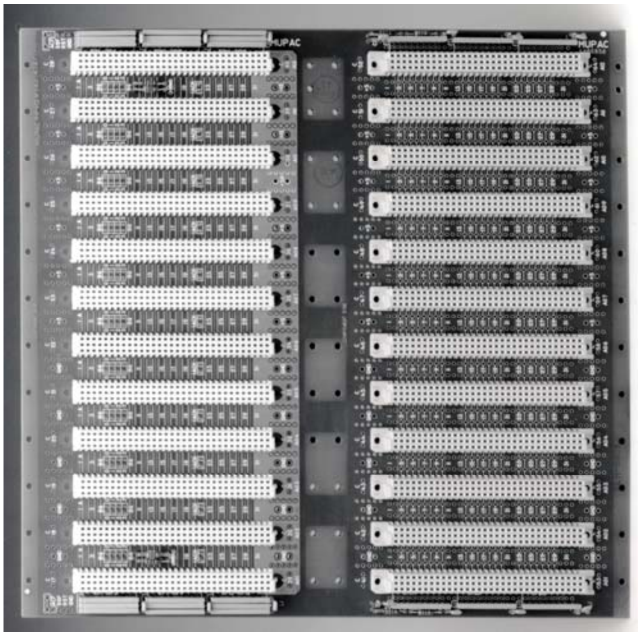


Figure 8-3 J1/J2 combination backplane with standard DIN 41612 connectors

Photo by Rob Bigelow.

8.1.1 VME64x Backplanes (††)

Backplanes must have a minimum set of features in order to be VME64x compliant. For example, a *3U backplane* must have:

- 160 pin J1 connectors.
- All assigned ground pins must be connected to the ground plane.
- All geographical address pins must be connected.
- All VME64 and VME64x defined signals must be bussed and terminated.
- Power connections for +5 V, +3.3 V, +/- 12 V, +/- V1/V2, VPC and +5V STDBY.

A *6U backplane* must also have:

- A combination J1/J2 backplane (to lower noise and power losses).
- If rear I/O is used, it must be designed in compliance with the IEEE 1101.11 rear I/O transition board standard.

Figures 8-4 and 8-5 show front and rear views of a seven slot VME64x backplane. This backplane also has optional J0 connectors. These are located between the J1 and J2 connectors.

A VME64x backplane must be used if the user intends to support +3.3 VDC power supplies, geographical addressing, VME64x live insertion, +/- V1/V2 or the backplane test & maintenance bus.

8.1.2 Backplane Terminators

Backplane terminators reduce signal reflections that are generated at the ends of the backplane. These reflections are caused by impedance mismatches. There are two types of termination networks: passive and active. A schematic diagram for passive networks is given in Chapter 6. They are made up of resistor networks which form a 194Ω termination at each end of a signal trace.

Bypass capacitors should be placed across passive terminators to reduce power supply noise. Some manufacturers also place Schottky diodes on AS*, DS0*, DS1* and SYSCLK* to improve impulse response.

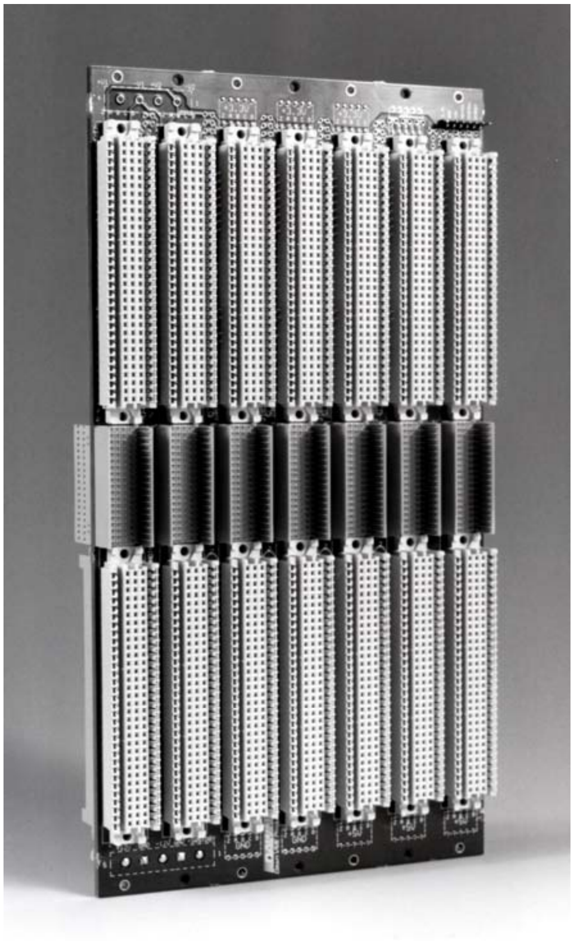


Figure 8-4 Front view of a seven slot VME64x backplane with J0 connector

Backplane courtesy of Dawn VME Products. Photo by Wade Peterson.

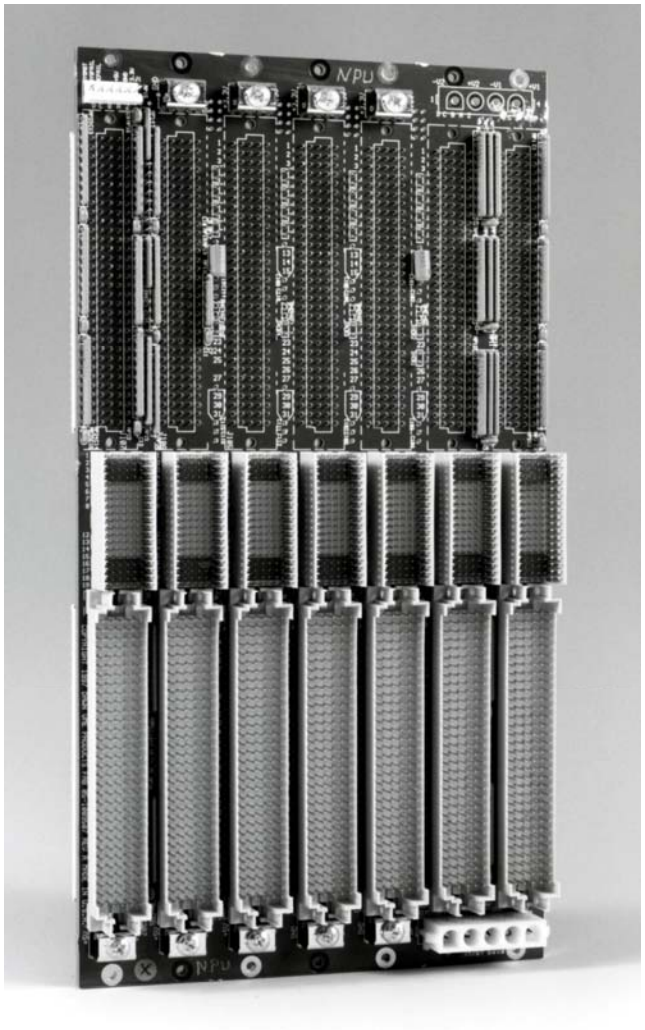


Figure 8-5 Rear view of a seven slot VME64x backplane with J0 connector

Backplane courtesy of Dawn VME Products. Photo by Wade Peterson.

As the name implies, *active terminators* use active components. They are usually formed from an amplifier with a low output impedance, coupled with a 194 Ω resistor in series with the signal trace. The main advantage of the active terminator is reduced power consumption. The active terminator can be designed with a very low quiescent current. Passive networks, however, require some biasing current. For example, a 6U, +5VDC *passive* backplane will draw about 1.3 amps from the system power supply. A 6U *active* backplane will draw less than 0.1 Amp.

Passive and active terminators are available in three configurations: on-board, off-board parallel and off-board right angle.

On-board terminators are the simplest. A passive on-board terminator is shown in Figure 8-6. The main advantage of this type is simplicity and low cost. The main disadvantage is they take up extra room at the ends of the backplane.

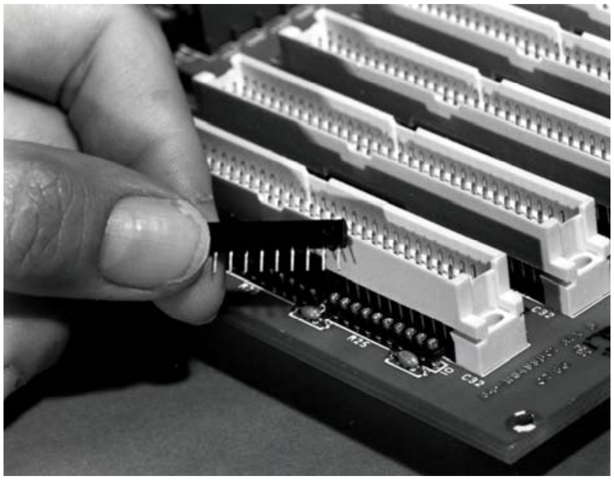


Figure 8-6 J2 passive, on-board / outboard terminator

Photo courtesy of Motorola Computer Group.

Passive on-board terminating backplanes can also be split into two types: *inboard* and *outboard* terminators. The *outboard terminator* is the most popular because the resistor packs reside at the outer edges of the backplane. However, one drawback of this technique is that resistor packs take up too much room at the end of the backplane. If they are placed onto a twenty-one slot backplane, they won't fit into a 19" subrack.

One solution to this problem is to use *inboard terminators*. Inboard terminators place the termination resistor packs between the two end slots of the backplane. For example, on a twenty-one slot backplane they are placed between slots one and two, and slots twenty and twenty-one. In this case each signal trace connects to the end connector pin, and then reverses direction back to the terminator.

Some VMEbus purists reject inboard terminators, citing small signal deformations that are caused when the signal reverses direction at the ends of the backplane. However, in the author's experience these deformations are so small as to be insignificant.

Some manufacturers use off-board terminators to solve the twenty-one slot backplane problem. Off-board terminators have two advantages: they allow twenty-one slot backplanes to fit within the 19" subrack and they enable a single backplane to be used with either passive or active terminators. They are available in parallel or right angle versions. The type used depends on the room available at the rear of the backplane. Figure 8-7 shows a J1 passive, off-board parallel terminator.

When evaluating or designing *off-board* terminators for the J2 backplane, make sure that connection points to the a and c rows are made available to the user. A correct J2 off-board terminator is shown in Figure 8-8. Without these connection points it would be impossible to connect an I/O cable to the two end slots of the backplane.

8.1.3 Power Connectors

Backplane power connectors usually come in one of four styles: FAST-ON, screw terminal, lug (bolt-on) terminal and quick-disconnect. When wiring the backplane verify that the power connectors are capable of handling sufficient current, especially on the +5 VDC and +3.3 VDC supplies. A good rule of thumb is that each double height VMEbus card slot can draw at least 7.2 amps at +5 VDC, and 12.0 amps at +3.3 VDC. That means that a 21 slot backplane could theoretically draw 151 amps at +5 VDC, and 252 amps at +3.3 VDC.

Longer backplanes will provide several power connection points. When evaluating or designing the wiring for VMEbus systems, verify that the power connections can handle the current requirements of the backplane. Overloaded connectors can cause voltage drops (and heat) which may cause the system to fail.

8.1.4 Power Monitor Connectors

Most commercial backplanes provide connection points for the ACFAIL* and SYSRESET* power monitor signals. The connectors for these are not standardized, and will vary from manufacturer to manufacturer. The most common types are FASTON terminals and wire-wrap posts. The power monitor connectors are optional, and are wired only when an external power monitor is used.

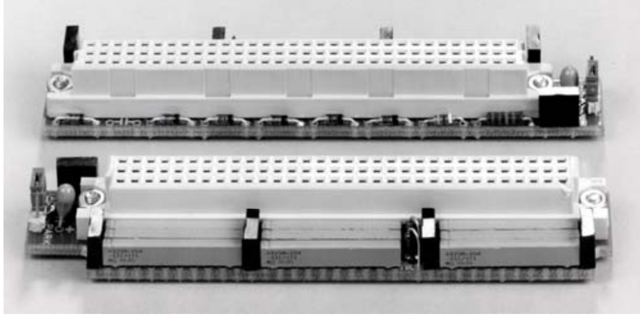


Figure 8-7 J1 passive, off-board parallel terminator

Photo courtesy of VERO Electronics, LTD.

8.1.5 Methods For Bypassing The Daisy-chains

The backplane must have some method for closing the bus grant and interrupt acknowledge daisy-chains in empty slots. There are several ways of doing this. Four common methods include daisy-chain jumpers, auto grant connectors, auto grant backplanes and bypass modules.

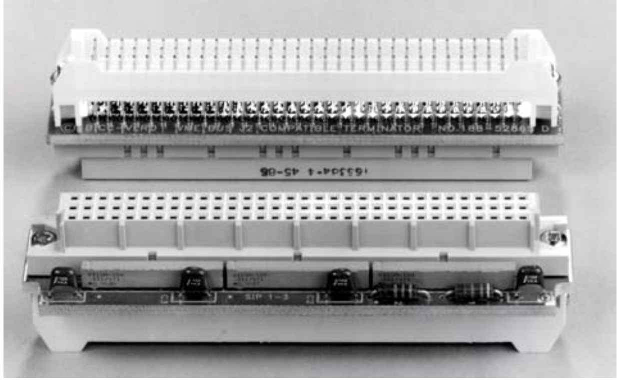


Figure 8-8 J2 passive, off-board terminator which provides a connector out the rear of the terminator

This allows connection to the user defined I/O pins in the two end slots.

Photo courtesy of VERO Electronics, LTD.

The *daisy-chain jumpers* are the simplest and least expensive way to bypass the daisy-chains. As shown in Figure 8-9, jumper pins are placed between the bus modules. These are usually located on both the front and the rear of the backplane. Shunts are installed on these pins whenever the card slot is empty, thereby bypassing the daisy-chains.

Backplanes with *auto grant connectors* are another method for closing the daisy-chains. An example of these connectors is shown in Chapter 7. It is compatible with standard DIN 41612 jack connectors, and provides an automatic switching mechanism for the bus grant and interrupt acknowledge daisy-chains. The auto grant connector has a unique mechanical switch inside the insulator cavity of a standard DIN 41612 connector. It is mounted to the backplane in the same manner as the 96 pin DIN. When a VMEbus module is installed, the switches on the bus grant and interrupt acknowledge daisy-chains open, thereby routing the daisy-chain signals through the card. When the module is removed, the switches close and pass the daisy-chain signals on the backplane.

The *auto grant backplane* is another way to bypass the daisy-chains. As shown in Figure 8-10, the auto grant backplane has active components located on the rear of the backplane. When a bus module is installed, it is sensed by the backplane and the daisy-chains are opened. When the module is removed, active logic bypasses the daisy-chains.

Another type of auto grant backplane uses a small external circuit board, which is installed on the rear of the backplane as in Figure 8-11. The advantage of this technique is that the auto grant hardware can be repaired without removing the backplane. Two disadvantages include cost, and the module's susceptibility to shock and vibration.

A fourth method for handling the daisy-chains is with a *bypass module*. As shown in Figure 8-12, the bypass modules are simply 'dummy' boards with a front panel. They have no active components on them, but they do close the daisy-chains. When a bus module is removed from the backplane the user replaces it with a bypass module. The advantage of this method is that it forces the user to cover up the empty hole in the front panel, and prevents air from escaping out of the chassis. The disadvantages of this method are that the user must have a quantity of bypass modules on hand, and is rather expensive.

8.1.6 User Defined I/O Pins (††)

The outer two rows of the DIN 41612 connector are used as user defined I/O pins. These pins are installed so that they run directly through to the rear of the backplane, and are not bussed. They are commonly called the *P2/J2 user defined pins*, and are usually used for I/O signals.

Under the VME64x standard user defined I/O pins are also available on the P0/J0 connector, as well as the 'z' and 'd' rows of the 160 pin connector. In fact, the VME64x standard boosted the number of user defined rear I/O pins from 64 to 205...a 141 pin (320%) increase. These were added because of the greater demands which are now placed on VMEbus systems. This is especially true in military and telecom applications.

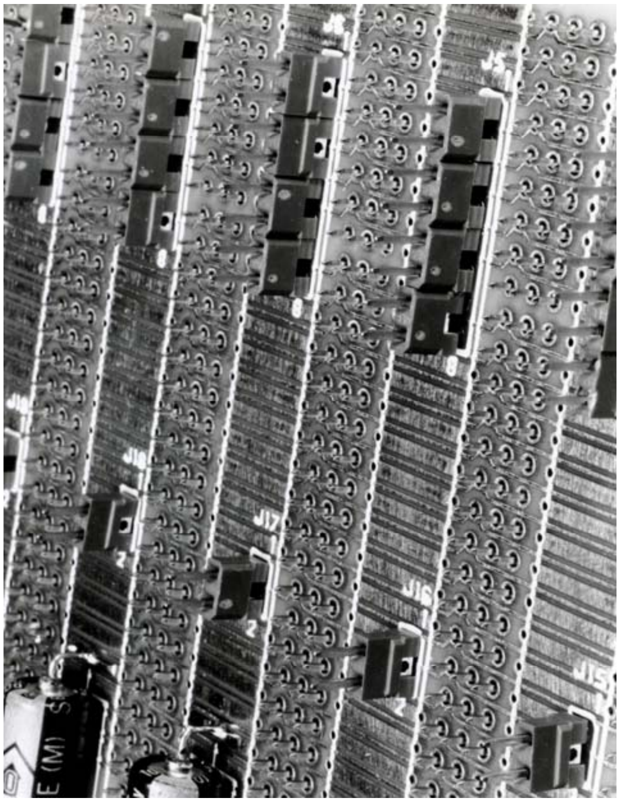


Figure 8-9 Backplane daisy-chain jumpers

Photo courtesy of Motorola Computer Group.

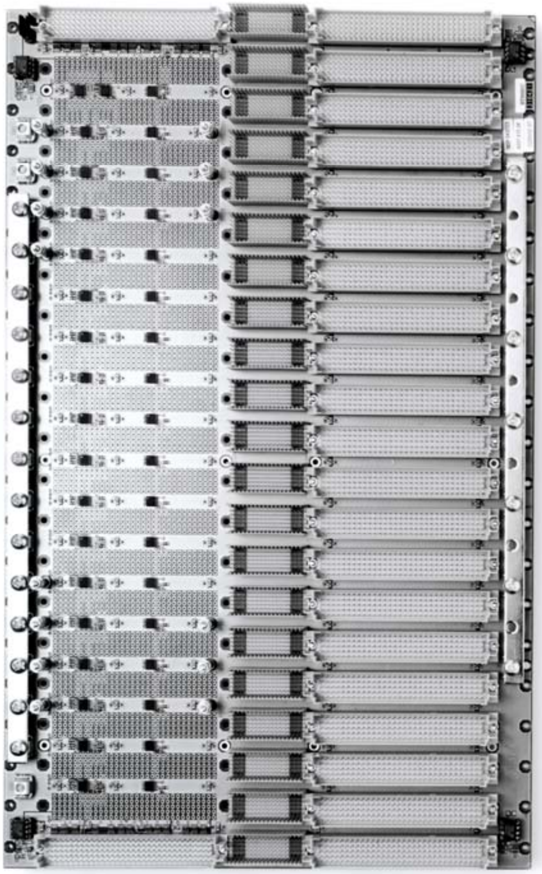


Figure 8-10 6U VME64x auto grant backplane (rear view)

Photo courtesy of VERO Electronics, LTD.

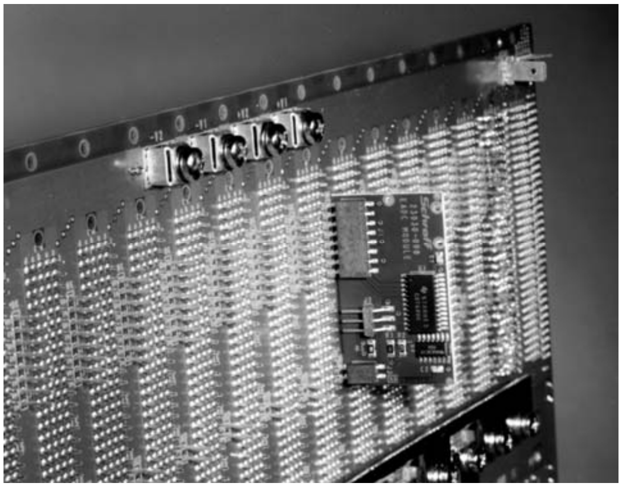


Figure 8-11 Auto grant circuit module

Photo courtesy of Schroff Inc.

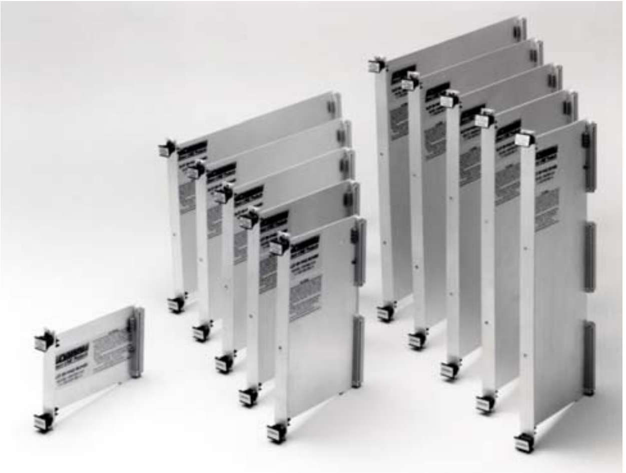


Figure 8-12 Bypass modules

Photo courtesy of Dawn VME Products.

For example, in many applications the use of front panel cables is frowned upon because:

- There is not very much room for connectors on front panels.
- Front panel cables are often susceptible to shock and vibration problems.
- Bus modules are harder to replace when cables are attached to the front panel.
- Systems with rear I/O look better.

I/O from any of these connectors can be routed:

- Out the rear of a bus module (when a J2 backplane is not used).
- Out the rear of the backplane using ribbon cable connectors.
- Onto a rear transition module.
- Bussed across the backplane.

Figure 8-13 shows how the P2 user defined I/O pins can be cabled directly from the P2 connector to a peripheral, such as a transition module. That photo shows a 64 pin insulation displacement connector and ribbon cable. When ribbon cables are used, only the two outer rows of pins on the DIN 41612 are connected. The center row of pins is not connected.

Figure 8-14 shows a similar arrangement, except this time a combination J1/J2 backplane with DIN 41612 connectors is used. In this example the backplane routes the outer two rows of the J2 connector directly to the rear of the panel. Connector shrouds, as shown in Figure 8-15, provide a mating surface for the connector, and protect the pins from damage.

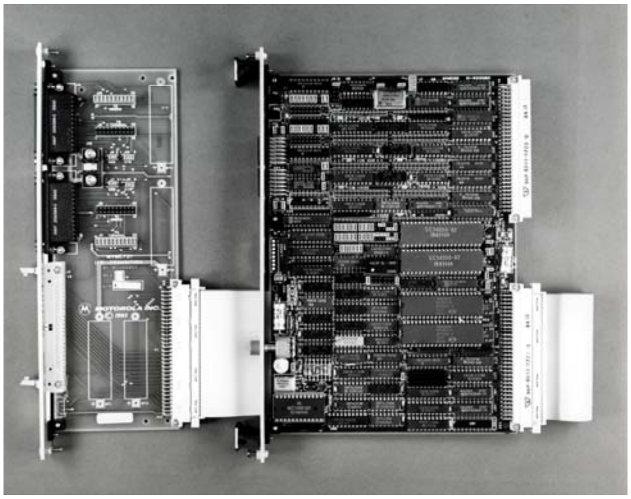


Figure 8-13 Direct cable connection to the P2 user defined pins

Photo courtesy of Motorola Computer Group.

It should also be noted that the rear connectors and shrouds are not necessarily standard on VMEbus backplanes. Some users prefer to mount the backplane up against a panel surface, which does not leave any room for I/O pins. Users are advised to specify these features when ordering a backplane.

8.1.7 J2 High Speed Backplane

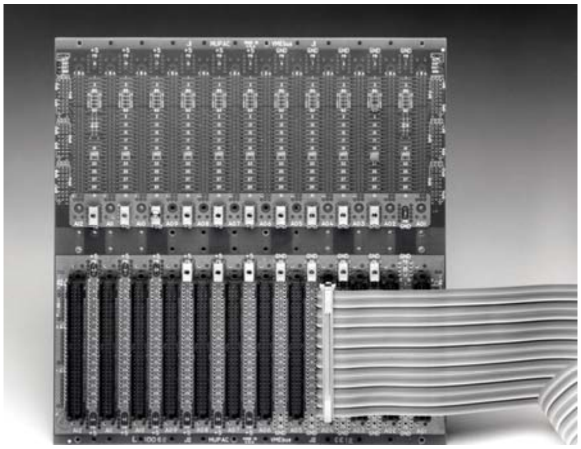


Figure 8-14 64 pin ribbon cable connection on the rear of the J2 backplane permits simple I/O cabling

Only rows a and c of the connector are attached to the cable. Row b, which is used by VMEbus, is not connected to the cable. Photo by Rob Bigelow.

A J2 *high speed* backplane is shown in Figure 8-2. That backplane has two rows of pins straddling rows a and c of the DIN 41612 connector. These are ground pins. The most common use of the high speed J2 backplane is to ground alternate conductors of ribbon cables attached to the user defined I/O pins. One or two 64 pin ribbon cables can be connected. One connector attaches to row a and to one row of ground pins, the other attaches to row c and the other row of ground pins. This grounds every other conductor on the ribbon cable. Grounding a cable in this manner reduces cross-talk between the conductors, reduces EMI and cuts cable inductance considerably.

