

INTERNATIONAL STANDARD

ISO/IEC 15776

First edition
2001-12

VME64bus – Specification



Reference number
ISO/IEC 15776:2001(E)

INTERNATIONAL STANDARD

ISO/IEC 15776

First edition
2001-12

VME64bus – Specification

© ISO/IEC 2001

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland



PRICE CODE **XF**

For price, see current catalogue

CONTENTS

FOREWORD	8
INTRODUCTION	9
1 General	13
1.1 Scope and object	13
1.2 Normative references	13
1.3 VMEbus interface system elements	14
1.4 VMEbus specification diagrams	20
1.5 Specification terminology	22
1.6 Protocol specification	24
1.7 System examples and explanations	25
2 Data transfer bus	25
2.1 Introduction	25
2.2 Data-transfer-bus lines	27
2.3 DTB modules – Basic description	38
2.4 Typical operation	64
2.5 Data-transfer-bus acquisition	73
2.6 DTB timing rules and observations	75
3 Data transfer bus arbitration	120
3.1 Bus arbitration philosophy	120
3.2 Arbitration bus lines	122
3.3 Functional modules	124
3.4 Typical operation	132
3.5 Race conditions between master requests and arbiter grants	141
4 Priority interrupt bus	141
4.1 Introduction	141
4.2 Priority interrupt bus lines	144
4.3 Priority interrupt bus modules – Basic description	146
4.4 Typical operation	159
4.5 Race conditions	165
4.6 Priority interrupt bus timing rules and observations	166
5 Utility bus	183
5.1 Introduction	183
5.2 Utility bus signal lines	183
5.3 Utility bus modules	183
5.4 System initialization and diagnostics	186
5.5 Power and ground pins	190
5.6 Reserved line	191
5.7 Auto slot ID	191
5.8 Auto system controller	198
6 Electrical specifications	199
6.1 Introduction	199
6.2 Power distribution	200
6.3 Electrical signal characteristics	201
6.4 Bus driving and receiving requirements	201

6.5	Backplane signal line interconnections	206
6.6	User defined signals.....	210
6.7	Signal line drivers and terminations	210
7	Mechanical specifications.....	212
7.1	Introduction.....	212
7.2	VMEbus boards.....	213
7.3	Front panels	217
7.4	Backplanes.....	220
7.5	Assembly of VMEbus subracks.....	222
7.6	Conduction cooled VMEbus systems.....	223
7.7	VMEbus backplane connectors and VMEbus board connectors	223
Annex A (normative)	Glossary of VMEbus terms	245
Annex B (normative)	VMEbus Connector/Pin description	251
Annex C (normative)	Manufacturer's board identification	255
Rule index.....		257
Figure 1 – System elements		15
Figure 2 – Functional modules and buses.....		21
Figure 3 – Signal timing notation		25
Figure 4 – Data transfer bus functional block diagram		28
Figure 5 – Block diagram – Master.....		39
Figure 6 – Block diagram – Slave.....		41
Figure 7 – Block diagram – Bus timer		43
Figure 8 – Block diagram – Location monitor		44
Figure 9 – Four ways in which 32 bits of data might be stored in memory.....		53
Figure 10 – Four ways in which 16 bits of data might be stored in memory.....		54
Figure 11 – Block diagram – Configuration ROM / Control & Status registers.....		59
Figure 12 – Example of a non-multiplexed address, single-byte read cycle		66
Figure 13 – Example of multiplexed address double-byte write cycle		67
Figure 14 – Example of non-multiplexed address quad-byte write cycle.....		69
Figure 15 – Example of an eight-byte block read cycle		70
Figure 16 – Data transfer bus master exchange sequence		74
Figure 17 – Address broadcast timing – All cycles.....		94
Figure 18 – A16, A24, A32 master, responding slave, and location monitor.....		95
Figure 19 – Master, slave, and location monitor – A16, A24 and A32 address broadcast timing		96
Figure 20 – Master, slave, and location monitor A16, A24, and A32 address broadcast timing		97
Figure 21 – Master, slave and location monitor – A64, A40, and ADOH address broadcast timing		98
Figure 22 – Master, slave, and location monitor data transfer timing.....		99
Figure 23 – Master, slave, and location monitor data transfer timing.....		101

Figure 24 – Master, slave and location monitor data transfer timing A40 multiplexed quad byte read, A40BLT multiplexed quad byte block read, MBLT eight byte block read	103
Figure 25 – Master, slave and location monitor data transfer timing	105
Figure 26 – Master, slave and location monitor data transfer timing	107
Figure 27 – Master, slave and location monitor data transfer timing A40 multiplexed quad byte write, A40BLT multiplexed quad byte block write, MBLT eight byte block write	109
Figure 28 – Master, slave and location monitor data transfer timing single-byte RMW cycles	111
Figure 29 – Master, slave and location monitor data transfer timing double-byte RMW cycles, quad-byte RMW cycles	112
Figure 30 – Address strobe inter-cycle timing	113
Figure 31 – Data strobe inter-cycle timing	113
Figure 32 – Data strobe inter-cycle timing	114
Figure 33 – Master, slave and bus timer data transfer timing timed-out cycle	114
Figure 34 – Master DTB control transfer timing	115
Figure 35 – Master and slave data transfer timing master responding to RETRY* line	116
Figure 36 – Master and slave data transfer timing master ignoring RETRY* line	117
Figure 37 – A40, MD32 read-modify-write	118
Figure 38 – Rescinding DTACK timing	119
Figure 39 – Arbitration functional block diagram	121
Figure 40 – Illustration of the daisy chain bus grant lines	123
Figure 41 – Block diagram – Arbiter	127
Figure 42 – Block diagram – Requester	130
Figure 43 – Arbitration flow diagram two requesters, two request levels	134
Figure 44 – Arbitration sequence diagram two requesters, two request levels	136
Figure 45 – Arbitration flow diagram two requesters, same request level	137
Figure 46 – Arbitration sequence diagram two requesters, same request level	140
Figure 47 – Priority interrupt bus functional diagram	142
Figure 48 – Interrupt subsystem structure – Single handler system	143
Figure 49 – Interrupt subsystem structure – Distributed system	144
Figure 50 – IACKIN*/IACKOUT* DAISY-CHAIN	146
Figure 51 – Block diagram – Interrupt handler	148
Figure 52 – Block diagram – Interrupter	151
Figure 53 – Block diagram – IACK daisy-chain driver	152
Figure 54 – Release of interrupt request lines by ROAK and RORA interrupters	156
Figure 55 – IACK daisy-chain driver and interrupter on the same board	158
Figure 56 – Two interrupters on the same board	159
Figure 57 – The three phases of an interrupt sequence	160
Figure 58 – Two interrupt handlers, each monitoring one interrupt request line	161
Figure 59 – Two interrupt handlers, each monitoring several interrupt request lines	162

Figure 60 – Typical single handler interrupt system operation flow diagram	164
Figure 61 – Typical distributed interrupt system with two interrupt handlers, flow diagram	165
Figure 62 – Interrupt handler and interrupter – Interrupter selection timing single-byte, double-byte and quad-byte interrupt acknowledge cycles	178
Figure 63 – IACK daisy-chain driver – Interrupter selection timing single-byte, double-byte, and quad-byte interrupt acknowledge cycles	178
Figure 64 – Participating interrupter – Interrupter selection timing single-byte, double-byte, and quad-byte interrupt acknowledge cycles	179
Figure 65 – Responding interrupter – Interrupter selection timing single-byte, double-byte, and quad-byte interrupt acknowledge cycles	179
Figure 66 – Interrupt handler – Status/ID transfer timing single-byte interrupt acknowledge cycle	180
Figure 67 – Interrupt handler – Status/ID transfer timing double-byte and quad-byte interrupt acknowledge cycles	180
Figure 68 – Responding interrupter – Status/ID transfer timing single-byte interrupt acknowledge cycle	181
Figure 69 – Responding interrupter – Status/ID transfer timing double-byte interrupt acknowledge cycle quad-byte interrupt acknowledge cycle	182
Figure 70 – IACK daisy-chain driver, responding interrupter and participating interrupter IACK daisy-chain inter-cycle timing	182
Figure 71 – Utility bus block diagram	184
Figure 72 – System clock driver timing	185
Figure 73 – Block diagram of power monitor module	186
Figure 74 – Power monitor power failure timing	187
Figure 75 – Power monitor system restart timing	187
Figure 76 – SYSRESET* and SYSFAIL* timing diagram	190
Figure 77 – Current rating for power pins	191
Figure 78 – CR/CSR auto ID slave initialization algorithm	195
Figure 79 – First slot detector (FSD)	198
Figure 80 – VMEbus signal levels	201
Figure 81 – Standard bus termination	208
Figure 82 – Subrack with mixed board sizes	226
Figure 83 – Single height board – Basic dimensions	227
Figure 84 – Double height board – Basic dimensions	228
Figure 85 – Connector positions on single and double height boards	229
Figure 86 – Cross-sectional view of board, connector, backplane, and front panel	230
Figure 87 – Optional enhanced DIN connector	231
Figure 88 – Component height, lead length and board warpage	232
Figure 89 – Single height, single width front panel	233
Figure 90 – Double height, single width front panel	234
Figure 91 – Front panel mounting brackets and dimension of single height boards	235
Figure 92 – Front panel mounting brackets and dimension of double height boards	236
Figure 93 – Single height filler panel	237

Figure 94 – Double height filler panel	238
Figure 95 – Backplane detailed dimensions of a J1 and a J2 backplane.....	239
Figure 96 – Detailed dimensions of a J1/J2 backplane	240
Figure 97 – "Off-board type" backplane terminations (viewed from top of backplane)	241
Figure 98 – "On-board type" backplane terminations (viewed from top of backplane).....	242
Figure 99 – 21 slot subrack	243
Figure 100 – Board guide detail.....	244
Table 1 – The eight categories of byte locations	29
Table 2 – Address alignment on bus	29
Table 3 – Signal levels during data transfers used to select which byte location(s) are accessed during a data transfer	31
Table 4 – Address modifier codes	33
Table 5 – Use of data lines to move data during nonmultiplexed data transfers.....	35
Table 6 – Use of the address and data lines for multiplexed data cycles.....	36
Table 7 – RULEs and PERMISSIONs specifying the use of the dotted lines by the various types of masters.....	40
Table 8 – Slaves – RULEs and PERMISSIONs specifying the use of the dotted lines by the various type of slaves.....	42
Table 9 – Use of the BTO() mnemonic specifying the time-out period of bus timers.....	43
Table 10 – Location monitors – RULEs and PERMISSIONs specifying the use of the dotted lines by the various types of location monitors	45
Table 11 – Mnemonics specifying addressing capabilities	46
Table 12 – Mnemonics specifying basic data transfer capabilities	48
Table 13 – Mnemonics specifying block transfer capabilities	51
Table 14 – The mnemonic that specifies read-modify-write capabilities	52
Table 15 – Transferring 32 bits of data using multiple-byte transfer cycles	54
Table 16 – Transferring 16 bits of data using multiple-byte transfer cycles	55
Table 17 – Mnemonic that specifies unaligned transfer capability	55
Table 18 – Mnemonics specifying address only capability	56
Table 19 – Configuration ROM/control & status registers: RULEs and PERMISSIONs or monitoring the dashed lines.....	60
Table 20 – Control and status register base definition	60
Table 21 – Configuration ROM definition.....	61
Table 22 – Timing diagrams defining master, slave, and location monitor operation (see Table 27 for timing values)	76
Table 23 – Definitions of mnemonics used in Tables 24, 25 and 26.....	78
Table 24 – Use of the address and data lines to select a byte group.....	78
Table 25 – Use of DS1*, DS0*, A1, A2, and LWORD* during the address phase of the various cycles.....	79
Table 26 – Use of the data lines to transfer data.....	80

Table 27 – Master, slave, and location monitor timing parameters	83
Table 28 – Bus-timer timing parameters (see also Table 32)	84
Table 29 – Master, timing RULEs and OBSERVATIONS	84
Table 30 – Slave, timing RULEs and OBSERVATIONS	89
Table 31 – Location monitor, timing OBSERVATIONS	93
Table 32 – BUS TIMER, timing RULEs	94
Table 33 – RULEs and PERMISSIONs specifying the use of the dotted lines by the various types of arbiters	128
Table 34 – RULEs and PERMISSIONs specifying the use of the dotted lines by the various types of requesters	131
Table 35 – RULEs and PERMISSIONs specifying the use of the dotted lines in Figure 51 by the various types of interrupt handlers	149
Table 36 – RULEs and PERMISSIONs specifying the use of the dotted lines in Figure 52 by the various types of interrupters	151
Table 37 – Use of the IH() mnemonic to specify interrupt request handling capabilities	153
Table 38 – Use of the I() mnemonic to specify interrupt request generation capabilities	153
Table 39 – Mnemonics specifying status/ID transfer capabilities	153
Table 40 – Mnemonics specifying interrupt request release capabilities	155
Table 41 – 3-bit interrupt acknowledge code	164
Table 42 – Timing diagrams defining interrupt handler and interrupter operation	167
Table 43 – Timing diagrams defining IACK daisy-chain driver operation	168
Table 44 – Timing diagrams defining participating interrupter operation	168
Table 45 – Timing diagrams that define responding interrupter operation	168
Table 46 – Definitions of mnemonics used in tables 47, 48 and 49	169
Table 47 – Use of addressing lines during interrupt acknowledge cycles	169
Table 48 – Use of the DS1*, DS0*, LWORD* and WRITE* lines during interrupt acknowledge cycles	170
Table 49 – Use of the data bus lines to transfer the Status/ID	170
Table 50 – Interrupt handler, interrupter and IACK daisy-chain driver timing parameters	171
Table 51 – Interrupt handler, timing RULEs and OBSERVATIONS	172
Table 52 – Interrupter, timing RULEs and OBSERVATIONS	174
Table 53 – IACK daisy-chain driver, timing RULEs and OBSERVATIONS	177
Table 54 – Module drive during power-up and power-down sequences	189
Table 55 – Bus voltage specification	200
Table 56 – Bus driving and receiving requirements	202
Table 57 – Bus driver summary	211
Table 58 – J1/P1 pin assignments	224
Table 59 – J2/P2 pin assignments	225

VME64bus – SPECIFICATION

FOREWORD

- 1) ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
- 3) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 15776 was prepared by subcommittee 26: Microprocessor systems, of ISO/IEC joint technical committee 1: Information technology.

International Standards are drafted in accordance with ISO/IEC Directives, Part 3.

Annexes A, B and C form an integral part of this International Standard.

