IEEE Draft Std P1275.4/D14 Standard for Boot (Initialization Configuration) Firmware Supplement for IEEE 896 (Futurebus+) Bus

18 August, 1994

DRAFT EDITION for SPONSOR BALLOT

Sponsored by the IEEE Bus Architecture Standards Committee
Prepared by the P1275 Working Group

This is an unapproved draft of a proposed IEEE Standard, subject to change. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities. If this document is to be submitted to ISO or IEC, notification shall be given to the IEEE Copyright Administrator. Permission is also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position. Other entities seeking permission to reproduce this document for standardization or other activities, or to reproduce portions of this document for these or other uses must contact the IEEE Standards Department for the appropriate license. Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, PO Box 1331
Piscataway, NJ 08855-1331 USA

Copyright © 1992-94 by the Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017 USA
All rights reserved.

Foreword by the Chairman of the Working Group

(This foreword is not a part of IEEE Draft Std P1275.4/D14, *Standard for Boot (Initialization Configuration) Firmware Supplement for IEEE 896 (Futurebus+) Bus*)

Introduction

Firmware is the ROM-based software that controls a computer between the time it is turned on and the time the primary operating system takes control of the machine. Firmware's responsibilities include testing and initializing the hardware, determining the hardware configuration, loading (or booting) the operating system, and providing interactive debugging facilities in case of faulty hardware or software.

Open Firmware is the firmware architecture defined by IEEE Std 1275-1994, *Standard for Boot (Initialization Configuration) Firmware, Core Requirements and Practices.* That standard is bus-independent, instruction-set-independent, and system-independent.

The core requirements and practices specified by IEEE Std 1275-1994 must be supplemented by system-specific requirements to form a complete specification for the firmware for a particular system. This standard establishes such additional requirements pertaining to the bus architecture defined by IEEE Std 896-1991, *Futurebus+ Standard*.

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Working Group Members

The following individuals were members of the Working Group at the time this document was produced:

William M. (Mitch) Bradley, Chairman, FirmWorks
David M. Kahn, Vice Chairman, Sun Microsystems, Inc.
John Rible, Draft Editor, FORCE COMPUTERS Inc.
Luan D. Nguyen, Secretary, IBM Corporation
Mike Tuciarone, FirmWorks
David L. Paktor, FORCE COMPUTERS Inc.
Paul Thomas, Sun Microsystems, Inc.
Ron Hochsprung, Apple Computer Inc.
Yongjae Rim, IBM Corporation
Martin Walsh, Sun Microsystems, Inc.
Thanos Mentzelopoulos, Antel Inc.

Contributors

The following individuals also have contributed to this document:

Mike Williams, SHL
Paul Fischer, FORCE COMPUTERS, Inc.

Sponsor Balloting Body

The following individuals were members of the Sponsor Balloting Body:

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Table of Contents

1. Overview 1
2. References 1
3. Terms 1
4. Bus characteristics 1
4.1. Physical address formats and representations 1
4.2. 3
4.2. FCode interpretation semantics 3
5. Bus nodes 3
5.1. Properties 3
5.2. Methods 4
6. Child nodes 5
6.1. Properties 5
6.2. Methods 5
7. User interface extensions 6
7.1. Bus-specific interpretations of standard commands 6
7.2. Bus-specific FCodes 6
7.3. Bus-specific user interface commands 6
Annex A. (Normative) Futurebus+ CSRs 8
A.1 CSR command name behavior 8
A.2 IEEE 896 core CSRs 8
A.3 IEEE 896 Futurebus+ CSRs 10
A.4 IEEE 896 ROM bus information block CSRs 10
A.5 IEEE 896 ROM root directory CSR 10

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1. Overview

This standard describes the application of IEEE Std 1275-1994, *Standard for Boot (Initialization Configuration) Firmware, Core Requirements and Practices*, to the bus defined by IEEE Std 896-1991, *Futurebus+ Standard*. This standard addresses both Futurebus+ monarch-capable devices that contain an IEEE 1275 FCode evaluator and Futurebus+ devices that are probed by such an evaluator.

