IEEE Standard for Futurebus+™/VME64 Bridge

Sponsor
Bus Architecture Standards Committee
of the
IEEE Computer Society

Approved December 13, 1994

IEEE Standards Board

Abstract: The logical protocal layer for bridging between a Futurebus+ system and a VME64 system in a tightly coupled fashion is defined. Transferring of both data and events is specified. The physical layer is not defined in this standard. Implementers are free to define and use the Bridge in the manner that best fits the application for optimum performance, functionality, and cost trade-offs. The Bridge provides for software transparency such that, at the application level, software is not aware of accesses that reach across a Bridge into the modules of another bus.
Keywords: Bus Bridge, Futurebus+, VMEbus, VME64

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1995 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1995. Printed in the United States of America.

ISBN 1-55937-510-8

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1014.1-1994

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE Standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Introduction

(This introduction is not a part of IEEE Std 1014.1-1994, IEEE Standard for Futurebus+™/VME64 Bridge.)

This standard was developed to provide a consistent logical interface between Futurebus+TM and VME64-based systems. It was developed by members of the IEEE Futurebus+ (P896.*x* and related standards) Working Groups and the VITA VME64 Working Group. These members formed the P1014.1 Working Group.

Special recognition is due to Mark Woodbury for the many hours spent designing the architecture and writing the Data Channel and CSR clauses, for creating the byte swap scheme, and for editing the complete document. Rick Spratt also deserves special recognition for architecting and writing the Event Channel, Dual Port Memory Channel, and Byte Swapping clauses.

At the time this standard was completed, the P1014.1 Working Group had the following membership:

Wayne Fischer, *Chair*
Richard Spratt, *Vice Chair*
Mark Woodbury*IEEE Standards Project Editor*

Ray Alderman
David Cook
Colin Davies
Ian Dobson
Frank Duffy
Samuel Duncan
Chris Eck
Bill Evertz
Paul Fulton
Mike Hasenfratz
Greg Hill
David James
Ernst Kristiansen
Jim Leahy
Thanos Mentzelopoulos
Douglas Patterson
Jack Regula
Larry Snow
Dean Van De Walker
David Wickliff
David Wright

The following persons were on the balloting committee:

Ray Alderman
Richard Ames
Keith Anthony
Harrison Beasley
Christos Bezirtzoglou
John Black
Paul Borrill
Charles Brill
Clyde Camp
Andy Cheese
Steven Cobb
David Cohen
Steven Corbesero
Stephen Deiss
Ian Dobson
Jean-Jacques Dumont
Samuel Duncan
Wilhelm Evertz
Wayne Fischer
Gordon Force
Paul Fulton
Julio Gonzalez Sanz
John Griffith
Michael Hayward
Kenneth Heck
Frank Horn
Ed Jacques
Horace Jones, Jr.
Ernst Kristiansen
Ralph Lachenmaier
Lak Ming Lam
Arthur Mansky
Karl McClure
Thanos Mentzelopoulos
Bruce Millard
Brian Morrison
Klaus Dieter Mueller
Gregory Novak
Elwood Parsons
Chandresh Patel
Steve Quinton
William Ramsay
Frederick Sauer
Donald Senzig
Patricia Smith
Richard Spratt
Michael Thompson
Joseph Trainor
Robert Tripi
Yoshiaki Wakimura
Eike Waltz
David Wickliff
Robert Widlicka
Mark Woodbury
David Wright
Yoshio Yamaguchi
Oren Yuen
Janusz Zalewski

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

When the IEEE Standards Board approved this standard on December 13, 1994, it had the following membership:

Wallace S. Read, *Chair*
Donald C. Loughry, *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril
Bruce B. Barrow
José A. Berrios de la Paz
Clyde R. Camp
James Costantino
Stephen L. Diamond
Donald C. Fleckenstein
Jay Forster*
Ramiro Garcia
Donald N. Heirman
Richard J. Holleman
Jim Isaak
Ben C. Johnson
Sonny Kasturi
Lorraine C. Kevra
E.G. "Al" Kiener
Ivor N. Knight
Joseph L. Koepfinger*
D. N. "Jim" Logothetis
L. Brace McClung
Marco W. Migliaro
Mary Lou Padgett
Arthur K. Reilly
Ronald H. Reimer
Gary S. Robinson
Leonard L. Tripp

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall
Richard B. Engelman
Robert E. Hebner

Rachel A. Meisel and Paula M. Kelty
IEEE Standards Project Editor

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1014.1-1994

CLAUSE PAGE

1. Overview 1

1.1 Scope 1
1.2 Purpose 1

2. References 1

3. Definitions 2

3.1 Special keywords 2
3.2 Bridge-specific definitions 2
3.3 Bus-specific definitions 2
3.4 CSR and ROM definitions 5

4. System model and architecture 8

4.1 Introduction 8
4.2 Single bridge model 8
4.3 Dual Bridge model 9
4.4 Bridge Interconnect Bus 10
4.5 Bridge transaction translator 10
4.6 Reference tables 11

5. Bridge control and status registers (CSRs) 17

5.1 Introduction 17
5.2 CSRs 22
5.3 ROM window 25
5.4 CSR specifications 25
5.5 ROM specification 27

6. Data Channel 27

6.1 Description 27
6.2 Specification 35
6.3 Data Channel ROM specification 59

7. Event Channel 65

7.1 Description 65
7.2 Specification 68
7.3 Event Channel ROM specification 85

8. Dual Port Memory Channel 86

8.1 Description 86
8.2 Specification 88
8.3 Dual Port Memory Channel ROM specification 97

9. Byte swapping 100

9.1 Introduction 100
9.2 Specification 102

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1014.1-1994

IEEE Standard for Futurebus+™/VME64 Bridge

1. Overview

1.1 Scope

This standard defines the logical interface for interconnecting two different buses based on Futurebus+™[1] and VME64 in a tightly coupled configuration. Such an interconnecting interface is called a Bridge. Interface between the Bridge and each of the buses is defined in accordance with the standard for the respective bus.

1.2 Purpose

The following goals were established for the Bridge:

a) Allow Futurebus+ and VME64 systems to coexist in a multiple bus-based system environment.
b) Allow modules on one bus to access resources on another bus, as if those resources resided locally.
c) Shield details of one bus from the other for transparent operations.
d) Allow for a variety of physical implementations.
e) Allow Bridge implementors latitude for optimization of performance, functionality, and cost trade-off.

2. References

This standard shall be used in conjunction with the following publications:

ISO/IEC 10857:1994 [ANSI/IEEE Std 896.1, 1994 Edition], Information technology—Microprocessor systems—Futurebus+—Logical protocol specification.[2]

IEEE Std 896.2-1991, IEEE Standard for Futurebus+—Physical Layer and Profile Specification (ANSI).[3]

IEEE Std 896.2a-1994, IEEE Standard for Futurebus+—Physical Layer and Profile Specification: Errata, Corrections, and Clarifications.

[1]Futurebus+™ is pending trademark registration by the Institute of Electrical and Electronics Engineers, Inc.
[2]ISO/IEC publications are available from the ISO Central Secretariat. ISO/IEC [ANSI/IEEE] publications are also available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.
[3]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

Copyright © 1995 IEEE All Rights Reserved

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

IEEE Std 896.3-1993, IEEE Recommended Practices for Futurebus+ (ANSI).

IEEE Std 1014-1987, IEEE Standard for a Versatile Backplane Bus: VMEbus (ANSI).

ISO/IEC 13213: 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information technology—Microprocessor systems—Control and Status Registers (CSR) Architecture for microcomputer buses.

IEEE Std 1275-1994, IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices (ANSI).

VITA 1.1994, VME64 Draft Specification, Revision 1.11, February 4, 1995.[4]

3. Definitions

3.1 Special keywords

3.1.1 may: A key word indicating flexibility of choice with no implied preference.

3.1.2 shall: A key word indicating a mandatory requirement. For example, designers *shall* implement all such mandatory requirements.

3.1.3 should: A key word indicating flexibility of choice with a strongly preferred implementation.

3.1.4 set: A key word indicating that a binary value of 1 is written to a bit or bits within a field.

3.1.5 clear: A key word indicating that a binary value of 0 is written to a bit or bits within a field.

3.1.6 reserved: A key word indicating that a field is undefined in the present specification but may be used in future versions of the specification. It shall *not* be used for any user-defined purpose such as a user- or device-specific function.

3.1.7 user-defined: A key word indicating that a field may be specified by the user for any specific functions chosen.

3.2 Bridge-specific definitions

3.2.1 Bridge: An interconnect between two or more buses that provides signal and logical protocol translation from one bus to another. The buses may adhere to different bus standards for mechanical, electrical, and logical operation (such as a bus Bridge from Futurebus+ to VME64).

3.2.2 Bridge Interconnect Bus (BIB): The medium used to connect two or more Bridge modules together.

3.2.3 Channel: A control or data path established between two buses that allows information to flow from one bus to the other. The Channels specific to this standard are the CSR Channels, the F2V and V2F Data Channels, the Event Channel, and the Dual Port Memory Channel.

3.3 Bus-specific definitions

3.3.1 address only transaction: A bus transaction that does not include a data phase. The only information transferred is contained within the connection phase and, in some cases, the disconnection phase.

3.3.2 arbitration: The process of selecting the next bus master.

[4]Copies of this draft standard are available from VITA, 10229 N. Scottsdale Rd., Suite B, Scottsdale, AZ 85253.

Copyright © 1995 IEEE All Rights Reserved
Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

3.3.3 backplane: An electronic circuit board and connectors used to interconnect other modules together electrically. The backplane connects selected pins of the connectors, thus providing the medium for the transfer of signals needed for the operation of the bus.