INTRODUCTION

The architectural concepts of VMEbus are based on the VERSAbus developed by Motorola in the late 1970s. Motorola's European Microsystems group in Munich, West Germany proposed the development of a VERSAbus-like product line of computers and controllers based on the Eurocard mechanical standard. To demonstrate the concept, Max Loesel and Sven Rau developed three prototype boards: (1) a 68000 CPU card, (2) a dynamic memory card, and (3) a static memory card. They named the new bus VERSAbus-E, which was later renamed "VME" by Lyman Hevle, then VP of the Motorola Microsystems Operation (and later the founder of VITA). VME is the acronym for VERSA-module Europe. Motorola, Mostek, and Signetics agreed to jointly develop and support the new bus architecture in early 1981.

John Black of Motorola, Craig McKenna of Mostek, and Cecil Kaplinsky of Signetics developed the first draft of the VMEbus specification. In October of 1981, at the Systems 81 trade show in Munich, West Germany, Motorola, Mostek, and Signetics announced their joint support for VMEbus, and placed Revision A of the specification in the public domain.

In August of 1982, Revision B of the VMEbus specification was published by the newly formed VMEbus Manufacturers Group (now VITA). This new revision refined the electrical specifications for the signal line drivers and receivers, and also brought the mechanical specifications more in line with the developing IEC 60297 standard, the formal specifications for Eurocard mechanical formats.

In the latter part of 1982, the French delegation of the International Electrotechnical Commission (IEC) proposed Revision B of the VMEbus specification as an international standard. The IEC SC47B subcommittee nominated Mira Pauker of Philips, France, as the chairperson of an editorial committee, formally starting international standardization of the VMEbus.

In March of 1983, the IEEE Microprocessor Standards Committee (MSC) requested authorization to establish a working group to standardize the VMEbus in the US. This request was approved by the IEEE Standards Board, and the P1014 Working Group was established. Wayne Fischer was appointed first chairman of the working group. John Black served as chairman of the P1014 Technical Subcommittee.

The IEC, IEEE, and VMEbus Manufacturers Group (now VITA) distributed copies of Revision B for comment, and received requests for changes to the document as a result. These comments made it clear that it was time to go forward past revision B. In December of 1983, a meeting was held that included John Black, Mira Pauker, Wayne Fischer, and Craig McKenna. It was agreed that a revision C should be created, and that it should take into consideration all comments received by the three organizations. John Black and Shlomo Pri-Tal of Motorola incorporated the changes from all sources into a common document. The VMEbus Manufacturers Group (now VITA) labeled the document Revision C.1 and placed it in the public domain. The IEEE labeled it P1014 Draft 1.2, and the IEC labeled it IEC 60821 Bus. Subsequent ballots in the IEEE P1014 group and in the MSC resulted in more comments, and required that the IEEE P1014 draft be updated. This work resulted in the ANSI/IEEE 1014-1987 specification.

In 1989, John Peters of Performance Technologies, Inc. (Rochester, NY) developed the initial concept of VME64: multiplexing address and data lines (A64/D64) on the VMEbus. This concept was shown for the first time in 1989 and placed in the VITA Technical Committee in 1990 as a performance enhancement to the VMEbus specification. In 1991, the PAR (Project Authorization Request) for P1014R (revisions to the VMEbus specification) was granted by the IEEE. Ray Alderman, Technical Director of VITA, co-chaired the activity with Kim Clohessy of DY 4 Systems (Nepean, Ontario, Canada).

At the end of 1992, the additional enhancements to VMEbus (A40/D32, Locked Cycles, Rescinding DTACK*, Autoslot-ID, Auto System Controller, and enhanced DIN connector mechanicals) required more work to complete this document. In 1992, the VITA Technical Committee suspended work with the IEEE and sought accreditation as a standards developer organization (SDO) with the American National Standards Institute. The original IEEE Par P1014R was subsequently withdrawn by the IEEE. The VITA Technical Committee returned to using the public domain VMEbus C.1 specification as its base level document to which it added new enhancements. This enhancement work was undertaken entirely by the VITA Technical Committee resulting in this document. The tremendous undertaking of the document editing was accomplished by Kim Clohessy of DY 4 Systems, the technical co-chair of the activity with great help from Frank Hom who created the mechanical drawings, and with exceptional contributions by each chapter editor.

Additional enhancements proposed to the VME64 Subcommittee have been placed in another VITA subcommittee: the VME64 Extensions Document. Two other activities began in late 1992: (1) BLLI (VMEbus Board-level Live Insertion Specifications), and (2) VSLI (VMEbus System-level Live Insertion with Fault Tolerance).

New activities begun in 1993 using the base-VME architecture involve the implementation of high-speed serial and parallel sub-buses for use as I/O interconnections and data mover subsystems. These architectures can be used as message switches, routers, and small multiprocessor parallel architectures.

VITA's application for recognition as an accredited standards developer organization of ANSI (American National Standards Institute) was granted in June 1993. Numerous other documents, including mezzanine, P2, and serial bus standards, have been placed with VITA as the Public Domain Administrator of these technologies.

VMEbus Specification Genealogy

VMEbus	Revision B and C.1 (Public Domain)
IEEE 1014-1987	Versatile Backplane Bus VMEbus
VITA 1-1994	VME64 Specification
IEEE 1096-1988	VSBbus Specification (IEEE)
IEC 60821:1991	VMEbus – Microprocessor system bus for 1 byte to 4 byte data
IEEE 1101.1	IEEE Standard for Mechanical Core Specifications for Microcomputers Using IEC 60603-2 Connectors
IEEE 1101.2	IEEE Standard for Mechanical Core Specification for Conduction-Cooled Eurocards

This standard was constructed through the many hours of hard work by the members of the VME64 Subcommittee (of the VITA Technical Committee) and the commitment of their companies to this standard.

NAME	COMPANY
Ray Alderman	PEP Modular Computers
Michael Humphrey	VITA
John Rynearson	VITA
Martin Blake	BICC-VERO UK
Kim Clohessy	DY 4 Systems, Inc.
Jing Kwok	DY 4 Systems, Inc.
Clarence Peckham	Heurikon Corporation
Dennis Terry	Heurikon Corporation
Wayne Fischer	Force Computers Inc.
Jack Regula	Consultant
Tad Kubic	Dawn VME Products, Inc.
Will Hamsher	AMP, Inc.
Doug Reubendall	AMP, Inc.
William Mahusen	Performance Technologies, Inc.
Thanos Mentzelopoulos	Ironics, Inc.
Joel Silverman	Radstone Technology Corp.
Colin Davies	Radstone Technology UK
Frank Hom	Electronic Solutions
Keith Burgess	Mizar
John Black	Micrology PBT
Greg Hill	Hewlett-Packard Co.
Sam Babb	Hewlett Packard Co.
Ben LaPointe	Motorola GEG
Chau Pham	Motorola MCG
Mac Rush	Motorola MCG
Mike Hasenfratz	Micro Memory, Inc.
Richard DeBock	Matrix Corp.
Richard O'Connor	Newbridge Microsystems
Michael Bryant	PEP Modular Computers
Dr. Chris Eck	CERN-Geneva
Mike Thompson	Schroff, Inc.
Eike Waltz	Schroff, Inc.
Tom Baillio	Mercury Computer Systems, Inc.
Steve Corbesero	Mupac Corp.
Dave Horton	Cypress Semiconductor
Mike Maas	Cypress Semiconductor
Mike Munroe	Hybricon Corp.

The following are acknowledged for their extra contributions as chapter editors.

Kim Clohessy	DY-4 Systems Inc.
Wayne Fischer	Force Computers, Inc.
Chau Pham	Motorola MCG
Frank Hom	Electronic Solutions
Richard DeBock	Matrix Corp.
Jing Kwok	DY 4 Systems, Inc.
Mike Hasenfratz	Micro Memory, Inc.

CANVASS BALLOT

Consensus for this standard was achieved by use of the Canvass Method.

The following organizations, recognized as having an interest in the standardization of VME64, participated in the Canvass Ballot process. Inclusion in this list does not necessarily imply that the organization concurred with the submittal of the proposed standard to ANSI.

767 AWACS	Micrology
Adept Technology	MITRE Corporation
AMP	Motorola Computer Group
AT&T Bell Laboratories	Mupac Corporation
Berg Electronics	Newbridge Microsystems
Bit 3 Computer	Object Technology Inc.
CERN	PEP Modular Computers
CSPI	Philips Ind. Automation
Cypress Semiconductor	Picosoft
Dawn VME Products	Radstone Technology
Dialogic Corporation	Schroff
Digital Equipment Corp.	Technology Consulting
DY 4 Systems	Texas Instruments
Electronic Solutions	VERO Electronics
Force Computers	VITA
Harting Elektronik	VME MEMBER
Heurikon Corporation	Winchester Electronics
Hewlett-Packard	
Hughes Aircraft Company	
Hybricon Corporation	
IBM	
IXTHOS	
Loral Western Devel. Lab	
Los Alamos Nat'l Lab	
Matrix Corporation	
Micro Memory	

VME64bus – SPECIFICATION

1 General

1.1 Scope and object

The VMEbus specification defines an interfacing system used to interconnect microprocessors, data storage, and peripheral control devices in a closely coupled hardware configuration. The system has been conceived with the following objectives:

- a) to allow communication between devices on the VMEbus without disturbing the internal activities of other devices interfaced to the VMEbus;
- b) to specify the electrical and mechanical system characteristics required to design devices that will reliably and unambiguously communicate with other devices interfaced to the VMEbus;
- c) to specify protocols that precisely define the interaction between the VMEbus and devices interfaced to it;
- d) to provide terminology and definitions that describe the system protocol;
- e) to allow a broad range of design latitude so that the designer can optimize cost and/or performance without affecting system compatibility;
- f) to provide a system where performance is primarily device limited, rather than system interface limited.

1.2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60297-1:1986, *Dimensions of mechanical structures of the 482,6 mm (19 in) series – Part 1: Panels and racks*

IEC 60297-2:1982, *Dimensions of mechanical structures of the 482,6 mm (19 in) series – Part 2: Cabinets and pitches of rack structures*

IEC 60297-3:1984, *Dimensions of mechanical structures of the 482,6 mm (19 in) series – Part 3: Subracks and associated plug-in units*

IEC 60297-4:1995, *Mechanical structures for electronic equipment – Dimensions of mechanical structures of the 482,6 mm (19 in) series – Part 4: Subracks and associated plug-in units – Additional dimensions*

IEC 60603-2:1995, *Connectors for frequencies below 3 MHz for use with printed boards – Part 2: Detail specification for two-part connectors with assessed quality, for printed boards, for basic grid of 2.54 mm (0.1 in) with common mounting features*

IEC 61076 (all parts), *Connectors with assessed quality, for use in d.c., low frequency analogue and digital high speed data applications*

IEEE 1101.2, *Standard for Mechanical Core Specifications for Conduction-Cooled Eurocards*

IEEE 1394, *Standard for a High Performance Serial Bus*

1.3 VMEbus interface system elements

1.3.1 Basic definitions

The VMEbus structure can be described from two points of view: its mechanical structure and its functional structure. The mechanical specification describes the physical dimensions of subracks, backplanes, front panels, plug-in boards, etc. The VMEbus functional specification describes how the bus works, what functional modules are involved in each transaction, and the rules which govern their behavior. The following informal definitions describe some basic terms used for both the mechanical and the functional structure of the VMEbus.

1.3.1.1 VMEbus mechanical structure

1.3.1.1.1

VMEbus backplane

printed circuit (PC) board with 96 or 160 pin connectors and signal paths that bus the connector pins

Some VMEbus systems have a single PC board, called the J1 backplane. It provides the signal paths needed for basic operation. Other VMEbus systems also have an optional second PC board, called a J2 backplane. It provides the additional 96 or 160 pin connectors and signal paths needed for wider data and address transfers. Still others have a single PC board that provides the signal conductors and connectors of both the J1 and J2 backplanes.

1.3.1.1.2

board

printed circuit (PC) board, its collection of electronic components, with either one or two 96 or 160 pin connectors that can be plugged into VMEbus backplane connectors

1.3.1.1.3

slot

position where a board can be inserted into a VMEbus backplane

If the VMEbus system has both a J1 and a J2 backplane (or a combination J1/J2 backplane) each slot provides a pair of 96 or 160 pin connectors. If the system has only a J1 backplane, then each slot provides a single 96 or 160 pin connector.

1.3.1.1.4

subrack

rigid framework that provides mechanical support for boards inserted into the backplane, ensuring that the connectors mate properly and that adjacent boards do not contact each other. It also guides the cooling airflow through the system, and ensures that inserted boards do not disengage themselves from the backplane due to vibration or shock

1.3.1.2 VMEbus functional structure

Figure 1 shows a simplified block diagram of the functional structure, including the VMEbus signal lines, backplane interface logic, and functional modules.

1.3.1.2.1

backplane interface logic

special interface logic that takes into account the characteristics of the backplane: its signal line impedance, propagation time, termination values, etc.

The VMEbus specification prescribes certain rules for the design of this logic based on the maximum length of the backplane and its maximum number of board slots.

1.3.1.2.2**functional module**

collection of electronic circuitry that resides on one VMEbus board and works together to accomplish a task

1.3.1.2.3**data transfer bus**

one of the four buses provided by the VMEbus backplane

The Data Transfer Bus allows Masters to direct the transfer of binary data between themselves and Slaves. (Data Transfer Bus is often abbreviated DTB.)

1.3.1.2.4**data transfer bus cycle**

sequence of level transitions on the signal lines of the DTB that result in the transfer of an address or an address and data between a Master and a Slave

The Data Transfer Bus cycle is divided into two portions, the address broadcast and then zero or more data transfers. There are 34 types of Data Transfer Bus cycles. They are defined later in this chapter.