2. References

[1] IEEE Std 1275-1994, *Standard for Boot (Initialization Configuration) Firmware, Core Requirements and Practices*

[2] IEEE Std 896.1-1991, *Futurebus+ Standard, Logical Protocol Specification;*

[3] IEEE Std 896.2-1991, *Futurebus+ Standard, Physical Layers and Profiles A, B, and F;*

[4] IEEE Std 1212-1991, *Control and Status Register Architecture;*

3. Terms

This standard uses technical terms as they are defined in the documents cited in clause 2, plus the following terms:

3.1 bus node. In the device tree, a descendant node that represents the interface, or "bridge," between a Futurebus+ bus and its parent (which could be another bus).

3.2 child node. In the device tree, a descendant node that represents a device plugged into a Futurebus+ bus.

3.3 core specification. Synonym for IEEE Std 1275-1994, *Standard for Boot (Initialization Configuration) Firmware, Core Requirements and Practices*, i.e., the standard that specifies the system-independent and bus-independent requirements for Open Firmware.

3.4 Open Firmware. The firmware architecture defined by IEEE Std 1275-1994 and its applicable supplements or, when used as an adjective, a software component compliant with such an architecture.

4. Bus characteristics

4.1. Physical address formats and representations

Futurebus+ defines a 64-bit address domain independent of the physical configuration of backplanes and boards within it. This domain consists of three non-overlapping regions: a low-data region with 32-bit addresses from 0 to efff.ffff, a high-data region with 64-bit addresses from 0000.0000.f000.0000 to ffff.ffff.efff.ffff, and a configuration region with 32-bit addresses from f000.0000 to ffff.ffff. Futurebus+ hardware distinguishes 32-bit addresses from 64-bit addresses, so the configuration region is distinct from the 0000.0000.f000.0000-to-0000.0000.ffff.ffff subrange of the high-data region. (See Table 1.) Standard devices are not permitted to generate 64-bit addresses in the range ffff.ffff.f000.0000 to ffff.ffff.ffff.ffff, so it is possible to encode all valid Futurebus+ addresses with 64 bits of information.

Table 1—Hybrid Addressing (IEEE Std 896.2-1991, pg. 29)

CM7*	ADDRESS FIELD FROM	TO	DESCRIPTION
0	0000 0000	EFFF FFFF	A32 memory 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 EFFF FFFF	A64 memory address space
1	FFFF FFFF F000 0000	FFFF FFFF FFFF FFFF	PROHIBITED (Use the A32 CSR space)

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Addresses within the low and high data regions are dynamically assigned to board functions, with no fixed correspondence between addresses and physical backplane "slots." Addresses within the configuration region are "geographic," with specific subranges of that region assigned to specific physical backplane "slots."

4.1.1. Numeric representation

The numeric representation of a Futurebus+ address shall consist of two cells denoted *phys.hi phys.lo*, each containing 32 bits of information, encoded as follows:

Configuration region addresses:

```
        bit#:  3322.2222.2222.1111.1111.1100.0000.0000
               1098.7654.3210.9876.5432.1098.7654.3210
phys.hi cell:  1111.1111.1111.1111.1111.1111.1111.1111
phys.lo cell:  1111.bbbb.bbbb.bbss.sssn.pppp.pppp.pppp
```

where

[bb.bbbb.bbbb] is the 10-bit *bus number* (0-1022; 1023 is the local bus);

[ss.sssn] is the 6-bit *module number* (0-61; 62 is the local module, 63 the local bus broadcast module). The [ss.sss] portion thereof is the five-bit geographic *slot number* and [n] is the one bit *node number*;

[pppp.pppp.pppp] is the 12-bit *offset* within the module identified by the other fields.

Low data region addresses:

```
        bit#:  3322.2222.2222.1111.1111.1100.0000.0000
               1098.7654.3210.9876.5432.1098.7654.3210
phys.hi cell:  0000.0000.0000.0000.0000.0000.0000.0000
phys.lo cell:  xxxx.pppp.pppp.pppp.pppp.pppp.pppp.pppp
```

where

[xxxx.pp...pp] is a 32-bit unsigned number denoting a 32-bit address in the range 0 - efff.ffff. At least one of the bits in [xxxx] shall be zero, as [1111] in this field denotes an address in the high data region.