3.3.4 big-endian: A specified ordering of bytes within a data structure where the low-order byte (byte 0) is placed in the most significant byte lane of that data structure.

3.3.5 board: An electronic circuit assembly that connects to a single slot on the backplane. It is removable from and replaceable to a backplane assembly via connectors. This is standard terminology for VME64, while Futurebus+ uses **module** synonymously.

3.3.6 bus tenure: The duration of a master's control of the bus; i.e., the time during which a module has the right to initiate and execute bus transactions.

3.3.7 bus transaction: An event initiated with a connection phase and terminated with a disconnection phase. Data may or may not be transferred during a bus transaction.

3.3.8 busy: If a slave is unable to accept a bus transaction from a master, it may issue a busy status to the master of the transaction. The master shall relinquish the bus and may then re-acquire the bus and retry the transaction after a suitable time interval.

3.3.9 byte: An ordered set of eight binary digits (bits).

3.3.10 byte lane: A data path formed by eight data lines and one parity line (if parity is implemented) that is used to carry a single byte of information among the system modules.

3.3.11 cache coherence: A system of caches is said to be coherent with respect to a cache line if each cache and main memory in the coherence domain observes all modifications of that same cache line. A modification is said to be observed by a cache when any subsequent read would return the newly written value.

3.3.12 cache memory: A buffer memory inserted between one or more processors and the bus, which is used to hold currently active copies of blocks of information from main memory.

3.3.13 coherence domain: A coherence domain is a region in a multiple-cache system in which cache consistency measures are enforced.

3.3.14 compelled data transfer: A technology-independent transfer mechanism in which the slave is compelled to provide a response before the master proceeds to the next transfer.

3.3.15 connected transaction: A transaction in which both the request and response are performed within the same bus transaction.

3.3.16 CSR: An abbreviation for *control and status register*.

3.3.17 deadlock: This occurs when modules are awaiting action that can only be performed by those waiting, and those waiting cannot perform that action.

3.3.18 doublet: An ordered set of two adjacent bytes.

3.3.19 DTB: An abbreviation for *data transfer bus*, which is used as a data path in the VMEbus.

3.3.20 Dual Port Memory: A set of registers (or memory) that are asynchronously accessible from each of two ports (or *buses*, in the context of a Bridge) without the requirement for arbitration on one bus while accessing the memory from the other bus.

3.3.21 F2V: A transaction that is originated at the Futurebus+ (the Bridge acts as a Futurebus+ slave), and has its destination at the VME64 (the Bridge acts as a VME64 master).

3.3.22 forward progress: A condition whereby a module is not blocked from performing the tasks necessary to achieve its goal. This condition is guaranteed only in the absence of **deadlock** or **starvation**.

3.3.23 generator: A module or device that initiates a bus request (such as an interrupt request) as the master to that request.

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1014.1-1994

3.3.24 handler: A module or device that responds to a bus request (such as an interrupt request) as the slave to that request.

3.3.25 little-endian: A specified ordering of bytes within a data structure where the low-order byte (byte 0) is placed in the least significant byte lane of that data structure.

3.3.26 livelock: A system condition that occurs when two or more modules acquire and release resources without any of them doing useful work for an indefinite period of time.

3.3.27 locking: A facility whereby a module is requested to guarantee exclusive access to an addressed resource, blocking other modules from accessing that data. This allows indivisible operations to be performed on addressed resources.

3.3.28 master: A module that has acquired control of the bus through the control acquisition procedure.

3.3.29 module: A collection of circuitry that is designed to perform a specific operation. This is standard terminology for Futurebus+, while VME64 uses **board** synonymously.

3.3.30 node: A set of control and status register addresses (including identification-ROM and reset command registers) that are initially defined in a 4 kbyte (minimum) initial node address space. Each node can be reset independently (a reset of one node has no effect on other nodes).

3.3.31 octlet: An ordered set of eight adjacent bytes.

3.3.32 operation: The total activity associated with the exchange of data or events.

3.3.33 packet data transfer: A very fast, but technology-dependent, noncompelled transfer mechanism used to transfer data from a source to a destination.

3.3.34 page: A unit of memory with a specific size defined as a power of 2.

3.3.35 PMR: An abbreviation for *page map register*.

3.3.36 posted transaction: A transaction in which the request and response are performed within different transactions.

3.3.37 quadlet: An ordered set of four adjacent bytes.

3.3.38 reflective memory: Memory that may physically reside in more than one location (on multiple modules) but that contains the identical information within that memory at all the physical locations.

3.3.39 request: A command generated by a requester, to initiate an action on a responder. For example, for a processor-to-memory read transaction, the request transfers the memory address and command from the processor to memory. In the case of a split transaction, the request would be a separate bus transaction. In the case of a connected transaction, the request would be the connection phase of a bus transaction.

3.3.40 requester: A module that initiates a transaction by sending a request (containing address, command, and sometimes data) is referred to as the requester.

3.3.41 responder: A module which completes a transaction by sending a response (containing the completion status and sometimes data) is referred to as the responder.

3.3.42 response: A reply generated by a responder, to complete a transaction initiated by a requester. For example, for a processor-to-memory read transaction, the response returns the data and status from the memory to the processor. In the case of a split transaction, the response would be a separate bus transaction. In the case of a connected transaction, the response would be the data and disconnection phases of a bus transaction.

3.3.43 retry: A mechanism whereby a transaction that (for whatever reason) could not complete in the current operation is attempted again at a later time.

3.3.44 round robin: A bus allocation rule whereby, after a module acquires and uses the bus, it will not be granted use of the bus again until all other modules currently requesting the bus at the same priority level have had a chance to use the bus.

Copyright © 1995 IEEE All Rights Reserved
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

3.3.45 scatter/gather: Data structures in memory that are sequentially ordered in virtual space may be sparcely ordered in physical space. In order to access this data structure with a physical device (such as a DMA controller), the device may need to redirect its address pointer to different physical pages of memory while transferring that data.

3.3.46 shared memory: The address space in the system accessible to all modules.

3.3.47 slave: A module that can be addressed and is capable of participating in, but not initiating, bus transactions.

3.3.48 split operation: A request to exchange data or an event that cannot be satisfied while all its participants remain interlocked. Rather, the participants record the requested action and, at a later time, when the exchange can be completed, re-establish a communication path to complete it.

3.3.49 starvation: A system condition that occurs when one or more modules are prevented from doing useful work, for an indefinite period of time, due to lack of access to the bus or to other system resources.

3.3.50 triplet: An ordered set of three adjacent bytes.

3.3.51 V2F: A transaction that is originated at the VME64 (the Bridge acts as a VME64 slave), and has its destination at the Futurebus+ (the Bridge acts as a Futurebus+ master).

3.3.52 vector: A packet of data that is passed, by an interrupt handler, to a processor in response to an interrupt acknowledge. This data is typically used to identify to an operating system which interrupt subroutine handler should be dispatched in response to the asserted interrupt.

3.3.53 window: A contiguous unit of addressing space that one bus utilizes to provide access to data on another bus.

3.4 CSR and ROM definitions

3.4.1 CSR entries

This standard describes a large number of control and status registers (CSRs) and fields within these registers. To distinguish register and field names from node states or descriptive text, the register names are always uppercase and the field name is always in italic. Thus, the notation STATE_CLEAR.*lost* is used to describe the *lost* field within the STATE_CLEAR register.

3.4.2 ROM entries

This standard describes a large number of ROM entries and fields within these entries. To distinguish ROM entry and field names from node states or descriptive text, the first character of the entry name is always capitalized and the field name is always in italic. Thus, the notation Node_Memory_Extent.*align* is used to describe the *align* fieldwithin the Node_Memory_Extent entry.

3.4.3 Addressing conventions

All CSRs and ROMs are quadlets and are quadlet-aligned. The address of a register (which is always a multiple of 4) is specified as the byte-offset from the beginning of the initial node space. When a range of register addresses is described, the ending address is the address of the last register, which is also a multiple of 4. These addressing conventions are illustrated in table 1.

Table 1— Addressing conventions

Offset	Register	Description
0	STATE_CLEAR	First CSR location
4–12	OTHER_REGISTERS	Next three CSR locations

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

3.4.4 Logic conventions

The notation “0”, when used in reference to a CSR ROM definition, implies a logical Boolean 0. The notation “1”, when used in reference to a CSR ROM definition, implies a logical Boolean 1.

3.4.5 Signal names

Where a group of bus lines are represented by the same letters, the lines within the group are numbered, e.g., AD0*, AD1*, AD2*, etc. In order to represent a group of lines or signals in more convenient form, the notation AD[63:0]* is used. Also, the notation AD[]* is used to refer to all of the lines within the group.

3.4.6 Register specification format

This standard precisely defines the format and function of CSRs. Some of these registers are read-only, some are read-write, and some have special side effects on writes. To accurately define the content and function of these CSRs, their specification includes the format (the sizes and names of bit field locations), the initial value of the register (if not zero), the value returned when the register is read, and the effect(s) when the register is written. An example register is illustrated in figure 1.