1.3.1.2.5**master**

functional module that initiates DTB cycles in order to transfer data between itself and a Slave module

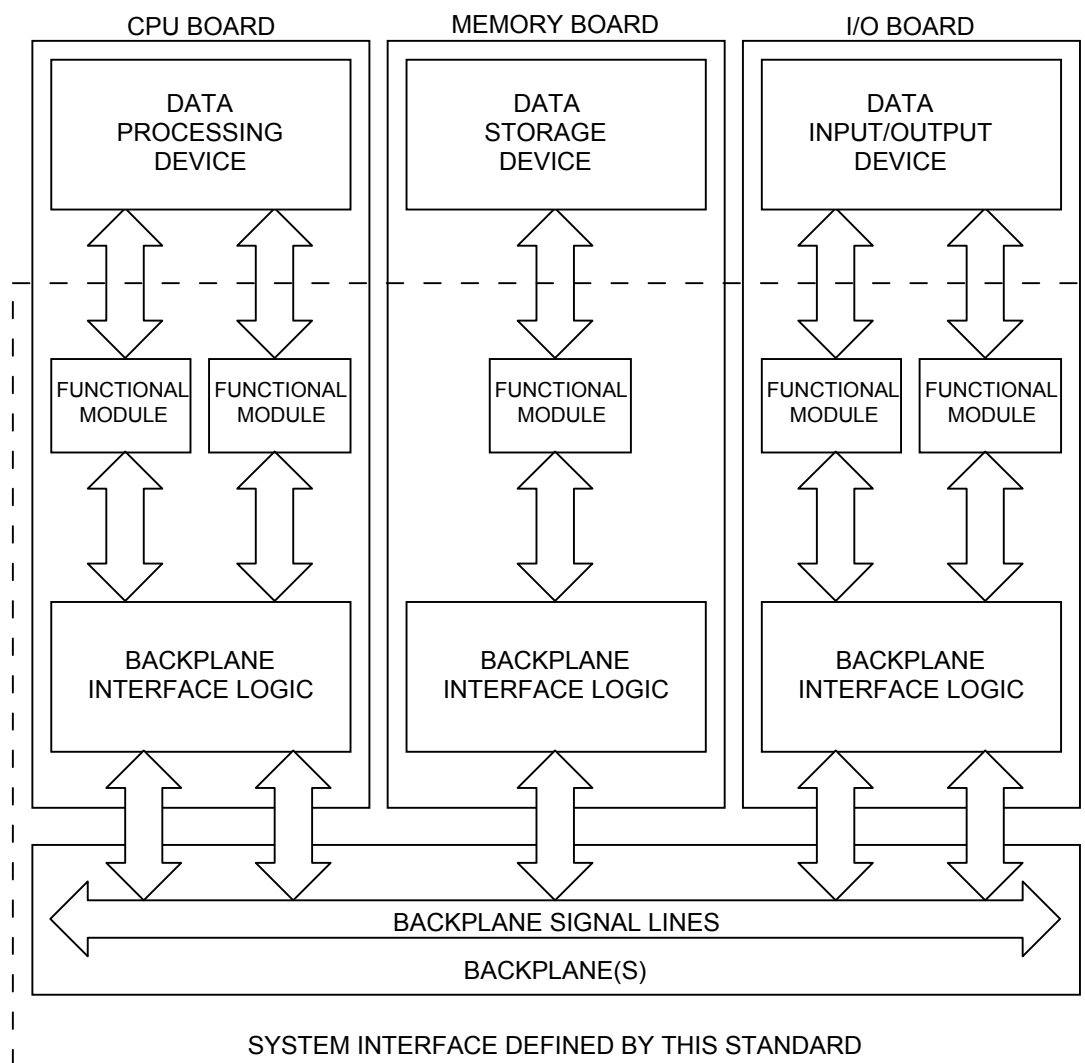


Figure 1 – System elements

1.3.1.2.6

slave

a functional module that detects DTB cycles initiated by a Master and, when those cycles specify its participation, transfers data between itself and the Master

1.3.1.2.7

location monitor

a functional module that monitors data transfers over the DTB in order to detect accesses to the locations it has been assigned to watch. When an access to one of these assigned locations occurs, the Location Monitor generates an on-board signal

1.3.1.2.8

bus timer

a functional module with a preset time-out period which terminates the DTB cycle if a transfer exceeds the time-out period. Without this module, if the Master tries to transfer data to or from a nonexistent Slave location it might wait forever. The Bus Timer prevents this by terminating the cycle

1.3.1.2.9

priority interrupt bus

one of the four buses provided by the VMEbus backplane

The Priority Interrupt Bus allows Interrupter modules to send interrupt requests to Interrupt Handlers.

1.3.1.2.10

interrupter

a functional module that generates an interrupt request on the Priority Interrupt Bus and then provides Status/ID information when the Interrupt Handler requests it

1.3.1.2.11

interrupt handler

a functional module that detects interrupt requests generated by Interrupters and responds to those requests by asking for Status/ID information.

1.3.1.2.12

status/ID

an eight, sixteen, or thirty-two-bit value returned by an interrupter to an interrupt handler during an interrupt acknowledge cycle

1.3.1.2.13

daisy-chain

a special type of VMEbus signal line that is used to propagate a signal level from board to board, starting with the first slot and ending with the last slot

There are four bus grant daisy-chains and one interrupt acknowledge daisy-chain on the VMEbus.

1.3.1.2.14

IACK daisy-chain driver

a functional module which activates the interrupt acknowledge daisy-chain whenever an Interrupt Handler acknowledges an interrupt request

This daisy-chain ensures that only one Interrupter will respond with its STATUS/ID when more than one has generated an interrupt request.

1.3.1.2.15

arbitration bus

one of the four buses provided by the VMEbus backplane

This bus allows an Arbiter module and several Requester modules to coordinate use of the DTB.

1.3.1.2.16**requester**

a functional module that resides on the same board as a Master or Interrupt Handler and requests use of the DTB whenever its Master or Interrupt Handler needs it

1.3.1.2.17**arbiter**

a functional module that accepts bus requests from Requester modules and grants control of the DTB to one Requester at a time

1.3.1.2.18**utility bus**

one of the four buses provided by the VMEbus backplane

This bus includes signals that provide periodic timing and coordinate the power-up and power-down of VMEbus systems.

1.3.1.2.19**CR/CSR**

a functional module that provides Configuration ROM information and Control and Status Registers. The module provides manufacturing and board ID and other important board information

The CSRs are used for software configuration of a VMEbus system.

1.3.1.2.20**system clock driver**

a functional module that provides a 16 MHz timing signal on the Utility Bus

1.3.1.2.21**serial bus**

a functional module that provides a bused 2 wire interface between cards in the backplane, independent of the other VMEbus modules

1.3.1.2.22**power monitor module**

a functional module that monitors the status of the primary power source to the VMEbus system, and signals when that power has strayed outside the limits required for reliable system operation

Since most systems are powered by an a.c. source, the Power Monitor is typically designed to detect drop-out or brown-out conditions on AC lines.

1.3.1.2.23**system controller board**

a board which resides in slot 1 of a VMEbus backplane and has a System Clock Driver, an Arbiter, an IACK Daisy-Chain Driver, and a Bus Timer; some also have a Power Monitor

1.3.1.3 Types of cycles on the VMEbus**1.3.1.3.1****read cycle**

a DTB cycle used to transfer 1, 2, 3, 4 or 8 bytes from a Slave to a Master

The cycle begins when the Master broadcasts an address and an address modifier. Each Slave captures the address modifier and address and checks to see if it is to respond to the cycle. If so, it retrieves the data from its internal storage, places it on the data bus and acknowledges the transfer. The Master then terminates the cycle.

1.3.1.3.2

write cycle

a DTB cycle used to transfer 1, 2, 3, 4 or 8 bytes from a Master to a Slave

The cycle begins when the Master broadcasts an address and address modifier and places data on the DTB. Each Slave captures the address and address modifier and checks to see if it is to respond to the cycle. If so, it stores the data and then acknowledges the transfer. The Master then terminates the cycle.

1.3.1.3.3

block read cycle

a DTB cycle used to transfer a block of 1 to 256 bytes from a Slave to a Master

This transfer is done using a string of 1, 2, or 4 byte data transfers. Once the block transfer is started, the Master does not release the DTB until all of the bytes have been transferred. It differs from a string of read cycles in that the Master broadcasts only one address and address modifier (at the beginning of the cycle). Then the Slave increments this address on each transfer so that the data for the next transfer is retrieved from the next higher location.

1.3.1.3.4

block write cycle

a DTB cycle used to transfer a block of 1 to 256 bytes from a Master to a Slave. The block write cycle is very similar to the block read cycle. It uses a string of 1, 2, or 4 byte data transfers. The Master does not release the DTB until all of the bytes have been transferred. It differs from a string of write cycles in that the Master broadcasts only one address and address modifier (at the beginning of the cycle). Then the Slave increments this address on each transfer so that the data from the next transfer is stored in the next higher location.

1.3.1.3.5

multiplexed cycle

a DTB cycle that transfers address information and/or data information using both the address and data buses. Multiplexed cycles are used in four cases.

- a) A64 – the full address bus and the full data bus are combined to create a 64 bit address.
- b) MBLT – the full address bus and the full data bus are combined to create a 64 bit data word.
- c) A40 – The full 24 bit address bus and the full 16 bit data bus on the P1/J1 connector are combined to create a 40 bit address. This mode is especially useful for 3U boards which have a J1 connector only.
- d) MD32 – The lower 16 address lines and the lower 16 data lines are combined to create a 32 bit data word. This mode is especially useful for 3U modules.

Multiplexed cycles are used in both basic transfers and block transfers. A64 and A40 basic transfers support 1 byte, 2 byte and 4 byte transfers. In addition MBLT cycles support 8 byte transfers.

A Multiplexed Cycle will have an Address phase that is separate from the Data phase. The Address phase may include (i.e. A64 and A40 cycles) or may not include (i.e. A32, A24 cycles) the use of the Data bus. The Data phase may include (MBLT, MD32 cycles) or may not include (i.e. D32, D16, D08(OE) cycles) the use of the Address bus.

Multiplexed block transfers can have 1 to 256 transfers between the Master and a Slave. With 8 byte transfers, up to 2,048 bytes can be transferred in one block.

1.3.1.3.6

read-modify-write cycle

a DTB cycle that is used to both read from, and write to, a Slave location without permitting any other Master to access that location

This cycle is most useful in multiprocessing systems where certain memory locations are used to provide semaphore functions.

1.3.1.3.7**address-only cycle**

a DTB cycle that consists of an address broadcast, but no data transfer

Slaves do not acknowledge ADDRESS-ONLY cycles and Masters terminate the cycle without waiting for an acknowledgment. No data strobes or acknowledge strobes are asserted in an ADDRESS-ONLY cycle.

1.3.1.3.8**address-only-with-handshake cycle**

a DTB cycle that consists of an address broadcast, but no data transfer

The addressed Slave responds in the same manner as a standard access cycle.

1.3.1.3.9**interrupt acknowledge cycle**

a DTB cycle, initiated by an Interrupt Handler, which reads a STATUS/ID from an Interrupter

An Interrupt Handler generates this cycle whenever it detects an interrupt request from an Interrupter and it has control of the DTB.

1.3.2 Basic VMEbus structure

The VMEbus interface system consists of backplane interface logic, four groups of signal lines called buses, and a collection of functional modules which can be configured as required. The functional modules communicate with each other using the backplane signal lines.

The functional modules defined in this document are used as vehicles for discussion of the bus protocol and need not be considered a constraint to logic design. For example, the designer might choose to design logic which interacts with the VMEbus in the manner described, but uses different on-board signals, or monitors other VMEbus signals. VMEbus boards might be designed to include any combination of the functional modules defined by this standard.

The VMEbus functional structure can be divided into four categories. Each consists of a bus and its associated functional modules which work together to perform specific duties. Figure 2 shows the VMEbus functional modules and buses. Each category is briefly summarized below.

1.3.2.1**data transfer**

devices transfer data over the Data Transfer Bus (DTB), which contains data and address pathways and associated control signals

Functional modules called Masters, Slaves, Interrupters, and Interrupt Handlers use the DTB to transfer data between each other. Two other modules, called Bus Timer and IACK Daisy-Chain Driver also assist them in this process.

1.3.2.2**DTB arbitration**

since a VMEbus system can be configured with more than one Master or Interrupt Handler, a means is provided to transfer control of the DTB between them in an orderly manner and to guarantee that only one Master controls the DTB at a given time. The Arbitration Bus modules (Requesters and Arbiter) coordinate the control transfer

1.3.2.3

priority interrupt

the priority interrupt capability of the VMEbus provides a means by which devices can request services from an Interrupt Handler

These interrupt requests can be prioritized into a maximum of seven levels. Interrupters and Interrupt Handlers use the Priority Interrupt Bus signal lines.

1.3.2.4

utilities

periodic clocks, initialization, and failure detection are provided by the Utility Bus

It includes a general purpose system clock line, a system reset line, a system fail line, an a.c. fail line, and two serial lines. Utilities also include power and ground pins.

1.4 VMEbus specification diagrams

As aids to defining or describing VMEbus operation, several types of diagrams are used, including:

- a) Timing diagrams that show the timing relationships between signal transitions. The times involved will have minimum and/or maximum limits associated with them. Some of the times specified on these diagrams specify the behavior of the backplane interface logic, while other times specify the interlocked behavior of the functional modules.
- b) Sequence diagrams that are similar to timing diagrams but show only the interlocked timing relationships of the functional modules. These diagrams are intended to show a sequence of events, rather than to specify the times involved. For example, a sequence diagram might indicate that module A cannot generate signal transition B until it detects module C's generation of signal transition D.
- c) Flow diagrams that show a stream of events as they would occur during a VMEbus operation. The events are stated in words and result from interaction of two or more functional modules. The flow diagram describes VMEbus operations in a sequential manner and, at the same time, shows interaction of the functional modules.