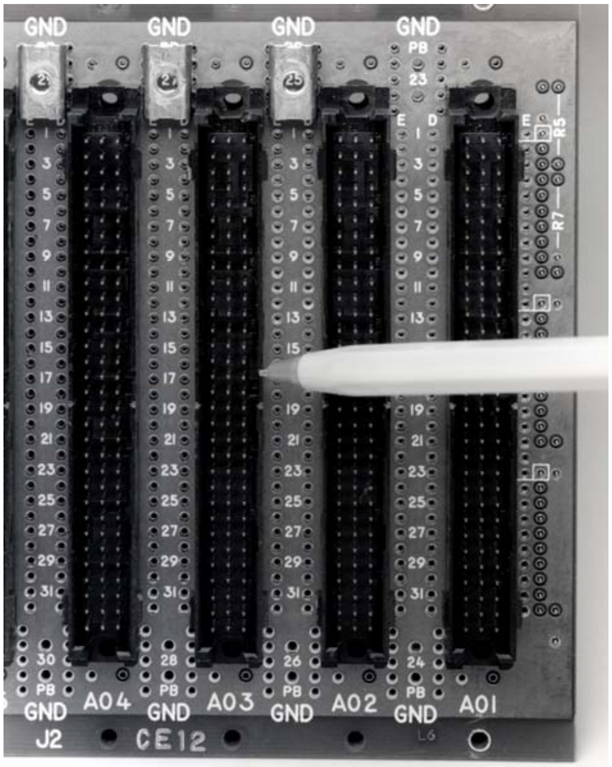


Figure 8-15 Rear connector shroud

Photo by Rob Bigelow.

Another use for the high speed J2 backplane is to provide extra signal paths. In many applications the 64 user defined I/O pins are not enough. The high speed J2 backplane allows the user to allocate all of the 64 I/O pins to signals. Grounds are all provided by the separate ground pins.

8.1.8 Rear I/O Transition Modules (††)

A popular new feature on VME64x backplanes is rear I/O transition module support. These are compatible with a standardized rear I/O transition module. They are a popular way to route cables out the rear of the VMEbus chassis.

As shown in Figure 8-16, the rear I/O transition modules plug directly into the P0/J0 and P2/J2 user defined pins from the rear of the backplane. They are popular for routing cables out of the back side of a VMEbus chassis. This eliminates a big cable mess in many systems.

The rear I/O transition modules are detailed in the IEEE 1101.11 specification. Before this standard, many manufacturers used transition modules, but most had incompatible features. For example, VMEbus backplanes didn't always provide clearances for these boards, nor did the boards themselves have standard depths

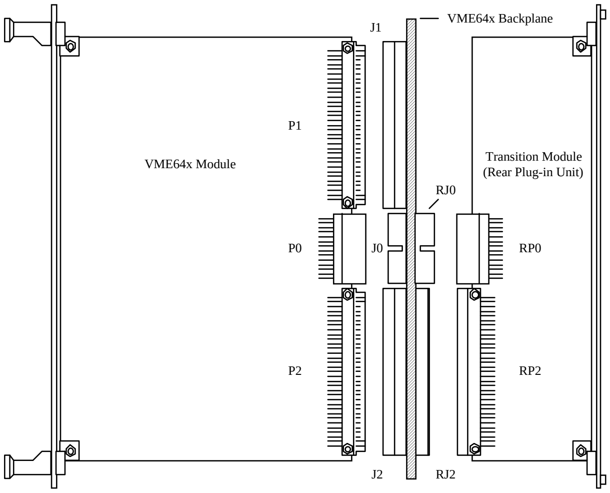


Figure 8-16 Rear transition module

Another reason for the rear transition module is to overcome the lack of rear mating connectors to the newer VMEbus 160 pin connector.

Very often the rear I/O transition module is a passive board that simply routes cable connections to the rear of the chassis. However, these modules can also have active components on them. In these cases, the transition module can obtain +5 VDC power and ground from the center row of the RJ2 connector.

The VME64x specification also named the various connectors on the rear of the backplane. For example, the 'RJ2' and 'RP2' connectors are located on the rear of the backplane. The same is true for the P0/J0 connector.

8.1.9 VME320 Backplane (†††)

In 1997 a modified VMEbus architecture called VME320 was developed by Bustronic™ Corporation and Arizona Digital, Inc. The VME320 architecture was designed for operation at over 320 Mbyte/second, and peak bandwidths of up to 500 Mbyte/second. At the time this handbook was printed, the VME320 architecture was experimental.

VME320 uses a new backplane design and bus protocol. It is a proprietary architecture, and is jointly owned by Bustronic™ and Arizona Digital. These companies have filed for patent protection in the United States. They have promised, however, that the backplane technology will be available on a royalty basis.

The VME320 backplane uses a 'star' interconnection method to speed up the VMEbus backplane itself. As shown in Figure 8-17, all of the interconnections are connected together at slot 11 on a 21 slot backplane. In a nutshell, the idea behind this backplane is that the leading edges of signals, propagating from slot-to-slot, will effectively pass through only one slot on the way to their destination. This allows tighter skew delays on the backplane, thereby speeding up the system.

A special protocol is needed to operate at the very highest bandwidths. At the time this handbook was printed, no protocol had yet been defined.

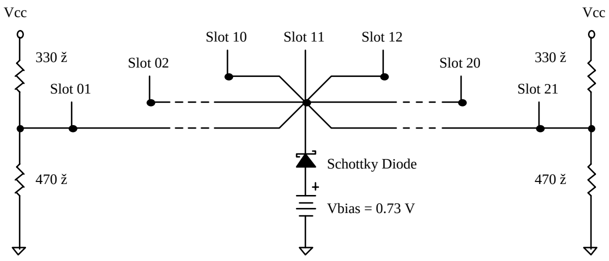


Figure 8-17 VME320 backplane uses a star interconnection method to speed up signal transitions across the VMEbus backplane

8.1.10 Installation And Adjustment Of Backplanes

When installing backplanes into a card cage, place a VMEbus module at each end of the backplane(s) before tightening the screws. This perfectly aligns the backplane in the sub-rack. Systems where the bus modules are hard to install may need adjustment. If the system is full of VMEbus modules, this usually means loosening and re-tightening the backplane with a screwdriver. This can be done with all of the cards installed in the system.

8.2 The System Power Supply

Every VMEbus system must have at least one power supply. The same supply is usually used for both the bus modules and the peripherals. There are two popular types of power supplies: chassis mounted and rack mounted.

An example of a chassis mounted power supply is shown in Figure 8-18. These are available from scores of vendors, and come in a wide variety of configurations. The photo also shows a popular way to mount the supply. It is attached to an aluminum rail behind the J1 backplane. The rail is designed so that it can swing up and away for servicing.

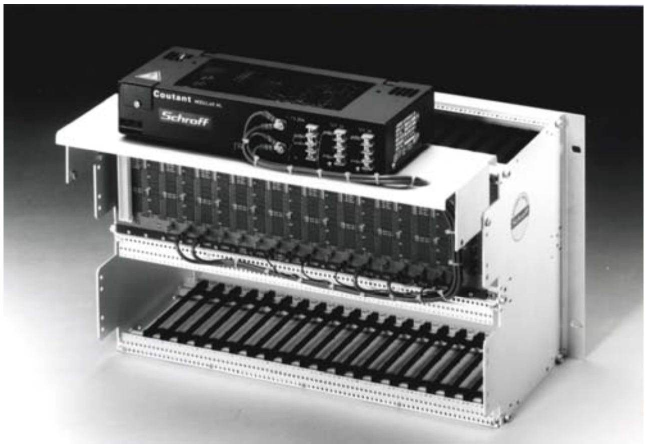


Figure 8-18 Open frame power supply

Photo courtesy of Schroff Inc.

One advantage of the chassis mounted supply is cost. The open frame power supply business is very competitive, and costs are quite reasonable. Also, as we'll also see shortly, the chassis mounted supplies do not require any VMEbus card slots. This allows a full 21 slot backplane to be used. One disadvantage of the chassis mounted supply is serviceability, as they can be quite difficult to replace in the field.

Another popular power supply is the Eurocard plug-in module. As shown in Figure 8-19, these slide into a VMEbus chassis just like a bus module. However, they do not plug into standard backplane connectors, as those connector pins cannot carry sufficient current. Instead, the plug-in units usually use connectors with spade terminals.

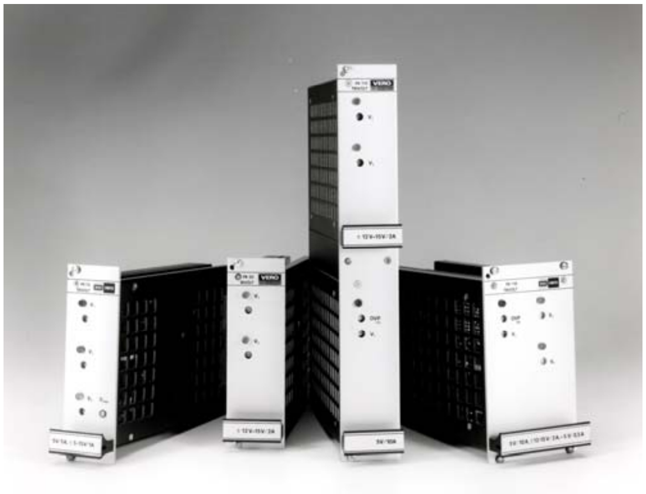


Figure 8-19 Eurocard plug-in supplies

Photo courtesy of VERO Electronics, LTD.

The wide variety of power supplies can make selection a bewildering task. When evaluating power supplies choose the features and options that you need, and then look at the power specifications. Some common features and options include:

- Input voltage selection (i.e. automatic or manual)
- Output voltages (+5 VDC, +3.3 VDC, etc.)
- Switch mode or linear power conversion
- Mounting configuration
- Special indicators (power on/off, voltage indicators, etc.)
- EMI filters
- Remote sensing
- Overload and over voltage protection
- Power fail detection
- Remote shut down
- CE compliance

Some of the important power specifications to look at are:

- Maximum output current rating
- Input voltage requirements
- Line regulation
- Load regulation
- Output ripple
- Transient response
- Temperature derating

8.2.1 Linear And Switching Supplies

Power supplies fall into two broad categories: linear and switch mode regulation. A linear supply first steps AC line voltage down using a transformer. The low voltage AC is then rectified, filtered and regulated. The main advantage to linear supplies is their simplicity and low noise output. Their main disadvantages are low efficiency and large size. Linear supplies usually cost more than switching supplies in larger wattages.

A switching supply directly rectifies and filters the AC input to high voltage DC. This DC voltage is then converted to high frequency AC (> 20 KHz) using a pulse-width-modulated regulator, and then stepped down to low voltage AC with a transformer. The output of this transformer is rectified and filtered. The main advantages of switching supplies are low cost per watt, availability in large output power ranges and high efficiency. Their main disadvantage is slightly higher output noise levels than linear supplies. In all-digital systems the switching supplies will usually work well. In high performance analog systems they may be too noisy. Evaluate the functions of your system before selecting the type of supply.

Switching supplies are smaller and more efficient because they use higher frequencies than linear types. A linear supply will transform a 50 or 60 Hz signal, but switching supplies operate in the 20 KHz - 1+ MHz range. The efficiency is the amount of power loss for every watt delivered to the load. Besides lower power costs, the higher efficiency supply will require less system cooling such as fans and air conditioners. They are also more compact because they can use much smaller transformers, capacitors and inductors.

Many switching supplies require a minimum load for proper regulation. A common minimum load is 10% of the rated load. For example, a +5 VDC supply rated at 50 amps could require a minimum load of about 5 amps. If less than 5 amps are drawn, the supply could behave abnormally. If you intend running your system over large load conditions, verify that the minimum load will always be present. Also remember that passive P1 backplane terminators draw about one amp.

8.2.2 Remote Sensing

One popular feature on many power supplies is a remote sense input. Remote sensing allows the power supply to regulate its output at the load, rather than at the output terminals of the supply. This enhances its load regulation by eliminating the voltage drops across supply leads. It is most commonly used by supplies with large loads (usually tens or hundreds of amps) or supplies which must be located a long distance from the VMEbus chassis.

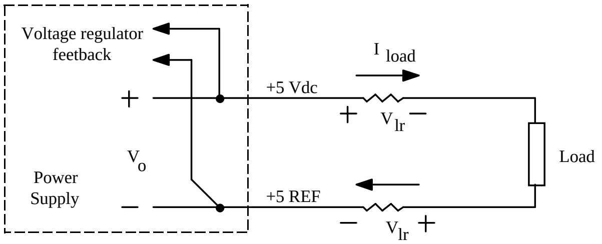


Figure 8-20 Power supplies with internal sense regulate their output voltage at the output terminals of the power supply. Current through the power supply leads cause voltage drops Vlr caused by the resistance in the wire.

To illustrate remote sensing, consider the schematic diagram of Figure 8-20. That diagram shows what happens when a power supply without remote sensing is connected to a load. Each of the wires connected to the load has some resistance. The voltage across each wire (V_{lr}) increases in proportion to the current delivered to the load (I_{load}). This results in a lower load voltage. For example, a power supply lead with resistance of 0.001 ohm will experience a voltage drop of 0.01 volt at 10 amps, but 0.1 volt at 100 amps (a fully loaded

VMEbus backplane can draw more than 151 amps at +5 VDC). As load current fluctuates, so will the voltage across the load.

The voltage drop across the leads can be reduced by using a power supply with a remote sense input as shown in Figure 8-21. That diagram shows how the supply senses the output voltage at the load and compensates for the resistance of the leads.

When wiring the VMEbus backplane, connect the sense wires near the center of the current load. This prevents the bus modules nearest to the sense point from receiving too high a voltage, and those farthest from the sense point receiving too low. The center of current load will vary from system to system. In a fully loaded backplane this will be in the center of the backplane. Some power supply manufacturers also require that a bypass capacitor (0.01 µF - 10 µF) be located at the sense lead connection point.

8.2.3 Overload And Over Voltage Protection

Most power supplies incorporate some sort of over voltage and over current protection. Over voltage protection shuts the power supply off if it accidentally attempts to drive its output voltage too high. This protection mechanism is usually independent of the supply's voltage regulator. If the regulator fails in an unprotected supply, it could raise the output voltage on +5 VDC to tens or hundreds of volts. This could destroy every VMEbus module and peripheral in the system. The added cost of over-voltage protection is usually worthwhile when compared to replacing a chassis full of boards.

Over current protection turns off the supply if it attempts to drive too much current. It shuts down if the user accidentally shorts a power supply, if power is inadvertently wired backwards or if a component fails in the system. This prevents damage to the power supply and the rest of the system.

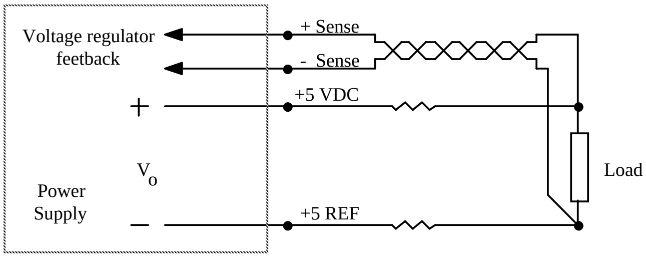


Figure 8-21 Power supplies with remote sense regulate their output voltage at the load.

The resistance of the power leads
prevent voltage drops at the system.

Use caution when selecting power supplies in systems with large capacity disk drives. These units can draw high current loads on +12 VDC or +24 VDC when spinning up, often two to three times the normal operating load. Most disk drive manufacturers will specify their current loading during and after spin-up.

8.2.4 AC Power Fail Detection

Some power supplies include AC power fail detection (refer to the section on power monitors in Chapter 5). Usually they have a TTL output which activates shortly after AC power is lost, but before their output voltage drops. VMEbus supports power fail detection using the ACFAIL* and SYSRESET* signals in conjunction with the power monitor functional module. Power fail sensing is not required by the VMEbus specification, however.

When selecting power supplies with AC power fail detection, verify that they conform to the VMEbus Power Monitor functional module timing and ACFAIL* interface specifications. Unless they are designed specifically for use with VMEbus, they probably will not conform. However, if they do not conform they can usually be made to work with some additional interface ICs.

8.2.5 Remote Shut-down

Some power supplies provide a remote shut-down feature. Usually, this is a TTL input which turns the supply on and off. This feature can be used to sequence power supplies in several chassis during power-up and power-down, to provide remote on/off switches and to facilitate software shut-down.

Software shut-down is desirable in some systems because it forces the user to execute a shut-down command from a terminal. This notifies the disk operating system that housekeeping tasks such as flushing disk caches or parking disk drive heads should be performed before the power goes down. Once this is done, the operating system then shuts itself off.

8.2.6 Power Supply Specifications

Power supply specifications should be evaluated before a supply is chosen for a VMEbus system. All power supplies are not alike, and some are totally inappropriate for VMEbus systems. Input voltage, load regulation, transient response, output ripple and temperature factors are but a few of the things that should be evaluated.

8.2.6.1 AC Line Regulation

The characteristics of the AC line voltage will affect a power supply's output. The most basic characteristic is line voltage. Table 8-1 shows the line voltage and frequency in a few selected countries. A complete list is given in an excellent pamphlet entitled "Electric Current Abroad, 1984 Edition" which is available from the US Department of Commerce, Bureau of Industrial Economics.

When selecting a power supply, verify that it will operate at the local line voltage and frequency where the system is to be installed. Many supplies include a switch to double the input voltage of the supply. While this helps in some countries, it does not provide for a universal voltage match. For example, in the United States a power supply manufacturer may rate their supply at 120 / 240 VAC, +/- 10%, 50 / 60 Hz. If the intended destination of the system is Japan, where the line is 100 / 200 VAC, +/- 10%, 50 / 60 Hz, it may not work. The voltage in Japan (which can drop to 90 VAC) may not be high enough to run the power supply. This may cause the supply's regulation to degrade, the efficiency of the supply to drop or a total failure altogether.

Besides the power supply, other AC equipment in the VMEbus system may also need to be matched to the local line voltage. Cooling fans and air conditioners are often overlooked when evaluating line voltage requirements. Even though they can accept a wide range of supply voltage, some equipment may not work correctly at both 50 Hz and 60 Hz. In Japan, for example, one side of a factory may be at 50 Hz, while the other end may be at 60 Hz. Mobile equipment would need to work at both ends of the building.

One way to match the input voltage is to place a step-up or step-down transformer between the AC input and the power supply. These transformers are available in a wide variety of voltages, frequencies and wattages. Some power conditioning equipment may also supply this feature.

Local conditions cause line voltages to vary over time. The factory environment is especially susceptible to these kinds of problems. Electric motors, welders and a wide variety of other equipment can cause problems. Line regulation is a specification given by power supply manufacturers, and measures the ability of the supply to maintain its output over variances in line voltages. Power utilities can only specify a voltage range at their utility pole step-down transformer. IR (resistive) voltage drops may cause wide fluctuations in voltage at the destination power equipment. Many power utilities in the United States rate voltage at 120 VAC +/- 5%, but even these tolerances are not always met, especially during times of peak load or bad weather.

Line regulation is defined as the change in DC output voltage over a given change in AC line voltage. It is normally specified as the +/- change from the nominal DC output voltage. This specification should be analyzed in terms of the VMEbus voltage tolerances (see Chapter 6).

For example, consider a factory environment where the AC line voltage may vary between 108 - 132 VAC (120 VAC +/- 10%). A power supply is selected with rated line regulation of +/- 0.1%. This means that the nominal +5 VDC output can vary +/- 0.1% away from this value or 5 VDC X 0.1% = +/- 0.005 VDC. The output voltage will vary between 4.995 and 5.005 VDC. While this is well within the VMEbus specification of 4.875 - 5.250 VDC, the line regulation must be added to the effects of load regulation, voltage drops in the power supply output leads, noise and other factors. When all of these are taken together, the power supply could drift out of tolerance.

Table 8-1 AC utility service in selected countries

Country	Line Frequency (Hz)	# Phases	Nominal Voltage	# Wires
Australia §	50	1,3	240 / 415	2,3,4
Belgium §	50	1,3	220 / 380	2,3,4
Canada	60	1,3	120 / 240	3,4
Denmark	50	1,3	220 / 380	2,3,4
England	50	1	240 / 480	2,3
	50	3	240 / 415	4
Finland	50	1,3	220 / 380	2,4
France §	50	1,3	220 / 380	2,4
Fed. Rep. Germany	50	1,3	220 / 380	2,4
Israel	50	1,3	230 / 400	2,4
Italy §	50	1,3	220 / 380	2,4
Japan	50 / 60	1,3	100 / 200	2,4
Mexico	60	1,3	127 / 220	2,3,4
Netherlands §	50	1,3	220 / 380	2,4
Norway	50	1,3	230	2,3
Scotland	50	1,3	240 / 415	2,4
Sweden	50	1,3	220 / 380	2,3,4
Switzerland	50	1,3	220 / 380	2,3,4
United States	60	1,3	120 / 240	2,3,4

Source: Electric Current Abroad, 1984 Edition, US Department of Commerce
(§) Denotes that some local power utilities will vary from given specifications.

Many VMEbus systems use some type of AC power line conditioning to protect their input power. There is a wide range of available products for power conditioning. These include simple transient noise suppressers, EMI / RFI filters, isolation transformers and uninterruptible power systems.

8.2.6.2 Load Regulation

The load regulation of a power supply is the change in DC voltage resulting from a change in load. It is usually specified as a change in output voltage during a load change between the minimum and maximum output currents. The load regulation specification is usually rated at lower frequencies. It shows the output voltage change when another VMEbus module is added to the system, but will not show how the supply reacts to high speed load changes (high speed response is rated by the transient response of the supply). Examples of high speed load changes include dynamic RAM refresh and disk drive 'head seek' operations.

8.2.6.3 Transient Response

The transient response of a power supply is an indication of how fast its regulator changes in response to a high speed load change. It is often specified as the amount of time it takes for the supply to compensate for the load change.

Most power supplies are too slow to react to the high frequency noise changes in VMEbus systems. These load changes often occur over several nanoseconds, whereas power supplies often react in hundreds of microseconds. This is one reason each VMEbus module must use local bypass capacitors to supply short bursts of energy to ICs.

8.2.6.4 Output Ripple

Output ripple is the same as output noise. It is usually specified as a peak-to-peak or RMS voltage over some frequency range. Ripple has several sources, some of them being line voltage feed-through, power line noise or noise generated by the power supply itself. For example, a switching power supply will have noise components at 50/60 Hz and 20+ KHz. These are from AC line power and switching noise, respectively.

The output ripple specifies the magnitude of the noise components. When selecting a supply, compare the ripple specification against the maximum allowable VMEbus noise (see Chapter 6).

8.2.6.5 Temperature Derating

Typically, power supplies are derated with temperature. As ambient temperature goes up, the components of the supply are less able to dissipate heat. When selecting a supply, determine the maximum temperature at which the system operates before selecting the wattage. Most manufacturers will rate a supply at room temperature. If higher temperatures are needed, you may need to derate the output power.

8.3 Wiring VMEbus Systems

Figure 8-22 shows typical wiring for a VMEbus system. Power enters the system through a power cord which has a neutral, hot and safety ground connection. This is typical of systems in the United States, Canada and some other countries. The hot wire (sometimes called the load wire) delivers 120 VAC with respect to neutral. The US National Electrical Code requires that neutral and safety ground be connected only at the power service entrance (usually a fuse box).

If the VMEbus system is to be shipped to a member country of the European Economic Union, then it will probably need to be certified under the CE low voltage power directive.

Safety ground reduces injuries or death caused by electrocution. The main idea is that short circuit and leakage currents can't create hazardous voltages on areas that humans touch. In a system without a safety ground, a short circuit could cause electricity to travel through a human body if it touches the chassis and an earth grounded floor. The safety ground prevents this by providing a low resistance path to ground. In addition, all circuits usually create some leakage current which may find its way to the chassis. This leakage current can also generate hazardous voltages.

Figure 8-22 shows the safety ground connected to the chassis (frame). This is often done near the point where power enters the system. All parts of the chassis must be connected to this point. Hinged doors, removable covers, fans, disk drive chassis and power supply frames should all be connected to this point with a length of green (or green/yellow) wire. Connect this wire to the chassis using a screw and star washer. Rivets are discouraged because they can electrically insulate themselves due to corrosion.

The European CE regulations have strict rules regarding how safety grounds may be connected together. Often, they require a terminal strip with separate terminals for each wire.

Neutral should never be connected to safety ground. Sometimes the safety ground prong is removed from the cord by the user, which makes it possible for hot and neutral to be reversed. This causes an unsafe condition since the chassis would be brought to line voltage.

The only time when safety ground should carry current is during a fault condition.

As Figure 8-22 shows, both the hot and neutral wires are fused and switched. In the advent of a failure the fuse should blow, preventing human injury and damage to the system.

Figure 8-22 shows a dual pole circuit breaker as the line protection element. These are designed so that both sides of the breaker open if either side draws too much current. This technique is usually required by the CE rules.

If the system only needs to be operated in the USA, Canada or Mexico, then a single circuit breaker or fuse could be used on the hot input lead. The neutral lead can remain unfused.

Two common types of fuses, called *fast blow* and *slow blow*, are available. A fast blow fuse senses the short term, instantaneous, current flow through the fuse. If the current exceeds its rating it will blow. A slow blow fuse monitors average current, and will tolerate brief over current conditions. In a VMEbus chassis the fuse should probably be a slow blow, since power supplies usually have a large inrush current shortly after they are turned on. Follow the power supply manufacturers recommendations when fusing the VMEbus system.