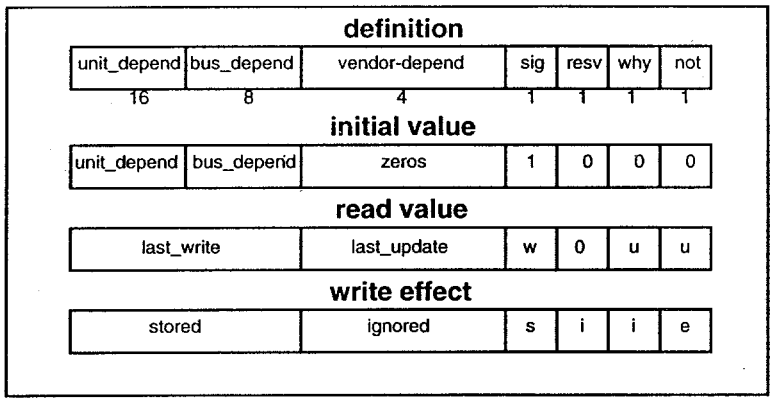


Figure 1— CSR field definitions

The fields within a register are named in its definition. These names are somewhat descriptive, but may be too short to be meaningful. The meaning of these fields is defined in the text; their capabilities should not be implied by their names. However, the following register-definition fields have generic meanings:

unit_dependent

Abbreviation *unit_depend*
Definition The meaning of this field shall be defined by the node’s unit architecture(s).

bus_dependent

Abbreviation *bus_depend*
Definition The meaning of this field shall be defined by the bus standard.

vendor_dependent

Abbreviation *vendor_depend*
Definition The meaning of this field shall be defined by the node’s vendor.

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Within a unit architecture, the unit_dependent fields may be defined to be vendor_dependent. Within a bus standard, the bus_dependent fields may be defined to be vendor_dependent.

The node registers defined in the CSR architecture shall be initialized when power is restored (a power_reset) or when a quadlet is written to its RESET_START register (a command_reset). For most registers, the initial value after a power_reset or command_reset is the same. When the initial CSR values differ, the two initial values are explicitly illustrated.

State in the CSRs need not be lost when the CSRs are initialized (in the sense that the previous values are irretrievable). An implementation may save the CSR contents in nonvolatile memory before power is lost; bus standards are expected to provide a power-failure warning, to support such state-save protocols. After a power_reset or command_reset initializes the CSRs to (possibly) new values, the previously saved CSR values may be recalled using bus- or vendor-dependent mechanisms.

The following read-value fields have a generic meaning:

last_write

Abbreviation	*w*
Definition	The value of the data field shall be the value that was previously written to the same register address.

last_update

Abbreviation	*u*
Definition	The value of the data field shall be the last value that was updated by node hardware.

The following write-effect fields have a generic meaning:

stored

Abbreviation	*s*
Definition	The value of the written data field shall be immediately visible to reads of the same register.

ignored

Abbreviation	*i*
Definition	The value of the written data field shall be ignored; it shall have no effect on the node's state.

effect

Abbreviation	*e*
Definition	The value of the written data field shall have an effect on the node's state, but is not immediately visible to reads of the same register.

The register description specifies the register's optionality (required or optional) as well as its read/write characteristics. A read-write register (RW) is expected to be read and written by software; a read-only register (RO) is expected to be only read by software; a write-only register (WO) is expected to be only written by software. Although reads of WO registers or writes of RO registers are not expected, the register definition still defines their results.

Copyright © 1995 IEEE All Rights Reserved

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

4. System model and architecture

4.1 Introduction

This standard describes how one Futurebus+-based subsystem and one VME64-based subsystem are bridged together in a single tightly coupled system. The logical structure for translating both data and events between the buses is defined. A Dual Port Memory is defined as an alternative method for passing data between the Futurebus+ and the VME64 bus subsystems. Included in this specification are layout, usage of CSR, initialization, error handling, and exception handling.

The Bridge model defined allows for physical and logical implementation flexibility. The Bridge can be a single mechanical module where a standard VME64 board plugs into a Futurebus+ Profile-specific module that slides into a single Futurebus+ slot; or a Bridge can be one module that plugs into a Futurebus+ slot and another module that plugs into a VME64 slot. The interconnect between the two modules is not defined and is implementation-specific. Following are descriptions of each configuration.

4.2 Single bridge model

The single Bridge is useful for applications where both Futurebus+ modules and VME64 modules reside in a single mechanical enclosure. This is possible since Futurebus+ Profile A (or other profiles with the same physical definition) modules are larger in height, depth, and width than VME64 modules. It is possible to build a single Bridge that slides into one or more Profile A slots that bridges to one or more VME64 modules. The circuitry that provides the Bridge electronics resides on a single printed circuit board with mechanical connectors for attachment to both Futurebus+ and VME64 backplanes. The physical and electrical interconnect are not defined in this specification.

Figure 2 illustrates the single Bridge module defined in this standard. On both the Futurebus+ and VME64 backplanes there can be multiple processors, memory, and I/O modules. The single Bridge connection logic to the Futurebus+ is called FB+/IF (Futurebus+ interface). The Bridge connection logic to VME64 is called VME64/IF (VME64 interface). The logic between both interfaces is called BTT (Bridge Transaction Translator). The BTT performs the transaction translation necessary to take the transaction of one bus and translate it to the transaction of the other bus. Transactions can be both data and events.

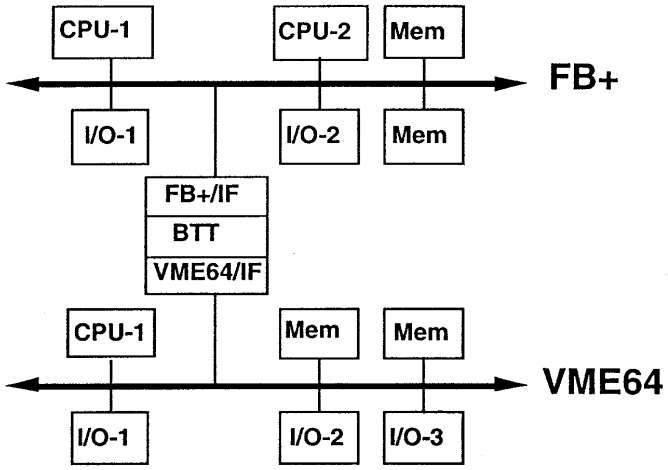


Figure 2— Single Bridge model

Copyright © 1995 IEEE. All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

One of the major goals of the Bridge is to provide transparency between the two buses such that one bus does not have to be aware of the other bus. In fact, at the application layer, the software should not have to be aware of the Bridge and that its transactions are communicating with modules on a different bus structure.

Clause 6 defines the logical implementation of how data transactions are mapped across the Bridge. Event transfers are defined in Clause 7. Clause 8 defines an alternate method for transferring data between both buses. Organization of the data is defined in Clause 9. The internal implementation is left up to the designer.

On the Futurebus+ side, each profile defines a set of CSRs that are required to identify, initialize, and control a module plugged into a Futurebus+ slot. Clause 5 defines additional CSRs required to control the Bridge from the Futurebus+ side.

4.3 Dual Bridge model

In some applications, individual chassis are used to house Futurebus+-based subsystems and VME64-based subsystems. Individual Bridge modules are plugged into the respective chassis for direct communications between each of the respective Futurebus+ and VME64 subsystems. Some kind of interconnect mechanism is required to provide communications between each of the Bridge modules.

Bus transactions that are directed to another bus are routed through two bus Bridges. First, the bus transactions are bridged to the interconnect. Second, the Bridge interconnect transaction is then bridged into the targeted bus. Figure 3 illustrates the dual Bridge model.

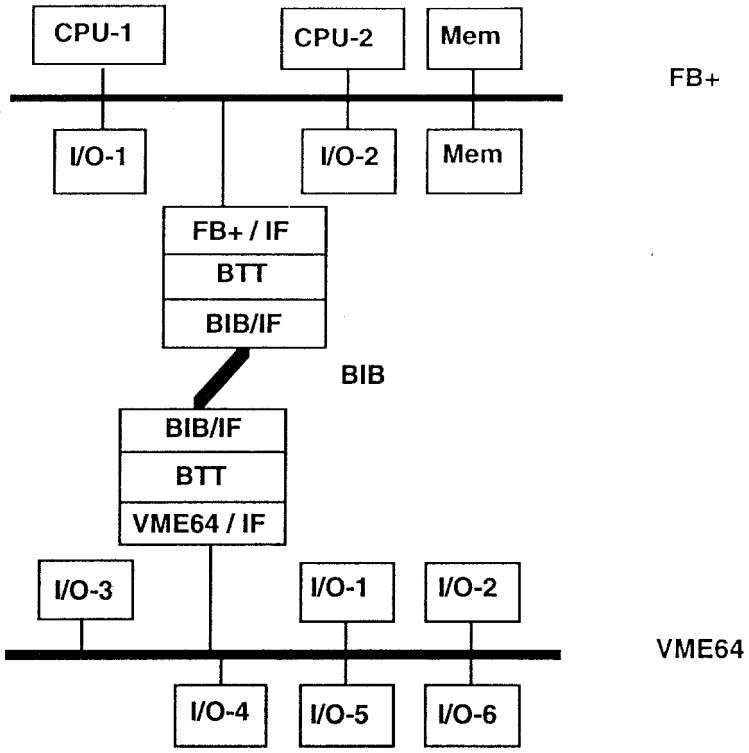


Figure 3— Dual Bridge model

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

On the Futurebus+ side there can be one or more processor, memory, and I/O modules, plus one or more Bridge modules. The Bridge translates Futurebus+ transactions from the Futurebus+ into transactions that are to be routed over the Bridge Interconnect Bus (BIB).

On the VME64 side there can be one or more processor, memory and I/O modules, plus a single Bridge module. The Bridge translates VME64 transactions to and from the BIB.

As in the Single Bridge model, FB+/IF provides the interface between Futurebus+ and the Bridge transaction translator (BTT). BIB/IF provides the BIB interface between the BTT and the BIB on both Bridges. VME interface is performed by the VME64/IF logic.