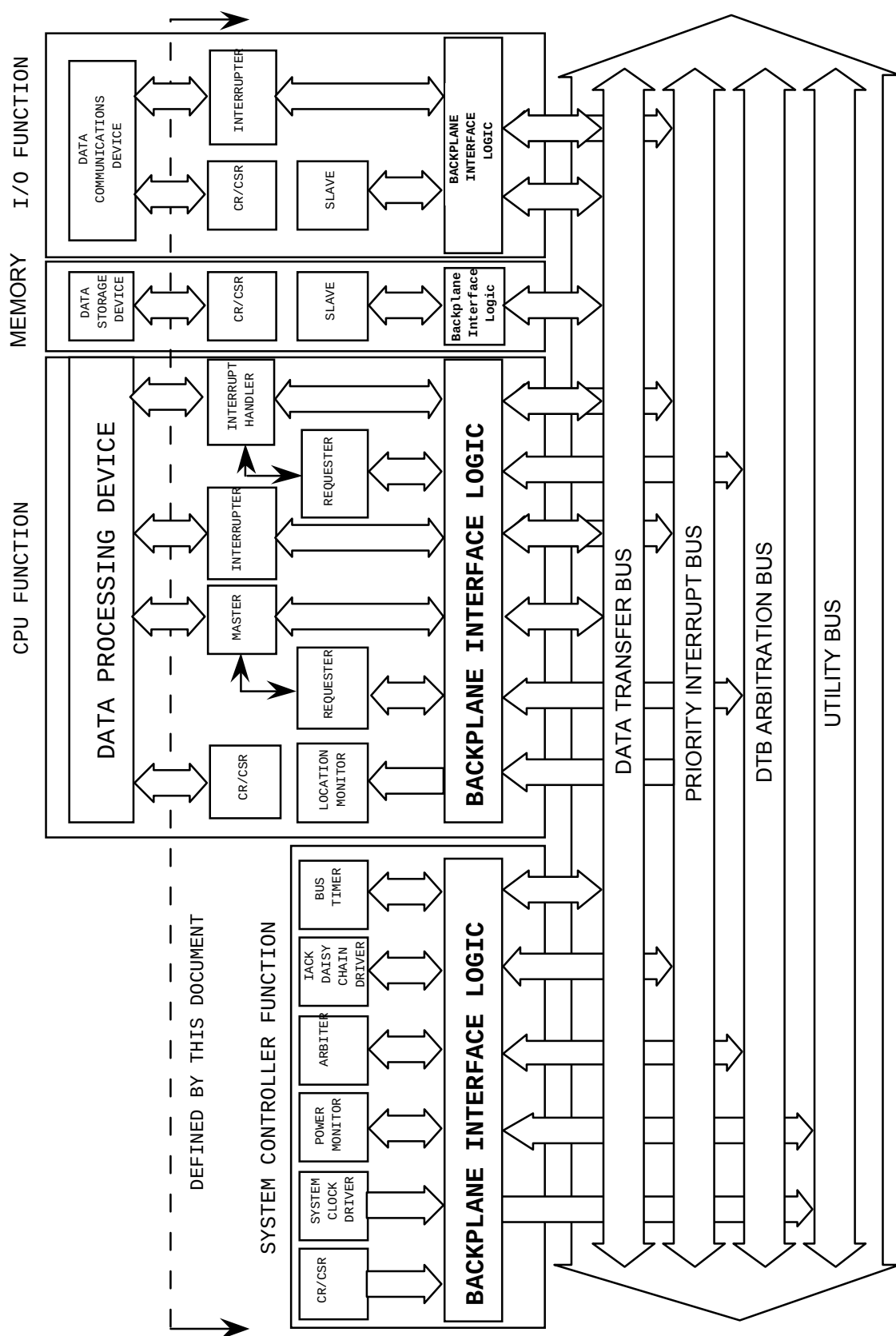


Figure 2 – Functional modules and buses

1.5 Specification terminology

To avoid confusion, and to make very clear what the requirements for compliance are, many of the paragraphs in this document are labelled with keywords that indicate the type of information they contain. The keywords are listed below

RULE
RECOMMENDATION
SUGGESTION
PERMISSION
OBSERVATION

Any text not labelled with one of these keywords describes the VMEbus structure or operation. It is written in either a descriptive or a narrative style. These keywords are used as follows:

RULE chapter.number

Rules form the basic framework of the VMEbus specification. They are sometimes expressed in text form and sometimes in the form of Figures, tables, or drawings. All VMEbus rules **MUST** be followed to ensure compatibility between VMEbus designs. Rules are characterized by an imperative style. The upper-case words **MUST** and **MUST NOT** are reserved exclusively for stating rules in this document and are not used for any other purpose.

RECOMMENDATION chapter.number

Wherever a recommendation appears, designers would be wise to take the advice given. Doing otherwise might result in some awkward problems or poor performance. While the VMEbus has been designed to support high performance systems, it is possible to design a VMEbus system that complies with all the rules, but has abysmal performance. In many cases, a designer needs a certain level of experience with the VMEbus in order to design boards that deliver top performance. Recommendations found in this document are based on this kind of experience and are provided to designers to speed their traversal of the learning curve.

SUGGESTION chapter.number

In the VMEbus specification, a suggestion contains advice which is helpful but not vital. The reader is encouraged to consider the advice before discarding it. Some design decisions that need to be made in designing VMEbus boards are difficult until experience has been gained with the VMEbus. Suggestions are included to help a designer who has not yet gained this experience. Some suggestions have to do with designing boards that can be easily reconfigured for compatibility with other boards, or with designing the board to make the job of system debugging easier.

PERMISSION chapter.number

In some cases a VMEbus rule does not specifically prohibit a certain design approach, but the reader might be left wondering whether that approach might violate the spirit of the rule, or whether it might lead to some subtle problem. Permissions reassure the reader that a certain approach is acceptable and will cause no problems. The upper-case word **MAY** is reserved exclusively for stating permissions in this document and is not used for any other purpose.

OBSERVATION chapter.number

Observations do not offer any specific advice. They usually follow naturally from what has just been discussed. They spell out the implications of certain VMEbus rules and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain rules, so that the reader understands why the rule must be followed.

1.5.1 Signal line states

The VMEbus specification describes its protocol in terms of levels and transitions on bus lines.

A signal line is always assumed to be in one of two levels or in transition between these levels. Whenever the term “high” is used, it refers to a high TTL voltage level. The term “low” refers to a low TTL voltage level. A signal line is “in transition” when its voltage is moving between these levels. (See clause 6 for voltage thresholds used on the VMEbus.)

There are two possible transitions which can appear on a signal line, and these are called “edges”. A rising edge is the time during which a signal level makes its transition from a low level to a high level. The falling edge is the time during which a signal level makes its transition from a high level to a low level.

Some bus specifications prescribe maximum or minimum rise and fall times for these edges. The problem with doing this is that board designers have very little control over these times. If the backplane is heavily loaded, the rise and fall times will be long. If it is lightly loaded, these times might be short. Even if designers know what the maximum and minimum loading will be, they still need to spend time in the lab, experimenting to find out which drivers will provide the needed rise and fall times.

In fact, rise and fall times are the result of a complex set of interactions involving the signal line impedances of the backplane, its terminations, the source impedance of the drivers, and the capacitive loading of the signal line. In order to trade off all of these factors the board designer would have to study transmission line theory, as well as certain specific parameters of drivers and receivers which are not normally found in most manufacturers' data sheets.

Recognizing all of this, the VMEbus standard doesn't specify rise and fall times. Instead, it specifies the electrical characteristics for drivers and receivers and specifies the backplane design. It also tells designers how the worst case bus loading will affect the propagation delay of these drivers so that they can ensure that the VMEbus timing is met before building a board. If VMEbus designers follow these propagation delay guidelines, their boards will operate reliably with other VMEbus compatible boards under worst case conditions.

System performance is influenced by the rise time of one of the open collector control signals. To improve the performance of the system, the concept of a rescinding signal has been introduced. A rescinding signal is an open collector type output that is initially driven high and then released back to open collector mode within a short period of time.

1.5.2 Use of the asterisk (*)

To help define usage, signal mnemonics have an asterisk suffix where required.

- a) An asterisk (*) following the signal name of signals which are level significant denotes that the signal is true or valid when the signal is low.
- b) An asterisk (*) following the signal name of signals which are edge significant denotes that the actions initiated by that signal occur on a high to low transition.

OBSERVATION 1.1

The asterisk is inappropriate for the asynchronously running clock line SYSCLK. There is no fixed phase relationship between this clock line and other VMEbus timing.

1.5.3 Keyword numbering

This standard is based on previous standards. To preserve keyword numbering, new keywords (rules, recommendation, suggestions, permissions, observations) have been numbered starting from the last available number. Hence, keywords as presented throughout this standard are not necessary in sequential order. See keyword cross-reference index to determine where a specific keyword is referenced.

1.6 Protocol specification

There are two layers of VMEbus protocol. The lowest VMEbus layer, called the backplane access layer, is composed of the backplane interface logic, the Utility Bus modules, and the Arbitration Bus modules. The VMEbus data transfer layer, is composed of the Data Transfer Bus and Priority Interrupt Bus modules. Figure 2 shows this layering.

OBSERVATION 1.2

The signal lines used by the data transfer layer modules form a special class because they are driven by different modules at different times. They are driven with line drivers that can be turned on and off at each board based upon signals generated in the backplane access layer. It is very important that their turn-on and turn-off times be carefully controlled to prevent two drivers from attempting to drive the same signal line to different levels. Special timing diagram notation is used in this document to specify their turn-on and turn-off times. It is shown in Figure 3.

There are two basic kinds of protocol used on the VMEbus: closed loop protocols and open loop protocols. Closed loop protocols use interlocked bus signals while open loop protocols use broadcast bus signals.

1.6.1 Interlocked bus signals

An interlocked bus signal is sent from a specific module to another specific module. The signal is acknowledged by the receiving module. An interlocked relationship exists between the two modules until the signal is acknowledged.

For example, an Interrupter can send an interrupt request which is answered later with an interrupt acknowledge signal (no time limit is prescribed by the VMEbus specification). The Interrupter doesn't remove the interrupt request until the Interrupt Handler acknowledges it.

Interlocked bus signals coordinate internal functions of the VMEbus system, as opposed to interacting with external stimuli. Each interlocked signal has a source module and a destination module within the VMEbus system.

The address strobe and data strobes are especially important interlocking signals. They are interlocked with the data transfer acknowledge and bus error signals and coordinate the transfer of addresses and data which are the basis for all information flow between modules in the data transfer layer.

1.6.2 Broadcast bus signal

A module generates a broadcast signal in response to an event. There is no protocol for acknowledging a broadcast signal. Instead, the broadcast is maintained for a minimum specified time, long enough to assure that all appropriate modules detect the signal. Broadcast signals might be activated at any time, irrespective of any other activity taking place on the bus. They are each sent over a dedicated signal line. Some examples are the system reset and a.c. failure lines. These signal lines are not sent to any specific module, but announce special conditions to all modules.

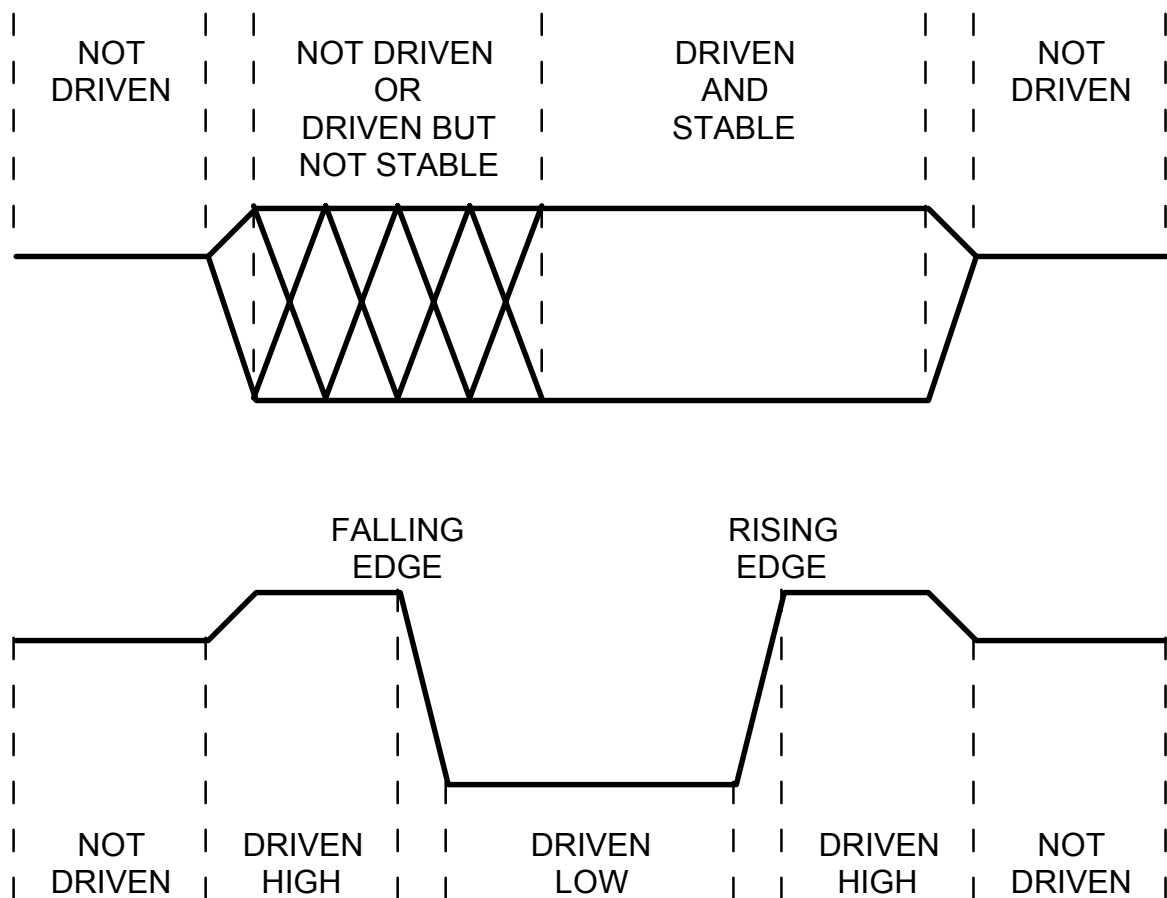


Figure 3 – Signal timing notation

1.7 System examples and explanations

A protocol specification describes, in detail, the behavior of the various functional modules. It discusses how a module responds to a signal without saying where the signal came from. Because of this, a protocol specification does not give the reader a complete picture of what is going on over the bus. To help the reader, the VMEbus specification provides examples of typical VMEbus operations. Each example shows one possible sequence of events; other sequences are also possible. In providing these examples, there is the danger that readers will assume that the sequence shown in the example is the only legal one. To help readers avoid this trap, all examples are given in a narrative style, using the present tense. This is in contrast to the imperative style used when giving rules for compliance with the VMEbus specification.

2 Data transfer bus

2.1 Introduction

This standard describes an asynchronous parallel Data Transfer Bus (DTB). Figure 4 shows a typical VMEbus system, including all of the DTB functional module types. Masters use the DTB to select storage locations provided by Slaves and to transfer data to or from those locations. Some Masters and Slaves use all of the DTB lines while others use only a subset.

Location Monitors monitor data transfers between Masters and Slaves. When an access is done to one of the byte location(s) that it monitors, a Location Monitor generates an on-board signal. For example, it might signal its on-board processor by means of an interrupt request. In such a configuration, if processor board A writes into a location of the global VMEbus memory that is monitored by processor B's Location Monitor, processor B will be interrupted.

After a Master initiates a data transfer cycle it waits for the responding Slave to respond before finishing the cycle. The asynchronous definition of the VMEbus allows a Slave to take as long as it needs to respond. If a Slave fails to respond because of some malfunction, or if the Master accidentally addresses a location where there is no Slave, the Bus Timer intervenes, allowing the cycle to be terminated.

2.1.1 Enhancements

This standard is based on VMEbus Rev C.1 and includes all the capabilities described in that standard. In addition the following enhancements have been added. For faster and wider transfers, the multiplexed block transfer (MBLT) feature is included in this standard. MBLT provides 64-bit data transfers by using both the data and address lines. MBLT protocols use the same basic asynchronous protocol as in D08(EO), D16 and D32 transfers.