High data region addresses:

```
        bit#:  3322.2222.2222.1111.1111.1100.0000.0000
               1098.7654.3210.9876.5432.1098.7654.3210
phys.hi cell:  hhhh.hhhh.hhhh.hhhh.hhhh.hhhh.hhhh.hhhh
phys.lo cell:  xxxx.pppp.pppp.pppp.pppp.pppp.pppp.pppp
```

where

[hh...hh.xxxx.pp...pp] is a 64-bit unsigned number denoting a 64-bit address in the range 0000.0000.f000.0000 - ffff.ffff.efff.ffff. At least one of the bits in [hh...hh.xxxx] shall be zero, as [11...11.1111] in these fields denotes an address in the configuration region. If all of the bits in [hh...hh] are zeroes, all of the bits in [xxxx] shall be ones, as [00...00.xxxx] denotes an address in the low data region otherwise.

4.1.2. Text representations

The text representation of a data address is "`address`", the ASCII hexadecimal representation of a 64-bit number.

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

The text representation of a configuration address is "`module,offset`", where *module* and *offset* are the ASCII hexadecimal representation of numbers. *Module* and *offset* are separated by a comma.

NOTE—The bus number is not required in the text representation of a child node's configuration address because it is implied by the child node's position in the device tree (i.e., the child's bus number is the same as the "**`bus#`**" property of the bus node for the local bus).

4.1.3. Conversion between representations

Conversion of hexadecimal numbers from text representation to numeric representation shall be case-insensitive. Leading zeros and internal place holders, e.g., as in 0012.3456, shall be ignored during conversion.

Conversion from numeric representation to text representation shall use the lower case forms of the hexadecimal digits in the range a..f, shall suppress leading zeros, and shall contain no internal place holders.

4.2. FCode interpretation semantics

Each Futurebus+ module may have an FCode program associated with it that will be evaluated as a result of executing **`probe-self`**. An IEEE 896.2-defined Root Directory entry, consisting of one of the key-value pairs described below, shall specify the address of any such associated FCode program. See IEEE Std 1212-1991, pgs. 98-101, for the specification of the directory entry formats.

8-bit key	24-bit Directory Entry value
`0x30`	Quadlet-aligned byte offset of the start of FCode within the 4K CSR space.
`0x70`	Quadlet offset (byte offset/4) of a quadlet within the 4K CSR space containing the 32-bit offset of the start of FCode within the Extended Units space.
`0xB0`	Indirect offset of a ROM directory leaf entry containing the FCode.

NOTE—The Futurebus+ specification requires only quadlet access to the CSR space, so FCode that exists there (key entries 0x30 and 0xB0, above) must be physically accessed by quadlet. Futurebus+ FCode evaluators shall accept FCode with a spread of 1 or 4.

5. Bus nodes

5.1. Properties

5.1.1. Core-specified properties for bus nodes

The following Open Firmware standard properties have special interpretations for Futurebus+.

"**`device_type`**" S

Standard *prop-name* to specify the implemented interface.

The meaning of this property is as defined in Open Firmware. A Standard Package conforming to this specification and corresponding to a device that implements a Futurebus+ interface shall implement this property with the string value "`futurebus+`".

"**`ranges`**" S

Standard *prop-name* to specify child-to-parent address mapping.

The meaning of this property is as defined in Open Firmware. The "**`ranges`**" property should specify the regions of the parent's address space that are "mapped through" to the Futurebus+. The format of child physical addresses is as specified in 4.1.1.

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

"**#size-cells**" S

Standard *prop-name* to specify the number of cells necessary to represent a physical address.

prop-encoded-array: Integer constant 2, encoded as with **encode-int**.

The value of "**#size-cells**" for Futurebus+ bus nodes is 2.

5.1.2. Bus-specific properties for bus nodes

"**bus#**" S

Standard *prop-name* to specify the number assigned to the Futurebus+ local to this node.

prop-encoded-array: Integer constant, encoded as with **encode-int**.