The BTT on the Bridge between Futurebus+ and BIB is relatively simple in that it provides more of a gate between the two respective buses, if the BIB implements the Futurebus+ logical layer.

The BTT function on the Bridge between BIB and VME64 is (or could be) the same as in the single Bridge model if the BIB implements the Futurebus+ logical layer.

4.4 Bridge Interconnect Bus

The Bridge Interconnect Bus (BIB) is, in effect, another bus. It is a point-to-point type of bus that connects one VME64 Bridge module and one Futurebus+ Bridge module.

4.5 Bridge transaction translator

The bridge transaction translator (BTT) is divided into six logical functions. As mentioned earlier, there are two major types of transactions performed by the bridge: data transfers and event transfers. For each type there shall be a logical path in each direction as follows:

— Futurebus+ to VME64 Data Channel (F2V)
— VME64 to Futurebus+ Data Channel (V2F)
— Futurebus+ to VME64 Event Channel (F2V)
— VME64 to Futurebus+ Event Channel (V2F)

For the Futurebus+ interface there shall be a set of CSRs that are accessible by the Futurebus+ interface for control and status of the bridge functions. IEEE Std 1275-1994,[5] the Open Firmware standard, has been added as an extension capability to the CSR capability. Open Boot provides a standardized software mechanism for interfacing to the CSRs. Although the main access to the CSRs is from Futurebus+, processors on the VME64 side can optionally read the CSRs for their own information. VME64 also provides Open Boot capability. For the Bridge, this Open Boot definition is the same for both buses.

The sixth channel is Dual Port Memory. Communications between the two buses can be via Dual Port Memory. Another term used for this capability is *reflective memory*. Whatever gets written into one side of the memory is available on the other side. Both sides of the Dual Port Memory activity is reflected on the other side's Dual Port Memory. No special mechanism within the Dual Port Memory is provided that notifies the other side whenever a written access is performed on the other port.

Figure 4 illustrates the six channels that make up the BTT functions.

[5]Information about references can be found in clause 2.

Copyright © 1995 IEEE All Rights Reserved
Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

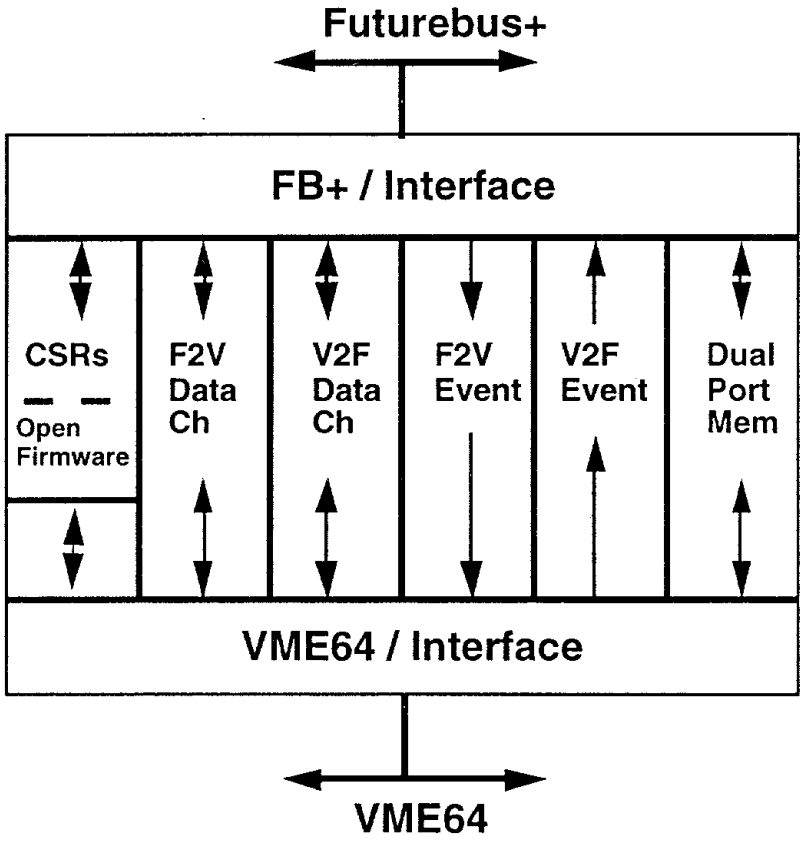


Figure 4— Six BTT channels

4.6 Reference tables

Tables and indicate which specifications are referenced for the logical layers for both Futurebus+ and VME64 Bridge implementations.

The required set is the minimum set of transactions and functions that are desirable for a Bridge from Futurebus+ to VME64. However, if a profile does not allow specific transactions, then they are not required to be implemented in a Bridge for that specific profile. In such a case, the user documentation shall clearly document the exceptions to this specification.

Copyright © 1995 IEEE All Rights Reserved

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1014.1-1994

Table 2— Bridge Futurebus+ logical layer referenced specifications

Function	Standard	Mandate	Comments
Arbitration Distributed Central	ISO/IEC 10857 : Chapter 5 Chapter 4	Profile-specific	At least one arbitration scheme shall be implemented.
Arbitrated messages	ISO/IEC 10857 : Chapter 5	Profile-specific	
Module address width 32 64	ISO/IEC 10857 IEEE Std 896.2	Required Recommended	
Module data path width 32 64 128 256	ISO/IEC 10857 IEEE Std 896.2	Required Recommended Optional Optional	
Split transactions	ISO/IEC 10857 : Chapter 6	Required for Masters	Bridge shall support split reads. Bridge may support split write requests.
Packet mode	ISO/IEC 10857	Optional	
CSR	ISO/IEC 13213 ISO/IEC 10857 IEEE Std 896.2 : Chapter 3	Required	All profile-specific and bridge-required registers shall be implemented
Boot Firmware	IEEE Std 1275	Optional	Software assist for reading and writing CSRs.
Tag bits and tag parity	ISO/IEC 10857	Profile-specific	
Message passing	ISO/IEC 10857	Not supported	

Function	**Bridge Channel**				
Standard: ISO/IEC 10857 IEEE Std 896.2	Data Channel F2V[a]	Data Channel V2F[b]	DP_Mem Slave[c]	Event Channel Slave	Event Channel Master
Transaction types:					
Read Unlock	Required	Required	Required	CSR access	Not supported
Write Unlock	Required	Required	Required	CSR access	Required
Address Only Unlock	Required[d]	Not supported	Not supported	Not supported	Not supported
Read Locked	Not supported	Not supported	Optional	Not supported	Not supported
Write Locked	Not supported	Not supported	Optional	Not supported	Not supported
Address Only Locked	Not supported	Not supported	Not supported	Not supported	Not supported

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 2— Bridge Futurebus+ logical layer referenced specifications (Continued)

Function	Standard	Mandate	Comments		
Read Partial	Required	Required	Required	Not supported	Not supported
Write Partial	Required	Required	Required	Not supported	Not supported
Read Partial Locked	Not supported	Not supported	Optional	Not supported	Not supported
Write Partial Locked	Not supported	Not supported	Optional	Not supported	Not supported
Read Response	Required	Required	Not Supported	Not supported	Not supported
Write Response	Optional	Required	Not Supported	Not supported	Not supported
Write No Acknowledge	Required	Required	Required	Not supported	Not supported
Read Invalid	Optional	Optional	Not supported	Not supported	Not supported
Write Invalid	Optional	Optional	Not supported	Not supported	Not supported
Read Shared	Not supported	Optional	Not supported	Not supported	Not supported
Copyback	Not supported	Optional	Not supported	Not supported	Not supported
Read Modified	Not supported	Optional	Not supported	Not supported	Not supported
Invalidate	Not supported	Optional	Not supported	Not supported	Not supported
Shared Response	Not supported	Optional	Not supported	Not supported	Not supported
Modified Response	Not supported	Optional	Not supported	Not supported	Not supported
Mask and Swap	Not supported	Not supported	Not supported	Not supported	Not supported
Fetch and Add	Not supported	Not supported	Not supported	Not supported	Not supported
Compare/Swap	Not supported	Not supported	Not supported	Not supported	Not supported

[a]For F2V transactions, it is required to accept a split read and optional to accept a split write. It is required to generate a read response, and generating a write response is required *only* if split writes are supported.
[b]Split transactions *should* support one or more splits (up to a maximum of 64).
[c]Split transactions shall not be supported to the Dual Port Memory.
[d]Beat error will cause address only transactions. The Bridge shall be able to handle this error case, but is not required to pass this transaction across the bridge to the VME64 bus.