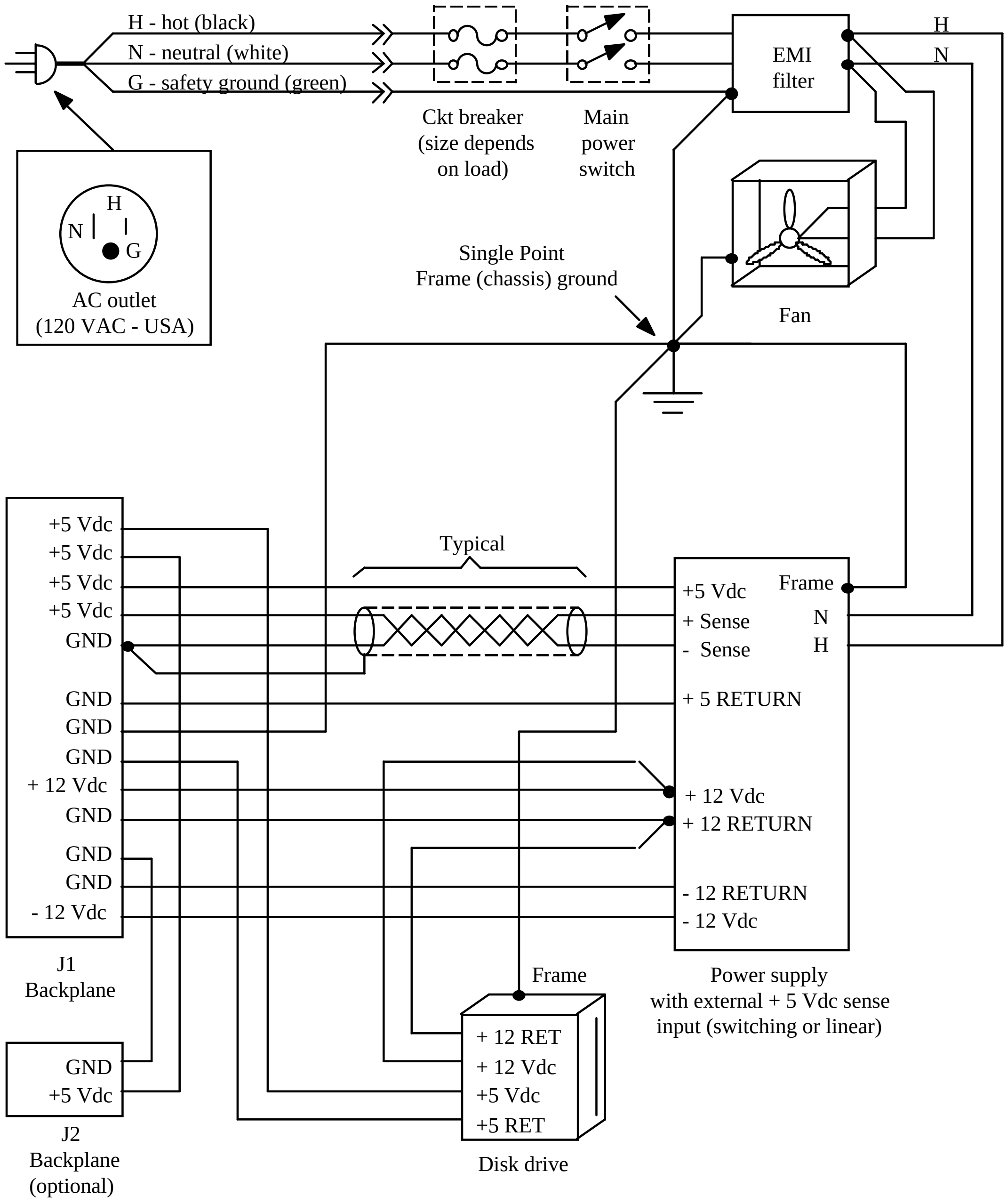


Figure 8-22 Typical VMEbus system wiring

A main power switch should be located after the fuse. Select the switch so that it exceeds the maximum expected voltage and current. Also, to be CE compliant the power switch should be a double-pole, single-throw type. However, in the USA, Canada and Mexico, a single-pole single-throw switch (on the hot lead) could be used.

Figure 8-22 shows an EMI / RFI filter which reduces the conducted emissions (i.e. noise) entering and leaving the system over the AC line. Power supplies sometimes have an internal filter. If the power supply does not have an EMI filter, one can be added.

Wiring the VMEbus chassis is not complicated on simple systems, but a few general rules should be followed. Figure 8-22 shows a supply with sense inputs which are used to remotely regulate the output voltage of the supply. In this example the power supply regulates the voltage at the J1 backplane rather than at the output terminals of the supply. This eliminates effects of voltage drop across the power supply leads running between it and the backplane. Here only the +5 VDC supply is remotely sensed. In many VMEbus systems +12 VDC and -12 VDC are not remotely sensed because they don't source much current.

External devices should tap power from the point where the voltage is sensed. Figure 8-22 shows an external disk drive which draws +5 VDC from the J1 backplane, and +12 VDC from the output of the supply. Disk drives use +5 VDC to power logic ICs, and +12 VDC drives motor.

A fully loaded VMEbus backplane can consume more than 151 amps at +5 VDC, and 252 amps at +3.3 VDC. +12 VDC and -12 VDC usually only require several amps.

A shielded, twisted pair should be used for the sense wires. This reduces magnetic and capacitive noise coupling from nearby wires. The shield should be connected at the backplane end only.

Twisting power supply leads together reduces magnetic coupling from stray magnetic sources, a prime source being the switching power supply. Also, minimize inductance by keeping the leads as short as possible. To reduce EMI generation a good rule of thumb is to keep the power leads less than 1/20th the wavelength of the highest frequency in the system.

When using two backplanes, power and ground for the J2 backplane should be tapped from the J1 backplane. One method, shown in Figure 8-22, uses wires running between the backplanes. Some manufacturers line up the power lugs on their J1 and J2 backplanes so that shorting bars can be used. A good rule of thumb is to use the shortest possible conductor between the two. This reduces noise voltages between the backplanes due to resistance and inductance of the conductor. Combination J1/J2 backplanes are also available, and operate at lower noise levels. Combination backplanes also require less wiring labor.

Figure 8-22 shows circuit GND connected to frame ground at the backplane. This was done because the VMEbus chassis is usually connected to some other system such as a terminal. Most power supplies isolate their input and output grounds, a requirement in more sophisticated systems. Without this conductor the VMEbus ground would be floating. Connecting this floating ground to, say, a terminal could cause a large ground current to flow between the VMEbus module and the external terminal. If the link were an RS-232C cable, large surge currents could cause a voltage differential on signal conductors because of inductance. The result is garbled data communication between the two. In larger systems with multiple VMEbus chassis and power supplies, circuit GND from all the supplies would typically be connected together at one point. This arrangement is called a star ground.

8.4 Temperature & Humidity

Temperature and humidity affect the performance and reliability of electronic systems. In situations where climate is well controlled, such as in air-conditioned office buildings, little attention is needed. However in severe environments, such as those found in industrial or military applications, climate control is an important factor.

8.4.1 Temperature

All VMEbus modules have minimum and maximum temperature ranges. These are usually specified in the manufacturer's data sheets as *operating temperature* and *storage temperature*. These should not be confused with *junction temperature*, which is the internal temperature of ICs.

VMEbus systems generate heat, and it is essential that temperatures inside the chassis don't get too hot or too cold. If they do, timing variations or component failure may cause the system to crash.

Thermal effects should be considered early in the system design. The first step is to know the ambient operating temperature of the system. In most cases this isn't known very well, and a survey is helpful. Temperature and humidity (chart) recorders are available for survey work.

A thermal analysis can be performed once the temperature extremes of the application are known. The goal is to estimate the maximum internal temperature of the system. Many books, computer programs and other tools are available for this task. However, most stress numerical techniques such as finite element analysis. However, practical experience (by the author) has shown that the fastest and most accurate analysis is done using the simple method shown in Figure 8-23.

The method of Figure 8-23 will not yield an exact maximum temperature, at least on paper. The method is very pragmatic: the maximum power of each system component is determined (from current consumption numbers), the air velocity at various points in the system is estimated with an air-flow sketch, a mock-up of the system is assembled and the temperature of critical components are measured. If hot spots are encountered the system is re-arranged. It's something of a brute force approach to thermal analysis, but it does yield accurate data in a relatively short amount of time.

Most VMEbus systems use convection or forced air cooling. In general, conduction cooling is more expensive than convection cooling. However, some applications require conduction cooling. For example, at high altitudes (greater than 10,000 ft) convection cooling becomes inefficient because of low air density. Conduction cooling should be considered for those applications.

The rest of this section will be devoted to convection and forced air cooling. Users interested in conduction cooling are directed to Chapter 7.

8.4.2 Estimating Power Dissipation

Most of the power delivered to a VMEbus module is eventually lost as heat. We could say that *all* of the power is lost as heat, but then again some of the power consumed by a module could be sent elsewhere over the user defined pins.

The average power dissipation from a bus module can be found by:

power (watts) = voltage (volts) x current (amps)

For example, if a VMEbus module consumes 3.00 A at 5.25 volts, then the power dissipation is: 5.25 volts x 3.0 amps = 15.8 watts.

The total power dissipated by an entire system can be quickly measured with an AC amp meter. By measuring the current delivered to the system, the total power dissipation can be found. For example, if 5.0 amps (RMS AC) is delivered to a system running at 110 VAC, then the total power dissipation is 110 VAC x 5.0 amps = 550 watts.

Power is additive. If two bus modules each consume 10 watts of power, then the combined power is 10 watts + 10 watts = 20 watts.

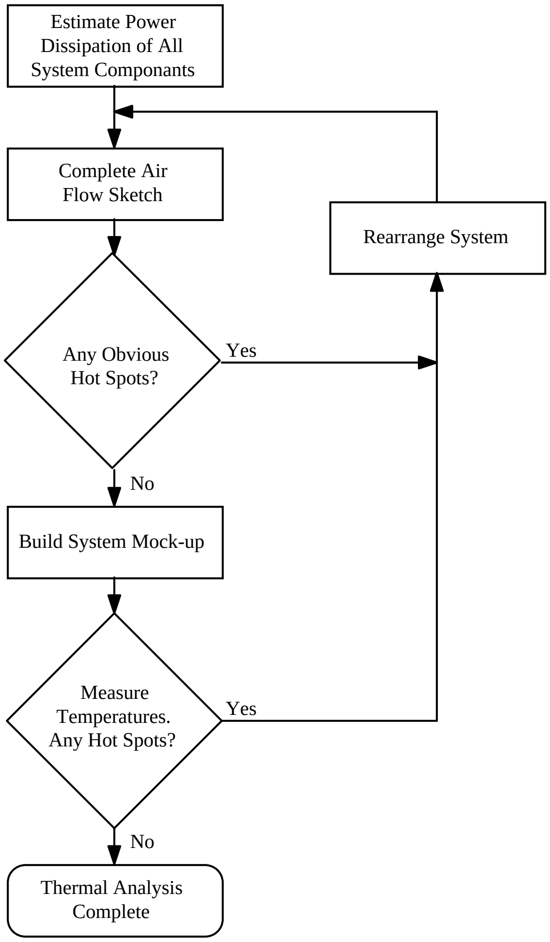

Estimate Power Dissipation of All System Componants
Complete Air Flow Sketch
Rearrange System
Any Obvious Hot Spots?
Yes
No
Build System Mock-up
Measure Temperatures. Any Hot Spots?
Yes
No
Thermal Analysis Complete

Figure 8-23 Procedure used for simple thermal analysis

Power dissipation goes up with voltage. Figure 8-24 shows the power consumption of a typical 6U VMEbus CPU when the supply voltage is changed from 4.5 VDC to 5.5 VDC. Power consumption at 5.25 volts is about 13% higher than at 4.875 volts.

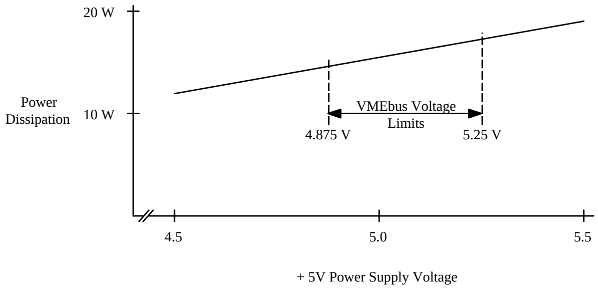


Figure 8-24 Power dissipation from a typical VMEbus module goes up with applied voltage

This graph was taken from a Motorola MVME 133-1 (6U) CPU module.

Use conservative, but reasonable, assumptions when estimating power loss in the system. For example, the maximum power consumption of a VMEbus module at 60° C is theoretically:

= pwr (+ 5V) + pwr (+ 12 V) + pwr (- 12 V) + pwr (+ 5V STDBY)

= (5.25V x 9.0 A) + (+ 12.6 VDC x 1.5 A) + (12.6 VDC x 1.5 A) + (5.25V x 1.5 A)

= 93 watts

While this is the worst case power dissipation, it's not reasonable because + 5 V STDBY and +/- 12 VDC rarely deliver full power in most systems. Furthermore, we haven't added in the +3.3 VDC or +/- V1/V2 supplies available in VME64x.

Use RMS (average) current rather than peak current for determining power dissipation for VMEbus modules or peripherals. For example, a hard disk drive consumes much more power right after the power is first turned on. That's because the drive requires more power to bring the disk platter up to full speed. If peak power were used for the thermal analysis, then much more cooling would be required. Unfortunately, there is no standard method of reporting power consumption among suppliers of VMEbus products. Some suppliers report peak current, and some average current.

Power supplies generate heat. Most power supply manufacturers will rate the efficiency of the supply. For example, if a switching power supply is 80% efficient, then for every watt delivered to the load the following power will be lost as heat:

Power dissipated by the supply = (1 - efficiency) x power delivered to the load

If, for example, the power supply efficiency is 80% and the power delivered to the load is 200 watts, then the power dissipated by the supply is: (100% - 80%) x 200 watts = 40 watts. As a general rule of thumb, switching power supplies are more efficient than linear supplies, at least at higher power levels.

8.4.3 Estimating Air Flow

The velocity of air at any point in the system can be estimated with an air-flow sketch. Figures 8-25 and 8-26 show some examples, and Table 8-2 gives rules of thumb for drawing them.

Air velocity is important because air carries away heat. Forced air convection can be as much as ten times more efficient than natural convection. The faster the air, the more heat that's removed. Very hot components should be located in areas with high air velocity.

Air temperature is important because air density is lower at high temperatures. That also means that the cooling ability of air is lower at high temperatures. Convection cooling rates should be derated at ambient temperatures above 50 ˚C (as compared to 25 ˚C). For highest efficiency, lower power components should be located upstream and the higher power components down stream (along the air flow path).

8.4.4 System Mock-up

The final step in the thermal analysis is to build a system mock-up and measure the temperatures at various points in the system. The mock-up is a thermal prototype of the system. In most cases the fastest mock-up can be made from off-the-shelf Eurocard packaging components. Prototypes of sheet metal parts can be quickly fabricated from cardboard and held in place with tape.

Once the mock-up is complete the temperature profile of the system can be taken. Figure 8-27 shows a simple meter that is very useful for this task. It uses a thermocouple bead that is very small and can be placed into tight areas. The purpose of the mock-up, of course, is to measure the temperature inside of the system at various locations, and to compare these temperatures with the airflow sketch.

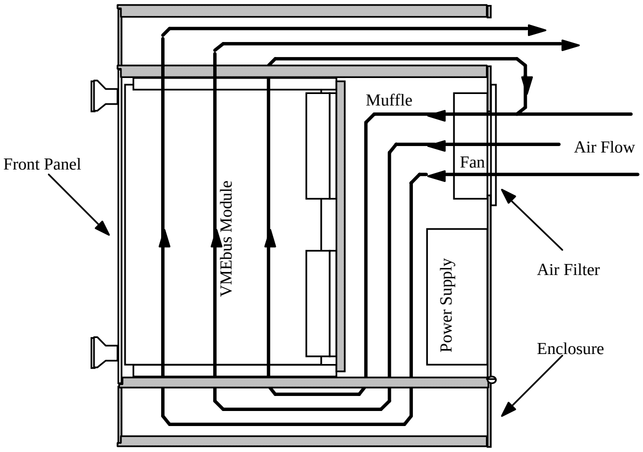
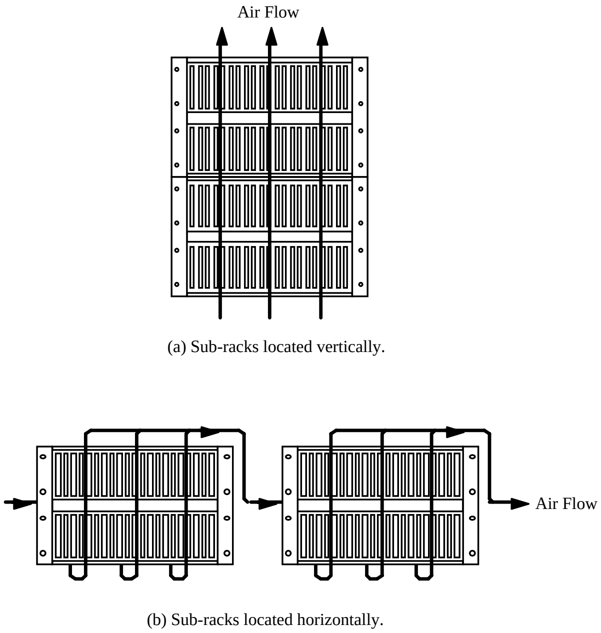


Figure 8-25 A system air flow sketch helps to identify restrictions and problem areas. It should be drawn to scale to identify the severity of restrictions.

Table 8-2 Rules of thumb for air flow sketches

- Air flow sketches provide *estimates*, not exact numbers. Use your intuition.
- The average air flow entering the system is equal to that leaving.
- Air flows along the path of least resistance.
- Use blank front panels in unused card slots.

- Hot air rises because it's lighter (unless the system is in zero gravity).
- Filters, grills, card cages and cabling restrict airflow.
- Air flow from bottom to the top is the most efficient. However, air flow from the floor picks up dirt. Collected dirt, cob webs and other build-up will reduce the ability of bus modules to dissipate heat (i.e. they will run hotter).
- Don't forget power supplies & peripherals in the air flow sketch.



(a) Sub-racks located vertically.

(b) Sub-racks located horizontally.

Figure 8-26 Consider the air-flow impact on other system components

If ambient temperatures above 50 C are expected, the thermal analysis should be performed at the maximum ambient temperature. Air density decreases with temperature and reduces the ability of air to carry heat away. Environmental chambers (ovens) are available for this task.

Several products are available specifically for thermal analysis of VMEbus systems. Figure 8-28 shows a load board for simulating thermal, signal and power loading of VMEbus card slots. When thermoanalyzing a system, these boards can be used for cooling studies, variable heat loads and hot-spot evaluation. They can also simulate full burn-in loads for high reliability systems.

8.4.5 Hot Spots

Hot spots are sometimes found during the thermal analysis. There are several ways of handling the problem if one is found.

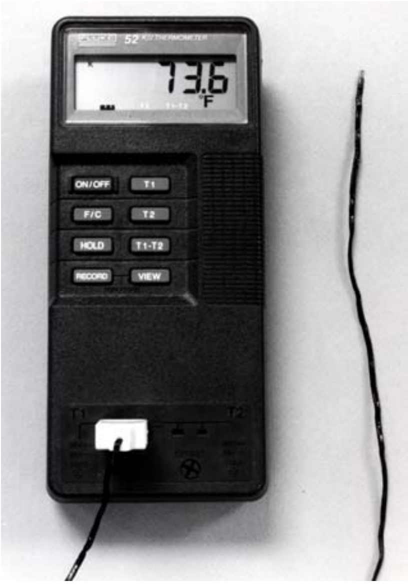


Figure 8-27 Meter with bead type thermocouple is very useful for measuring temperatures

This meter also has dual thermocouple inputs. Dual thermocouples are useful for measuring the temperature rise between two points in the air flow. Photo by Wade Peterson.

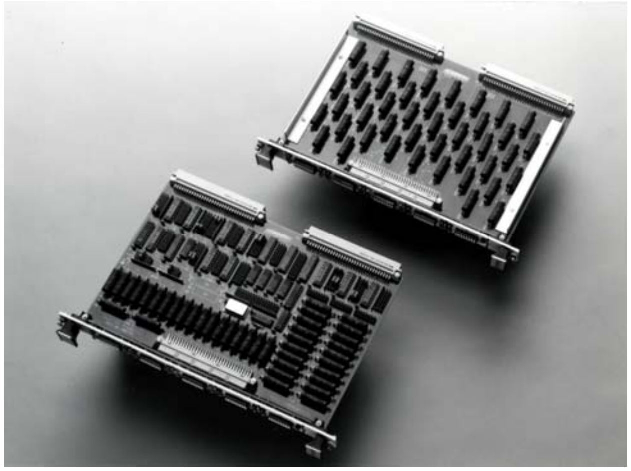


Figure 8-28 Active and passive load boards

Photo courtesy of Dawn VME Products.

If the hot-spot occurs only at very extreme ambient temperatures, consider using an over-temperature alarm. If the system gets too hot, as would happen if a fan or air conditioner failed, then the power supply can be shut off or an alarm activated. Over temperature alarms don't need to be complicated. Simple bi-metal thermostats are available at low cost.

The most common way of cooling a system component is to increase the air flowing past it. For example, the Eurocard fan tray in Figure 8-29 could be used. Another way of increasing air velocity is to restrict some airflow elsewhere in the system. Experimentation on the system mock-up with bits of cardboard will verify which method works best.

When forced air fans are used some sort of filtering mechanism is recommended. Forced air means greater air flow, but also more dirt and dust flow. Unfortunately, layered dirt on PC boards and other system components will reduce their capability to dissipate heat. Tables 8-3 and 8-4 give rules of thumb for adding filters and fans.

Air conditioning is also available for VMEbus systems. Table 8-5 lists rules of thumb for air conditioners.

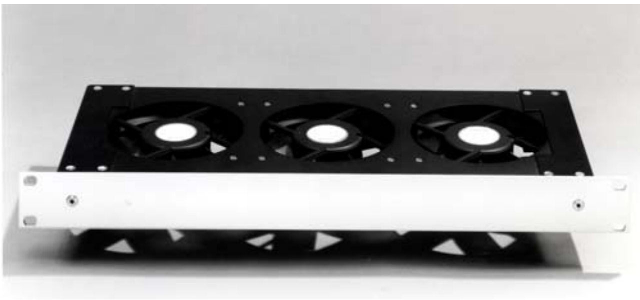


Figure 8-29 Fan Tray

Photo courtesy of Schroff Inc.

Table 8-3 Rules of thumb for air filters

- If forced air is used, then consider using a filter at the air inlet.
- Air filters get dirty and should be cleaned or replaced regularly.
- Dust, pollen, spider webs and other foreign matter will coat the VMEbus modules after a few years. These act as insulators and increase the temperature of the boards. Dirty boards should be vacuumed annually.

Table 8-4 Rules of thumb for fans

- If an AC fan is used, then:
  1) If possible, run the AC wiring away from DC power supplies and cabling. This minimizes electrical noise coupling into the system power supply.
  2) All AC fans don't work at all voltages and frequencies. If the system is to be used in another country, then make sure that the fan will operate with available power.
- Fans make acoustical noise. If acoustical noise is a problem, then:
  1) Don't use a fan that's too big. If multiple fans are used, then consider removing one or more of them.
  2) Air carries sound. A fan blowing into a system will generate less acoustical noise than one pointing out.
  3) Compare fans from two or three suppliers and listen to the difference. Speed, efficiency and bearing quality all affect the noise level.
  4) Fan speeds (and noise) increase if air flow is restricted.
  5) Variable speed fans are available. The speed of these fans are controlled by a thermostat located at the system air flow exit. Therefore, when more heat gets generated, the fans run faster (and generate more noise).
  6) Locate fans where they can't be heard. If a fan is located in the back of a system, it can't be heard as well from the front.
  7) If interior space is available, an expansion chamber can be used as a muffler. This cuts the transmitted noise.
  8) Sound absorbing materials can be placed inside the card cage to muffle noise.
  9) Fan vibration can be amplified by resonating sheet metal. This can be minimized by (a) tightening loose screws, (b) isolating vibrating parts with rubber grommets or (c) dampening large sheet metal parts (i.e. attach a weight to lower the resonate frequency of the part).

Table 8-5 Rules of thumb for air conditioners

- Consider using an air conditioner if heat build-up, humidity or system contamination are problems.
- Set the air conditioner's thermostat to a relatively high temperature such as 40 or 50 C. This prevents excessive condensation, saves energy and extends the lifetime of the unit.

- Air conditioners are fine, but they do have drawbacks:
  1) When humidity is high, condensation builds up on everything cold. If the door on a cooled system is opened, then everything gets wet.
  2) Condensation can be minimized by raising the temperature of the system.
  3) When the air conditioner compressor kicks on, it generates a large noise spike on the AC power line. This line should be isolated as much as possible from the VMEbus power supply.
- Chill plates (peltier devices) can often be used instead of air conditioners.

8.4.6 Humidity & Moisture Control

High humidity is a problem in some systems. Moisture, especially when combined with other contaminates (such as salt spray), conducts electricity. Sensitive circuits, such as high impedance amplifiers or crystal oscillators, will fail if moisture condenses on ICs or signal traces.

Long term humidity can cause more serious damage to the system. Corrosion is accelerated when moisture is present. Fungus growth can cause permanent damage to PC boards, power supplies, fans and other components.

Condensation occurs when moist air is cooled below the dew point. The same thing happens when you drink a cold glass of lemonade on a hot, humid day. Water droplets condense on the outside of the glass. The same is true for the VMEbus system...if the system temperature falls below the dew point, then condensation will form on the chassis.

Moisture evaporates when air temperature increases. That's because warmer air holds more moisture than cooler air. One good example is a clothes dryer. Clothes are tumbled through heated air. That allows moisture to evaporate faster, and the clothes dry quicker.

Intermittent system failures have been linked to humidity problems. For example, one system was found to resist start-up. The system wouldn't start right away, but eventually (after about five minutes) would always spring to life. The curious thing about the problem was that it only occurred in the morning. If the system was started later in the day, it was fine. It turned out that moisture condensed on one of the bus modules at night. If started in the morning, the bus module would fail. If the power was left on for about five minutes the ICs would warm up and evaporate the moisture. Subsequent re-starts worked fine.