Additionally, 64-bit addressing modes have been added to the existing A16, A24 and A32 modes. As memory densities increase, a single VME board could use the complete A32 address range. A64 also provides support for RISC and CISC processors with full A64 addressing.

For enhanced support of P1/J1 only systems (e.g. 3U VMEbus cards), a 40-bit address mode has been included. This mode provides access to a one terabyte address space using only the P1/J1 connector. A multiplexed data cycle (MD32) has been added to allow 32-bit transfer on P1 only. MD32 uses the data bus and the address bus for 32 bit transfers. A block transfer mode (A40BLT) allows 8-, 16- and 32-bit transfers. These modes are grouped naturally with the A24 and A16 address modes. Although intended for P1/J1 only systems nothing precludes its use in full P1/P2 systems (6U VMEbus cards).

Five lock commands were added to provide multiport resource locking. This feature better supports the RISC and CISC microprocessors with a lock line output capability. A series of operations can now be performed on a resource without interference from another processor.

The Configuration ROM/Control & Status Registers (CR/CSR) provides processor, software and operating system independence for initialization, test and configuration of a board plugged into a VME backplane. The board's manufacturer and identification are also contained in the ROM area. A board's Control and Status Registers (CSR) can be implemented in the most expedient method deemed necessary by the board designer. Use of jumpers and other manual configuration are no longer needed.

VME cycles are defined to have either one phase or two phases. Single phase cycles are typically writes where both address and data are presented on the bus at the same time. Read cycles are divided into two phases, where the address is first broadcast to all Slaves followed by a data phase where the selected Slave presents data to the bus for the Master to read. For D08(E), D08(O), D16, D32 and MD32 reads there is an address broadcast phase followed by a data phase with a single data transfer. In BLT (except the first write cycle), MBLT and A40BLT cycles there is an address broadcast phase followed by a data phase with multiple data transfers. All A40 and A64 transactions also have two phases. Except for address only cycles (without handshake), all VME transfers end with one or more combinations of DTACK*, RETRY*, and BERR* being asserted low.

NOTE The Rules 2.61 through 2.65 and 2.67 are not used in this document to avoid confusion with other VMEbus specifications.

2.2 Data transfer bus lines

The Data Transfer Bus lines are grouped into three categories:

Addressing Lines	Data Lines	Control Lines
A[31..1]	D[31..0]	AS*
D[31..0]	A[31..1]	WRITE*
LWORD*	LWORD*	BERR*
AM[5..0]		DTACK*
DS0*		DS0*
DS1*		DS1*
		RETRY*

OBSERVATION 2.1

The two data strobes (DS0* and DS1*) serve a dual function.

- The levels of these two data strobe lines are used to select which byte(s) are accessed.
- The edges of the data strobes are also used as timing signals which coordinate the transfer of data between the Master and Slave.

OBSERVATION 2.86

A[31..1] and LWORD* serve a dual function.

- During the address broadcast phase of all cycles, all or some of these signals are used to carry addressing information.
- During the data transfer phase of 64-bit block transfers, all of these signals are used to carry data.
- During the data transfer phase of A40 and A40BLT cycles, some of these signals may be used to carry data.

OBSERVATION 2.87

The data lines D[31..0] serve a dual function.

- During the data transfer of all cycles, all or some of these signals are used to carry data.
- During the address broadcast phase of cycles where A40 and A64 address modes are used, all or some of these signals are used to carry addressing information.

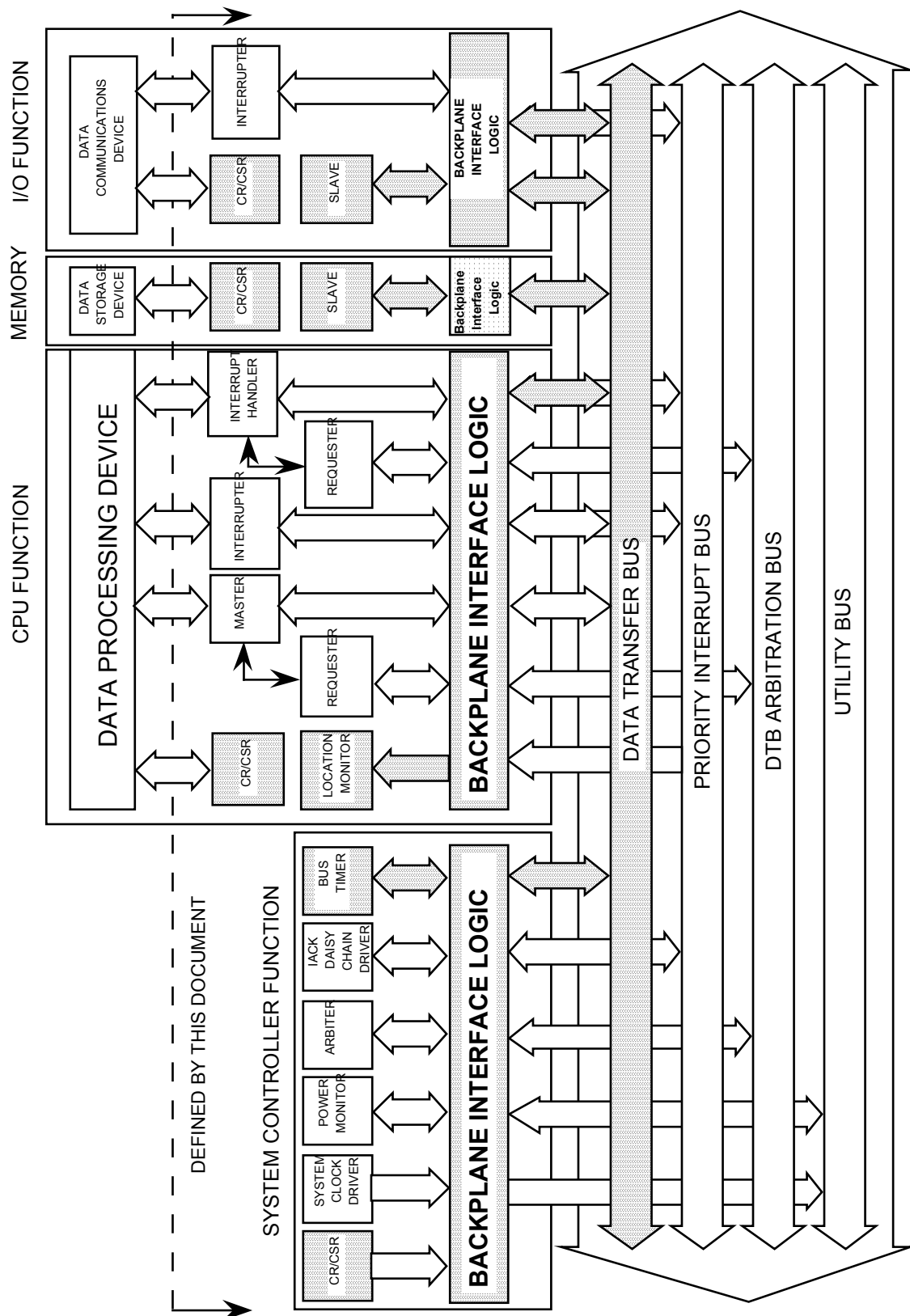


Figure 4 – Data transfer bus functional block diagram

2.2.1 Addressing lines

The smallest addressable unit of storage is a byte location. Each byte location is assigned a unique binary address. Each byte location can be assigned to one of eight categories, according to the least significant bits of its address. (See Table 1.)

Table 1 – The eight categories of byte locations

Category in a 4-byte group	Category in an 8-byte group	Byte address
Byte(0)	Byte(0)	XXXXXX.....XXXXXX000
Byte(1)	Byte(1)	XXXXXX.....XXXXXX001
Byte(2)	Byte(2)	XXXXXX.....XXXXXX010
Byte(3)	Byte(3)	XXXXXX.....XXXXXX011
Byte(0)	Byte(4)	XXXXXX.....XXXXXX100
Byte(1)	Byte(5)	XXXXXX.....XXXXXX101
Byte(2)	Byte(6)	XXXXXX.....XXXXXX110
Byte(3)	Byte(7)	XXXXXX.....XXXXXX111

NOTE 1 Byte 0 is selected by DS1* and Byte 1 is selected by DS0*.

NOTE 2 Note that in A40 mode, Byte(1) of the 40-bit address uses D[15..08]. This reduces the cost of implementation of mixed A64 and A40 bus capabilities.

A set of byte locations whose address differs only in the two least significant bits, is referred to as a 4-byte group or a Byte(0-3) group. Some or all of the bytes in a 4-byte group can be accessed simultaneously during DTB cycles.

A set of byte locations whose address differs only in the three least significant bits, is referred to as an 8-byte group or a Byte(0-7) group. All of the bytes in an 8-byte group are always accessed simultaneously. The 8-byte group is signaled by unique AM codes.

Assignment of the address bytes and bits for A16, A24, A32, A40 and A64 is shown in Table 2.

Table 2 – Address alignment on bus

Address mode	D[31..24]	D[23..16]	D[15..8]	D[7..0]	A[31..24]	A[23..16]	A[15..8]	A[7..1]
A16							A[15..8]	A[7..1]
A24						A[23..16]	A[15..8]	A[7..1]
A32					A[31..24]	A[23..16]	A[15..8]	A[7..1]
A40			A[31..24]	A[39..32]		A[23..16]	A[15..8]	A[7..1]
A64	A[63..56]	A[55..48]	A[47..40]	A[39..32]	A[31..24]	A[23..16]	A[15..8]	A[7..1]

When accessing a 4-byte group, Masters use address lines A[63..2] or a subset of these lines to select which 4-byte group is accessed. Four additional lines, DS1*, DS0*, A1 and LWORD*, are used during the address broadcast phase to select which byte locations within the 4-byte group will be accessed during the data transfer. Using these four lines, a Master can access 1, 2, 3 or 4 byte locations simultaneously as shown in Table 3.

OBSERVATION 2.2

In cycles where both data strobes are driven low, one data strobe might go low slightly after the other. In this case the signal levels indicated in Table 3 are the final levels.

OBSERVATION 2.3

Given the 4 signal line levels shown in Table 3, there are 16 possible combinations of levels. Of these 16, there are two illegal combinations that are not used (see Table in RULE 2.1a).

RULE 2.1a

Masters **MUST** ensure that none of the following combinations are generated during the address phase:

DS1*	DS0*	A1	LWORD*
high	low	high	low
low	high	high	low

PERMISSION 2.1

Masters that generate Byte(1-2) Read or Byte(1-2) Write cycles MAY generate either of the two combinations described in RULE 2.1a briefly as transition states (i.e. while one data strobe has fallen but the other has not).

OBSERVATION 2.4

Whenever a Master drives LWORD* low and A1 high it drives both data strobes low. (Any other combination is illegal.) VMEbus board designers can take advantage of this to simplify the logic on Slaves.

PERMISSION 2.2

To simplify the required logic, Slaves which respond to Byte(1-2) Read and Byte(1-2) Write cycles MAY be designed without logic to distinguish between these cycles and the two illegal cycles described in RULE 2.1a.

As will be explained in 2.2.2, Masters use special address modifier codes to indicate that they wish to access an 8-byte group. Address lines A[63..3] or a subset of these lines are used during the address broadcast phase to select which 8-byte group will be accessed during the data transfer. DS0*, DS1*, A1 and LWORD* lines are not used to select byte-locations when selecting an 8-byte group. During the MBLT data transfer phases, LWORD* and A[31..1] are used to carry data.

Also special address modifier codes are used to indicate A40 transfer modes. During A40 modes, A[15..1] and LWORD* are used along with D[15..0] to transfer a 4-byte group.

RULE 2.69

Whenever a Master identifies a cycle as an access to an 8-byte group by using one of the 64-bit block transfer AM codes (see 2.2.2), it **MUST** drive DS0*, DS1*, LWORD*, A1 and A2 low during the address phase.

Table 3 – Signal levels during data transfers used to select which byte location(s) are accessed during a data transfer

Type of cycle	DS1*	DS0*	A1	LWORD*	A2
Address-Only with No Handshake	high	high	Note 1	Note 1	high or low
Single even byte transfers					
Byte(0) Read or Write	low	high	low	high	high or low
Byte(2) Read or Write	low	high	high	high	high or low
Single odd byte transfers					
Byte(1) Read or Write	high	low	low	high	high or low
Byte(3) Read or Write	high	low	high	high	high or low
Double byte transfers					
Byte(0-1) Read or Write	low	low	low	high	high or low
Byte(2-3) Read or Write	low	low	high	high	high or low
Quad byte transfers					
Byte(0-3) Read or Write	low	low	low	low	high or low
Single byte block transfers					
Single Byte Block Read or Write	<-----	Note 2	----->	high	high or low
Double byte block transfers					
Double Byte Block Read or Write	low	low	Note 3	high	high or low
Quad byte block transfers					
Quad Byte Block Read or Write	low	low	low	low	high or low
Single byte RMW transfers					
Byte(0) Read-Modify-Write	low	high	low	high	high or low
Byte(1) Read-Modify-Write	high	low	low	high	high or low
Byte(2) Read-Modify-Write	low	high	high	high	high or low
Byte(3) Read-Modify-Write	high	low	high	high	high or low
Double byte RMW transfers					
Byte(0-1) Read-Modify-Write	low	low	low	high	high or low
Byte(2-3) Read-Modify-Write	low	low	high	high	high or low
Quad byte RMW transfers					
Byte(0-3) Read-Modify-Write	low	low	low	low	high or low
Unaligned transfers					
Byte(0-2) Read or Write	low	high	low	low	high or low
Byte(1-3) Read or Write	high	low	low	low	high or low
Byte(1-2) Read or Write	low	low	high	low	high or low
Eight Byte Access					
Byte(0-7)	low	low	low	low	low

NOTE 1 During Address-Only cycles, both data strobes are maintained high, but the A1 and LWORD* lines might be either high or low.

NOTE 2 During single byte block transfers, the two data strobes are alternately driven low. Either data strobe might be driven low on the first transfer. If the first accessed byte location is Byte(0) or Byte(2), then DS1* is driven low first. If the first accessed byte location is Byte(1) or Byte(3), then DS0* is driven low first. A1 is valid only on the first data transfer (i.e. until the Slave drives DTACK* or BERR* low the first time) and might be either high or low depending upon which byte the single byte block transfer begins with. If the first byte location is Byte(0) or Byte(1), then A1 is low. If the first byte location is Byte(2) or Byte(3), then A1 is high.