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 3— Bridge VME64 logical layer referenced specifications

Function	Standard	Mandate	Comments
Arbitration	VITA 1.1994		
Prioritized		Required	
Round robin		Required	
Single-level		Required	
Module address width			
For F2V transactions			VME64 as master
A16		Required	
A24		Required	
A32		Required	
A40		Not supported	
A64		Recommended	
For V2F transactions			VME64 as slave
A16		Not supported	
A24		Required	
A32		Required	
A40		Not supported	
A64		Recommended	
Module data path width			
D08		Required	
D16		Required	
D32		Required	
D64		Recommended	
Interrupt levels			
Bridge as requester asserts IRQ[7:1]			
Release via ROAK		Required	
Release via RORA		Required	
Bridge as handler ack's IRQ[7:1]			
Release via ROAK		Required	
Release via RORA		Not supported	
Transaction type *hex*			

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 3— Bridge VME64 logical layer referenced specifications (Continued)

Function		Standard	Mandate	Comments
AM[5:0]	3F	VITA 1.1994	Optional	A24 supervisory block transfer (BLT)
	3E		Required	A24 supervisory program access
	3D		Required	A24 supervisory data access
	3C		Optional	A24 supervisory 64-bit block transfer (MBLT)
	3B		Optional	A24 nonprivileged block transfer (BLT)
	3A		Required	A24 nonprivileged program access
	39		Required	A24 nonprivileged data access
	38		Optional	A24 nonprivileged 64-bit block transfer (MBLT)
	37		Not supported	A40 block MD32 transfer
	35		Not supported	A40 lock command
	34		Not supported	A40 single access
	32		Optional	A24 lock command
	2F		Optional	Configuration ROM/CSR
	2D		Required	A16 supervisory access
	2C		Optional[5]	A16 lock command
	29		Required	A16 nonprivileged access
	0F		Optional	A32 supervisory block transfer (BLT)
	0E		Required	A32 supervisory program access
	0D		Required	A32 supervisory data access
	0C		Optional	A32 supervisory 64-bit block transfer (MBLT)
	0B		Optional	A32 nonprivileged block transfer (BLT)
	0A		Required	A32 nonprivileged program access
	09		Required	A32 nonprivileged data access
	08			A32 nonprivileged 64-bit block transfer (MBLT)
	05		Optional[a]	A32 lock

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 3— Bridge VME64 logical layer referenced specifications (Continued)

Function	Standard	Mandate	Comments
04		Optional[a]	A64 lock
03		Optional	A64 block transfer (BLT)
01		Optional	A64 single transfer
00		Optional	A64 64-bit block transfer (MBLT)
Access types		**Slave**	**Master**
Single-byte access		Required	Required
Doublet-aligned access		Required	Required
Doublet-unaligned access		Required	Optional
Triplet-aligned access		Required	Optional
Triplet-unaligned access		Required	Optional
Quad-byte access		Required	Required
Eight-byte access		Optional	Optional
Read-modify-write (RMW)		Atomicity not supported	Transactions should be handled as a separate read followed by a separate write cycle.
VME64 LCK		Optional	
Retry		Optional	

[a]Locks are optionally supported for the Dual Port Memory only.

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

5. Bridge control and status registers (CSRs)

5.1 Introduction

This clause defines the control and status registers (CSRs) required for implementing the VME64/Futurebus+ Bridge interface. This information complements the standard CSR Architecture defined in ISO/IEC 13213 :1994 and IEEE Std 896.2-1991 .

5.1.1 CSR Architecture standard node registers

The CSR Architecture standard assigns address ranges for standard node registers (node space, internal space, and memory space). Figure 5 depicts the address assignments for 32-bit physical address buses. The 64-bit CSR Architecture node space is not used for access to the VME64. Refer to 3.2 in IEEE Std 896.2-1991 for specific definitions of the standard node CSR registers.

5.1.2 Extended addressing

The **extended** addressing is based on 32-bit and 64-bit physical address spaces, which are both split into separate available and register address spaces. The register address space is partitioned into bus offset addresses, and each bus is partitioned into node offset addresses. Although a large number of nodes are mapped to the register address space, the register address space is only a small portion of the total physical address space. The 32-bit extended address-space layout is illustrated in figure 5.

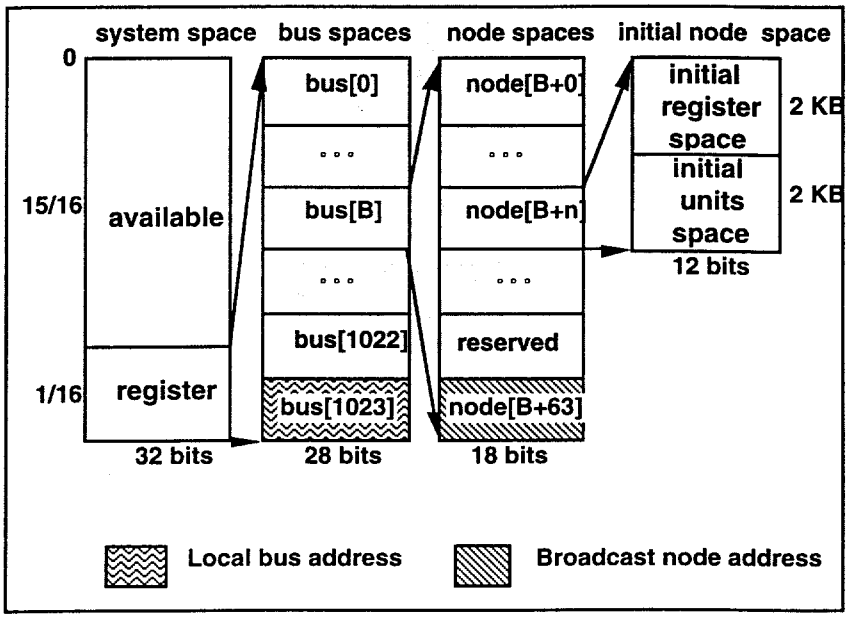


Figure 5— 32-bit extended addresses

Futurebus+ nodes shall support A32 and may support A64 addressing. Futurebus+ modules that support only A32 can coexist on the same backplane with modules that are using A64 without fear of aliasing. Since all Futurebus+ modules are required to support A32, all transactions with 32-bit equivalents shall be done with A32 transactions. A node that is A64 internally shall be designed to recognize a bus address of less than 3.75 Gbyte (the bottom 15/16 of 32-bit address space), whose address ranges are specified in table 4, and convert it to A32.

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 4— 32-bit extended addresses

CM7*	Address field		Description
From	To		
0	0000 0000	EFFF FFFF	A32 address space
0	F000 0000	FFFF FFFF	Node CSR space
1	0000 0000 0000 0000	0000 0000 EFFF FFFF	Prohibited (use the A32 equivalent)
1	0000 0000 F000 0000	FFFF FFFF FFFF FFFF	A64 address space

Within the register space, up to 64 node addresses are allocated to one bus, and up to 1024 bus addresses are allocated to a system. Since some of these allocated addresses have special uses, this limits the system to 1023 buses with 62 nodes on each bus.

The node's initial address space is limited to 4 kbytes, which contains two address spaces: the node's initial register space and initial units space. The address of the node is specified by a 16-bit *node_id* value. The 10 most significant bits define the *bus_id* and the 6 least significant bits define the *offset_id*. Software is expected to configure the *node_id* values so that all nodes on one bus have the same *bus_id* value and a bus-unique *offset_id* value.

One of the *bus_id* values (1023) has a special meaning—all slaves respond to this fixed local-bus address as well as to their dynamically configured global-bus address. Software is expected to use the fixed local-bus address to access nodes on the local bus before the global-bus address has been assigned. After the global-bus addresses have been assigned, software is expected to use global-bus addresses to access all nodes.

5.1.3 CSR Architecture node registers

Each node has 4 kbytes of CSR node space. The 4 kbytes of space is divided into a block of 2 kbytes of *assigned* (initial register space) space and a block of 2 kbytes of *available* (initial units space) space. Bridge modules use both the assigned and available spaces, placing CSR-standardized registers in assigned space and bus Bridge-specific registers in available space.

Of the 2 kbytes in assigned node space, the first 512 bytes are labeled *specified*. The addresses labeled *restricted* are reserved for future use. The ROM window is a 1 kbyte window into the node's ROM, pointed to by the internal_address register.

The specified registers are of three types:

- CSR Architecture core registers (bytes 0-511)
- Bus-dependent registers (bytes 512-1023)
- ROM window entries (bytes 1024-2047)

Refer to ISO/IEC 13213 :1994 for a detailed definition of the CSR Architecture core registers.

Bus-dependent registers are defined by each bus standard. Futurebus+ defines its bus-dependent registers in clause 7 of ISO/IEC 10857 :1994.

5.1.4 Initial node space

A node initially responds to a 4 kbyte initial node space. Half of the initial node space (2 kbytes) is register space and the other half is allocated to unit-dependent specifications. One quarter of the 2 kbyte register space is allocated to CSRs (as defined by this standard), one quarter of the register space is allocated to bus-dependent registers (which are restricted by this standard), and the other half provides an address window to the initial addresses in the node's ROM, as illustrated in figure 6.

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

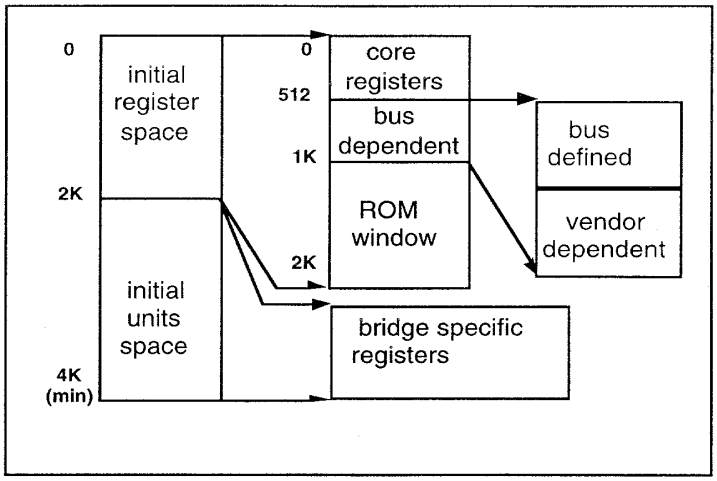


Figure 6— Initial node space components

Read transactions to the initial register space shall be supported and shall have no side effects. This restriction is intended to simplify state-save, diagnostics, and fault-retry software, which can safely read (or reread) any address within the initial register space (without unexpected, and sometimes unrecoverable, side effects).

The bus standard shall define the first part of the bus-dependent portion of the initial node space, and shall allocate the remainder for vendor-dependent uses. The bus standard shall specify the number of vendor-dependent registers at the end of the bus-dependent portion of the initial node space.