Table 8-6 lists several ways to control humidity in VMEbus systems. More information about moisture and fungus control can be found in the (US) military standard MIL-T-152 entitled *Moisture and Fungus Resistance of Communication, Electronic and Associated Electrical Equipment*.

Table 8-6 Humidity control in VMEbus systems

Humidity Solution	Comments
Air Conditioning	Air is cooled and moisture forms in an air conditioner's evaporator coil. Water is drained away from the system. De-humidifiers (that provide no cooling) are also available for this purpose. Air conditioners are perhaps the best way to insure uniform humidity in the system. However, they do have their problems. They are expensive to install and operate, they are noisy (acoustical and electrical) and they aren't very reliable. If a cooled cabinet is used, then condensation tends to form on everything when the door is opened on a hot, humid day.
Conformal Coating	Conformal coatings on PC boards reduce the effects of humidity. Moisture forms on top of the coating and is insulated from IC leads and PC board traces. Conformal coatings also prevent corrosion and fungus growth problems.
Sealed System	Sealed systems are less susceptible to humidity effects, especially when combined with a desiccant. Unfortunately, they usually require conduction cooled boards. Temperature cycling also tends to pump moisture in and out of sealed systems. Temperature variations cause pressure changes in the system, and air tends to be sucked in or pushed out. Die-cast aluminum parts, for example, are quite porous and allow slight air exchange. Moisture can collect inside the sealed unit, sometimes even causing several inches (cm) of water to form on the bottom. The term *'hermetic seal'* refers to seals that don't pump moisture in or out. However, hermetically sealed systems are very expensive.
Humidity Sensors	Relative humidity sensors are available. These sensors warn of dew point conditions by audible or visual alarm.
Power Cycling	Less condensation forms on warm systems. Leaving the system power on keeps it warm and prevents condensation.
System Location	More condensation forms in the cooler part of a room or building. Locate the system away from cool, humid areas (such as a concrete floor).

8.5 Disk Drive Mounting

Disk drive mounting kits are available for chassis or rack mounting. Figure 8-30 shows a Eurocard rack mounted hard/floppy disk drive package.

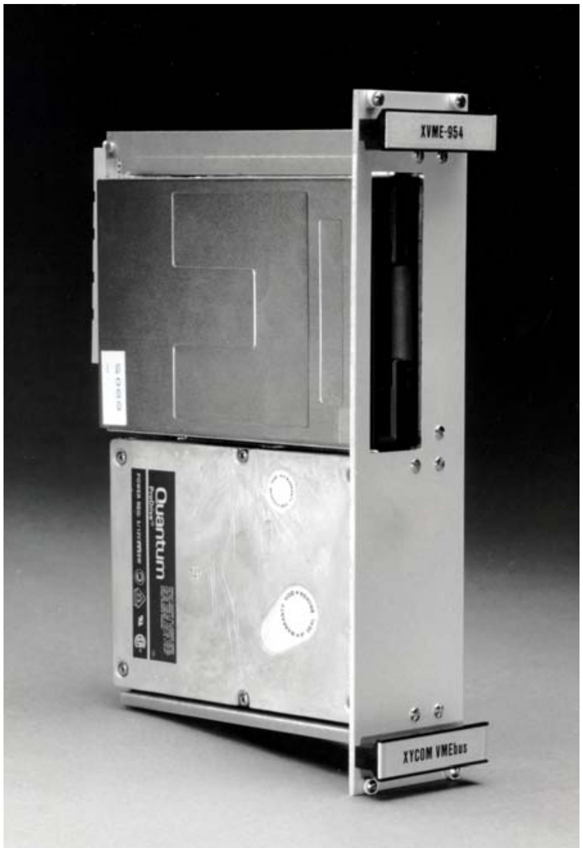


Figure 8-30 Hard disk / floppy disk unit for VMEbus

Photo courtesy of Xycom, Inc.

8.6 Bus And Card Adapters

Bus and card adapters are a popular way to intermix microcomputer buses. Usually, communications between buses is done over an RS-232C port, an IEEE-488 interface or some other style of network. The problem with these schemes is they require a lot of software overhead and have relatively low data transfer rates. Bus and card adapters are inherently fast, and often eliminate the need to write communication software.

Bus adapters directly couple VMEbus and other microcomputer buses. *Card adapters* allow cards from other buses (such as IBM-PC/XT or STD bus) to fit into a VMEbus card cage.

8.6.1 Bus Adapters

Bus adapters are available to connect VMEbus with other buses such as ISA bus, PCI bus, Multibus™, Q-Bus and many others. For example, Figure 8-31 shows an adapter which enables the IBM-PC/AT ISA bus to act as a bus master in a VMEbus system. This allows a wide variety of VMEbus peripherals to be used by an IBM-PC host system. Figure 8-32 shows a VMEbus-to-VMEbus adapter which enables a VMEbus card cage to be extended beyond its 21 slot maximum.

Bus adapters use either *memory windowing* or *shared memory* techniques to pass data between systems.

8.6.1.1 Memory Windowing Adapters

Memory windowing maps a remote system's memory into a local address space as shown in Figure 8-33. Read or write cycles to the bus adapter generate read or write cycles in the remote system's memory. Since the remote system appears as a peripheral to the local system, no communication software needs to be written. Memory and peripherals in the remote system appear as the same devices in the local system.

Two VMEbus systems with adapters, using the memory windowing scheme, can directly communicate with each other in a bi-directional manner. Each system can access the other's memory space without the use of shared memory. Unfortunately, there are some subtle system issues that need to be addressed in order to do this.

One area of concern is *deadlock*, which may occur in systems where two processes are interlinked. Deadlock happens when masters on both systems attempt to access the other at exactly the same time. If neither end of the bridge has the ability to 'back-off' from a bus cycle, then can both get stuck in a *deadly embrace*. This causes both systems to crash.

There are two ways of solving this problem:

- With software semaphore registers on the bus adapters.
- By using the VME64 RETRY* function.

Figure 8-31 IBM-PC/AT ISA bus-to-VMEbus adapter

These modules allow the IBM-PC/AT to operate as a VMEbus master.

Photo courtesy of Bit-3 Computer Corp.

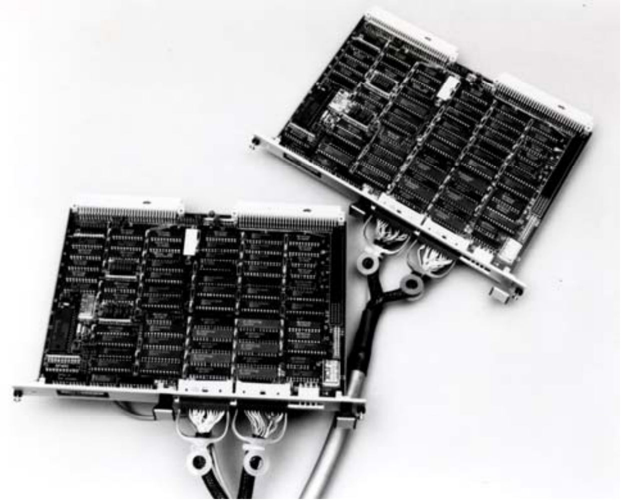


Figure 8-32 VMEbus-to-VMEbus adapter

These modules allow a VMEbus card cage to be extended beyond the 21 slot maximum. Photo courtesy of Bit-3 Computer Corp.

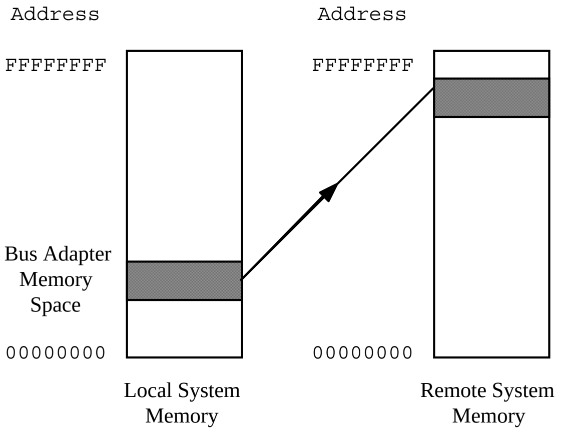


Figure 8-33 Memory windowing maps a remote system's memory into a local address space

Read or write cycles to the bus adapter will generate read and write cycles to the remote system's memory space.

The traditional way of handling this problem has been with software semaphore registers on each end of the bus adapter pair. Whenever a VMEbus module needs to access the other system, it sets a bit in the adapter semaphore register. The bus adapters then arbitrate for the destination bus, and set another bit in the semaphore register when it is done. The VMEbus module can then proceed with its bus cycles in the far system.

This method works well, and is very reliable. However, it is quite time consuming, as it takes several bus cycles to complete the arbitration process with the semaphore registers. It is also very difficult to do with DMA (Direct Memory Access) controllers, which can't twiddle bits as well as a CPU. The solution to this problem came with the introduction of the RETRY and LOCK functions in the VME64 specification.

The VME64 specification re-defined the RESERVED pin (P2/J2 pin B3) as RETRY*. This allows bus cycles to be aborted and attempted at a later time. It is intended for use in multi-ported applications such bus bridges and dual-ported memories.

When a slave board asserts RETRY*, it is indicating that the VMEbus master should retry the cycle again at a later time. In the VMEbus-to-VMEbus adapter example, the bus adapter asserts the RETRY* signal whenever deadlock occurs. Refer to Chapter 2 for more information on the functionality of the RETRY* signal pin.

The VME64 LOCK commands can also be used by the bus adapter to prevent the far system from accessing the near system. This allows boards in the near system to operate without interruption from the far system. When a CPU issues a LOCK command on the backplane, the bus adapter would recognize the command, and issue a RETRY* signal to all masters who attempt to access the near system. Refer to Chapter 2 for more information about the LOCK commands.

8.6.1.2 Shared Memory Adapters

Another technique used by bus adapters creates a shared memory space on one end of the bus adapter, as in Figure 8-34. Both microcomputer buses can access this memory as if it were local. The advantage of shared memory is that bus bandwidth is not gobbled up by inter-system communication. This technique is sometimes called reflective memory.

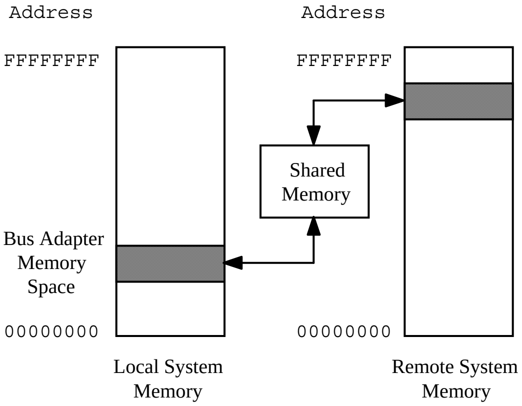


Figure 8-34 Some bus adapters use shared memory to buffer inter-crate communications

This reduces bus bandwidth requirements. One variation of this technique is sometimes called reflective memory.

8.6.2 Card Adapters

Card adapters allow modules from other buses (IBM-PC/XT ISA, PCI, etc.) to fit into a VMEbus card cage. Card adapters are very useful when I/O functions are available only on a bus structure other than VMEbus. Figures 8-35 and 8-36 show card adapters for IBM-PC/XT ISA and PCMCIA buses.

8.7 Side Buses

Many applications require a secondary channel to VMEbus. These are called side buses, and they allow several modules to share data over a private backplane. VMEbus traffic is reduced as a result. Some of the more popular side buses include:

- VMXbus
- VSBbus, ANSI / IEEE 1096-1988
- Front Panel Data Port, VITA 17-199x
- Crossbar switches (RACEway, SKYchannel, Myrinet)
- VMSbus (now obsolete)

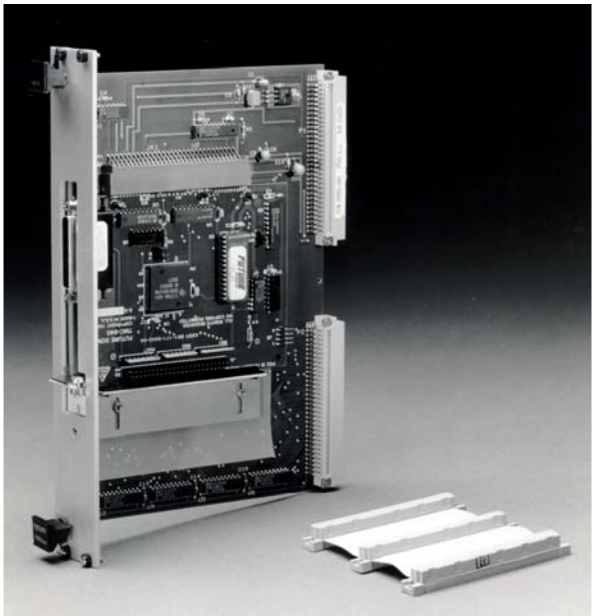


Figure 8-35 VMEbus card adapter to IBM-PC/XT ISA bus

Photo courtesy of Xycom, Inc.

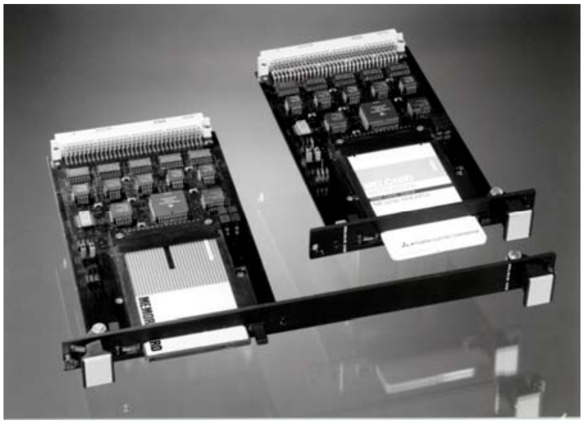


Figure 8-36 VMEbus card adapter to PCMCIA bus.

Photo courtesy of EKF Elektronik GmbH.

CPU modules often use side buses for private memory or I/O accesses. Memory expansion, I/O channel, graphics channel and inter-processor communication are just a few applications for the side buses.

For example, a single VMEbus CPU module may not be able to hold enough memory on its card. With a side bus, memory can be expanded without affecting

VMEbus bandwidth. Figure 8-37 shows such an example.

8.7.1 VMXbus

The VMXbus is a secondary channel to VMEbus. It uses its own backplane connected to the P2 user defined pins, which standard J2 backplanes do not bus. The VMXbus backplane can be "piggy backed" onto a J2 backplane or incorporated into one as shown in Figure 8-38. Up to six modules can reside on VMXbus.

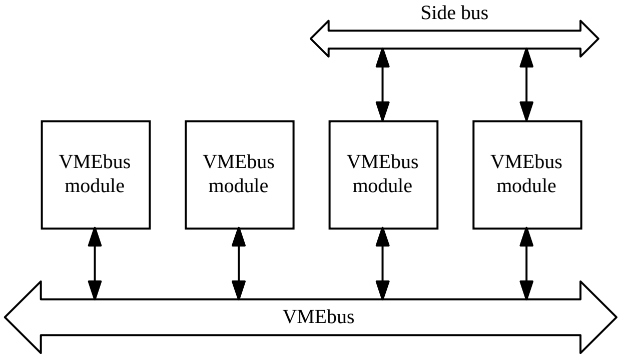


Figure 8-37 A side bus used for a private communication channel between modules

This reduces VMEbus traffic.

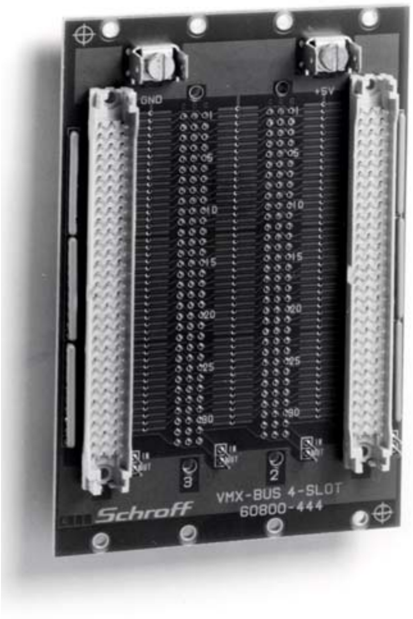


Figure 8-38 Combination VMEbus J2 and VMXbus backplane

This special backplane busses the VMEbus signals on the center row (row B) pins and VMXbus on the outer two rows (rows A and C). Note the bus grant daisy-chain jumpers on the bottom of the backplane.

Photo courtesy of Schroff Inc.

VMXbus has a 24-bit address and a 32-bit data bus. It has no power or ground connections, as all bus modules use the J1 and J2 power pins. The address bus is multiplexed. This was done because there are only 64 user defined I/O pins on the P2/J2 connector, and multiplexing reduces the number of pins needed. The data bus, however, is non-multiplexed.

A single interrupt handler, and any number of interrupt requesters, may reside on VMXbus. In response to an interrupt, the handler polls all the interrupters and services the highest priority one (there is no automatic arbitration or prioritization). The interrupt release mechanism is a RORA type (release on register access), similar to VMEbus RORA. Since there is no interrupt acknowledge cycle, automatic STATUS/ID bytes (interrupt vectors) are not used.

Multiprocessing is supported on the bus via a single level arbiter. This arbiter is similar to the VMEbus single level arbiter. Any module may request the bus using an open collector bus request line. During arbitration, a daisy-chain is passed from the first module (called the primary master) to the other modules in the system. The VMXbus bus grant daisy-chain is similar to the VMEbus bus grant daisy-chain. Other features of the bus include a read-modify-write cycle, block transfer cycle and an address-only cycle.

An upgrade called MVMX32bus was later developed by Motorola Inc., but never underwent formal standardization. Like VMXbus, MVMX32bus uses the P2 user defined pins, accommodates up to six modules, uses a single level arbiter and interrupt system, and supports read-modify-write, block transfer cycle and address-only cycles. Added features include a 32-bit address range, alternate and I/O addressing spaces (similar to VMEbus short I/O) and a special cacheable read cycle. The two buses are not compatible, however.

MVMX32bus timing was originally designed for a ribbon cable backplane. These are simply a piece of 64 pin ribbon cable connected between the J2 connectors of adjacent bus modules. A photograph of one is shown in Figure 8-39. Termination networks at the ends of the backplane reside on the bus modules. Ridged backplanes can also bue used, however.

8.7.2 VSBbus

The VME Subsystem Bus (VSBbus) provides a secondary communication path on the P2/J2 user defined pins. It has a 32-bit address bus, a 32-bit data bus, a single level interrupt structure, a single level bus arbitration, dynamic bus sizing, read-modify-write capability, block transfer and an address-only cycle. It can support a maximum of six card slots, at bandwidths up to 32 Mbyte/second.

This side bus is defined by the ANSI/IEEE Std. 1096-1988 entitled *VSB Specification*.

VSBbus is not compatible with VMXbus, and is marginally compatible with MVMX32bus. One major improvement of this bus over the VMXbus or MVMX32bus is its ability to handle STATUS/ID transfers during interrupt acknowledge cycles.

VSBbus requires a printed circuit backplane. It can be a special VMEbus / VSBbus backplane, like that shown in Figure 8-40, or a modular 'overlay' type, like the one shown in Figure 8-41. Ribbon cable backplanes cannot be used with VSBbus.

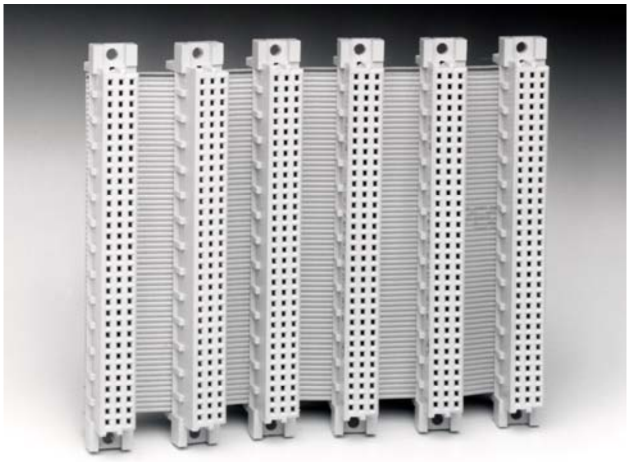


Figure 8-39 MVMX32bus backplane

An MVMX32bus backplane can be as simple as this 64 pin ribbon cable connected between the J2 backplane connectors. Photo by Rob Bigelow.

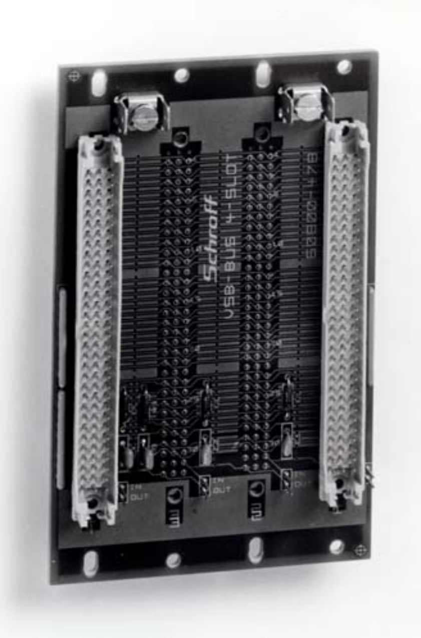


Figure 8-40 Combination VMEbus J2 and VSBbus backplane

Photo courtesy of Schroff Inc.

8.7.3 Front Panel Data Port (FPDP)

The front panel data port (FPDP) is a side bus that connects VMEbus modules together through a front panel ribbon cable. It is intended for high speed data transfers between two or more VMEbus boards, with minimum latency and precisely known data rates.

At the time of printing of this handbook, the FPDP is a preliminary standard defined under VITA 17-199x.

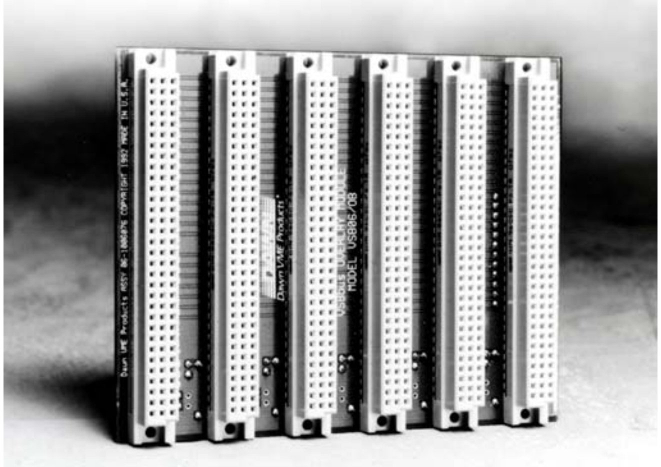


Figure 8-41 VSBbus overlay module

Photo courtesy of Dawn VME Products.

The FPDP bus is a 32-bit parallel synchronous bus wired by means of an 80 conductor ribbon cable, which is routed between VMEbus front panels. The P1/J1 and P2/J2 connectors are not affected.

A single master synchronizes transfers with a free running clock. The transfers occur in one direction only. However, interfaces can be made to support bi-directional operation under software control.

The FPDP bus protocol does not include address or arbitration cycles, so the data transfer rate is throttled by the free running clock. A mechanism is provided so that a receiving board may suspend transmissions from the sending board. A mechanism is also provided to allow synchronization of the receiver to the transmitted data stream.

8.7.4 Crossbar Switches

One popular type of VMEbus side bus called the *crossbar switch*. There are three types of crossbar switches used in conjunction with VMEbus. These include:

- RACEway Interlink, ANSI/VITA 5-1994
- SKYchannel, VITA 10-1995
- Myrinet, VITA 26-199x

The details of these topologies are beyond the scope of this handbook. However, we'll briefly describe some of the features included in all of them.

Crossbar switch overlay backplanes can be plugged into the back of standard VMEbus J2 backplanes. For example, VMEbus modules with a RACEway interface can communicate over a RACEway overlay backplane through the P2/J2 user defined I/O pins. The overlay board usually contains the crossbar switch, which is formed from one or more integrated circuits.

A common crossbar switch has a six port topology like the one shown in Figure 8-42. It can be used as part of a larger network, which is sometimes called a *fat-tree* topology. In its most basic form there are two channels, called A and B, which are also called *parents*. Four channels, called C, D, E and F are also known as *children*. Each of the parents and children can communicate with any other member of the family using the crossbar.

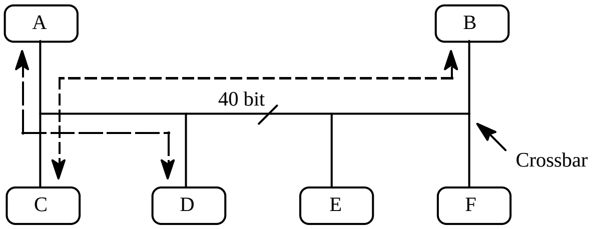


Figure 8-42 Crossbar switch topology

Each family member (i.e. port) is composed of a 40-bit parallel bus, which can support burst transfers (in one direction) at over 160 Mbytes/second. When a message is sent from one family member to another, it includes destination (addressing) information. The crossbar then routes the message using an optimizing algorithm.

Each crossbar family can be expanded to connect to other families. This forms a scalable network that offers:

- Dynamic configurability.
- Scalable architecture
- Parallelism

The crossbar routing mechanisms support *dynamic configurability*, thereby creating a re-routable and reliable network system.

The crossbar architectures are *scalable*, meaning that families of crossbars can be added as the need arises.

Finally, the crossbar architectures exhibit *parallelism*. That means that communication can take place between many nodes at the same time. Standard buses, such as VMEbus, allow only one communication path at any given time. This makes the crossbar switches inherently faster than VMEbus.