NOTE 3 During a double byte block transfer, the data strobes are driven low on each data transfer. A1 is valid only on the first data transfer (i.e. until the Slave drives DTACK* or BERR* low the first time) and might be either high or low depending upon what double byte group the double byte block transfer begins with. If the first double byte group is Byte(0-1), then A1 is low. If the first double byte group is Byte(2-3), then A1 is high.

Table 3 (continued)

An example of a single byte block transfer cycle which starts with Byte(2) is given below:

		DS1*	DS0*	A1	LWORD*
First data transfer	Byte (2)	low	high	high	high
	Byte (3)	high	low	X	X
	Byte (0)	low	high	X	X
	Byte (1)	high	low	X	X
Last data transfer	Byte (2)	low	high	X	X
X = high or low					

An example of a double byte block transfer cycle which starts with Byte(2-3) is given below:

		DS1*	DS0*	A1	LWORD*
First data transfer	Byte (2-2)	low	low	high	high
	Byte (0-2)	low	low	X	X
	Byte (2-3)	low	low	X	X
	Byte (0-1)	low	low	X	X
X = high or low					

2.2.2 Address modifier lines

There are 6 address-modifier lines. The Master uses these lines to pass additional binary information to the Slave during data transfers. Table 4 lists all of the 64 possible address modifier (AM) codes and classifies each into one of three categories:

Defined
Reserved
User defined

The defined address modifier codes can be further classified into five categories as follows.

- A16 addressing AM codes, which indicate that address lines A[15..2] are being used to select a Byte(0-3) group.
- A24 addressing AM codes, which indicate that address lines A[23..2] are being used to select a Byte(0-3) group or address lines A[23..3] for selecting a Byte(0-7) group.
- A32 addressing AM codes, which indicate that address lines A[31..2] are being used to select a Byte(0-3) group or address lines A[31..3] for selecting a Byte(0-7) group.
- A40 addressing AM codes, which indicate that address lines A[23..2] and data lines D[15..0] are being used to select a Byte(0-3) group.
- A64 addressing AM codes, which indicate that address lines A[31..2] and data lines D[31..0] are being used to select a Byte(0-3) group or address lines A[31..3] and data lines D[31..0] for selecting a Byte(0-7) group.

RULE 2.2

Except for the user-defined codes, the codes defined in Table 4 **MUST NOT** be used for purposes other than those specified.

RULE 2.3

VMEbus Slaves boards **MUST NOT** respond to reserved address-modifier codes.

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 the use of these codes is defined.

PERMISSION 2.3

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

Table 4 – Address modifier codes

HEX CODE	ADDRESS MODIFIER						FUNCTION
	5	4	3	2	1	0	
3F	H	H	H	H	H	H	A24 supervisory block transfer (BLT)
3E	H	H	H	H	H	L	A24 supervisory program access
3D	H	H	H	H	L	H	A24 supervisory data access
3C	H	H	H	H	L	L	A24 supervisory 64-bit block transfer (MBLT)
3B	H	H	H	L	H	H	A24 non-privileged block transfer (BLT)
3A	H	H	H	L	H	L	A24 non-privileged program access
39	H	H	H	L	L	H	A24 non-privileged data access
38	H	H	H	L	L	L	A24 non-privileged 64-bit block transfer (MBLT)
37	H	H	L	H	H	H	A40BLT
36	H	H	L	H	H	L	Reserved
35	H	H	L	H	L	H	A40 lock command (LCK)
34	H	H	L	H	L	L	A40 access
33	H	H	L	L	H	H	Reserved
32	H	H	L	L	H	L	A24 lock command (LCK)
31	H	H	L	L	L	H	Reserved
30	H	H	L	L	L	L	Reserved
2F	H	L	H	H	H	H	Configuration ROM/Control & Status Register (CR/CSR)
2E	H	L	H	H	H	L	Reserved
2D	H	L	H	H	L	H	A16 supervisory access
2C	H	L	H	H	L	L	A16 lock command (LCK)
2B	H	L	H	L	H	H	Reserved
2A	H	L	H	L	H	L	Reserved
29	H	L	H	L	L	H	A16 non-privileged access
28	H	L	H	L	L	L	Reserved
27	H	L	L	H	H	H	Reserved
26	H	L	L	H	H	L	Reserved
25	H	L	L	H	L	H	Reserved
24	H	L	L	H	L	L	Reserved
23	H	L	L	L	H	H	Reserved
22	H	L	L	L	H	L	Reserved
21	H	L	L	L	L	H	Reserved
20	H	L	L	L	L	L	Reserved
L = low-signal level							H = high-signal level

Table 4 (continued)

HEX CODE	ADDRESS MODIFIER						FUNCTION
	5	4	3	2	1	0	
1F	L	H	H	H	H	H	User-defined
1E	L	H	H	H	H	L	User-defined
1D	L	H	H	H	L	H	User-defined
1C	L	H	H	H	L	L	User-defined
1B	L	H	H	L	H	H	User-defined
1A	L	H	H	L	H	L	User-defined
19	L	H	H	L	L	H	User-defined
18	L	H	H	L	L	L	User-defined
17	L	H	L	H	H	H	User-defined
16	L	H	L	H	H	L	User-defined
15	L	H	L	H	L	H	User-defined
14	L	H	L	H	L	L	User-defined
13	L	H	L	L	H	H	User-defined
12	L	H	L	L	H	L	User-defined
11	L	H	L	L	L	H	User-defined
10	L	H	L	L	L	L	User-defined
0F	L	L	H	H	H	H	A32 supervisory block transfer (BLT)
0E	L	L	H	H	H	L	A32 supervisory program access
0D	L	L	H	H	L	H	A32 supervisory data access
0C	L	L	H	H	L	L	A32 supervisory 64-bit block transfer (MBLT)
0B	L	L	H	L	H	H	A32 non-privileged block transfer (BLT)
0A	L	L	H	L	H	L	A32 non-privileged program access
09	L	L	H	L	L	H	A32 non-privileged data access
08	L	L	H	L	L	L	A32 non privileged 64-bit block transfer (MBLT)
07	L	L	L	H	H	H	Reserved
06	L	L	L	H	H	L	Reserved
05	L	L	L	H	L	H	A32 lock command (LCK)
04	L	L	L	H	L	L	A64 lock command (LCK)
03	L	L	L	L	H	H	A64 block transfer (BLT)
02	L	L	L	L	H	L	Reserved
01	L	L	L	L	L	H	A64 single transfer access
00	L	L	L	L	L	L	A64 64-bit block transfer (MBLT)
L = low-signal level				H = high-signal level			

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.

2.2.3 Data lines

VMEbus systems can be built with a backplane configuration that provides either 16 data lines, D[15..0], or 32 data lines, D[31..0]. Backplane configurations that provide 16 data lines allow a Master to access up to 2 or 4 byte locations simultaneously, while those with 32 data lines allow it to access up to 8 byte locations simultaneously. When the Master has selected 1, 2, 3, 4 or 8 byte locations, using the method described in 2.2.1 and 2.2.2, it can transfer binary data between itself and those locations. Table 5 shows how the data lines and the address lines are used to access byte locations.

RULE 2.70

When accessing Byte(0-7), LWORD* **MUST** carry the least significant bit of Byte(3) and A7 **MUST** carry the most significant bit of Byte(3). When accessing Byte(0-3) in MD32 Mode, LWORD* **MUST** carry the least significant bit of Byte(1) and A7 **MUST** carry the most significant bit of Byte(1). (See Table 6.)

PERMISSION 2.4

The data sender (Master for a write cycle; Slave for a read cycle) **MAY** drive data lines which are not used to transfer data.

Table 5 – Use of data lines to move data during nonmultiplexed data transfers

During the following types of cycles...	the data lines are used to transfer data as shown below:			
	D[31..24]	D[23..16]	D[15..8]	D[7..0]
Address-Only	<----- no bytes transferred ----->			
Single even byte transfers Byte(0) Read, Write or RMW Byte(2) Read, Write or RMW			Byte(0) Byte(2)	
Single odd byte transfers Byte(1) Read, Write or RMW Byte(3) Read, Write or RMW				Byte(1) Byte(3)
Double byte transfers Byte(0-1) Read, Write or RMW Byte(2-3) Read, Write or RMW			Byte(0) Byte(2)	Byte(1) Byte(3)
Quad byte transfers Byte(0-3) Read, Write or RMW	Byte(0)	Byte(1)	Byte(2)	Byte(3)
Single byte block transfers Single Byte Block Read or Write			<----- Note 1 ----->	
Double byte block transfers Double Byte Block Read or Write			<----- Note 2 ----->	
Quad byte block transfers Quad Byte Block Read or Write	Byte(0)	Byte(1)	Byte(2)	Byte(3)
Unaligned transfers Byte(0-2) Read or Write Byte(1-3) Read or Write Byte(1-2) Read or Write	Byte(0)	Byte(1) Byte(1) Byte(1)	Byte(2) Byte(2) Byte(2)	Byte(3)
*NOTE RMW stands for Read-Modify-Write.				

15776-2001

Table 5 (continued)

Examples

- a) During single byte block transfers, data is transferred 8 bits at a time over D[15..8] or D[7..0]. One example of this is given below:

	D[31..24]	D[23..16]	D[15..8]	D[7..0]
First data transfer			Byte(2)	Byte(1)
			Byte(0)	Byte(3)
			Byte(2)	Byte(1)
Last data transfer			Byte(2)	Byte(3)

- b) During double byte block transfers, data is transferred 16 bits at a time over D[15..0]. One example of this is given below:

	D[31..24]	D[23..16]	D[15..8]	D[7..0]
First data transfer			Byte(2)	Byte(3)
			Byte(0)	Byte(1)
			Byte(2)	Byte(3)
			Byte(0)	Byte(1)
			Byte(2)	Byte(3)
			Byte(0)	Byte(1)
Last data transfer			Byte(2)	Byte(3)
			Byte(0)	Byte(1)

Table 6 – Use of the address and data lines for multiplexed data cycles

A[31..24]	A[23..16]	A[15..8]	A[7..1]	D[31..24]	D[23..16]	D[15..8]	D[7..0]
LWORD*							
Multiplexed Quad Byte transfers (MD32)							
Byte (0-3) Read, Write or RMW							
		Byte(0)	Byte(1)			Byte(2)	Byte(3)
Multiplexed Quad Byte Block transfers (MD32)							
Byte (0-3) Read or Write							
		Byte(0)	Byte(1)			Byte(2)	Byte(3)
Multiplexed Eight Byte Block Transfer (MBLT)							
Byte (0-7) Read or Write							
	Byte(0)	Byte(1)	Byte(2)	Byte(3)	Byte(4)	Byte(5)	Byte(6)
							Byte(7)

2.2.4 Data transfer bus control lines

The following signal lines are used to control the transfer of data over the data transfer lines:

AS*	Address Strobe
DS0*	Data Strobe Zero
DS1*	Data Strobe One
BERR*	Bus Error
DTACK*	Data Transfer Acknowledge
RETRY*	Retry
WRITE*	Read/Write

2.2.4.1 AS*

A falling edge on AS* informs all Slaves that the addressing information on all or some of the A[31..1] lines, LWORD* and AM(5..0) are stable and can be captured.

2.2.4.2 DS0* and DS1*

In addition to their function in selecting byte locations for data transfer, as described in 2.2.1, the data strobes also serve additional functions. On write cycles, the first falling edge of a data strobe indicates that the Master has placed valid data on the data bus. On read cycles, the first rising edge tells the Slave that it can remove valid data from the data bus.

When using the 64-bit or 40-bit addressing mode, the first falling edge of a data strobe indicates that the Master has placed valid addressing information on the data lines.

OBSERVATION 2.7

VMEbus Masters are not permitted to drive either of the data strobes low before driving AS* low. However, due to the fact that AS* might be more heavily loaded on the backplane than the data strobes, Slaves and Location Monitors might detect a falling edge on a data strobe, before they detect the falling edge on AS*.

PERMISSION 2.5a

Slaves and Location Monitors that do not support block transfer capability as described in 2.3.7 MAY be designed to capture AM[5..0], A[31..1] and LWORD* when they detect a falling edge on a data strobe instead of on the falling edge of AS*.

OBSERVATION 2.8

VMEbus Slaves and Location Monitors that capture the address on the falling edge of the data strobe(s) need not monitor AS*.

OBSERVATION 2.9a

To perform block read and write cycles as described in 2.3.7 or to take full advantage of address pipelining as described in 2.4.2, a Slave should capture the address on the falling edge of the data strobe (DS0* or DS1*) that falls first while AS* is low.

2.2.4.3 DTACK*

On a Write cycle, the Slave drives DTACK* low to indicate that it has successfully received all of the data called for by the transfer type. On a read cycle, the Slave drives DTACK* low to indicate that it has placed data on the data bus. During A64, A40, MBLT, and A40BLT address phases, DTACK* is driven low by the Slave to indicate it can receive or send data in the data phase. For lock commands the Slave drives DTACK* low indicating it has accepted the lock command and has locked out the other ports.

2.2.4.3.1 Rescinding DTACK*

Slaves are allowed to drive DTACK* high before releasing it at the end of a cycle, or during BLT, MBLT and A40BLT operations. Driving DTACK* high, versus letting DTACK* be pulled high by the backplane termination resistors provides less delay at the end of the cycle or data transfer and improves system performance. Figure 36 shows the timing parameters for Rescinding DTACK*. The timing parameter values are listed in Table 27.

PERMISSION 2.14

A selected Slave MAY drive DTACK* high for a short time, as specified by Timing Parameter 30A at the end of a cycle and during BLT, MBLT and MD32 data transfer phases after the data strobes go high.

2.2.4.4 BERR*

BERR* is driven low by the Slave or by the Bus Timer to indicate to the Master that the data transfer or address broadcast was unsuccessful or an error was detected. For example, when a Master tries to write to a location that contains Read-Only Memory, the responding Slave should drive BERR* low. When the Master tries to access a location that is not provided by any Slave, the Bus Timer drives BERR* low after a time-out period has expired.

RECOMMENDATION 2.2

Design Slaves to drive BERR* low when they detect an uncorrectable error in the data they retrieve from their internal storage during a read cycle. Also, design Slaves to drive BERR* low when a bus operation tries to write to devices that are read only.

PERMISSION 2.6

VMEbus Slaves MAY be designed without a BERR* driver.

2.2.4.5 WRITE*

WRITE* is a level-significant signal line. It is used by the Master to indicate the direction of data transfer operations. When WRITE* is driven low, the data transfer direction is from the Master to the Slave. When WRITE* is driven high, the data transfer direction is from the Slave to the Master.

2.2.4.6 RETRY*

RETRY* can be driven low by the responding Slave to indicate to the Master that the requested data transfer cannot be executed at this time. The Master that supports RETRY* should retry the data transfer at a later time. This operation may be used to break deadlock situations.