5.1.5 Indirect space

A node shall provide an indirect space and a standardized ROM shall be located at the lowest indirect space addresses. The first 1 kbyte of indirect address space is directly mapped to a 1 kbyte portion of the node's 4 kbyte initial node address space. This simplifies inexpensive nodes, whose total ROM size is less than 1 kbyte in size. When larger ROMs are implemented, critical information, such as the type of node and bus-dependent parameters, shall be located at the beginning of ROM. This directly mapped ROM address space is illustrated in figure 7.

In addition to the ROM, the indirect space may contain unit-dependent registers, which may have read and write capabilities. When accessing these registers, reads shall have no side effects.

Other indirect addresses, those with addresses at or above 1024 in the indirect space, shall be indirectly accessed through the INDIRECT_ADDRESS and INDIRECT_DATA register pair. When accessed indirectly, a write to the INDIRECT_ADDRESS register establishes the indirect address; a following read of the INDIRECT_DATA register returns a quadlet from this prespecified indirect address. In a multiprocessor environment, higher level locks are expected to ensure that this sequence is performed indivisibly. This indirect access model (which is used on indirect addresses above 1 kbyte) is illustrated in figure 8.

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

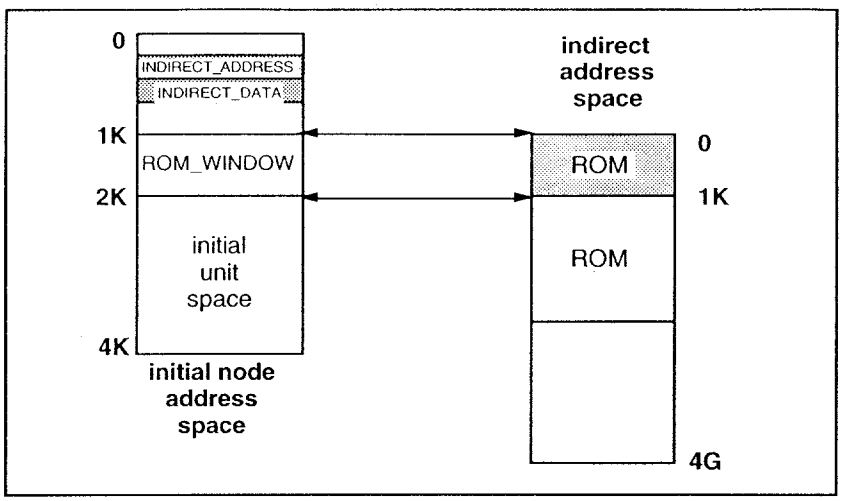


Figure 7— Indirect space mapping (address less than 1 kbytes)

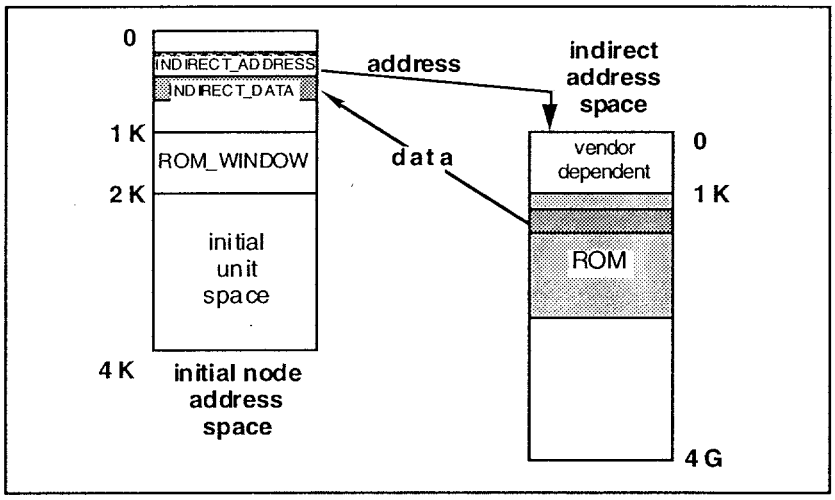


Figure 8— Indirect space mapping (address greater or equal to 1 kbyte)

Software is always expected to access the first 1 kbyte of the indirect address space by accessing an address within the node ROM_WINDOW. When the value of the INDIRECT_ADDRESS is less than 1 kbyte, the value returned from a read of the INDIRECT_DATA register and the effect of a write to the INDIRECT_DATA register are both vendor-dependent.

5.1.6 Bridge register locations

Figure 9 and table 5 provide a summary of the registers (required and optional) for the VME64 to Futurebus+ Bridge. These registers shall all be accessible from the Futurebus+. It is recommended that these registers not be read or write accessible from the VME64. This specification does not define VME64 access to these registers. If this access is provided, the implementation shall provide a mechanism to disable that register access.

The location of a node-control register is defined by its byte-offset from the beginning of the initial register space. The CSRs needed to support the basic and optional capabilities are listed in table 6.

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

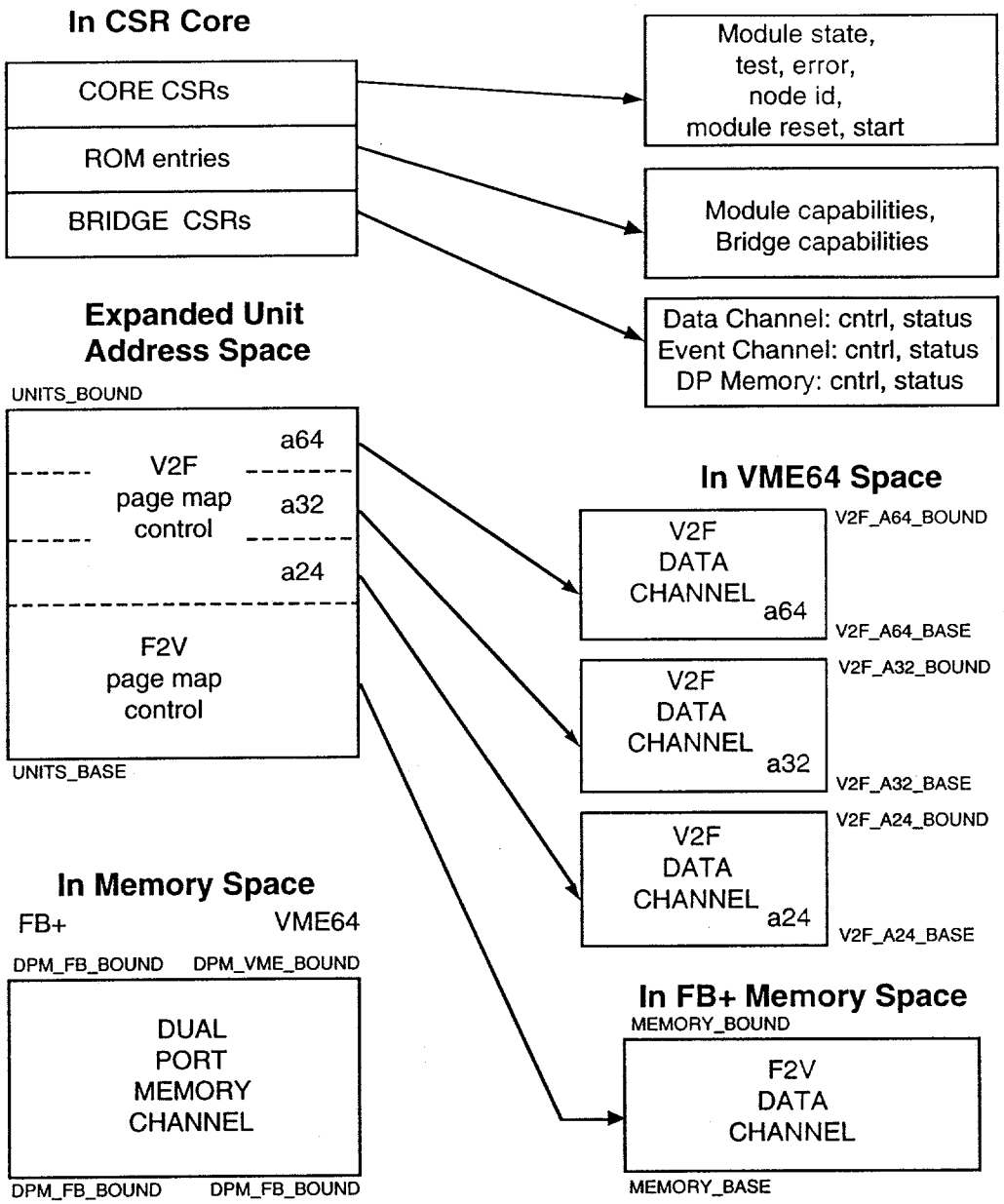


Figure 9— Bridge-specific CSR map

The full Bridge register set includes the following:

— **CSR Architecture standard node registers.** These configuration and status registers conform with ISO/IEC 13213:1994 (IEEE Std 1212-1991) and are required by IEEE Std 896.2-1991. These registers are all contained in the initial register space.
— **ROM entry.** The ROM contains Bridge module information in the initial register space, and also Bridge unit information in the initial register space and, optionally, in the indirect space.
— **Bridge-specific registers.** These registers consist of the CSRs required by this Bridge specification. They include the page map registers (PMRs), failing address registers, and event port registers. The registers are contained in the initial register space and, optionally, in the available system space.

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

— **VME64 memory Data Channel.** This space provides the actual access from Futurebus+ to the VME64. This window may be placed in either available system space or in memory space.
— **Dual Port Memory.** This space describes optional resident memory that is shared by the Futurebus+ and the VME64.