8.7.5 VMSbus (obsolete)

VMSbus is generally considered to be obsolete. The bus was never well supported, so the SERCLK and SERDAT* pins were redefined as SERA and SERB in VME64.

VMSbus is a synchronous serial bus. Like the VMXbus and VSBbus, it provides a secondary communication path between VMEbus modules. Unlike these other buses, however, the VMSbus is used for short messages rather than large memory transfers. It also provides a flexible way to pass interrupts between modules, and supports fault tolerant systems.

VMSbus may be used to connect several bus modules, sub-racks or multiple chassis. Its maximum clock speed is 2.9 Mbits/second when used over short distances. If longer runs are required, the clock speed may be slowed down. Signal lines SERCLK and SERDAT* are used to transfer data.

VMSbus uses a collision detection method for arbitration. If two modules attempt to send data at the same time, one message will be garbled. The module whose message was garbled then stops transmitting and lets the other complete its message.

Once VMSbus has been arbitrated, a control message frame is broadcast by a functional module called a controller. The controller (there may be many of these) initiates a message transfer between functional modules called talkers and listeners. The selected talker then sends a short data packet (up to eight bytes) to a listener. The VMSbus specification does not regulate the content of these messages.

Two VMSbus ICs were available from Signetics Corporation. These were the SCB68171 Very Little Serial Interface Chip (VLSIC) and the SCB68173 VMSbus controller (VMSCON). The SCB68171 interfaces one or more SCB68173 bus controllers to VMSbus. The SCB68173 contains a VMSbus controller, talker, listener and four general purpose flags. The 680XX compatible devices were fully programmable and could generate an interrupt to a host CPU.

8.8 Mezzanine Modules

Mezzanine modules are a popular way to customize a VMEbus card. These cards generally fit on top of (or beside) a common base board. This allows users to 'mix-and-match' the I/O functions that they need.

The mezzanine modules have become very popular in recent years. That's because they save a great deal of hardware development time, and inventory. Software development is also faster and less expensive.

While the mezzanine module concept has been around for many years, the VMEbus mezzanine industry has been fragmented because of all of the types of mezzanine cards. This situation has improved steadily as new standards have been adopted.

There is currently a plethora of mezzanine standards that are available. At the time of printing of this handbook in 1997 there were over 30 types available for VMEbus modules. However, two standards seem to be gaining a substantial foothold. These are the CMC/PMC and IP standards.

8.8.1 CMC / PMC Mezzanine Standards

CMC stands for Common Mezzanine Card, and defines a *mechanical* standard for mezzanine cards. The CMC standard does not define the electrical interface other than for the connector and voltage types. It is currently defined under the IEEE P1386.

The CMC standard is defined for VME64, Multibus I, Multibus II, Futurebus+ and generic platforms. This means that CMC mezzanines can be used with any of these systems, thereby increasing its user base. Figure 8-43 shows the main features of a VME64 CMC module. These include:

- A common mechanical layout for 3U and 6U VME64 modules.
- Capability for single and double-wide mezzanine cards on 6U modules.
- Four, 64 pin connectors.
- Mezzanine modules are parallel to the baseboard, and only use a signal slot.
- A front panel system including filler panel and EMC gasket (for front panel I/O).
- A voltage keying pin for both +3.3 and +5 VDC systems.

PMC stands for PCI mezzanine card. It defines the electrical interface for the CMC mechanical system, and is based on the popular PCI bus interface protocol. The PMC attaches to the CMC host module, which is why it is called a CMC/PMC mezzanine. Figure 8-44 shows an example of a PMC module, and Figure 8-45 shows the dimensions.

Single and double wide PMC cards are available on VME64. Each 6U VME64 host can hold one double or two single PMC cards. Each 3U VME64 host can hold one single PMC card.

One advantage of the PMC mezzanine is that little or no additional logic is needed when a PCI bus already exists on the baseboard. The mezzanine is simply connected up to the local PCI bus.

The peak bandwidth of the 33 MHz PCI bus is 132 Mbyte/second. The four 64-pin connectors carry the following signals to the PMC card:

- Two connectors carry the 32-bit PCI bus.
- One connector carries the 64-bit PCI expansion.
- One connector carries user defined I/O.

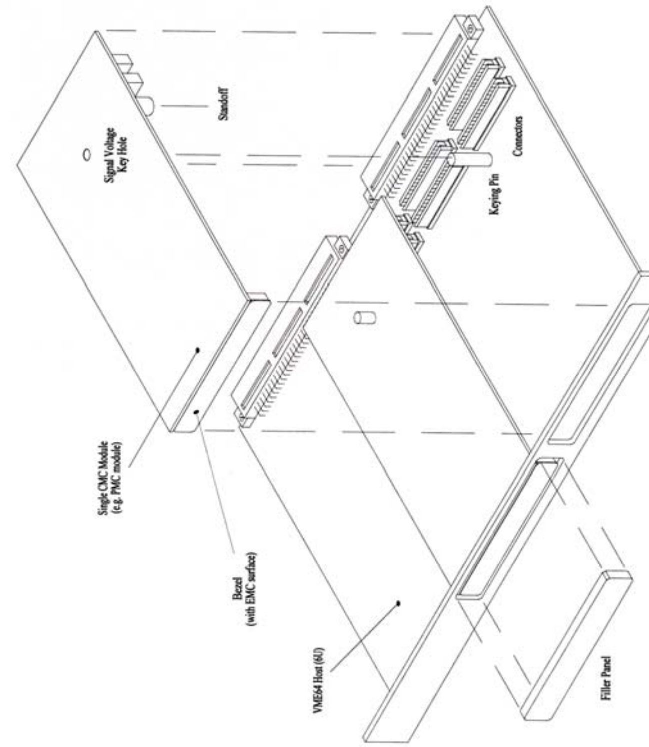


Figure 8-43 Main features of a VME64 CMC module

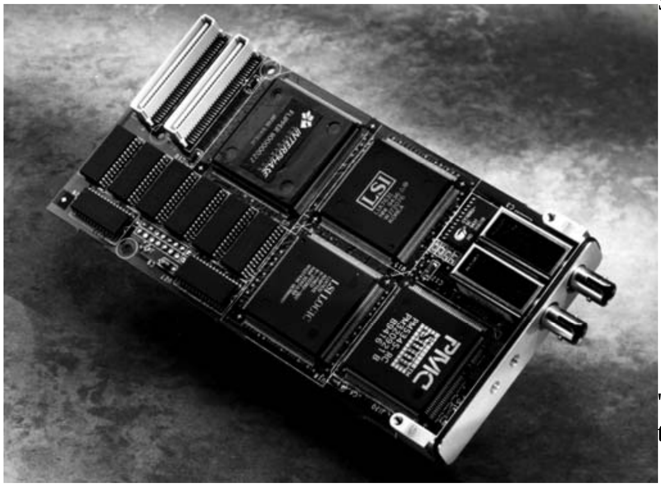


Figure 8-44 Single-wide PMC module

Photo courtesy of Interphase Corporation.

The PMC I/O connector can be wired directly to the VME64 P2/J2 user defined I/O pins, or to the P0/J0 connector. I/O can also be brought out the front panel bezel of the PMC.

A voltage keying pin is included on the CMC baseboard. This pin defines whether the baseboard logic operates at +3.3 VDC or +5 VDC. Voltage keying holes are also located on the PMC module. This only allows PMC modules with compatible voltage levels to be placed onto the CMC baseboard.

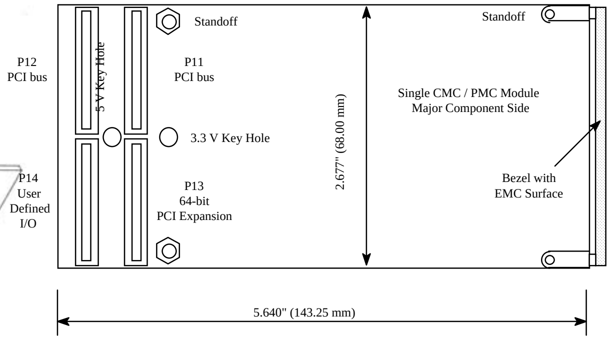


Figure 8-45 Dimensions and features of a single-wide PMC module

8.8.2 IP Mezzanine Modules

IP mezzanine modules are very popular for simple I/O such as analog-to-digital, digital-to-analog and thermocouple converters. Up to four IP modules can be placed on a 6U VMEbus module. It is standardized under the ANSI/VITA 4-1996. Figure 8-46 shows an IP module, and Figure 8-47 shows the carrier module. Some features of the IP module include:

- Simple I/O circuit.
- Synchronous interface at 8 or 32 MHz.
- Up to a 32-bit data path.
- 64 Mbyte/second continuous data rate.
- Autoconfiguration PROM
- Two interrupt levels per card.
- Two DMA channels per card.
- I/O can be routed to the front or the rear of the VMEbus module.

'IP' is short for IndustryPack™, which is a registered trademark of Greenspring Computers.

8.9 Rapid Development Modules

Design changes are commonplace during most system integrations. In addition, custom peripherals and I/O interfaces are often needed. Rapid development modules, such as 3U and 6U wire-wrap boards, are very useful when time-to-market is crucial.

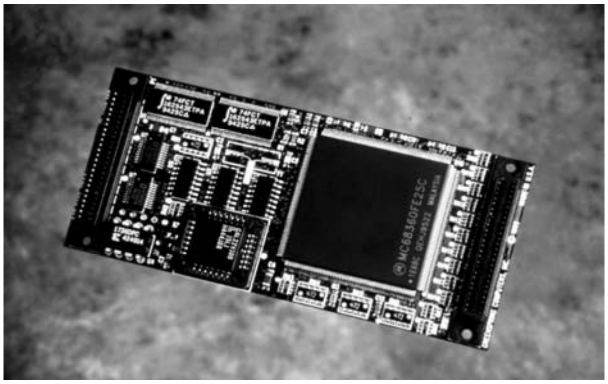


Figure 8-46 IP module

Photo courtesy of Greenspring Computers.

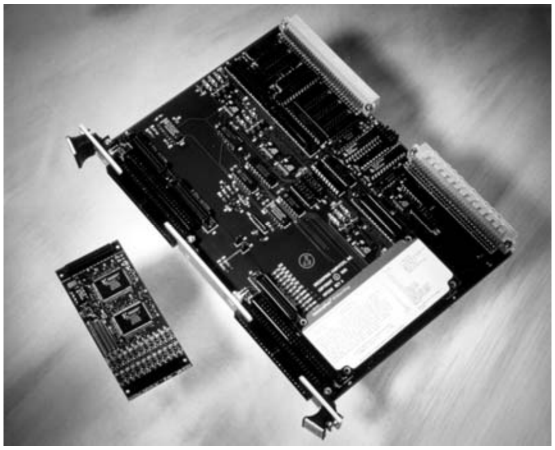


Figure 8-47 6U IP carrier module

Photo courtesy of Greenspring Computers.

One popular development module is the hybrid prototyping board. A typical board is shown in Figure 8-48. These generally have slave and interrupter interfaces with large prototyping areas.

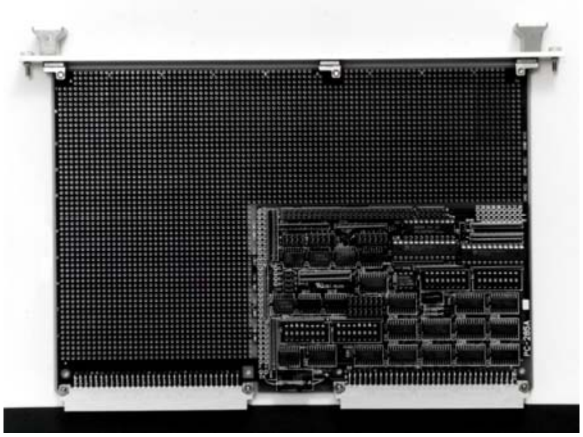


Figure 8-48 VMEbus prototyping module with D16:A24 slave and interrupter interface

Photo courtesy of Ampere Inc. (Tokyo, Japan).

Another excellent rapid development tool is the side card module as shown in Figure 8-49. A DIN connector, which carries local bus signals, is mounted on the lower edge of a single high (3U) CPU board. Customized side cards are attached to form a double high (6U) bus module.

Custom side card modules are much faster and less costly to develop than smart peripherals designed from scratch. For example, a system integrator may be able to purchase every bus module except for a specialized I/O interface. With the side card configuration the integrator designs only the peripheral portion of the interface. CPU hardware, VMEbus interface and software tools are ready-to-go. In some applications the time-to-completion has been cut by 80%.

Side card configurations are also available in double wide 3U styles. Modules are formed by using a right angle DIN connector and mounting the side card parallel to the CPU.

8.10 Debugging Tools

Inevitably, there are times where things don't go right, especially during the system integration phase of a project. The resolution of software and hardware bugs are perhaps the system integrator's most challenging job. Traditional debugging tools include volt-ohm meters, oscilloscopes, emulators and logic analyzers. In addition, there have been many tools developed specifically for VMEbus. Some, like extender cards and debug monitors, are very simple. Others, like anomaly triggers and bus analyzers, are specialized tools that provide powerful debugging techniques.

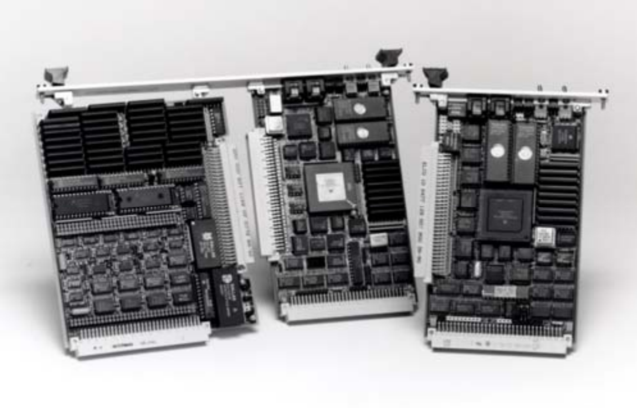


Figure 8-49 Side card arrangement cuts time-to-market for custom peripherals

Custom application modules are attached to a 3U CPU to form a 6U bus module. Development time is cut because there is no need to design the hardware or software for the CPU portion of card.

Photo courtesy of Mizar, Inc.

8.10.1 Debug Monitors

One of the most widely used debugging tools is the firmware debug monitor. The simplest debug monitors allow users to generate read and write cycles. The more sophisticated versions also have assemblers, disassemblers, disk support, memory test and a plethora of other functions.

The purchase of a debug monitor is strongly recommended for any system design. They are inexpensive, and can save a lot of time. The debug monitor allows the integrator to bypass much of the software complexity of high level operating systems and languages, thereby simplifying the debugging tasks.

8.10.2 Extender Cards

One good VMEbus debugging accessory is an extender card. Examples of standard and 160 pin versions are shown in Figures 8-50 and 8-51. These units simplify the connection of test instruments to bus modules, and allow direct access to components.

By strict interpretation of the VMEbus specification, extender cards are not permitted because they violate the maximum signal trace lengths allowable on any signal (19.7"). The extra length sometimes causes systems to fail due to altered bus timing and noise (from impedance mismatches). Fortunately, they usually work and are a convenient debugging tool.

8.10.3 Anomaly Triggers

An anomaly trigger is a board that monitors bus activity and automatically screens for common timing violations on VMEbus cycles. One popular anomaly trigger, shown in Figure 8-52, is the VBAT module made by Ultraview. This board constantly monitors the bus signals and provides real time detection of up to 28 timing protocol violations. One or more of the 37 LEDs on the front panel illuminates if a violation is detected. Each LED is associated with a RULE in the VMEbus specification. This permits rapid debugging of system problems.

Each LED on the Ultraview VBAT module also has a trigger output associated with it. This permits logic analyzers and oscilloscopes to be triggered on an offending bus fault condition.

A VMEbus anomaly trigger is not only useful for finding system bugs, it can also be a valuable board evaluation unit. The anomaly trigger can be used to verify bus compliance during the board selection phase of a system integration project.

8.10.4 Analyzers

VMEbus analyzers are logic analyzers that are tailored specifically to VMEbus. There are two types: timing analyzers and state analyzers. Timing analyzers use an internal clock to latch the state of each VMEbus signal. State analyzers use the VMEbus handshaking lines to latch the state of each VMEbus signal. Figure 8-53 shows a typical VMEbus analyzer.

VMEbus analyzers are useful for hardware and software troubleshooting. Hardware handshaking and timing problems can be tracked down with either the timing or state analyzers. Most analyzers also come with sophisticated state triggering mechanisms that permit debugging of very complex software problems.

8.10.5 Stimulators

A bus stimulator can be used to initiate the various bus protocols used by VMEbus. Read/write cycles, interrupt requests, arbitration and utility bus cycles can be generated with a bus stimulator. This simplifies the debugging of VMEbus modules, and permits rapid analysis of many hardware bugs.

8.10.6 Ownership Monitors

An ownership monitor shows which master has control of VMEbus at any given time. This permits rapid debugging of arbiter and requester designs.

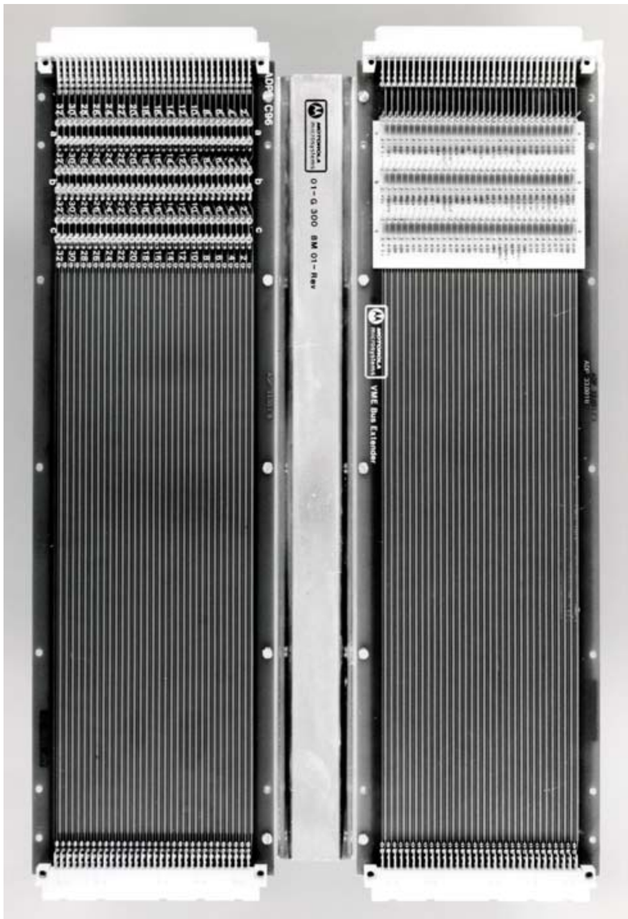


Figure 8-50 Standard VMEbus extender card

Photo courtesy of Motorola Computer Group.

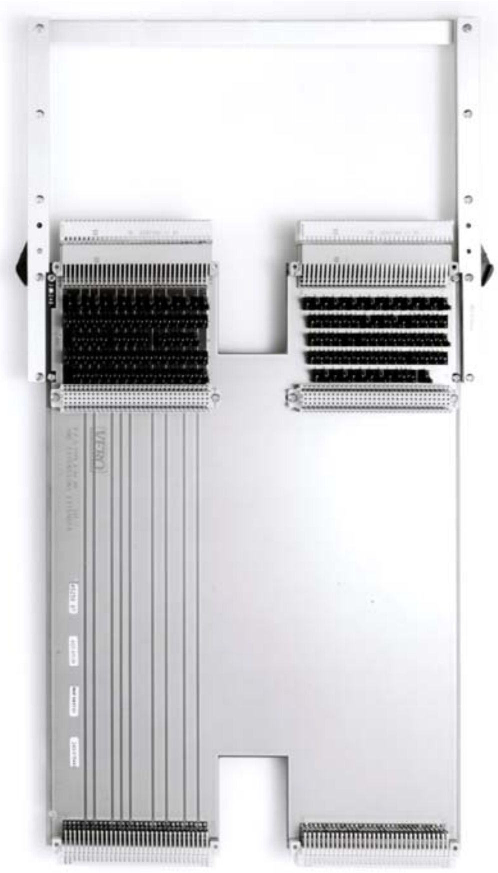


Figure 8-51 VME64x extender card with 160 pin connectors

Photo courtesy of VERO Electronics, LTD.

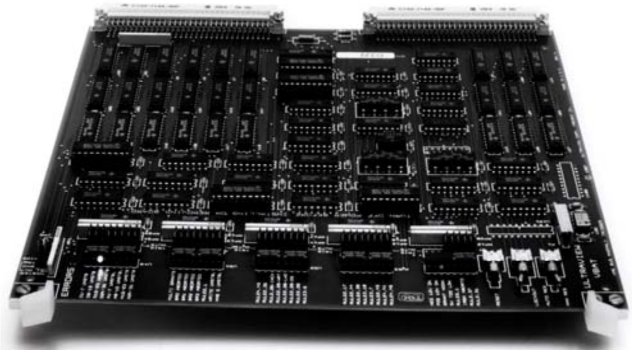


Figure 8-52 VMEbus anomaly trigger

Photo courtesy of Ultraview Corporation.

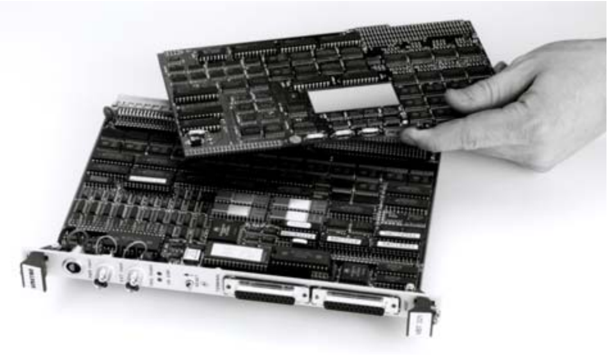


Figure 8-53 VMEbus analyzer

Photo courtesy of VMEtro, Inc.

The ownership monitor can also be used for static loading of multiprocessors during system integration. A histogram of bus activity can be generated for each processor, and the software 'load' on each processor adjusted accordingly.

8.10.7 Surveillance Monitors

A system surveillance monitor analyzes critical system parameters and allows the VMEbus system to phone home if it is not functioning correctly. One such product monitors system temperature, voltage and bus activity. An onboard modem is available for basic communication. Users can dial-up the board and monitor the system. The surveillance monitor can also phone home when a critical parameter jeopardizes system integrity.

Applications requiring remote surveillance include telecommunication, seismic, radar and mission control systems.

8.10.8 Power Supervisor

A power supervisor is a module that monitors DC power supply voltages. This prevents the destruction of expensive bus modules.

8.10.9 Signal Display

A signal display module uses front panel LEDs to display the state of VMEbus signals. These boards are usually inexpensive, and are surprisingly effective.

8.10.10 Load Boards

A load board is a bus module that places an electrical load on the power supply. Most load boards are simply VMEbus modules populated with resistors. They are used to monitor the effectiveness of the power distribution system (over wide ranges of bus loading and number of cards), and to measure the cooling effectiveness of the system.

8.11 Chassis Layout

Figure 8-54 to 8-64 show successful VMEbus chassis layouts.

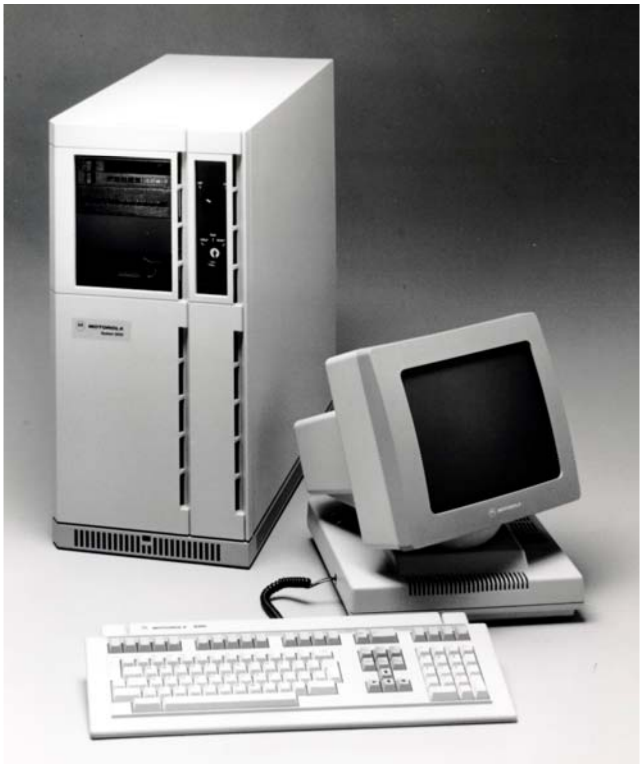


Figure 8-54 VMEbus Unix™ development system in a tower package

Photo courtesy of Motorola Computer Group.

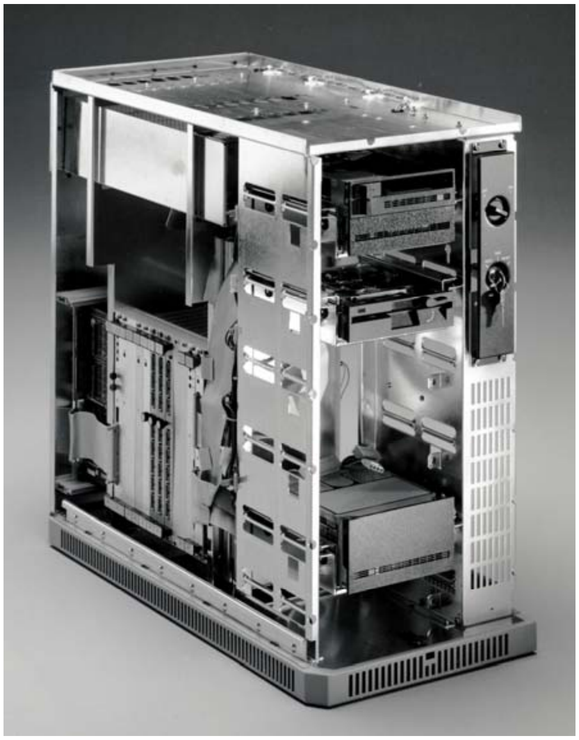


Figure 8-55 Internal view of VMEbus tower system

Photo courtesy of Motorola Computer Group.