The Master waits for a short period of time before trying the operation again. In the interim, the bus can be used for other transactions.

See 2.3.13 for rules, recommendations, permissions and observations governing the use of the RETRY* line.

2.3 DTB modules – Basic description

The DTB protocols define the methods used to transfer data. Five addressing modes are provided:

- a) A16 (16-bit)
- b) A24 (24-bit)
- c) A32 (32-bit)
- d) A40 (40-bit)
- e) A64 (64-bit).

The capabilities of Masters, Slaves, Location Monitors and CR/CSRs are described by a list of mnemonics that show what cycle types they can generate, accept or monitor respectively. (This will be described in more detail later in this chapter.)

Subclauses 2.3.1 through 2.3.5 provide block diagrams for the five types of DTB functional modules: Master, Slave, Location Monitor, Bus Timer and CR/CSR.

RULE 2.7

Output signal lines shown with solid lines in Figures 5 through 8 **MUST** be driven by the respective module, unless it would always drive them high.

OBSERVATION 2.11

IF an output signal line is not driven,
THEN terminators on the backplane ensure that it is high.

RULE 2.8

Input signal lines shown with solid lines in Figures 5 through 8 **MUST** be monitored and responded to in the appropriate fashion.

OBSERVATION 2.12

RULES and PERMISSIONS for driving and monitoring signal lines shown with dotted lines in Figures 5 through 8 are given in Tables 8, 9, and 11.

2.3.1 Master

The block diagram of the Master is shown in Figure 5. The dotted lines in the diagram show signals whose use varies among the various types of Masters. Table 7 specifies how the various types of Masters use these lines. Further information about how the various types of Masters drive the address line, the data lines, LWORD*, DS0*, DS1* and A1 is given in Tables 3, 24, 25.

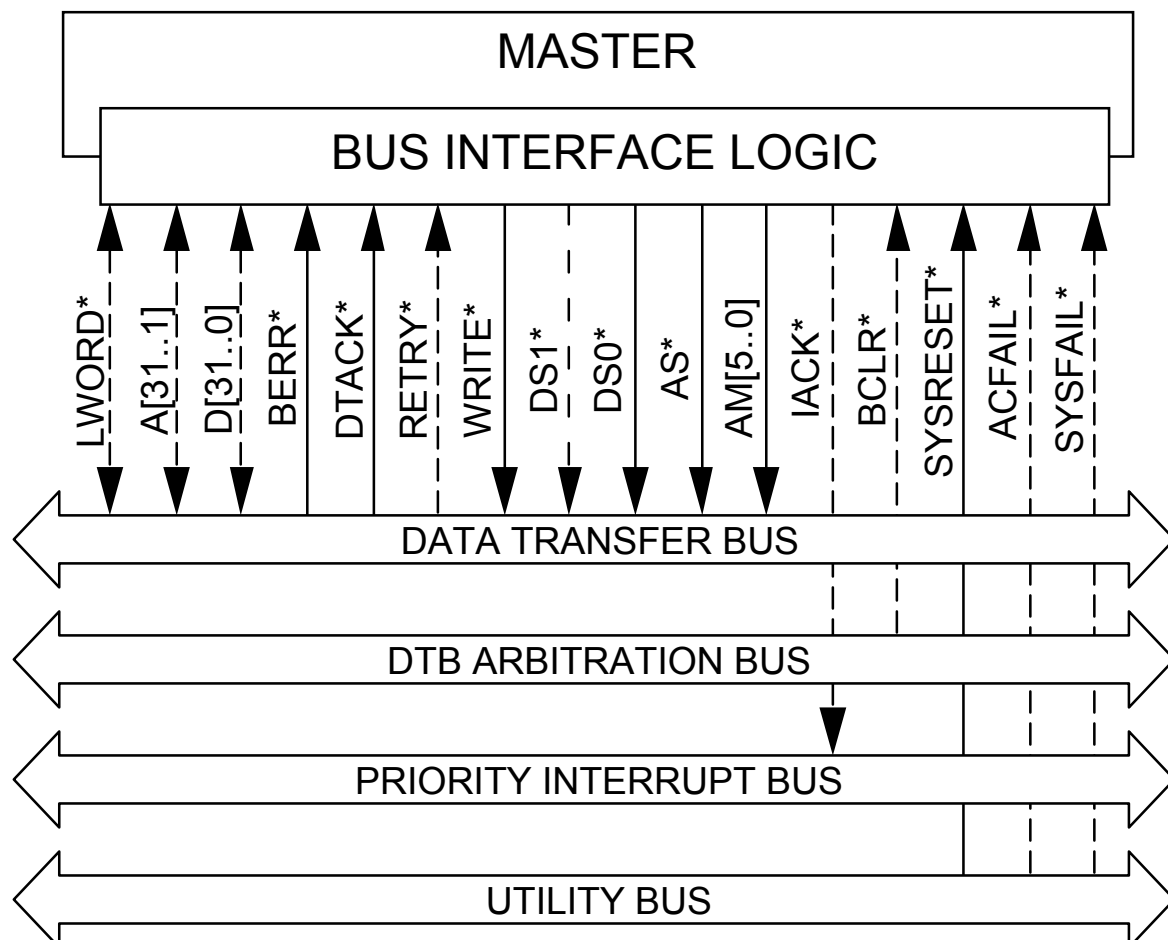


Figure 5 – Block diagram – Master

Table 7 – RULES and PERMISSIONS specifying the use of the dotted lines by the various types of masters

Type of master	Use of dotted lines
D08(O)	MUST monitor and drive D[7..0] MAY monitor and drive LWORD* MAY monitor and drive D[31..8]
D08(E0)	MUST drive DS0* or DS1, but not both low on the same data transfer MUST monitor and drive D[15..0] MAY monitor and drive LWORD* MAY monitor and drive D[31..16]
D16 Single & BLT	MUST drive DS1* MUST monitor and drive D[15..0] MAY monitor and drive LWORD* MAY monitor and drive D[31..16]
D32 Single & BLT	MUST drive DS1* MUST drive LWORD* MUST monitor and drive D[31..0] MAY monitor LWORD*
MD32	MUST drive DS1* MUST monitor and drive LWORD* MUST monitor and drive D[15..0] MUST monitor and drive A[15..1]
MBLT	MUST drive DS1* MUST monitor and drive LWORD* MUST monitor and drive D[31..0] MUST monitor and drive A[31..1]
A16	MUST drive A[15..1] MAY drive A[31..16]
A24	MUST drive A[23..1] MAY drive A[31..24]
A32	MUST drive A[31..1]
A40 & A40BLT	MUST drive DS1* MUST drive A[23..1] MUST drive D[15..0] MUST drive LWORD*
A64	MUST drive DS1* MUST drive A[31..1] MUST drive D[31..0] MUST drive LWORD*
ALL	MAY monitor RETRY* MAY monitor BCLR* (see clause 3) MAY monitor ACFAIL* (see clause 5) MAY monitor SYSFAIL* (see clause 5) MUST NOT drive IACK* low
NOTE 1 The mnemonics A16, A24, A32, A40 and A64 are defined in Table 11. NOTE 2 The mnemonics D08(E0), D16, D32 and MD32 are defined in Table 12. NOTE 3 The mnemonics BLT, MBLT and A40BLT are defined in Table 13.	

2.3.2 Slave

The block diagram of the Slave is shown in Figure 6. The dotted lines in the diagram show signals whose use varies among the various types of Slaves. Table 8 shows how the various types of Slaves use these lines. Further information about how the various types of Slaves drive the data lines is given in Table 27.

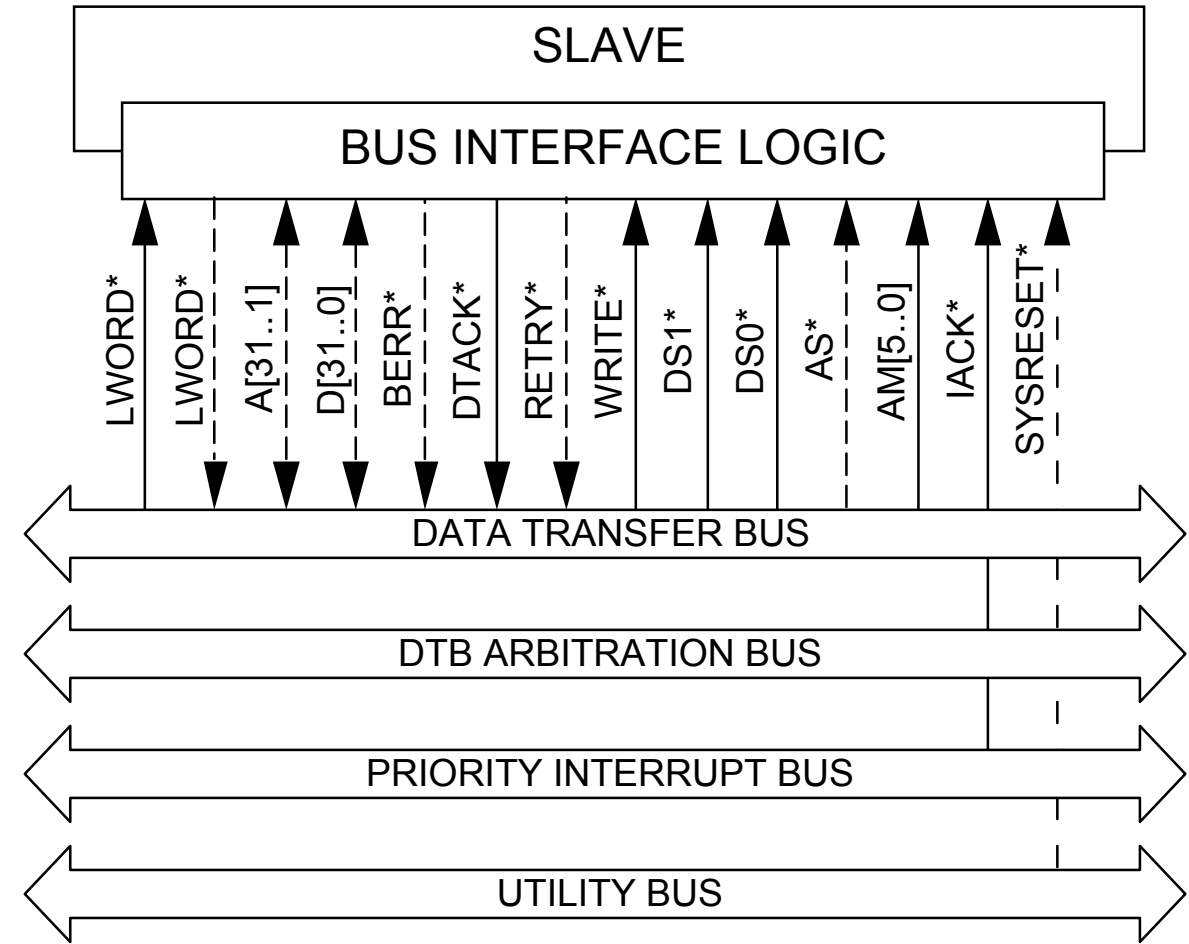


Figure 6 – Block diagram – Slave

Table 8 – Slaves – RULEs and PERMISSIONs specifying the use of the dotted lines by the various types of slaves

Type of slave	Use of dotted lines
D08(O)	MUST monitor and/or drive D[7..0] MAY monitor AS* MAY monitor or drive D[31..8]
D08(E0) and D16	MUST monitor and/or drive D[15..0] MAY monitor AS* MAY monitor or drive D[31..16]
D32	MUST monitor and/or drive D[31..0] MAY monitor AS*
MD32	MUST monitor and/or drive A[15..1] MUST monitor and/or drive LWORD* MUST monitor and/or drive D[15..0] MUST monitor AS*
MBLT	MUST monitor and/or drive D[31..0] MUST monitor and/or drive A[31..1] MUST monitor AS* MUST monitor and/or drive LWORD*
A16	MUST monitor A[15..1] MAY monitor A[31..16]
A24	MUST monitor A[23..1] MAY monitor A[31..24]
A32	MUST monitor A[31..1]
A40 & A40BLT	MUST monitor D[15..0] MUST monitor A[23..1] MUST monitor AS*
A64	MUST monitor A[31..1] MUST monitor D[31..0] MUST monitor AS*
ALL	MAY drive RETRY* MAY drive BERR* MAY monitor SYSRESET*
NOTE 1 The mnemonics D08(O), D08(E0), D16 and D32 are defined in Table 12. NOTE 2 The mnemonics BLT, MBLT and A40BLT are defined in Table 13. NOTE 3 The mnemonics A16, A24, A32, A40 and A64 are defined in Table 11.	

2.3.3 Bus timer

The block diagram of the Bus Timer is shown in Figure 7. Bus Timers can be designed to drive BERR* low after various periods of time. Table 9 shows how the BTO() mnemonic is used to describe the various types of Bus Timers.

OBSERVATION 2.13

The dashed DTACK* and BERR* lines shown in Figure 7 allow designers to implement a bus timer in one of two ways:

- to drive BERR* low when the first data strobe stays low for longer than the bus time-out period, regardless of the levels on the DTACK* and BERR* lines;
- to drive BERR* low when the first data strobe stays low for longer than the bus time-out period, but only when both DTACK* and BERR* are high at the point of time-out.