5.2. Methods

5.2.1. Core-specified methods for bus nodes

A standard package implementing the "futurebus+" device type shall implement the following Open Firmware methods with physical address representations as specified in 4.1.1.

open	(-- okay?)	Prepare this device for subsequent use.
close	(--)	Close this previously-**open**ed device.
map-in	(phys.lo phys.hi size -- virt)	Map the specified region, return a virtual address.
map-out	(virt size --)	Destroy mapping from previous **map-in**.
dma-alloc	(size -- virt)	Allocate a memory region for later use.
dma-free	(virt size --)	Free memory allocated with **dma-alloc**.
dma-map-in	(virt size cacheable? -- devaddr)	Convert virtual address to device bus DMA address.
dma-map-out	(virt devaddr size --)	Free DMA mapping set up with **dma-map-in**.
dma-sync	(virt devaddr size --)	Synchronize (flush) DMA memory caches.
probe-self	(arg-str arg-len reg-str reg-len fcode-str fcode-len --)	Interpret FCode as a child of this node.
decode-unit	(addr len -- phys.lo phys.hi)	Convert unit text string to physical address.
encode-unit	(phys.lo phys.hi -- addr len)	Convert physical address to unit text string.

5.2.2. Bus-specific methods for bus nodes

None.

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

6. Child nodes

6.1. Properties

6.1.1. Core-specified properties for child nodes

The following Open Firmware standard properties have special interpretations for Futurebus+.

"**reg**" S

Standard *prop-name* to define the package's registers.

The meaning of this property is as defined in Open Firmware. The first entry of the "**reg**" property shall specify the region of the address space that contains the Futurebus+ configuration region for this node.

6.1.2. Bus-specific properties for child nodes

Standard Packages corresponding to devices that are children of a Futurebus+ shall implement the following properties.

"**has-fcode**" S

Bus specific *prop-name* to denote that the node was created from FCode.

prop-encoded-array: <none>

The presence of this property indicates that this device node was not created automatically from information in configuration registers, but rather involved the evaluation of an FCode program.

"**module-vendor-id**" S

Bus specific *prop-name* to identify the manufacturer.

prop-encoded-array: a byte array encoded as with **encode-bytes**.

This property shall contain the IEEE-assigned company_id, consisting of six hexadecimal digits, of the vendor that manufactured the module, as described in IEEE 896.2a-1994 3.2.5.1.

"**node-capabilities**" S

Bus specific *prop-name* to specify the node capabilities.

prop-encoded-array: a byte array encoded as with **encode-bytes**.

This property shall contain bytes 1..3 of the NODE_CAPABILITIES ROM entry, consisting of the 24 assigned and reserved bits that indicate various capabilities of the node, as defined in IEEE 896.2a-1994 3.2.5.3.

"**node-memory-extent**" S

Bus specific *prop-name* to specify a bus-visible address space.

prop-encoded-array: a byte array encoded as with **encode-bytes**.

This property shall contain bytes 1..3 of the NODE_MEMORY_EXTENT ROM entry, consisting of the 24 assigned and reserved bits that indicate the size and alignment requirements of an address region mapped into Futurebus+ data space, as defined in IEEE 896.2a-1994 3.2.5.5.1.

"**node-units-extent**" S

Bus specific *prop-name* to specify a bus-visible address space.

prop-encoded-array: a byte array encoded as with **encode-bytes**.

This property shall contain bytes 1..3 of the NODE_UNITS_EXTENT ROM entry, consisting of the 24 assigned and reserved bits that indicate the size and alignment requirements of an address region mapped into Futurebus+ data space, as defined in IEEE 896.2a-1994 3.2.5.5.2.

6.2. Methods

There are no special requirements on the methods for Futurebus+ *child nodes* beyond those described in the core specification.

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7. User interface extensions

This section describes commands which are only applicable to Futurebus+ devices or have special meaning when used with Futurebus+ devices.

7.1. Bus-specific interpretations of standard commands

None.

7.2. Bus-specific FCodes

None

7.3. Bus-specific user interface commands

An Open Firmware-compliant user interface on a system with a built-in Futurebus+ bus should implement the commands listed here and the user interface commands listed in Annex A.

`bsn>pa` (bus# slot# node# -- phys.lo phys.hi)
Convert the given bus, slot, and node numbers into a Futurebus+ configuration region base address.

`csr!` (quad offset --)
Store *quad* at the address equal to the base of the current CSR region plus *offset*.

`csr@` (offset -- quad)
Read *quad* from the address equal to the base of the current CSR region plus *offset*.

`fb+c!` (byte phys.lo phys.hi --)
Store *byte* at the Futurebus+ byte-aligned address *phys.lo phys.hi*.

`fb+c@` (phys.lo phys.hi -- byte)
Read *byte* from the Futurebus+ byte-aligned address *phys.lo phys.hi*.