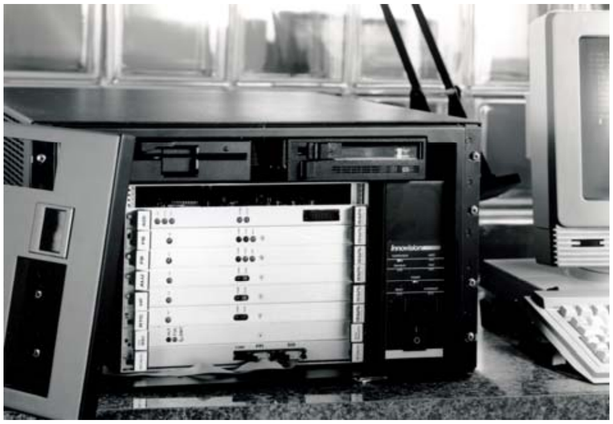


Figure 8-56 Internal view of Unix™ software development system

Note horizontal board placement. Photo courtesy of Heurikon Corporation.

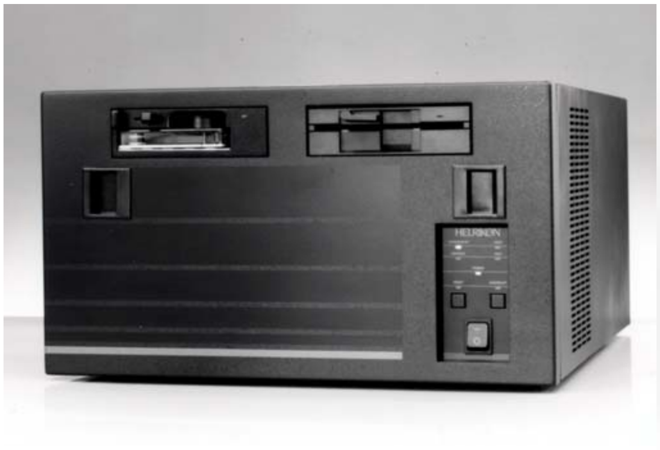


Figure 8-57 External view of Unix™ software development system

Photo courtesy of Heurikon Corporation.

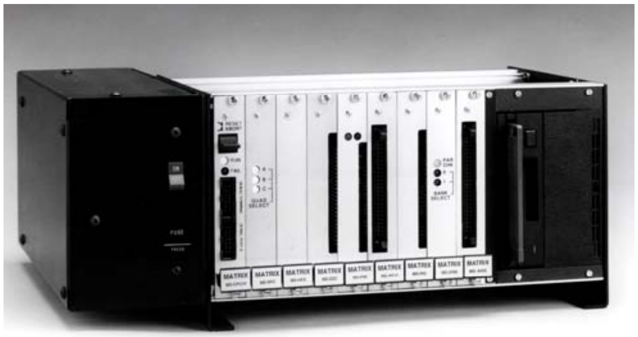


Figure 8-58 Single height VMEbus system with 3 1/2" floppy and Winchester disk drives

Photo courtesy of Matrix Corporation.

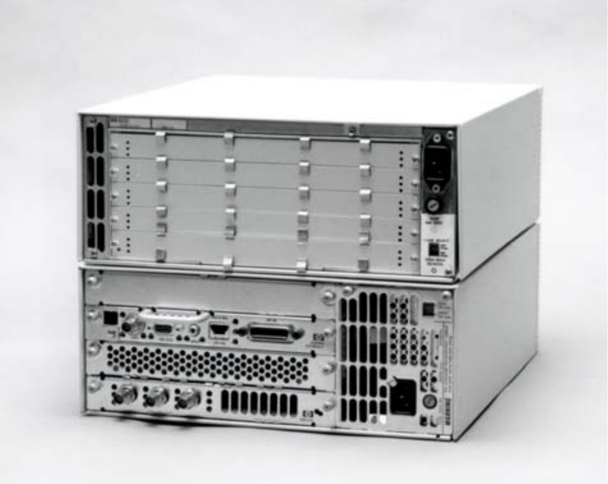


Figure 8-59 Rear view of VMEbus expander (top) for Hewlett Packard HP 350 computer (bottom)

Units like this allow users to add VMEbus modules to proprietary architectures. Photo courtesy of Hewlett Packard.

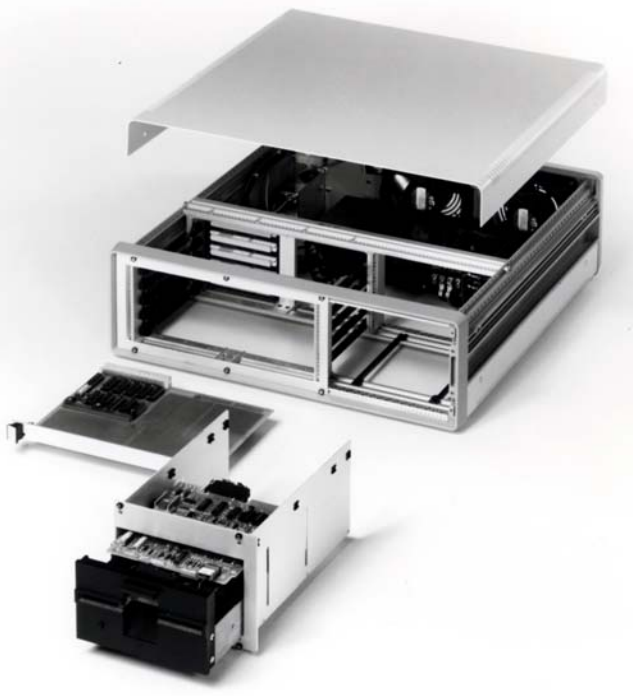


Figure 8-60 Eurocard modular construction

Photo courtesy of Schroff, Inc.

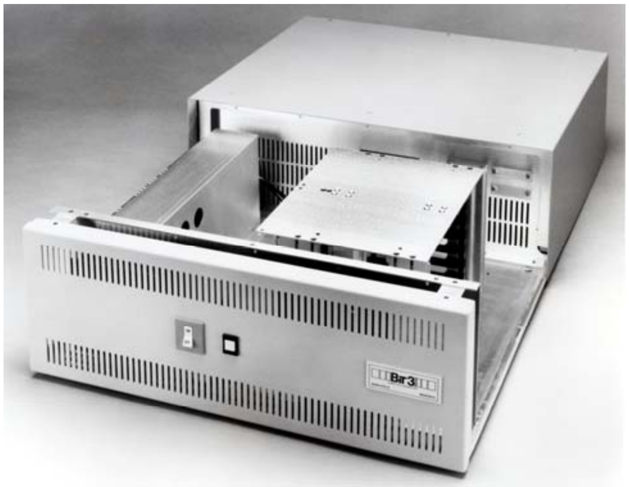


Figure 8-61 7 slot, 6U desk top chassis with cooling fans, power supply and cabling.

Photo courtesy of Bit-3 Computer Corporation.

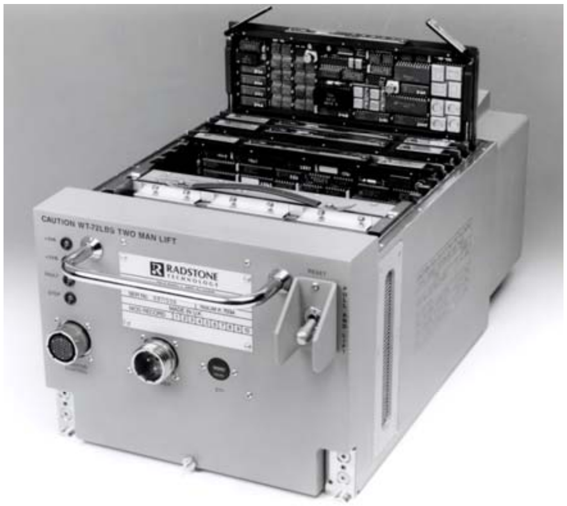


Figure 8-62 6U conduction cooled ATR (Air Transport Rack) system

Photo courtesy of Radstone Technology PLC.

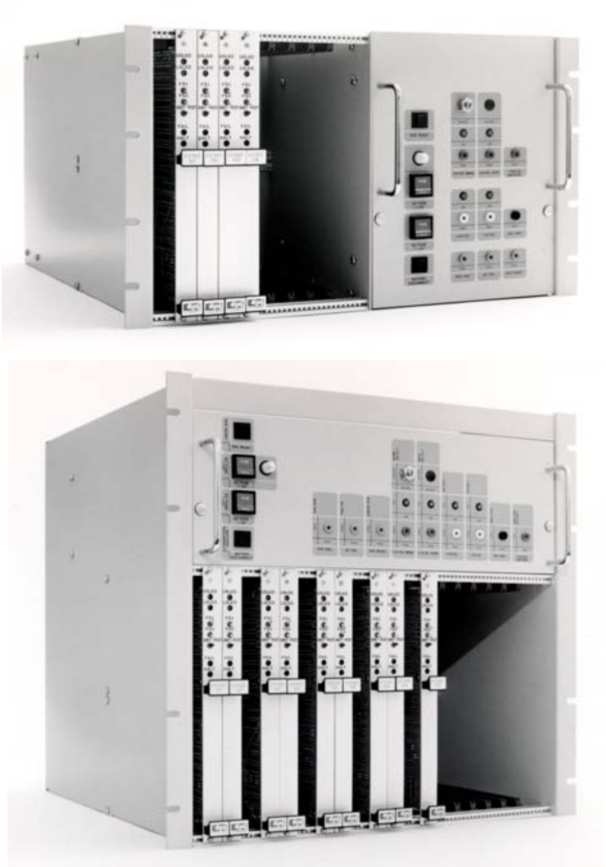


Figure 8-63 9 and 20 slot 6U VMEbus chassis with power supply

Combination of custom and off-the-shelf packaging makes for rugged, functional and good looking package.

Photo courtesy of DY-4 Systems.

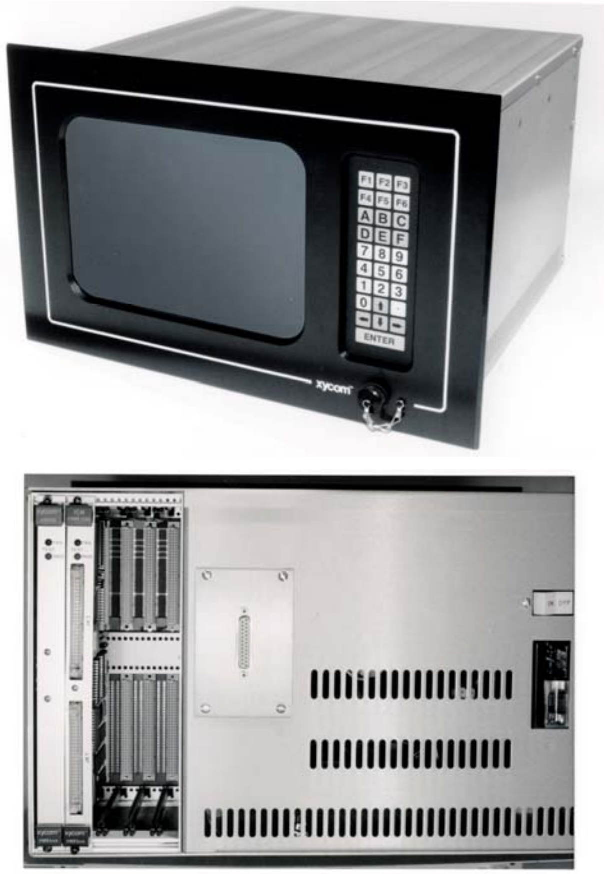


Figure 8-64 Front and rear view of a 19" rack mount NEMA terminal

With integral 5 slot VMEbus backplane, keypad, power supply and cabling. This package allows the user to integrate a sophisticated industrial control system into a single package.

Photo courtesy of Xycom Inc.

8.12 References

ATM Cells Bus (ACB) on VME / VITA 22-199x (Draft 0.1) VITA, Scottsdale, AZ 1997

Boioli, Roberto et al. VME-Multibus II Interface Adapter For Protocol Conversion And For Monitoring And Discriminating Accesses On The Multibus II System Bus US Patent No.: 5,218,690 1993

Chester, Michael. "VMEbus Standard Extension for Board Instruments" *Electronic Products*. September 1, 1987

Common Mezzanine Card Family: CMC / IEEE P1386 (Draft 2.0) IEEE, Piscataway, NJ 1995

Conduction Cooled PCI Mezzanine Card (CCPMC) / VITA 20-199x (Draft 1.3) VITA, Scottsdale, AZ 1997

Electric Current Abroad, 1984 Edition, US Department of Commerce, Bureau of Industrial Economics 1984

FPDP Specification / VITA 17-199x (Draft 1.3) VITA, Scottsdale, AZ 1997

Ginsberg, Gerald L. Electronic Equipment Packaging Technology Van Nostrand Reinhold, New York, NY 1992

Gish, Kevin S. et al. VME Slave Tester US Statutory Invention Registration No. H1444 1995

IP Modules / VITA 4-1995 (Draft 1.0.d.0) VITA, Scottsdale, AZ 1995

IP I/O Mapping to VME64x / VITA 4.1-199x (draft 0.7a) VITA, Scottsdale, AZ 1996

Korn, Heidi. "Military VME: A Design Study," *Proceedings of the Buscon/88 East Conference* CMC Norwalk, CT October 1988

Larson, Larry L. and Vukovic, Philip M. "The PC And VMEbus Or Multibus Processor," *Control Engineering*, June 1987.

Larson, Larry L. and Vukovic, Philip M. "Pooling The Resources Of Different Computing Systems" *Electronic Products* Dec 15, 1987

Lawler, Edward P. Circuit For Controlling Data Transfer From SCSI Disk Drive To VMEbus US Patent No. 5,410,678 1995

Maresh, Anthony J. et al. Computer Bus Interconnection Device. US Patent No. 5,083,259 1992

MacKenna, Craig. "VMS: How This Serial Bus Works - And How To Use It" *VMEbus Systems* Jan-Feb 1987

McKim, Dale W. I/O Interface Between VMEbus And Asynchronous Serial Data Computer. US Patent No. 5,414,814 1995

MVMX32bus Design Specification / Revision 0 Motorola Semiconductor Products Inc., Phoenix, AZ August 1985

MXbus II SideCard Interface Specification Mizar Inc. 1992

Patterson, Doug. "Board and System Issues In Militarized and Mil-spec Applications" *Proceedings of the Buscon/88 East Conference* CMC Norwalk, CT USA. October 1988, pp 353-357.

PCI Local Bus Specification / Revision 2.0 PCI Special Interest Group 1993

Peterson, Wade D. "VME64: Taking The VMEbus Architecture Into The 21st Century" *VMEbus Systems* August 1996

Peterson, Wade D. "VME64x: The VME64 Extensions Standard" *VMEbus Systems* August 1997

Physical and Environmental Layers for PCI Mezzanine Cards: PMC / IEEE P1386.1 (Draft 2.0). IEEE, Piscataway, NJ 1995

Schroff VMEbus Manual of Backplanes Schroff GMBH, Pforzheim West Germany 1987

Tobias, Michael R. et al. Byte Swapping Apparatus For Selectively Reordering Bytes Of An N-Bit Word Communicated Between An AT Computer And VMEbus US Patent No. 5,265,237 1993

Townsend, Bruce L. et al. Voice Bus For A Computer Backplane US Patent No. 5,495,583 1996

Van Doren, Thomas P. Grounding and Shielding Electronic Instrumentation University of Missouri - Rolla. 1988

VMSbus Specification Manual / Revision B1 VITA, Scottsdale, AZ 1985

VMXbus Specification Manual / Revision B VITA, Scottsdale, AZ 1984

VSB Specification Manual, Revision C / ANSI/IEEE Std. 1096-1988. VITA, Scottsdale, AZ 1988

Waltz, Eike. "On the Mechanical Reliability of VME Electronics Packaging" *VMEbus Systems*, Summer 1985

Chapter 9
Introduction to VXIbus

VMEbus Extensions for Instrumentation (VXIbus) is a superset of VMEbus, and is used for modular instruments. There are thousands of VXIbus cards available from scores of vendors. While most of these cards are oriented toward instrument applications, there are some tailored specifically for industrial and process control systems.

9.1 VXIbus History & Objectives

VXIbus was conceived in response to requests for instrumentation modules for VMEbus, particularly from the US Department of Defense. Reduced size of Automated Test Equipment (ATE) was a major factor driving the defense establishment toward VXIbus. Also, the high bandwidth of VMEbus was considered a benefit, particularly for digital test and digital signal processing applications. The United States Air Force, Navy and Army created programs addressing these issues are known as MATE, CASS and IFTE respectively.

The most serious impediment to VMEbus based instrumentation was the lack of standards beyond that of the VMEbus standard. In June of 1987, technical representatives from Colorado Data Systems, Hewlett Packard, Racal Dana, Tektronix and Wavetek formed an ad hoc committee to engineer an open instrumentation bus. The new bus used a collection of popular standards as its core. These included VMEbus, the Eurocard standard, IEEE 488 and several others. In July of 1987 they announced an agreement to support a common architecture of VMEbus Extension for Instrumentation, named the VXIbus.

The goal of VXIbus was to define a technically sound modular instrument standard that was based on VMEbus, open to all manufacturers and completely compatible with present industry standards. The VXIbus specification was intended to be used by designers interested in generating compatible components for systems. The writers of the first specification had the following objectives in mind:

- To allow communication among devices in an unambiguous fashion.
- To allow for physical size reduction of standard rack & stack instrumentation.
- To allow for software cost reduction by the use of common interfaces.
- To provide for higher system throughput for test systems.
- To provide test equipment which could be used in military test systems.
- To provide the ability for virtual instruments.
- To define the implementation of multi-module instruments.

By the end of 1987, Revision 1.1 of the VXIbus Systems Specifications was completed and standardization had begun through the IEEE P1155 working group. The five companies who had collaborated to define the specification signed a legal agreement creating the unincorporated VXIbus Consortium. The consortium provided a way for the competitors to work cooperatively and in accordance with the National Cooperative Research Act of 1984 (USA). The purpose of the consortium was to develop and maintain an open architecture for modular instruments. The consortium would not involve itself in the certification, definition or sale of products.

In 1988 the VXIbus consortium grew to ten members. In June 1988 the first operational multi-vendor VXIbus system was successfully demonstrated to the United States Air Force (who later adopted the bus as part of their MATE IAC Standard). That same summer, Revision 1.2 of the VXIbus specification was completed and adopted by the IEEE P1155 working group. Changes included in Revision 1.2 included semi-sync protocol timing, near field radiated emissions, power supply conducted emissions and other specification clarifications. Two new sections, Dynamic Configuration and Shared Memory Protocols were also added.

In 1989 the VXIbus consortium completed Revision 1.3 of the specification. Specific enhancements included changes to EMC and mechanical assemblies, system initialization, messages based device operation and the removal of the shared memory protocols.

The consortium eventually completed revision 1.4 of the specification. This was used as the basis for the current standard, known as the IEEE 1155-1992. In many ways, it is much better than the VMEbus specification because it covers more areas of system integration. The standard includes:

- Electromagnetic compatibility
- System power
- Device operation

- Communication protocols
- System resources
- Modular instruments
- IEEE 488 interfaces
- Dynamic configuration
- Implementation of the VMEbus specification (Revision C.1, Chapters 1-7)

9.2 VXIbus Mechanical Architecture

Like VMEbus, VXIbus uses Eurocard mechanical packaging. Figure 9-1 shows examples of bus modules and enclosures.

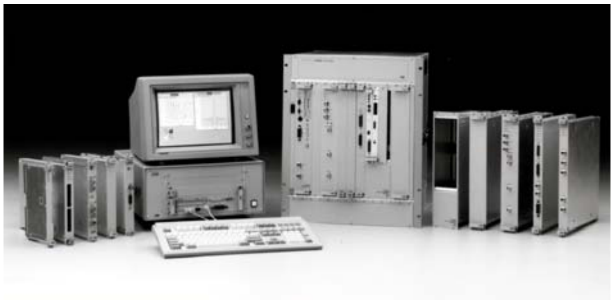


Figure 9-1 VXIbus system components

Shown are 'C' size modules (left), 'D' size modules (right) and the system mainframe (center). Note the standard VMEbus modules installed in the mainframe. Photo courtesy of Tektronix.

9.2.1 VXIbus Cards & Card Sizes

VXIbus specifies the four bus module sizes shown in Figure 9-2. The A and B modules are the same size as 3U and 6U VMEbus cards. The C and D sized modules are larger, and are standard Eurocard form factors.

Any size VXIbus module may be placed in a card cage of a larger size. For example, the A and B size cards may be used in a C or D mainframe. This allows the use of standard VMEbus modules, and provides for extremely flexible systems. Figure 9-3 shows a 'D' size VXIbus module with EMC shield.

The size of VXIbus instrument depends upon the application. Each of the connectors on the module provides the functions shown in Figure 9-4. The P1 and P2 connectors carry VMEbus signals. The outer two rows of the P2 connector (the VMEbus user defined pins) provide additional instrumentation signals for precision timing, triggering and module identification. Also included are an analog summing bus and a 12 pin local bus. The P3 connector has additional clock and triggering functions as well as a 24 pin local bus.

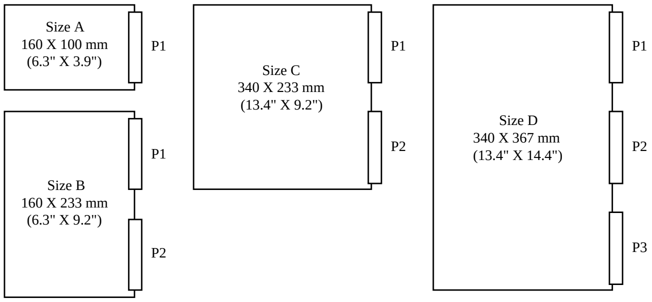


Figure 9-2 VXIbus card styles

Size 'C' is the most popular VXIbus module. Sizes A and B are the standard VMEbus card sizes.

Front panel lockout keys are required on VXIbus front panels if the module uses the local bus. These keys, shown in Figure 9-5, insure compatibility between adjacent modules. The P2 key, located at the top of the front panel, insures compatibility of the local bus signals located on the P2 connector. Similarly, the P3 key insures compatibility of signals located on the P3 connector.

9.2.2 VXIbus Mainframe

VXIbus modules are placed into a mainframe like the one shown in Figure 9-6. Mainframes contain a card cage, backplane, power supply and other system resources. The standard spacing between cards is 1.2" (VMEbus is 0.8"). This allows for a maximum of thirteen card slots in the VXIbus mainframe.

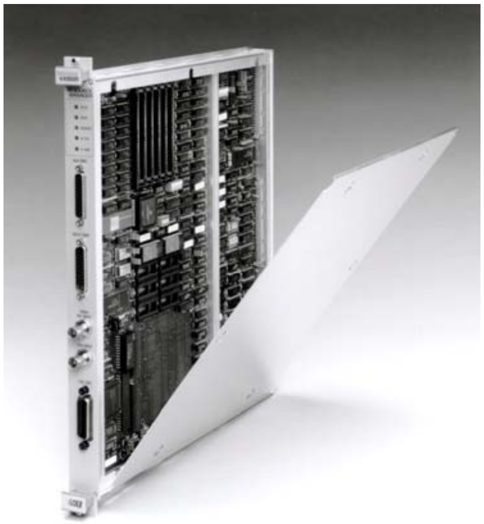


Figure 9-3 'D' size VXIbus module with EMC shield

Photo courtesy of Tektronix.

The 1.2" spacing allows for large analog components, daughter boards, shielding on both sides of the module and a chassis shield between modules. A module can be a printed circuit card, or an enclosed chassis assembly that contains several boards. Some instruments may use several slots if extra space is needed.

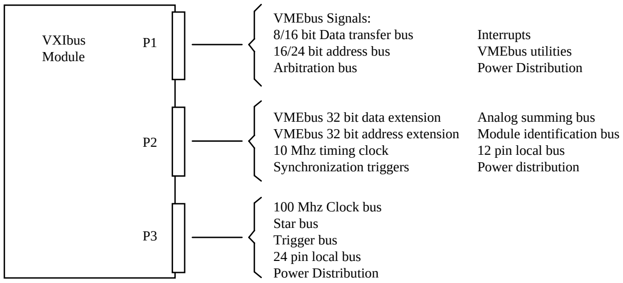


Figure 9-4 Functions located on the three VXIbus connectors

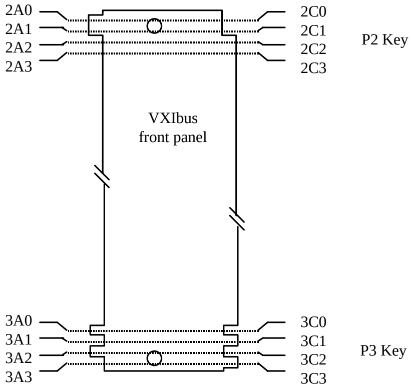


Figure 9-5 Lockout keys on VXIbus front panels

These keys are required on modules that use the local bus. They are used to prevent electrical damage to adjacent bus modules.

9.2.3 VXIbus Backplane

Both the 'C' and 'D' VXIbus backplane styles are available. These are shown in Figure 9-7. VXIbus backplanes are available with between one and thirteen card slots, with connectors located on 1.2" centers. The VXIbus spacing is wider than VMEbus (which is 0.8"), and allows for better EMI shielding and board cooling.