5.2 CSRs

The CSR address space shall be allocated as shown in table 5.

Table 5— CSR address space

Area name	Register offset address range
Core CSRs	0–508
Futurebus+ reserved CSRs	512–1020
ROM entries	1024–2044
Initial unit space	2048–4092

Complete lists of the minimum required Bridge register set for the Futurebus+ to VME64 Bridge are shown in table . Registers not listed are considered optional. Core CSRs shall, at a minimum, implement support for the CSRs that are required by the specific profile implemented. If a module implements optional functionality it shall do so following requirements specified in IEEE Std 896.2-1991, Section 3.

A mandate of *required* means that the register, field, or bit shall be readable, whatever its value. Some bits, like capability bits, are additionally required to be set to “1”, meaning that capability is required.

All nonimplemented bits, fields, and registers shall return “0” when read.

The capability bit SPLIT_ENABLE is a protocol requirement and cannot be disabled on a Futurebus+ to VME64 Bridge module. Therefore, that bit shall be hard-wired to “1”.

5.2.1 Summary of Bridge-specific CSRs

Table provides a summary and reference for all the required and optional registers specified for the Futurebus+/ VME64 Bridge.

5.2.2 CSR Architecture core registers

Core registers control all units on a node, for Reset and other node-wide functions. Registers are addressed as 4 byte *quadlets*. Core register quadlets should be written atomically. Bus Bridges use the following core registers.

STATUS_BITS:	Primary node status
NODE_IDS:	Global address of this node
STATUS_CLEAR and STATUS_SET:	Affects status bits
INTERNAL_ADDR, INTERNAL_DATA:	ROM address and data
UNIT_EXTENSION:	Defines more memory for units

Copyright © 1995 IEEE All Rights Reserved
Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 6— Bridge CSR locations

Offset	Register name	Mandate	Functional uses
0	STATE_CLEAR	Required	State and control information
4	STATE_SET	Required	Sets state set bits
8	NODE_IDS	Required	Node identification
12	RESET_START	Required	Command_reset
16–44	Profile-specific	Specific	ISO/IEC 13213 :1994 definitions
48–52	UNITS_BASE	Required	For F2V PMRs
56–60	UNITS_BOUND	Required	
64–68	MEMORY_BASE	Required	For F2V VME64 access
72–76	MEMORY_BOUND	Required	For F2V VME64 access
80	INTERRUPT_TARGET	Not used	Broadcast/nodecast interrupt
84	INTERRUPT_MASK	Not used	Broadcast/nodecast interrupt
88–764	Profile-specific	Specific	ISO/IEC 13213 :1994 definitions
1024–2044	ROM_WINDOW	Required	Access to ROM space
2048	F2V_DATA_CHNL_CONTROL	Required	State and control information
2052	F2V_DATA_CHNL_STATUS	Required	Error and status information
2056–2060	F2V_DATA_CHNL_ERR_ADDR	Required	Address of failing VME transaction
2064	V2F_A24_BASE	Required	For V2F Futurebus+ access by
2068	V2F_A24_BOUND	Required	VME64 devices in each address
2072	V2F_A32_BASE	Required	space. Pages (found in units space)
2076	V2F_A32_BOUND	Required	map contiguously from A24 through
2080–2084	V2F_A64_BASE	Optional	A64 space.
2088–2092	V2F_A64_BOUND	Optional	
2096	V2F_DATA_CHNL_CONTROL	Required	State and control information
2100	V2F_DATA_CHNL_STATUS	Required	Error and status information
2104–2108	V2F_DATA_CHNL_ERR_ADDR	Required	Address of failing VME transaction
2112–2116	Reserved	Specific	Future IEEE Std 1014.1 definitions
2120	EVENT_CHNL_STATUS	Required	Futurebus+ event generation
2124	EVENT_CHNL_CLEAR	Required	
2128	FEG_ARM/DISARM	Required	
2132	FEG_CSR_SYSRESET	Required	SYSRESET*
2136	FEG_CSR_SYSFAIL	Required	SYSFAIL*
2140	FEG_CSR_ACFAIL	Required	ACFAIL*
2144	FEG_CSR_EVENT_ERROR	Required	Event Channel error
2148	FEG_CSR_DATA_ERROR	Required	Data Channel error
2152	FEG_CSR_DPM_ERROR	Required	Dual Port Memory error
2156	FEG_CSR_IRQ7	Required	IRQ7
2160	FEG_CSR_IRQ6	Required	IRQ6

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 6— Bridge CSR locations (Continued)

Offset	Register name	Mandate	Functional uses
2164	FEG_CSR_IRQ5	Required	IRQ5
2168	FEG_CSR_IRQ4	Required	IRQ4
2172	FEG_CSR_IRQ3	Required	IRQ3
2176	FEG_CSR_IRQ2	Required	IRQ2
2180	FEG_CSR_IRQ1	Required	IRQ1
2184–2188	FEG_IRQ7_ADDR	Required	Write address for IRQ7
2192–2196	FEG_IRQ6_ADDR	Required	Write address for IRQ6
2200–2204	FEG_IRQ5_ADDR	Required	Write address for IRQ5
2208–2212	FEG_IRQ4_ADDR	Required	Write address for IRQ4
2216–2220	FEG_IRQ3_ADDR	Required	Write address for IRQ3
2224–2228	FEG_IRQ2_ADDR	Required	Write address for IRQ2
2232–223	FEG_IRQ1_ADDR	Required	Write address for IRQ1
2240–2244	FEG_SYS_ADDR	Required	Write addr for system events
2248	VEG_STATUS	Required	VEG Status register for all events
2252	VEG_SYSRESET	Required	SYSRESET*
2256	VEG_SYSFAIL	Required	SYSFAIL*
2260	VEG_ACFAIL	Required	ACFAIL*
2264	VEG_IRQ7	Required	IRQ7
2268	VEG_IRQ6	Required	IRQ6
2272	VEG_IRQ5	Required	IRQ5
2276	VEG_IRQ4	Required	IRQ4
2280	VEG_IRQ3	Required	IRQ3
2284	VEG_IRQ2	Required	IRQ2
2288	VEG_IRQ1	Required	IRQ1
2292	DPM_CONTROL	Optional	Dual Port Memory Control register
2296	DPM_STATUS	Optional	Dual Port Memory Status register
2300–2304	DPM_FB_BASE	Optional	FB+ start address location
2308–2312	DPM_FB_BOUND	Optional	FB+ address bounds
2316–2320	DPM_FB_ERR_ADDR	Optional	FB+ Dual Port Memory error addr
2324–2328	DPM_VME_BASE	Optional	VME start address location
2332–2336	DPM_VME_BOUND	Optional	VME address bounds
2332–4092	Reserved	Specific	Future Std 1014.1 definitions
Units space:			
xx0	F2V_DATA_CHNL_PAGE_00	Required	First VME64 PMR
xx16–	F2V_DATA_CHNL_PAGE_01		Next A24 V2F PMR
	xx is defined by the UNITS_BASE and UNITS_BOUND.		

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 6— Bridge CSR locations (Continued)

Offset	Register name	Mandate	Functional uses
yy0 yy8– 	V2F_A24_DATA_CHNL_PAGE_00 V2F_A24_DATA_CHNL_PAGE_01 yy is defined by the UNITS_BASE offset by (number of F2V pmr's * 8) V2F_A32_DATA_CHNL_PAGE_00 V2F_A64_DATA_CHNL_PAGE_00	Required Required	First A24 V2F PMR Next A24 V2F PMR First A32 PMR First A64 PMR
Memory space:			
zz0– 	FB_DPM zz is defined by FB_DPM_BASE and FB_DPM_BOUND.	Optional	First Dual Port Memory location
vv0– 	vv is defined by the MEMORY_BASE and MEMORY_BOUND.	Required	First VME64 memory location

5.3 ROM window

The ROM window supports the general ROM format as described in ISO/IEC 13213 :1994, clause 7. This includes the following:

General ROM format:	Required ROM entries
Root directory:	Module-specific characteristics
Unit subdirectory:	Bridge-specific information

Refer to ISO/IEC 13213:1994 for definitions of the information contained in the unit subdirectory. Table 7 provides a summary of the Futurebus+/VME64 Bridge-specific entries.

5.4 CSR specifications

The specifications of the registers contained in the Futurebus+ to VME64 Bridge are described in the following subclauses.

5.4.1 CSR Architecture core

These core registers specify where configuration registers and VME64 memory will be located within a Futurebus+ system.

5.4.1.1 State Clear register

The STATE_CLEAR register is defined in ISO/IEC 13213:1994.

5.4.1.2 State Set register

The STATE_SET register is defined in ISO/IEC 13213:1994.

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

Table 7— Bridge-specific ROM entries

Offset	Entry name	Optional uses
Bridge unit ROM entries:		
00	F2V_Data_Chnl_Config	Capabilities, priority, write-splits, VME transaction types supported
04	F2V_Pages_Available	FB+ Data Channel pages available
08	F2V_Page_Sizes_Supported	FB+ Data Channel allowable page
12	F2V_Data_Buffer_Size	FB+ Data Channel buffer size
16	V2F_Data_Chnl_Config	Capabilities, VME transaction types supported, priority
20	V2F_Pages_Available	FB+ Data Channel pages available
24	V2F_Page_Sizes_Supported	FB+ Data Channel allowable page
28	V2F_Data_Buffer_Size	FB+ Data Channel buffer size
32–36	V2F_AM_Codes	VME64 Address Modifier codes supported
40	V2F_Cache_Line_Size	Specifies the size of the cache
Event Channel:		
44	Event_Config	Event Channel capabilities
44–60	Reserved	
Dual Port Memory:		
64	DPM_Config	Dual Port Memory configuration
68	DPM_Memory_Size	Size of Dual Port Memory
72–76	DPM_AM_Codes	AM codes supported by Dual Port Memory
80–256	Reserved	Future ISO/IEC 31213:1994 definitions

5.4.1.3 Node IDs register

The NODE_IDS register is defined in ISO/IEC 13213:1994.