`fb+d!` (quad.lo quad.hi phys.lo phys.hi --)
Store *quad.lo quad.hi* at the Futurebus+ quadlet-aligned address *phys.lo phys.hi*.

`fb+d@` (phys.lo phys.hi -- quad.lo quad.hi)
Read *quad.lo quad.hi* from the Futurebus+ quadlet-aligned address *phys.lo phys.hi*.

`fb+l!` (quad phys.lo phys.hi --)
Store *quad* at the Futurebus+ quadlet-aligned address *phys.lo phys.hi*.

`fb+l@` (phys.lo phys.hi -- quad)
Read *quad* from the Futurebus+ quadlet-aligned address *phys.lo phys.hi*.

`fb+w!` (w phys.lo phys.hi --)
Store *w* at the Futurebus+ doublet-aligned address *phys.lo phys.hi*.

`fb+w@` (phys.lo phys.hi -- w)
Read *w* from the Futurebus+ doublet-aligned address *phys.lo phys.hi*.

`pa>bsn` (phys.lo phys.hi -- bus# slot# node#)
Convert a Futurebus+ configuration region base address into bus, slot, and node numbers.

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

select-fb+csrs (phys.lo phys.hi --)

Set the current CSR region (4K bytes) to the one specified by *phys.lo phys.hi*, and map it in.

If there is a different current CSR region, unmap it and map in the new region. If there is no current CSR region, map in the new region.

unselect-fb+csrs (--)

Unmap the current CSR region (4K bytes).

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Annex A. (Normative) Futurebus+ CSRs

This annex lists the Futurebus+ CSR names defined in IEEE std 896.2-1991 along with their configuration region offset and Open Firmware command name. If the user interface commands giving access to the Futurebus+ CSRs are implemented, as recommended in 6.3, they shall have the names and behavior described below.

NOTE—In most cases, the command name has been formed from the first letter of each of the words in the name given in IEEE std 896.2, subclauses 3.2.3 through 3.2.5.

A.1 CSR command name behavior

A.1.1 Single CSR access commands

CSR command-names without "**[]**" in them (*single-command-names*) refer to a single quadlet within the selected configuration region. Their behavior, similar to that of **value** words, is described by the following glossary entry prototypes.

single-command-name (-- value)

Push the *value* of the named CSR onto the stack.

Read the quadlet *value* from the current CSR base address plus the offset of the named CSR and push it onto the stack.

to *single-command-name* (value --)

Store *value* into the named CSR.

The quadlet *value* is popped off the stack and stored at the current CSR base address plus the offset of the named CSR.

A.1.1 Array CSR access commands

CSR command names with "**[]**" in them (*array-command-names*) refer to more than one quadlet within the selected configuration region, and therefore require an index value *n* to specify the quadlet being accessed. Their behavior, similar to that of **value** words, is described by the following glossary entry prototypes.

NOTE—IEEE 896 allows some of the regions covered by this class of CSRs to be byte addressed. Open Firmware only supports quadlet access to maintain portability.

array-command-name (n -- value)

Push the *n*th quadlet of the named CSR onto the stack.

Read the quadlet *value*, located at the current CSR base address plus the offset of the named CSR plus 4 times *n*, and push it onto the stack.

to *array-command-name* (value n --)

Store *value* into the *n*th quadlet of the named CSR.

Store the quadlet *value* at the current CSR base address plus the offset of the named CSR plus 4 times *n*.

A.1.3 Configuration region selection

The command **map-fb+csrs** is used to set the base address of the configuration region that will be accessed by the CSR command names that follow.

A.2 IEEE 896 core CSRs

CSR Name	Offset	Command name
STATE_CLEAR	0	**fb+sc**
STATE_SET	4	**fb+ss**
NODE_IDS	8	**fb+ni**
RESET_START	12	**fb+rs**

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.2 IEEE 896 core CSRs, continued