An active portion of the backplane, called the system resource manager, must be located in the first slot. This is called the slot 0 backplane position. This board provides bus arbitration, system reset, bus timer, timing functions and module ID control.

Figure 9-6 VXIbus mainframe

Photo courtesy of Schroff Inc.

Also note that the VXIbus card slots begin numbering at zero, and that VMEbus card slots begin numbering at one.

Active buffers for the ECL compatible CLK10, CLK100 and SYNC100 signals are located on the backplane. These are used to increase timing precision.

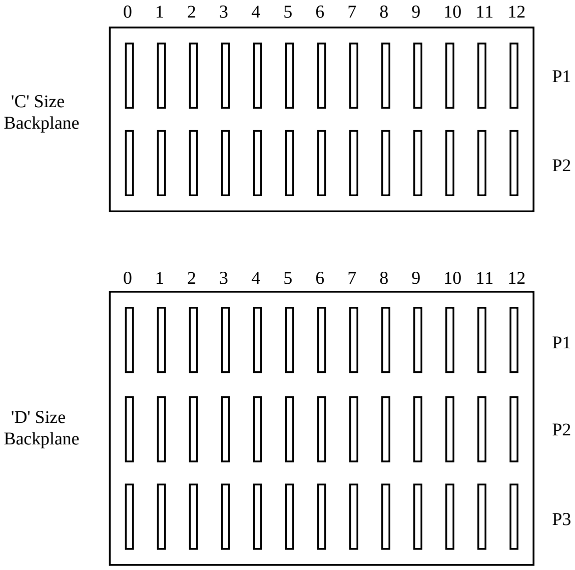


Figure 9-7 'C' and 'D' size, thirteen slot VXIbus backplanes

The VXIbus backplane specification provides for optional connector shields. Connector shields surround the DIN 41612 connectors, and prevent electromagnetic leakage through the rear of the instrument. The shield enclosing the VXIbus instrument contacts with the connector shield, and allows for a tight EMC enclosure. The connector shield is tied to circuit ground on the backplane.

9.2.4 VXIbus System Cooling

The VXIbus specification requires that forced air must enter the bottom of the module, and exit from the top. Module and mainframe cooling capability must be included in product specifications in order to determine cooling interoperability between mainframes and modules. VXIbus does not specify air flow in or out of the VXIbus mainframe. The external air entry and exit is not specified.

9.2.5 VXIbus Electromagnetic Compatibility (EMC)

The VXIbus Specification has many requirements regarding the electromagnetic compatibility (EMC) of modules. Included are conducted emissions, conducted susceptibility, radiated emissions and radiated susceptibility. Suggested test methods are also given.

9.3 VXIbus Electrical Architecture

As a minimum, each module must connect to the P1 connector. If extra functions are needed, it may also use the P2 and P3 connectors.

The VMEbus specification defines all the pins of P1 and the center row pins of P2. The P2 outer row pins are user defined. VXIbus defines the outer rows of P2 and completely defines P3. Tables 9-2 to 9-5 define the pinouts for the VXIbus P2 and P3 connector. The P1 connector is identical to the VMEbus P1 connector, with pinouts given in Chapter 1 of this book.

The minimum VXIbus configuration needs only P1 to operate. P1 is the VXIbus backbone that provides a VMEbus microcomputer bus. The P2 and P3 connectors expand the VXIbus system to include new voltages, additional power, a sub-system bus, system synchronization, a dedicated analog bus and automatic configuration capability.

VXIbus can be logically grouped into eight sub-buses and a few reserved pins. These sub-buses are given in Table 9-1.

Table 9-1 The VXIbus sub-buses

Sub bus	Type
VMEbus	Global
Trigger bus	Global
Analog sum bus	Global
Power distribution bus	Global
Clock & synchronization bus	Unique
Star bus	Unique
Module identification bus	Unique
Local bus	Private

Each sub-bus adds a new dimension of instrumentation capability. The bus type strongly affects how to best use the bus in an automated test environment. Global buses are accessible by every VXIbus module. Unique buses go from the Slot 0 Resource Manager to individual bus modules on a one-to-one basis. Private buses are local buses between adjacent modules.

Table 9-2 VXIbus P2 connector, slot 0

VXIbus P2/J2 Connector, Slot 0			
Pin Number	Row A	Row B	Row C
1	ECLTRG0	+5 VDC	CLK10+
2	-2V	GND	CLK10-
3	ECLTRG1	RSV1	GND
4	GND	A24	-5.2V
5	MODID12	A25	LBUSC00
6	MODID11	A26	LBUSC01
7	-5.2V	A27	GND
8	MODID10	A28	LBUSC02
9	MODID09	A29	LBUSC03
10	GND	A30	GND
11	MODID08	A31	LBUSC04
12	MODID07	GND	LBUSC05
13	-5.2V	+5 VDC	-2V
14	MODID06	D16	LBUSC06
15	MODID05	D17	LBUSC07
16	GND	D18	GND
17	MODID04	D19	LBUSC08
18	MODID03	D20	LBUSC09
19	-5.2V	D21	-5.2V
20	MODID02	D22	LBUSC10
21	MODID01	D23	LBUSC11
22	GND	GND	GND
23	TTLTRG0*	D24	TTLTRG1*
24	TTLTRG2*	D25	TTLTRG3*
25	+5V	D26	GND
26	TTLTRG4*	D27	TTLTRG5*
27	TTLTRG6*	D28	TTLTRG7*
28	GND	D29	GND
29	RSV2	D30	RSV3
30	MODID	D31	GND
31	GND	GND	+24V
32	SUMBUS	+5 VDC	-24V

Table 9-3 VXIbus P2 connector, slots 1 - 12

VXIbus P2/J2 Connector, Slots 1 - 12			
Pin Number	Row A	Row B	Row C
1	ECLTRG0	+5 VDC	CLK10+
2	-2V	GND	CLK10-
3	ECLTRG1	RSV1	GND
4	GND	A24	-5.2V
5	LBUSA00	A25	LBUSC00
6	LBUSA01	A26	LBUSC01
7	-5.2V	A27	GND
8	LBUSA02	A28	LBUSC02
9	LBUSA03	A29	LBUSC03
10	GND	A30	GND
11	LBUSA04	A31	LBUSC04
12	LBUSA05	GND	LBUSC05
13	-5.2V	+5 VDC	-2V
14	LBUSA06	D16	LBUSC06
15	LBUSA07	D17	LBUSC07
16	GND	D18	GND
17	LBUSA08	D19	LBUSC08
18	LBUSA09	D20	LBUSC09
19	-5.2V	D21	-5.2V
20	LBUSA10	D22	LBUSC10
21	LBUSA11	D23	LBUSC11
22	GND	GND	GND
23	TTLTRG0*	D24	TTLTRG1*
24	TTLTRG2*	D25	TTLTRG3*
25	+5V	D26	GND
26	TTLTRG4*	D27	TTLTRG5*
27	TTLTRG6*	D28	TTLTRG7*
28	GND	D29	GND
29	RSV2	D30	RSV3
30	MODID	D31	GND
31	GND	GND	+24V
32	SUMBUS	+5 VDC	-24V

Table 9-4 VXIbus P3 connector, slot 0

VXIbus P3/J3 Connector, Slot 0			
Pin Number	Row A	Row B	Row C
1	ECLTRG2	+24V	+12V
2	GND	-24V	-12V
3	ECLTRG3	GND	RSV4
4	-2V	RSV5	+5V
5	ECLTRG4	-5.2V	RSV6
6	GND	RSV7	GND
7	ECLTRG5	+5V	-5.2V
8	-2V	GND	GND
9	STARY12+	+5V	STARX01+
10	STARY12-	STARY01-	STARX01-
11	STARX12+	STARX12-	STARY01+
12	STARY11+	GND	STARX02+
13	STARY11-	STARY02-	STARX02-
14	STARX11+	STARX11-	STARY02+
15	STARY10+	+5V	STARX03+
16	STARY10-	STARY03-	STARX03-
17	STARX10+	STARX10-	STARY03+
18	STARY09+	-2V	STARX04+
19	STARY09-	STARY04-	STARX04-
20	STARX09+	STARX09-	STARY04+
21	STARY08+	GND	STARX05+
22	STARY08-	STARY05-	STARX05-
23	STARX08+	STARX08-	STARY05+
24	STARY07+	+5V	STARX06+
25	STARY07-	STARY06-	STARX06-
26	STARX07+	STARX07-	STARY06+
27	GND	GND	GND
28	STARX+	-5.2V	STARY+
29	STARX-	GND	STARY-
30	GND	-5.2V	-5.2V
31	CLK100+	-2V	SYNC100+
32	CLK100-	GND	SYNC100-

Table 9-5 VXIbus P3 connector, slots 1 - 12

VXIbus P3/J3 Connector, Slots 1 - 12			
Pin Number	Row A	Row B	Row C
1	ECLTRG2	+24V	+12V
2	GND	-24V	-12V
3	ECLTRG3	GND	RSV4
4	-2V	RSV5	+5V
5	ECLTRG4	-5.2V	RSV6
6	GND	RSV7	GND
7	ECLTRG5	+5V	-5.2V
8	-2V	GND	GND
9	LBUSA12	+5V	LBUSC12
10	LBUSA13	LBUSC15	LBUSC13
11	LBUSA14	LBUSA15	LBUSC14
12	LBUSA16	GND	LBUSC16
13	LBUSA17	LBUSC19	LBUSC17
14	LBUSA18	LBUSA19	LBUSC18
15	LBUSA20	+5V	LBUSC20
16	LBUSA21	LBUSC23	LBUSC21
17	LBUSA22	LBUSA23	LBUSC22
18	LBUSA24	-2V	LBUSC24
19	LBUSA25	LBUSC27	LBUSC25
20	LBUSA26	LBUSA27	LBUSC26
21	LBUSA28	GND	LBUSC28
22	LBUSA29	LBUSC31	LBUSC29
23	LBUSA30	LBUSA31	LBUSC30
24	LBUSA32	+5V	LBUSC32
25	LBUSA33	LBUSC35	LBUSC33
26	LBUSA34	LBUSA35	LBUSC34
27	GND	GND	GND
28	STARX+	-5.2V	STARY+
29	STARX-	GND	STARY-
30	GND	-5.2V	-5.2V
31	CLK100+	-2V	SYNC100+
32	CLK100-	GND	SYNC100-

9.3.1 VXIbus Clock Bus

The clock bus provides two clocks and a synchronization signal. A 10 MHz clock (CLK10) is located on the P2 connector, and a 100 MHz clock (CLK100) with a synchronization signal (SYNC100) is located on the P3 connector. CLK10, CLK100 and SYNC100 are differential ECL signals, and are located at the ends of the P2 and P3 connectors. They are isolated as much as possible with AC grounds (GND, +5V, -2V, etc.) to minimize noise jitter. The CLK10 and CLK100 signals are synchronized with each other.

Both clocks and SYNC100 are sourced from the Slot 0 resource module, and are buffered on the backplane. Figure 9-8 shows interconnect scheme for the CLK10 signal.

The reason for the backplane buffers is to provide a high degree of inter-module isolation, and to relax the clock loading rules for the modules. A common problem with high precision timing in backplane systems is that the loading on each clock signal changes with the number of modules installed in the system. Buffers permit each clock signal to be loaded identically, regardless of the number of modules in the system. For example, CLK10 timing is guaranteed to be +/- 8 ns between slot 0 and any other module. Between any two modules it is guaranteed to be +/- 2 ns.

For example, analog-to-digital conversion (A/D) performance is dependent on the A/D's clock phase jitter. The effect of the other modules on the A/D's measurement performance is eliminated with the VXIbus clock isolation as compared to a typical single line clock architecture.

Some VXIbus Slot 0 modules permit the CLK10 signal to be derived from an external source such as a rubidium standard. This permits exact timing, even among multiple VXIbus mainframes.

The CLK100 signal provides an even higher level of timing performance (100 MHz). The SYNC100 signal is used in conjunction with CLK100 to synchronize multiple modules with respect to a given CLK100 rising edge. This provides very tight time coordination between modules, and permits high performance simultaneous or sequential measurements on multiple boards.

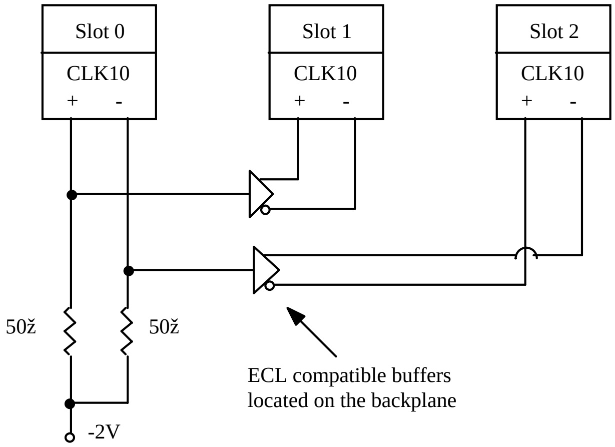


Figure 9-8 VXIbus CLK10 interconnection scheme

Similar to the CLK100 interconnect.

The VXIbus CLK100 signal has the tightest synchronization tolerances of any of the industry standard backplane buses.

9.3.2 VXIbus Star Bus

Located on the P3 connector is the high speed star bus. The star bus is composed of two star lines (STARX & STARY) that are connected between each module slot and the Slot 0 resource manager as shown in Figure 9-9. STARX and STARY are bi-directional, differential ECL compatible lines. Slot 0 can be viewed as the center of a star fish with twelve legs. Each leg has a module at the end of the leg with a STARX and a STARY running the length of the leg.

STARX and STARY have maximum timing skews of 2.0 ns. between any two STAR signals, and a maximum delay of 5.0 ns. between slot 0 and any module.

The star bus provides high speed communication between modules. This communication may be a clock with a synchronization signal or various forms of triggering. For example, a high speed clock and a start/stop signal can be acquired by any module and routed to the slot 0 module. The slot 0 module can provide a cross matrix switch that can selectively route the clock signals to other modules like A/Ds, D/As and logic analyzers. This tightly couples the timing signals.

9.3.3 VXIbus TTL Trigger Bus

The VXIbus trigger bus can be subdivided into eight TTL trigger lines (TTLTRG*) and six ECL trigger lines (ECLTRG). All of the TTLTRG* lines and two of the ECLTRG lines are located on the P2 connector. The four remaining ECLTRG lines are on P3.

The VXIbus trigger bus is used for intermodule communication. Any module, including slot 0, may drive and receive information on these lines. They are general purpose, and may be used for triggering, handshaking, clocking or data transmission. Figure 9-10 shows how the bus modules are interconnected by the trigger bus.

The trigger lines are held in the negated state by the backplane terminators until allocated by the user's program and asserted by a VXIbus module. The primary difference between the TTLTRG* and ECLTRG lines is speed of operation. The maximum clock speed for TTLTRG is 12.5 MHz as compared to 62.5 MHz for ECLTRG.

All TTLTRG* signals are open collector TTL style logic, and are bussed the length of the backplane. The terminators are identical to VMEbus.

In TTL timing transmissions it is recommended that the falling edge be used instead of the rising edge. This is because open collector rise times are greater than fall times, especially in fully loaded systems. The reason for the difference between rise and fall times is that open collector TTL signals are pulled high by the backplane termintors. Any capacitance on the signal lines causes them to rise at a slow rate.

Users can define their own protocols, or use one of six defined by the VXIbus specification. The six defined protocols include TTLTRG* synchronous trigger, TTLTRG* semi-synchronous trigger, TTLTRG* asynchronous trigger, TTLTRG* clock transmission, TTLTRG* data transmission and TTLTRG* start/stop.

9.3.3.1 VXIbus TTLTRG* Synchronous (SYNC) Trigger Protocol

The SYNC trigger protocol is a single line broadcast trigger that does not require an acknowledge from any acceptor. The maximum trigger repetition rate is 12.5 MHz.

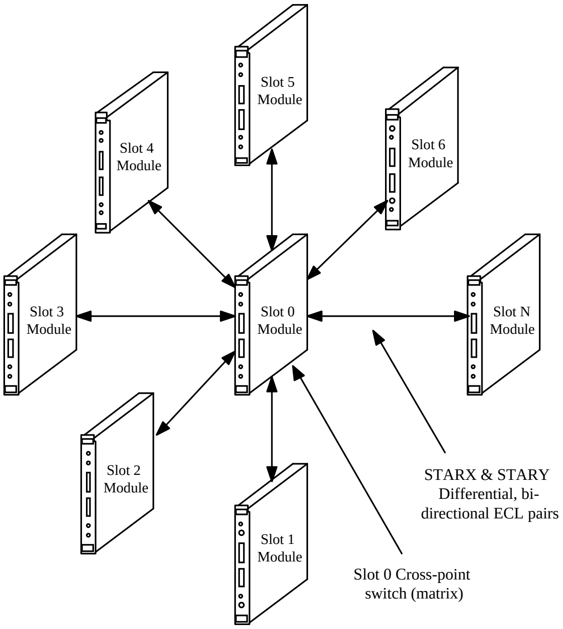


Figure 9-9 The VXIbus Star bus

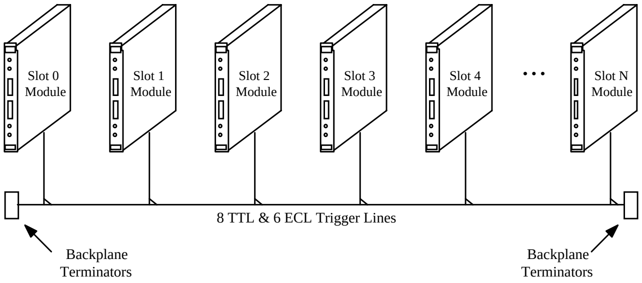


Figure 9-10 VXI Trigger Bus

9.3.3.2 VXIbus TTLTRG* Semi-synchronous (SEMI-SYNC) Trigger Protocol

The SEMI-SYNC trigger protocol is a single line broadcast, multiple acceptor handshake protocol. A single source asserts the line, and if needed the acceptors assert the same line and hold it until ready for the next operation. Using the wire-or features of the open collector bus, the bus stays low until the output of the source and all acceptors becomes negated. This insures that all devices have acknowledged completion (with a handshake).

An advantage of the SEMI-SYNC protocol is that the source device and all acceptor devices use a single trigger line. Since the data transfer rate is determined by the slowest acceptor, no data is lost. The data rate will automatically increase whenever the slowest devices are not involved in the handshake. Given the fastest source and acceptors, the maximum handshake rate is 12.5 MHz.

For those who are familiar with the low level workings of the IEEE 488 bus, the SEMI-SYNC protocol is similar to the IEEE 488 data transfer using the talker's DAV (Data Valid) and the listener(s) NDAC (Not Data Accepted) lines.

9.3.3.3 VXIbus TTLTRG* Asynchronous (ASYNC) Trigger Protocol

The TTLTRG* ASYNC trigger protocol is a two line, single source, single acceptor protocol. Using a pair of TTLTRG* lines, the source initiates an operation by asserting the lower numbered TTLTRG* line, while the acceptor acknowledges on the higher numbered TTLTRG* line as shown in Figure 9-11. This is a useful trigger mode for handshaking between VXIbus modules and external instruments or VXIbus systems. Under ideal conditions, the maximum trigger repetition rate is 12.5 MHz.

9.3.3.4 VXIbus TTLTRG* Clock Transmission

A TTLTRG* line can be used for clock transmission from 0 Hz to 12.5 MHz.

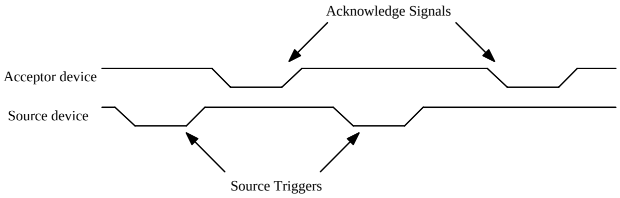


Figure 9-11 VXIbus asynchronous trigger protocol (ASYNC) used by TTLTRG* and ECLTRG

9.3.3.5 VXIbus TTLTRG* Data Transmission

One or more TTLTRG* lines can be grouped together for parallel data transmission. One of these lines is used as a clock and the other for data transfer. The data may be synchronized to either the rising or falling edge of the clock (or both).

For example, using eight TTLTRG* lines, one for falling edge clocking and the other seven for data transfer, the VXIbus subsystem maximum data transfer rate is 12.5 million (seven bit) bus cycles per second.

9.3.3.6 VXIbus TTLTRG* Start/stop (STST) Protocol

The STST trigger protocol provides a method for starting and stopping module clusters synchronously. One TTLTRG* line is driven from the Slot 0 resource manager and its state signifies the start or stop operation to the other modules. All participating modules respond to this line synchronously at the next CLK10 rising edge.

9.3.4 VXIbus ECL Trigger Bus

The ECLTRG lines are bussed the length of the VXIbus backplane and are terminated. All of the ECLTRG lines are located at the end of the 'A' rows of the P2 and P3 connectors, and are isolated with AC grounds (GND, +5V, -2V, etc.) to minimize jitter noise. Any module, including the Slot 0 resource manager, may drive and receive information from these lines. All are ECL compatible, and have 50Ω termination networks at each end. The asserted state is defined as logical HIGH.

9.3.4.1 VXIbus ECLTRG Protocols

Users can define their own ECLTRG protocols, or use one of the six defined by the VXIbus specification. The six defined protocols include:

- ECLTRG synchronous trigger (62.5 MHz)
- ECLTRG semi-synchronous trigger (50 MHz)
- ECLTRG asynchronous trigger (62.5 MHz)
- ECLTRG clock transmission (62.5 MHz)
- ECLTRG data transmission (62.5 MHz)
- ECLTRG extended start/stop (ESTST).

All of the trigger modes are analogous to the TTLTRG* protocols, except for the ECLTRG extended start/stop protocol.

9.3.4.2 VXIbus ECLTRG Extended Start/stop protocol (ESTST)

In the extended start/stop protocol, the ECLTRG lines are used in conjunction with the STST lines to synchronize P2 and P3 modules. All ESTST timing is synchronized to the CLK100 and the SYNC100 signals, and are qualified by an ECLTRG line. One ECLTRG line informs an ESTST module which SYNC100 signal is valid. The ESTST module still uses the TTLTRG* to determine start or stop operation when triggered by a valid CLK100 edge.

The benefit of the ESTST protocol is the fixed time relationship that exists between the P2 STST and the P3 ESTST modules. This is because CLK10 and CLK100 are synchronized, and the Slot 0 module can lock the CLK100 triggering to the CLK10 triggering in a fixed relationship. As a result, higher speed P3 instruments can be tightly coupled to lower speed P2 instruments.

9.3.5 VXIbus Local Bus

The local bus is a private communication link between adjacent modules. As shown in Figure 9-12, most modules have two separate local buses, one to the left module and one to the right. The exceptions are the slot 0 and slot 12 modules (which are located at the ends of the backplane).

Each 'D' size module has 36 local bus lines on each side of the bus module. Twelve of the lines are located on the P2 connector, and 24 on the P3. Each signal on the local bus is rated at a maximum voltage of +/- 42V, and a maximum current of 500 mA. Each pin is classified using the ranges shown in Table 9-6.

Table 9-6 Classes of local bus pins

Class	-V Limit	+V Limit
TTL	-0.50V	5.50V
ECL	-5.46V	0.00V
ANALOG LOW	-5.50V	5.50V
ANALOG MED	-16.0V	16.0V
ANALOG HIGH	-42.0V	42.0V

The purpose of the local bus is to eliminate the need for front panel cable jumpers. This makes system integration and repair much simpler.

A good application for the local bus is a digital signal processing system (DSP). An A/D converter, located in slot 1, samples data and passes it to a DSP module in slot 2. The DSP processes the data and passes it to a D/A converter in slot 3. The local bus allows private, intermodule communication without affecting VMEbus traffic, and results in a higher system performance (as compared to using VMEbus for data transfers).

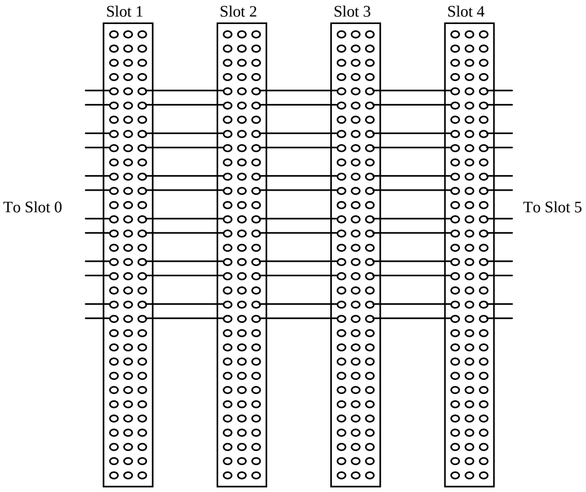


Figure 9-12 VXIbus P2 local bus backplane connections

9.3.6 VXIbus Analog SUMBUS

The VXIbus SUMBUS is an analog summing node that is connected to all instruments. It can be used both as an analog broadcast network and as a complex waveform generator.

The backplane layout of the SUMBUS is designed so that it is located away from digital and other active signals. The SUMBUS is located at the end of the P2 connector (pin A32), and is shielded by three AC grounds. It is terminated with 50Ω resistors at each end of the backplane. Clamping networks prevent voltage excursions from exceeding +/- 3.0 Volts.

It is intended that any module be able to drive or receive from the SUMBUS line. Each module can drive this line using an analog current source driver, and receive through a high impedance receiver. This permits complex waveform generation as shown in the example of Figure 9-13. In this example two modules (A & B) drive the SUMBUS line with complex waveforms. The two waveforms add together to form a third SUMBUS waveform, which can be monitored by any bus module. The technique is analogous to digital wire-or circuits.