5.4.1.4 Reset Start register

The RESET_START register is defined in ISO/IEC 13213:1994.

5.4.2 Bus-dependent

These registers specify configuration parameters specific to the Futurebus+/VME64 Bridge.

5.4.2.1 Data Channel

The Data Channel configuration registers are described in 6.1 and defined in 6.2.

5.4.2.2 Event Channel

The Event Channel configuration registers are described in 7.1 and defined in 7.2.

5.4.2.3 Dual Port Memory Channel

The Dual Port Memory configuration registers are described in 8.1 and defined in 8.2.

Copyright © 1995 IEEE. All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

5.5 ROM specification

The Bridge-specific capabilities information is contained in the ROM with the following specific definitions.

5.5.1 Module-specific characteristics

Refer to ISO/IEC 13213:1994.

5.5.2 Data Channel

The Data Channel configuration entries are defined and described in 6.3.

5.5.3 Event Channel

The Event Channel configuration entries are defined and described in 7.3.

5.5.4 Dual Port Memory

The Dual Port Memory configuration entries are defined and described in 8.3.

6. Data Channel

6.1 Description

The Bridge Data Channel routes data transfers between the Futurebus+ and the VME64 buses. This clause describes and specifies the requirements that the standard imposes on VME64 and Futurebus+ Bridge Data Channel designers.

These requirements are associated with the ability to route transactions between the VME64 and the Futurebus+. This is managed by a set of control and status registers (CSRs) and page map registers (PMRs) that provide address translation and bus cycle control information. The layout of these registers is specified in 5.2 to ensure that a common set of functions are supported and that the management of those functions is common to a level that allows interoperability among different Bridge implementations.

This standard requires that Bridge resident VME64 masters and slaves include certain capabilities that might not be mandatory in the VME64 specification itself. This was done to assure an acceptable level of interoperability and performance. This clause describes the functions and specifies the requirements imposed on Bridge resident VME64 masters and Bridge resident VME64 slaves.

6.1.1 Futurebus+ to VME64 (F2V) Data Channel

This Channel allows modules that reside on Futurebus+ to access resources that reside on VME64. A window, specified by the memory base and bound registers, provides access to the VME64 from Futurebus+. Within this window, PMRs provide cycle specific information needed by the VME64 bus master and allow for scatter/gather capability. Each page specified has configuration information needed to generate the unique VME64 address with address modifier (AM) code, data type, and byte swap mode chosen. Thus, once a page is configured, the application wishing to use the VME64 resource does not need to perform any functions other than to read or write to a specific Futurebus+ location.

While this specification is intended to be Futurebus+ profile-independent, there is a common subset of functionalities that must be adhered to in order to maintain operability with the VME64 bus. These are as follows:

Copyright © 1995 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

— **Split read operations.** To maintain availability of the two buses and prevent possible deadlock situations, read operations (to the Bridge as a Futurebus+ slave) are required to be split.
— **Split write operations.** As VME64 bus response time is indeterminate, write acknowledge response times may be long, causing reductions in the overall bandwidth of both the VME64 and Futurebus+. Therefore write operations should be immediately acknowledged. For profiles that support it, the Write No Acknowledge transaction should be used. For those profiles that do not support Write No Acknowledge, the Write Unlock transaction may acknowledge immediately (with no split), and complete the transaction at a later time (error handling is discussed in 6.1.3). For applications that require a positive acknowledge, the write transaction shall be split to prevent opportunities for deadlock.
— **No VME read-modify-write (RMW) transaction.** Due to the complexity of interlocked operations using RMW across a Bridge, RMW operations are not supported through the Data Channel for F2V transactions. Subclause 6.1.5 describes alternatives for providing the functions normally associated with the locked transactions.
— **Locked transactions.** Locked operations are not supported across the Data Channel. This function is only supported by the Bridge via the optional Dual Port Memory and is described in clause 8.
— **Bridge in the cache-coherent domain.** Since the VME64 has no provisions for cache coherency, coherency between processors on the different buses cannot be maintained. A system can be built with the Futurebus+ in the cache domain, and data can be moved into (and out of) that domain with the usage of the cache-invalidate transactions.
— **Sequential ordering.** Data transferred across the F2V Data Channel shall maintain a strong sequential ordering while on the Bridge. While transactions may be queued (as in the case of split operation), all transactions accepted by the Futurebus+ F2V channel of the Bridge will be presented to the VME64 master in the exact sequential order in which they were accepted. This sequential ordering may be lost once the data leaves the Bridge, dependent on the specific system implementations. Also, this ordering may be lost with respect to F2V transactions and event transactions.
— **VITA 1.1994.** VME64 Bridge slaves shall comply, where applicable, with all the requirements specified in VITA 1.1994. When VME64 Bridge slaves implement the optional functions described by VITA 1.1994, they shall also comply, where applicable, with the requirements specified for these options in VITA 1.1994.
— **ISO/IEC 10857:1994, IEEE Std 896.2-1991.** Futurebus+ Bridge slaves shall comply, where applicable, with the requirements specified in ISO/IEC 10857 :1994. Further, it is expected that a Bridge product will be built to conform to a specific profile as described in IEEE Std 896.2-1991 or other profiles and, as such, shall comply with all the requirements of the profile specified.

6.1.1.1 Mapping from F2V

Access to the VME64 bus is provided via a window residing in Futurebus+ memory space. This space is specified by the MEMORY_BASE and MEMORY_BOUND registers. Within this window, each page of memory can be individually controlled to have unique characteristics for generating a specific VME64 transaction request. Figure 10 represents the control flow and data access for Futurebus+ to VME64 accesses. Each successive page in the memory space is controlled by a successive PMR located in the units space.

Copyright © 1995 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994

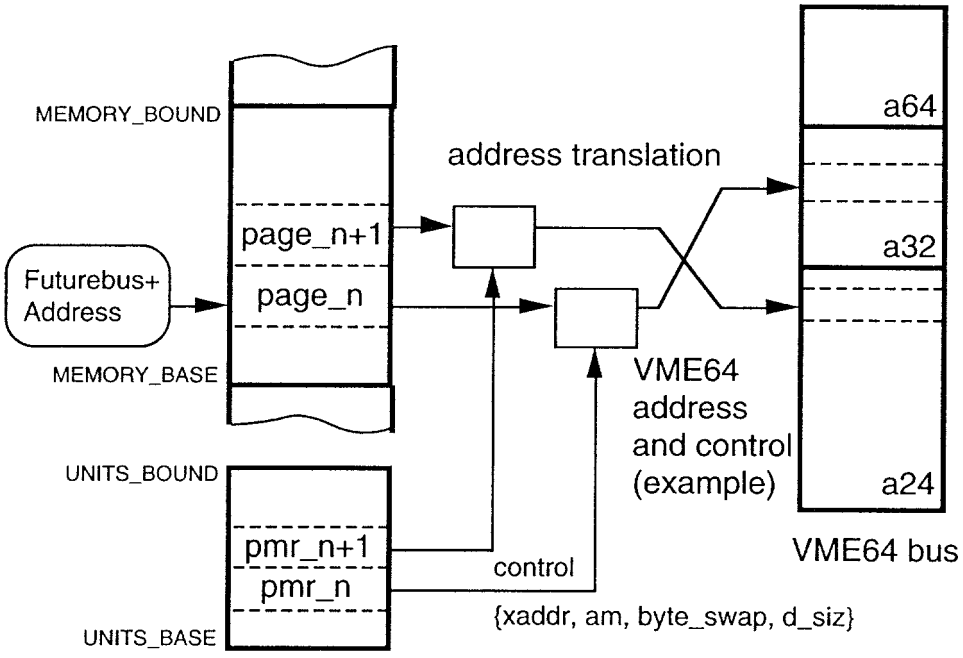


Figure 10— F2V address mapping translation

6.1.1.2 F2V Data Channel registers

The Futurebus+ to VME64 (F2V) data port registers provide the control to enable, map, and access memory or registers on VME64 from Futurebus+. The Bridge provides mapping translation registers to allow addresses on the Futurebus+ to be translated to addresses on the VME64 without having to explicitly map the two buses one-to-one. Also, control and configuration information is contained in each mapping translation register (i.e., page table entry) to more clearly define the cycle type requested on the VME64. The address modifier control lines AM[5:0] are an example of the unique requirements of the VME64. For each page of memory, these registers provide the following functions:

- Configure its operation.
- Make available operational status.
- Set the address range and the AM codes that allow its associated VME64 master to determine what it is to assert for a VME64 cycle.
- Select the byte swap mode to be used.
- Allow the translation of the Futurebus+ address into A16, A24, A32 or A64 VME64 address on a per page format with provision for scatter/gather addressing into the VME64 data bus. The control of each page allows for unique VME64 transaction type for each page specified.

Upon initialization or modification of a PMR, the Bridge controller (host) should clear the valid field before modifying the contents of the register, to avoid unintended transactions from occurring. After the register contents have been modified, then the valid field may be re-enabled.

This register set contains two categories. The first is the configuration information common to all VME64 Bridge master transactions. The second set specifies the unique configuration information for each page defined in the bridge system. Subclause 6.2.2 specifies the implementation details of these registers.

Copyright © 1995 IEEE All Rights Reserved

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:49:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1014.1-1994