Table 9 – Use of BTO() mnemonic specifying the time-out period of bus timers

The following mnemonic	when applied to a	means that it
BTO(x)	Bus Timer	drives BERR* low when the first data strobe stays low for longer than $x \mu\text{s}$, without DTACK* being asserted low

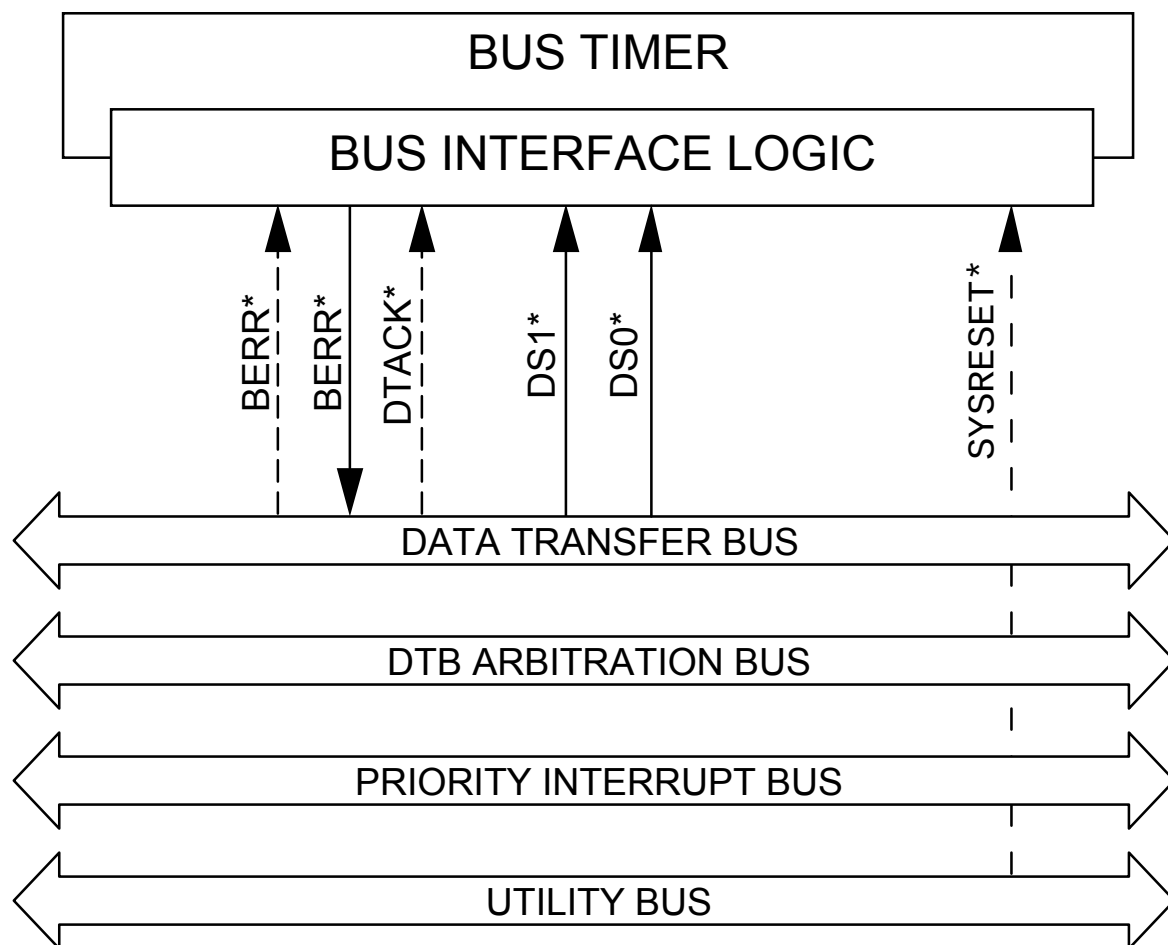


Figure 7 – Block diagram – Bus timer

2.3.4 Location monitor

The block diagram of the Location Monitor is shown in Figure 8. The dotted lines in the diagram show signals whose use varies among the various types of Location Monitors. Table 10 shows how the various types of Location Monitors use these lines.

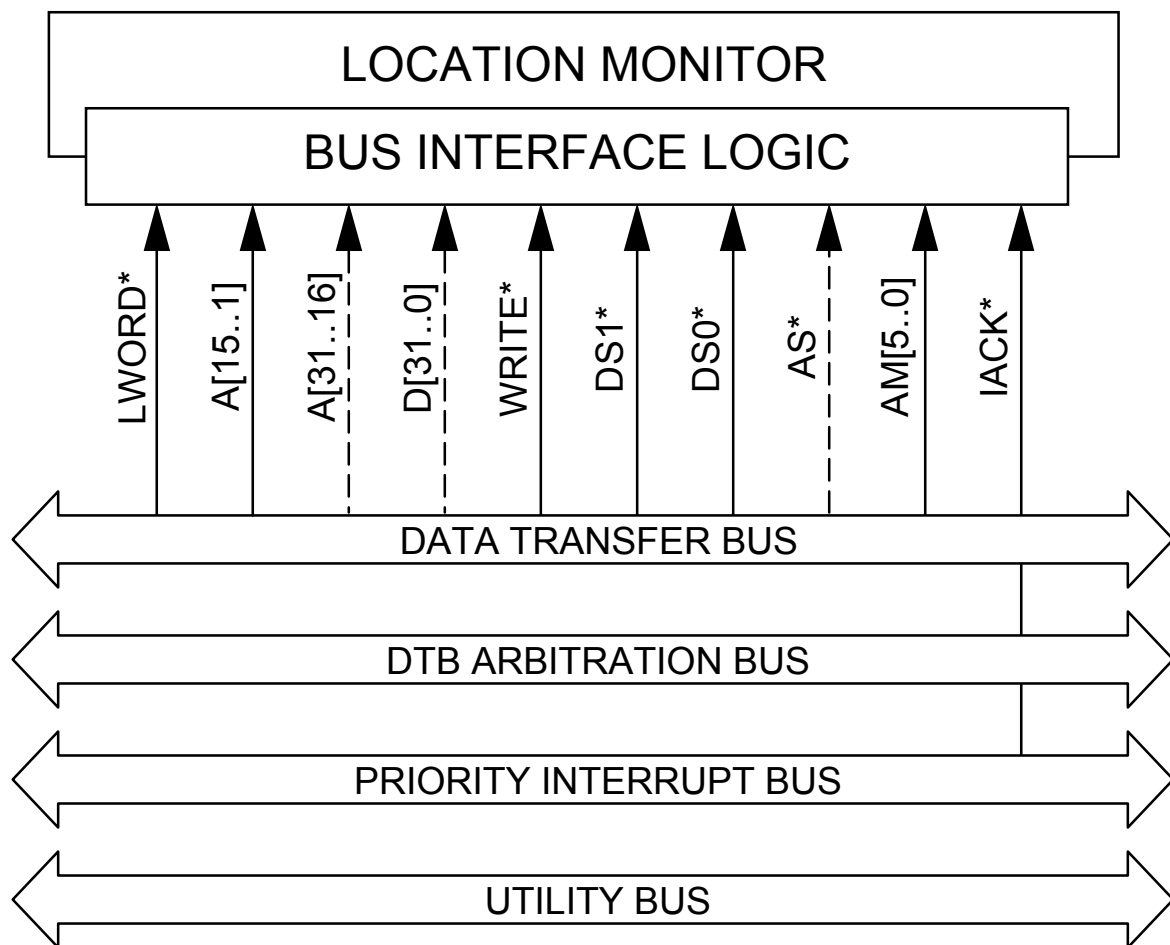


Figure 8 – Block diagram – Location monitor

Table 10 – Location monitors – RULEs and PERMISSIONs specifying the use of dotted lines by various types of location monitors

Type of slave	Use of dotted lines
A16	MUST monitor A[15..1] MAY monitor A[31..1]
A24	MUST monitor A[23..1] MAY monitor A[31..24]
A32	MUST monitor A[31..1]
A40	MUST monitor A[23..1] MUST monitor D[15..0] MUST monitor AS*
A64	MUST monitor A[31..1] MUST monitor D[31..0] MUST monitor AS*
ALL	MAY monitor AS*
NOTE The mnemonics A16, A24, A32, A40 and A64 are defined in Table 11.	

2.3.5 Addressing phases and modes

Masters broadcast an address over the DTB at the beginning of each cycle. This broadcast address might be a 16-bit, a 24-bit, a 32-bit, a 40-bit or a 64-bit address, depending on the capabilities of the Master broadcasting it. 16-bit addresses are referred to as A16 addresses, 24-bit addresses are referred to as A24 addresses, 32-bit addresses are referred to as A32 addresses, 40-bit addresses are referred to as A40 addresses and 64-bit addresses are referred to as A64 addresses.

Table 11 shows the various mnemonics used to describe the addressing capabilities and how each is used to describe Masters, Slaves, Location Monitors and CR/CSRs.

The Master broadcasts an Address Modifier (AM) code along with each address to tell Slaves whether the address is A16, A24, A32, A40 or A64.

RULE 2.9

Slave boards **MUST** decode all of the address modifier lines.

RULE 2.71

When broadcasting a 64-bit address, A64 Masters **MUST** drive the least significant 32-bits of the address on LWORD* and A[31..1] and the most significant 32-bits of address on D[31..0]. Refer to Table 2.

OBSERVATION 2.14a

Decoding all the address modifier lines allows a Slave to differentiate A16, A24, A32, A40 or A64 address mode as well as the specific operation within each address mode.

OBSERVATION 2.15a

In addition to the five modes of addressing described here, there is a sixth mode that is used on interrupt acknowledge cycles (see clause 4). These interrupt acknowledge cycles can be distinguished from data transfer cycles by the fact that IACK* is low instead of high.

RULE 2.10

Whenever a Master broadcasts an address, it **MUST** ensure that IACK* is high.

PERMISSION 2.7

The Master MAY either drive IACK* high during the address broadcast or it MAY leave IACK* undriven (the bus terminators will then hold it high).

RULE 2.11

Slaves **MUST NOT** respond to DTB cycles when IACK* is low.

OBSERVATION 2.88

Systems may include a mixture of A16, A24, A32, A40 and A64 Masters and Slaves.

Table 11 – Mnemonics specifying addressing capabilities

The following mnemonic	when applied to a	means that it
A16	Master Slave Location Monitor	MUST generate cycles with A16 addresses MUST accept cycles with A16 addresses MUST monitor cycles with A16 addresses
A24	Master Slave Location Monitor CR/CSR	MUST generate cycles with A24 & A16 addresses MUST accept cycles with A24 addresses MUST monitor cycles with A24 addresses MUST monitor cycles with A24 addresses
A32	Master Slave Location Monitor	MUST generate cycles with A32, A24 & A16 addresses MUST accept cycles with A32 addresses MUST monitor cycles with A32 addresses
A40	Master Slave Location Monitor	MUST generate cycles with A40, A24 & A16 addresses MUST accept cycles with A40 addresses MUST monitor cycles with A40 addresses
A64	Master Slave Location Monitor	MUST generate cycles with A64, A32, A24 & A16 addresses MUST accept cycles with A64 addresses MUST monitor cycles with A64 addresses

RULE 2.72

A64 Masters **MUST** include the A32, A24 and A16 capabilities.

RULE 2.73

A40 Masters **MUST** include the A24 and A16 capabilities.

RULE 2.74

A32 Masters **MUST** include the A24 and A16 capabilities.

RULE 2.75

A24 Masters **MUST** include the A16 capability.

SUGGESTION 2.6a

Do not assume that the above RULEs are known to the readers of product specifications. Rather, specify an A64 Master product as a "A64, A32, A24 and A16 Master". Specify an A32 Master product as a "A32, A24 and A16 Master". Similarly, specify an A24 Master product as a "A24 and A16 Master".

OBSERVATION 2.89

A64 and A32 Masters MAY include A40 capability for interoperability with 3U products.

2.3.6 Basic data transfer capabilities

There are five basic data transfer capabilities associated with the DTB: D08(O) (Odd byte only), D08(EO) (Even and Odd byte), D16, D32 and MD32. These capabilities allow flexibility when interfacing different types of processors and peripherals to the bus.

Eight-bit processors can be interfaced to the bus as D08(EO) Masters. Sixteen-bit processors can be interfaced to the bus as D16 Masters. The D16 Slave is useful for interfacing 16-bit memory devices or 16-bit I/O devices to the DTB.

Many existing peripheral chips have registers that are only 8 bits wide. While these chips often have several of these registers, they cannot provide the contents of two registers simultaneously when a D16 Master attempts to access two adjacent locations with a double-byte read cycle. These 8-bit peripheral ICs can be interfaced to the DTB as D08(O) Slaves, which provide only Byte(1) and Byte(3) locations and respond only to single-odd-byte accesses. This simplifies the D08(O) Slave's interface logic, since single-odd-byte accesses always take place over D[7..0].

RULE 2.76

D16 Slaves **MUST** include D08(EO) capability.

RECOMMENDATION 2.3

D16 Masters should include D08(EO) capability.

RULE 2.77

D32 and MD32 Slaves **MUST** include D16 and D08(EO) capabilities.

RECOMMENDATION 2.4

D32 and MD32 Masters should include D16 and D08(EO) capabilities.

RECOMMENDATION 2.5

MBLT Masters should include D32, D16 and D08(EO) capabilities.

SUGGESTION 2.9

Do not assume that the above RULEs are known to the readers of product specifications. Specify products that have the D32 capability as "D32, D16 and D08(EO)". Similarly, specify D16 products as "D16 and D08(EO)".

RULE 2.4

D08(O), D08(EO), and D16 Slaves **MUST NOT** respond with a falling edge on DTACK* during a quad byte cycle if they do not have quad-byte capability.

RULE 2.5

D08(O) and D08(EO) Slaves **MUST NOT** respond with a falling edge on DTACK* during a double-byte cycle if it does not have double-byte capability.

SUGGESTION 2.8a

Design Slaves are to respond with a falling edge on BERR* in the following situations:

- a) when a D08(O), D08(EO) or D16 only Slave is requested to do a quad-byte cycle;

- b) when a D08(O) or D08(E0) Slave is requested to do a double-byte cycle;
- c) when a D08(O), D08(E0) or D16 Slave is requested to do an unaligned transfer (that is, a triple-byte transfer or a double-byte Byte(1-2) transfer);
- d) when a D08(O) only Slave is requested to do an even single-byte transfer (that is, a single-byte Byte(0) or Byte(2) transfer).

Table 12 shows the various mnemonics used to describe the basic data transfer capabilities and how each is used to describe Masters, Slaves and Location Monitors.

Table 12 – Mnemonics specifying basic data transfer capabilities

The following mnemonic	when applied to a	means that it
D08(E0)	Master Slave Location Monitor	MUST generate the following cycles: MUST accept the following cycles: MUST monitor the following cycles: Single byte read cycles: Byte(0) Read Byte(1) Read Byte(2) Read Byte(3) Read Single byte write cycles: Byte(0) Write Byte(1) Write Byte(2) Write Byte(3) Write
D08(O)	Slave	MUST accept the following cycles: Single byte read cycles: Byte(1) Read Byte(3) Read Single byte write cycles: Byte(1) Write Byte(3) Write
D16	Master Slave Location Monitor	MUST generate the following cycles: MUST accept the following cycles: MUST monitor the following cycles: Double byte read cycles: Byte(0-1) Read Byte(2-3) Read Double byte write cycles: Byte(0-1) Write Byte(2-3) Write
D32	Master Slave Location Monitor	MUST generate the following cycles: MUST accept the following cycles: MUST monitor the following cycles: Quad byte read cycle: Byte(0-3) Read Quad byte write cycle: Byte(0-3) Write
MD32	Master Slave Location Monitor	MUST generate the following cycles: MUST accept the following cycles: MUST monitor the following cycles: Multiplexed Quad byte read cycle: Byte(0-3) Read Multiplexed Quad byte write cycle: Byte(0-3) Write
NOTE (EO) is Even and Odd; (O) is Odd only.		