CSR Name	Offset	Command name
INDIRECT_ADDRESS	16	**fb+ia**
INDIRECT_DATA	20	**fb+id**
SPLIT_TIMEOUT_HI	24	**fb+sth**
SPLIT_TIMEOUT_LO	28	**fb+stl**
ARGUMENT	32	**fb+a**
TEST_START	40	**fb+tst**
TEST_STATUS	44	**fb+tss**
UNITS_BASE_HI	48	**fb+ubah**
UNITS_BASE_LO	52	**fb+ubal**
UNITS_BOUND_HI	56	**fb+uboh**
UNITS_BOUND_LO	60	**fb+ubol**
MEMORY_BASE_HI	64	**fb+mbah**
MEMORY_BASE_LO	68	**fb+mbal**
MEMORY_BOUND_HI	72	**fb+mboh**
MEMORY_BOUND_LO	76	**fb+mbol**
INTERRUPT_TARGET	80	**fb+it**
INTERRUPT_MASK	84	**fb+im**
CLOCK_VALUE_HI	88	**fb+cvh**
CLOCK_VALUE_LO	92	**fb+cvl**
CLOCK_TICK_PERIOD_HI	96	**fb+ctph**
CLOCK_TICK_PERIOD_LO	100	**fb+ctpl**
CLOCK_STROBE_ARRIVED_HI	104	**fb+csah**
CLOCK_STROBE_ARRIVED_LO	108	**fb+csal**
CLOCK_STROBE	112	**fb+cs**
Reserved for IEEE 1212	116	
CLOCK_REFERENCE_HI	120	**fb+crh**
CLOCK_REFERENCE_LO	124	**fb+crl**
MESSAGE_REQUEST	128..188	**[]fb+mrq**
MESSAGE_RESPONSE	192..252	**[]fb+mr**
Reserved for IEEE 1212 extensions	256..380	
ERROR_HI	384	**fb+eh**
ERROR_LO	388	**fb+el**
FAILING_ADDRESS_HI	392	**fb+fah**
FAILING_ADDRESS_LO	396	**fb+fal**
BYTE_LANE_IN_ERROR	400	**fb+blie**
DATA_HOLD	404..432	**[]fb+dh**
TAG_HOLD	436	**fb+th**
TIME_STAMP_HI	440	**fb+tsh**
TIME_STAMP_LO	444	**fb+tsl**
Reserved for IEEE 896 extensions	448..476	
Reserved for vendor extensions	480..508	

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.3 IEEE 896 Futurebus+ CSRs

CSR Name	Offset	Command name
LOGICAL_COMMON_CONTROL	512	**fb+lcc**
LOGICAL_MODULE_CONTROL	516	**fb+lmc**
BUS_PROPAGATION_DELAY	520	**fb+bpd**
COMPETITION_SETTLING_TIME	524	**fb+cst**
TRANSACTION_TIMEOUT	528	**fb+tt**
MESSAGE_PASSING_SELECT_MASK_HI	532	**fb+mpsmh**
MESSAGE_PASSING_SELECT_MASK_LO	536	**fb+mpsml**
BUSY_RETRY_COUNTER	540	**fb+brc**
BUSY_RETRY_DELAY	544	**fb+brd**
ERROR_RETRY_COUNTER	548	**fb+erc**
ERROR_RETRY_DELAY	552	**fb+erd**
Reserved for IEEE 896 extensions	556..764	
Reserved for profile extensions	768..1020	

A.4 IEEE 896 ROM bus information block CSRs

CSR Name	Offset	Command name
ROM_HEADER	1024	**fb+rh**
BUS_HEADER (ASCII “0896”)	1028	**fb+bh**
PROFILE_ID_HI	1032	**fb+pih**
PROFILE_ID_LO	1036	**fb+pil**
MODULE_LOGICAL_CAPABILITY	1040	**fb+mlc**
NODE_CAPABILITIES_EXT	1044	**fb+nce**
COMPETITION_INTERNAL_DELAY	1048	**fb+cid**
PACKET_SPEED	1052	**fb+ps**
MESSAGE_FRAME_SIZE	1056	**fb+mfs**
BUSY_RETRY_COUNTER_CAPABILITY	1060	**fb+brcc**
BUSY_RETRY_DELAY_CAPABILITY	1064	**fb+brdc**
ERROR_RETRY_COUNTER_CAPABILITY	1068	**fb+ercc**
ERROR_RETRY_DELAY_CAPABILITY	1072	**fb+erdc**
Reserved for IEEE 896 extensions	1076..1084	

A.5 IEEE 896 ROM root directory CSR

CSR Name	Offset	Command name
ROOT_DIRECTORY_HEADER	1088	**fb+rdh**

Copyright © 1992-1994 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.