9.3.7 VXIbus Module Identification Bus

The module identification bus (MODID01 - MODID12) can be used to identify the slot number of a VXIbus instrument. It is used for automatic system configuration, and for determining the location of failed bus modules.

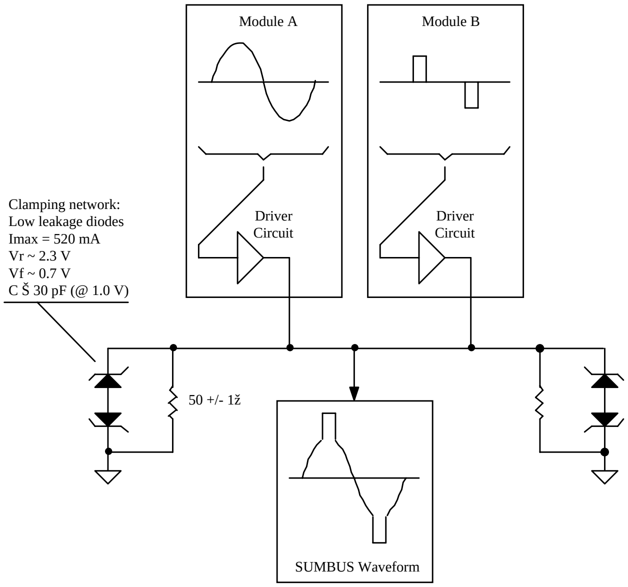


Figure 9-13 Example showing SUMBUS terminators and waveform generation

The SUMBUS signal is the summation of waveforms generated by one or more VXIbus modules. The feature is useful in many instrumentation systems that require complex waveform generation.

The twelve MODID lines are sourced from the Slot 0 module and are individually routed to slots 1 through 12 as shown in Figure 9-14. Each module connects its MODID line to the MODID* bit in its Status/control Register. The presence of a module in any slot can be detected by asserting a MODID line, and checking the Status/control Register of the same module. When the asserted MODID line and Status/control Register agree, the module's location is found.

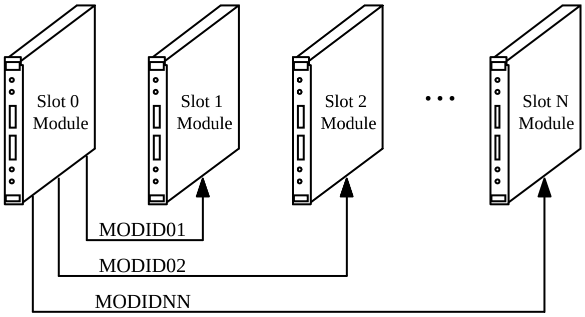


Figure 9-14 VXIbus module identification bus

The MODID bus can be used to identify the slot number of a VXIbus instrument. It is used for automatic configuration of the system, and for determining the location of failed bus modules.

9.3.8 VXIbus Power Distribution

Each 'D' size VXIbus module can consume up to 268 watts of power. This power is delivered from the backplane as seven different regulated voltages. Table 9-7 summarizes the power available on each connector.

The VMEbus specification defines +5 VDC, + 5VSTDBY, +12 VDC and -12 VDC on the P1 and P2 connectors. The VXIbus specification adds +24V, -24V, - 5.2V and -2V to the P2 and P3 connectors. +/- 24V is usually used for analog signals, -5.2V and -2V are used for ECL logic. These voltages meet most instrumentation needs. Lower voltages like +/- 15V can be obtained from the +/- 24V power supplies with on-board regulators or DC-DC converters.

9.4 VXIbus Software Architecture

The VXIbus architecture allows a wide range of instruments, interface cards and computers from different manufacturers to coexist in the same card cage. VXIbus does not define a specific system hierarchy or topology, nor does it specify the type of microprocessor, operating system or interface to a host computer. VXIbus does define a foundation for compatibility between different manufacturers.

There are many topologies possible within the VXIbus architecture. These include single CPU systems, multiple CPU systems with distributed intelligence and stand alone mainframes. Each of these topologies have different communication needs which are met by a layered set of communication protocols as shown in Figure 9-15.

Table 9-7 VXIbus power distribution for A, B, C and D size modules

Voltage	P1		P1 & P2		P1 & P2 & P3	
	# Pins	Watts	# Pins	Watts	# Pins	Watts
GND	8	-	26	-	40	-
+5	3	15W	7	35W	12	60W
+5STDBY (§)	1	7.5W	1	7.5W	1	7.5W
+ 12	1	12W	1	12W	2	24W
- 12	1	12W	1	12W	2	24W
+ 24			1	24W	2	48W
- 24			1	24W	2	48W
- 5.2			5	26W	10	52W
- 2			2	4W	6	12W

(§) Used when main system power is off.

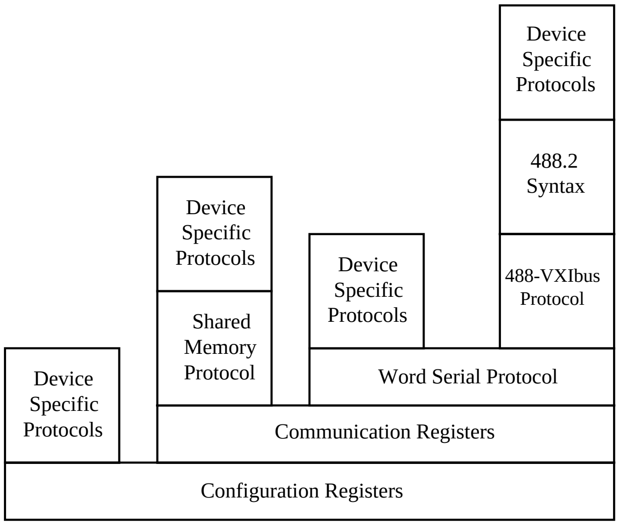


Figure 9-15 VXIbus communication layers

In order to support automatic system and memory configuration, the VXIbus standard specifies a minimum capability on each device regardless of its function. A VXIbus device, being the lowest level of the communication hierarchy, has a set of configuration registers that are accessible from the P1 connector. These registers allow the system to identify the device, its class, model, manufacturer, address space, memory requirements and other housekeeping data. VXIbus instruments with only this minimum level of capability are called *register based devices*.

VXIbus also defines *memory based devices* that can be identified as RAM, ROM or other memory types. This allows contiguous blocks of memory to be defined (based on speed or memory type).

If a higher level of communication is desired in the system, VXIbus defines a class of devices called *message based devices*. In addition to the configuration registers mentioned above, message based devices are required to have communication registers that are accessible to other modules in the system. Each device in the system can then use specific communication protocols such as the VXIbus Word Serial Protocol to communicate with other devices.

Communication between VXIbus instruments is based on a hierarchical device relationship involving *commanders* and *servants*. Commanders are devices capable of communicating with servants. If the servants are message based devices, such communication can take place using commands which are described by the word serial protocols. Communication with register based servants is device dependent, and may vary between systems. The commander/servant hierarchy can be nested since a message based device may be a commander as well as a servant (to the next level above it).

Instruments may share an interface with a host computer. An IEEE-488 VXIbus interface is an example of such a device. VXIbus defines this as a function specific message based device with communication protocols that support IEEE-488 functions. This device may be shared by the instruments needing to be accessed by a host computer. Other interfaces (such as RS-232 or Ethernet™) can be implemented in a similar fashion.

The VMEbus multi-master architecture must be supported regardless of the topology of the VXIbus system. This requires a minimal amount of system related functions that are to be performed by the slot 0 resource manager. The following are some of the functions a resource manager may need to perform:

- Address map configuration. This configures the A24 and A32 address maps and assigns blocks of addresses to the devices requiring them.
- System hierarchy. In multiprocessor systems the potential exists for two or more processors to control the same device. The commander/servant hierarchies determined by the resource manager prevent this type of conflict.
- Allocating shared system resources. The resource manager sets up shared blocks of memory for communication or other hardware resources (such as trigger buses).
- Self test and diagnostics.
- Initialization of all commanders.

9.4.1 VXIbus Device Operation

Devices are the lowest logical component in the VXIbus system. Normally a single VXIbus module will be a device. However, multi-board devices and multi-device boards are permitted. There is a unique logical address associated with each device in the system.

Examples of devices include computers, multimeters, multiplexers, oscillators, operator interfaces and counters. VXIbus devices may be categorized into four classes by their supported protocols: message based devices, register based devices, memory devices and extended devices. [Two other types called *hybrid devices* and *non-VXIbus devices*, refer to VMEbus compatible modules].

Message based devices support the VXIbus configuration and communication protocols. This category includes only devices with commander and/or command based servant elements. Message based devices are those with local intelligence that require some level of communication capability. Examples include digital multimeters, spectrum analyzers, display controllers and IEEE-488 interfaces.

Register based devices support VXIbus register maps, but no VXIbus communication protocols. Typically register based devices are simple, inexpensive modules such as switches, digital I/O, serial I/O or any other function that requires little or no local intelligence.

Memory devices have configuration registers, and contain certain attributes of the memory device such as type and access time, but have no other VXIbus defined register or protocols. RAM and ROM cards fit into this category.

Extended devices are special purpose VXIbus modules that have configuration registers so they can be identified by the system. This category of devices allows for the definition of future device classes.

Hybrid devices are VMEbus compatible modules that know about VXIbus devices, and have the ability to communicate with them or make use of them. They do not have to comply with the VXIbus device requirements. Existing VMEbus boards fit into this category.

Non-VXIbus devices are VMEbus modules that do not comply with or use any of the VXIbus requirements.

9.4.2 VXIbus Device Configuration

All VXIbus devices must have configuration registers located within the A16 address space. These registers are D16 compatible, with the contents shown in Figure 9-16. The base address of the configuration registers are placed at:

Base address = (V • 64) + 49152

where [V] is the logical address of the device. Logical addresses are between 0 and 255, and are set by a switch. Modules that support the optional dynamic configuration use a memory device for the logical address.

Modules that support dynamic configuration are required to set their logical address to 0xFF after power-up. Since multiple modules would have the same address after power-up (0xFFC0), the MODID select lines are used as address qualifiers. During power-up initialization each module is individually selected using the MODID lines, and the logical address is updated by writing to the base address occupied by logical address 0xFF. The module then immediately assumes the new logical address.

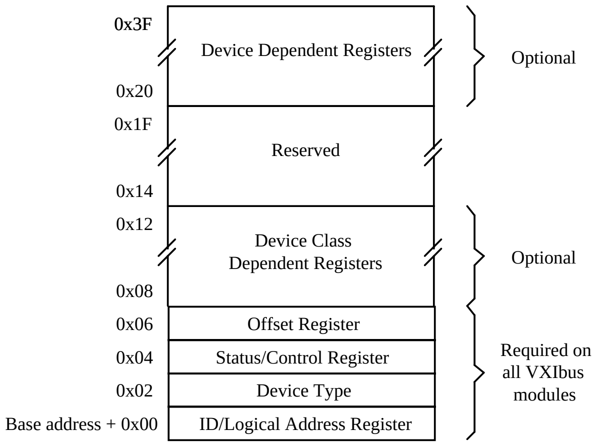


Figure 9-16 VXIbus configuration registers

The 32 VXIbus configuration registers allow a variety functions including:

- Logical address
- Manufacturer identification code
- Address space type (A16, A24, A32)
- Device class (register device, memory device, etc)
- Required memory size (if used)
- Module ID (MODID) line status
- Module self-test status
- VMEbus SYSFAIL* status and control
- Module reset
- Offset register (for selecting the base address of on-board memory)
- Device dependent registers (use determined by device class)

9.4.3 VXIbus Software Protocols

There are two general forms of VXIbus devices: register based and message based. Register based devices do not respond to any pre-defined protocols (the device's registers are entirely card dependent). Message based devices, however, operate with several pre-defined protocols.

In VXIbus terminology there are two types of instruments: *commanders* and *servants*. A *commander* is a message based device that is also a bus master, and can control one or more servants. A *servant* is a device that is controlled by a commander. There are message based and register based servants. This terminology has its roots in IEEE-488 (GPIB/HPIB) terminology.

Since register based devices are entirely card dependent, they are not discussed here (see the manufacturer of the bus module for details). Message based devices, however, use standardized protocols. They are generally capable of independently executing complex commands, and may also control other devices in a hierarchical instrumentation system. Message based servants communicate using the VXIbus message based device protocols.

Message based commanders have interfaces capable of exercising control over other devices. Message based commanders communicate using the VXIbus message based device protocols.

The protocols for commander/servant communication involve the servant's protocol, response, data and signal registers. These registers are a subset of the VXIbus device class dependent configuration registers. The simplest communications use the data and response registers to transfer data in a word serial fashion. This mode is defined as the *word serial* protocol.

The word serial protocol is the most basic method of communication defined for message based devices. It is simple to implement, and provides the required communication capability. Once communication is established between message based devices with the word serial protocol (and proper notification is given to all device), the devices may progress to the more sophisticated longword (32-bit) and extended longword (48-bit) serial protocols.

The serial protocols are based on a generalized model of a full duplex UART (Universal Asynchronous Receiver/Transmitter). Each implementation utilizes bi-directional data and response registers. The data registers are categorized as full duplex, because reads and writes are totally independent. Each write to the data registers is interpreted as a command, unless a previous command has defined it as data. Commands may contain embedded data or may require supporting data to be transferred in succeeding writes. Valid data written to or read from the registers are qualified with the write ready bit or the read ready bit in the response register.

For specific information regarding the VXIbus serial protocols, the user is encouraged to refer to the VXIbus Specification.

9.5 Using VMEbus Cards in VXIbus Mainframes

VMEbus modules can be used in VXIbus mainframes. However, there are several issues that must be resolved in order to achieve a successful system integration. Some of the more common issues include:

- VXIbus implementation of the VMEbus specification.
- Mechanical packaging ramifications.
- P2 user defined pins.
- Autoconfiguration protocol.
- Support of software (message based) protocols.

9.5.1 VXIbus Implementation Of The VMEbus Specification

The VXIbus specification uses a subset of the VMEbus specification for data transfer protocols. However, a subset of VMEbus protocols relating to data transfer, arbitration and interrupts has been specified. This may limit the utility of some standard VMEbus products in a VXIbus system.

Here are a few VMEbus options that are required under VXIbus:

- VXIbus requires that slaves must assert DTACK* or BERR* within 20 µS after a data strobe is asserted. It further requires that all slaves must negate DTACK* or BERR* within 5 µS after both data strobes are negated. Almost all VMEbus modules will conform to this requirement. If in doubt, however, refer to the manufacturer's data sheet for each VMEbus module in the system and confirm the bus timing of DTACK* and BERR*.
- The VXIbus specification requires that the bus timer shall conform to the VMEbus BTO(≥ 100) bus timer specifications. This means that the bus timer will not assert BERR* until at least 100 µS has passed since a data strobe has been asserted. For more information about the BTO(X) mnemonic, refer the section on VMEbus bus timers.
- The BTO(100) capability allows for inter-crate bus links. These links often take more than 20 µS to respond to chassis-to-chassis bus cycles.
- The VXIbus specification *recommends* that a four level priority arbiter be used. Any type of arbitration method *may* be used, however.
- The VXIbus specification requires that every mainframe must provide a power monitor (this is an option under VMEbus). It further recommends that the power monitor assert ACFAIL* a minimum of 8 ms before SYSRESET* is asserted, and a minimum of 10 ms before +5 VDC drops below 4.875 volts.

9.5.2 VMEbus/VXIbus Mechanical Packaging Ramifications

The VXIbus mechanical specification provides for the installation of 3U and 6U VMEbus modules. In the VXIbus specification these are referred to as A and B size instruments. It does not specify the way in which mechanical conformance is to be achieved. The installation of VMEbus modules into VXIbus mainframes must have several design goals:

- Installation of 3U/6U VMEbus modules into 'C' and 'D' size VXIbus mainframes.

- Extension of the VMEbus front panel to the 340 mm deep VXIbus front panel.
- EMC shield.
- Handling the P2 user defined pins (if used on the VMEbus module).

It is possible to design a VXIbus mainframe so that 3U and 6U VMEbus modules can be mounted directly onto the VXIbus backplane. A variety of off-the-shelf Eurocard packaging solutions exist. A standard VXIbus backplane can be used in this configuration if the VMEbus module does not use the P2 user defined pins. If the P2 user defined pins are used, then a hybrid backplane is required. A hybrid backplane includes some slots that are reserved only for VMEbus modules, and allow the P2 user defined pins to be carried out the rear of the backplane.

Another popular way of installing VMEbus modules in a VXIbus backplane is with the module adapter kit as shown in Figure 9-17. The VMEbus front panel is removed, and the card screwed to the EMC shield. The entire assembly can then be placed into a standard VXIbus mainframe, in any card slot. Again, the VMEbus module cannot use the P2 user defined pins, but I/O from the rear of the VMEbus module can be cabled through the VXIbus front panel.

Another style of adapter is shown in Figure 9-18. This module has a VMEbus sub-rack mounted in front of the VXIbus backplane. Standard VMEbus modules are installed in the usual fashion. P2 user defined I/O pins can also be used as well.

Another practical alternative is to use a prototyping module or an interface card as shown in Figure 9-19. Both modules implement a VMEbus interface, as well as the VXIbus extensions and mechanical packaging (including shields).

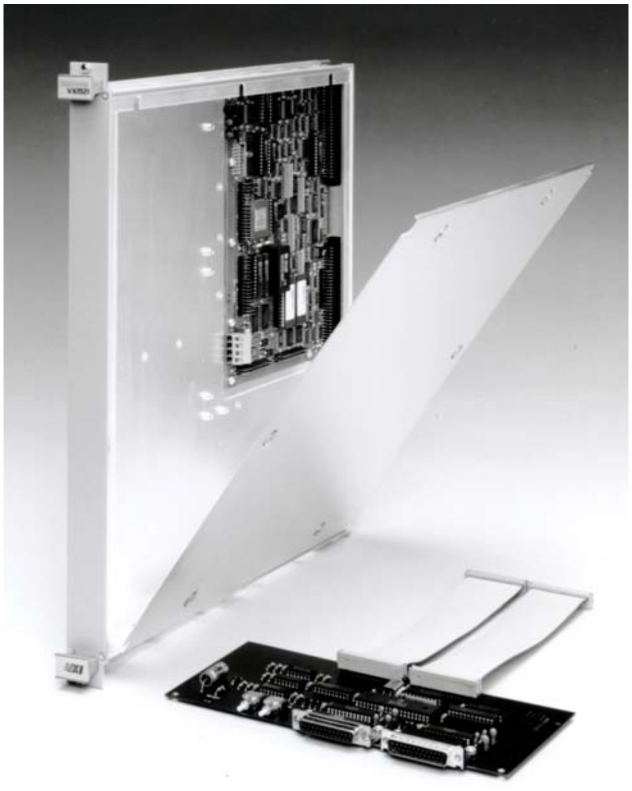


Figure 9-17 VXIbus adapter kit

VXIbus adapter kit allows a standard VMEbus module to be placed into a VXIbus mainframe. Photo courtesy of Tektronix.

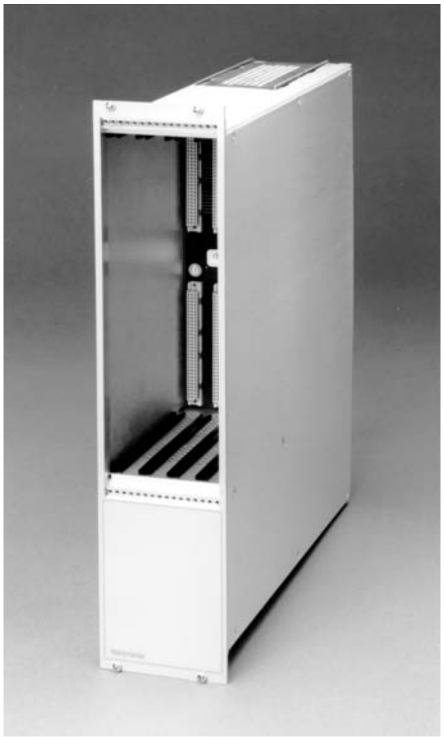


Figure 9-18 VXIbus adapter kit uses a VMEbus sub-rack mounted in front of the VXIbus backplane

Photo courtesy of Tektronix.

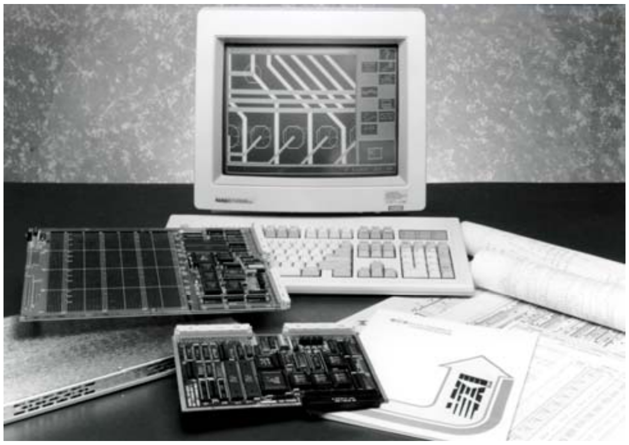


Figure 9-19 VXIbus prototyping modules

Shown are a VXIbus Prototyping Module (middle-left) and a VXIbus Interface Card (bottom-middle). Note the shield located below the Prototyping Module. Photo courtesy of ICS Electronics Corporation.

9.6 For More Information About VXIbus

Additional sources of information about VXIbus are given in Chapter 1 of this book. Included are The VXIbus Consortium, sources for the VXIbus Specification, VXIjournal Magazine, VXIbus integrated circuits and additional information about VMEbus.

9.7 References

Anschlimann, Larry D. et al. Apparatus And Method For Adapting Cards Designed For A VMEbus For Use In A VXIbus System US Patent No. 5,175,536 1992

Goss, Frank G., "Using VMEbus Boards in VXIbus Systems" *VMEbus Systems* December 1988

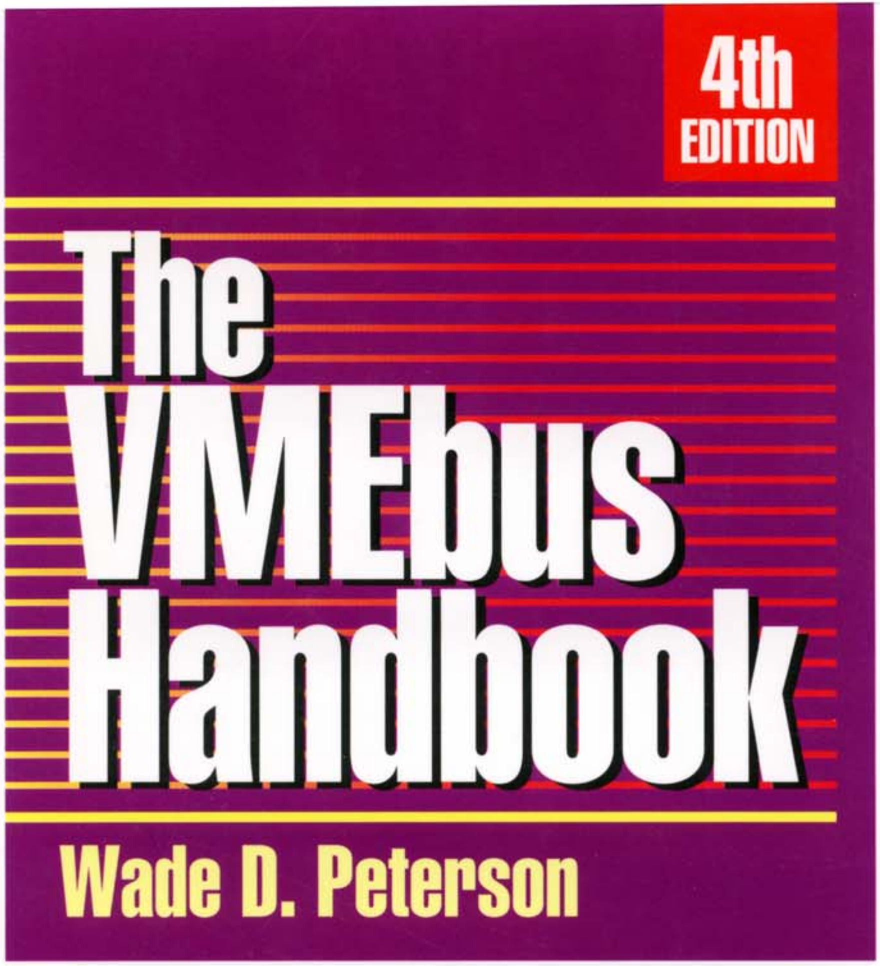
Haworth, David A., "Using VXIbus, A Guide to VXIbus Systems" Tektronix Inc., 1990. Portions reprinted with permission

IEEE Standard for VMEbus Extensions for Instrumentation: VXIbus / IEEE Std 1155-1992. IEEE, New York, NY 1993

Klahn, Louis J. Jr., "The VXIbus - An Idea Whose Time Has Come" *VMEbus Systems* December 1988

Miles, Michael D. Modular Instrument Chassis US Patent No. 5,528,455 1996

Index



4th EDITION

The VMEbus Handbook

Wade D. Peterson

Wade Peterson is an independent engineering consultant who specializes in the design of VMEbus boards, integrated circuits, systems and software. His VMEbus experience covers a wide range of applications, with a special emphasis on industrial automation. Previously he served as Manager of Technology for MTS Systems Corporation in Cary NC, and as a system integrator for the same company in Minneapolis MN. Until 1987 he was a VMEbus board designer for Mizar Incorporated in St. Paul MN. Wade has written numerous articles and papers, is a named inventor on four patents in the areas of industrial automation and sensor systems, and holds a BEE degree in Electrical Engineering from the University of Minnesota. He also presents a regular seminar for VITA entitled ***UNDERSTANDING VME64*** throughout the United States, Canada and Europe.

VITA
Open Standards, Open Markets

ISBN 1-885731-08-6