Proposal for Working Draft

T13 1248D

Revision 5
28 October 1997

Information Technology - 1394 to AT Attachment - Tailgate

This is an internal working document of T13, a Technical Committee of Accredited Standards Committee NCITS. As such, this is not a completed standard and has not been approved. The contents may be modified by the T13 Technical Committee. The contents are actively being modified by T13. This document is made available for review and comment only.

Permission is granted to members of NCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of NCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any commercial or for-profit replication or republication is prohibited.

Technical Editor:

Jonathan L. Hanmann
Western Digital Corporation
8105 Irvine Center Drive
Irvine Ca, 92618
USA

Tel: 714 932-5189
Fax: 714 932-6010
Email: JHanmann@dt.wdc.com

Reference number
ANSI X3.*** - 199x
Printed June, 30, 2000 1:21PM

Other Points of Contact:

T13 Chair
Gene Milligan
Seagate Technology
OKM 251
10323 West Reno (West Dock)
P.O. Box 12313
Oklahoma City, OK 73157-2313
Tel: 405-324-3070
Fax: 405-324-3794

T13 Vice-Chair
Pete McLean
Maxtor Corporation
2190 Miller Drive
Longmont, CO 80501
Tel: 303-678-2149
Fax: 303-682-4811

NCITS Secretariat
Administrator Standards Processing
1250 Eye Street, NW Suite 200
Washington, DC 20005
Tel: 202-626-5738
Fax: 202-638-4922
Email: NCITS@ITIC.NW.DC.US

T13 Reflector
Internet address for subscription to the T13 reflector: majordomo@dt.wdc.com
Send email to above account and include in BODY of text, on a line by itself the following:
"subscribe t13 [your email address]"
Internet address for distribution via T13 reflector: T13@dt.wdc.com

T13 Anonymous FTP Site
fission.dt.wdc.com
T13 directory is: "/ x3t13 "

T13 mailings
Global Engineering
15 Inverness Way East
Englewood, CO 80112-5704
Tel: 303-792-2181 or 800-854-7179
Fax: 303-792-2192

DOCUMENT STATUS

Revision 0 - 4 January 1997

Document created from revision 0.4 of ad-hoc 1394 to Tailgate specification proposed by Compaq Computer. Document rev 0.4 was reformatted to meet ANSI editorial requirements with the intent to transfer the development of this technology from the ad-hoc 1394 tailgate group to the ANSI X3T13 AT Attachment Group for further development and standardization.

D97107R0 Revision A-E - 15 January 1997

Edits performed to bring document more in conformance with accepted X3T13 standards guidelines and rules. Changes also made per tailgate editors meeting during X3T10 week in Dallas (6 January 1997). Most changes where editorial rather than technical. Some technical changes where made and are listed below. Since the charter of the tailgate editor's meeting was to make only editorial changes the technical changes should not be considered as finalized until review and acceptance by the working group.

1. Security model and commands changed. Many of those at the editors meeting, including the technical editor, had an incorrect understanding of the security related modifications to the SBP-2 standard. This invalidated some of the command and model definitions for the tailgate security. In order to bring the tailgate standard into compliance with the SBP-2 standard some changes where made. The opportunity was taken at this time to further simplify the tailgate security model and commands.
2. *tailgate_status* field added to the status block as per discussion in San Jose tailgate meeting. Table provided by Steve Finch (SSI).

D97107R1 - 10 February 1997

Edits in accordance with January Disk Boys meeting in San Jose. Also made changes in cover in accord with comments from January X3T13 plenary meeting.

1. Changed "device" to "logical unit" in security model section.
2. Clarified that tailgate only supports a single login per logical unit.
3. Moved statement that dual logical unit tailgates shall implement a mechanism to prevent starving one logical unit at the expense of a faster more active logical unit. This was in the model section. It was moved to a new section in the protocol section. This permits the model section to be completely informative.
4. Corrected Abort Task Set figures and text per January Disk Boys meeting.
5. Added table that documents the processing and actions performed by the tailgate for all reset functions.

D97107R2 - 6 March 1997

Edits in accordance with February Disk Boys meeting. An additional change based upon Disk Boys reflector traffic was also made.

1. Updated security mode state diagram based upon prior requests to break out each transition condition. Additional improvements made for missing transitions in states 0 and 1. Discussion on Bus Reset processing also required an additional state and appropriate transitions.
2. Replaced <TBD> with REQUESTED ABORTED per prior Disk Boys agreement on correct status for Dummy ORBs.
3. Editorial change in step 1 of Abort Task Set to improve readability of sentence.
4. Added a read only bit in the ATA Timing Capability CSR to indicate whether the tailgate supports Power Management.
5. Added one read only bit and one read/write bit in the ATA Timing Select CSR to indicate power state and control power for each logical unit.

D97107R3 - 7 April 1997

Edits in accordance with March 11 Disk Boys meeting.

1. Corrected byte count on PIO/DMA transfers in model section to correctly reflect ATAPI defined functionality.
2. Updated definition of PIO as per Disk Boys discussion.
3. Renamed Power Management (Control) bits in to better clarify their purpose. Also updated text to match renamed bits. Fixed cosmetic issue with use of fonts.
4. Added code 6 (Target Reset Completed) to Tailgate Status byte in Status block per request from Symbios Logic and subsequent discussion at Disk Boys meeting.

1248D Revision 4 - 7 August 1997

1. Editorial changes to comply with ANSI format.
2. Changed document to reflect T13 project number.
3. Change ORB size in logical unit characteristcs entry (key type/value = $3A_{16}$). Value changed to be in quadlets rather than bytes (8 instead of 32).
4. Eliminated definitions for obsolete and retired as they are not used within this document.
5. Eliminated references to ANSI C and SCSI-3 Architectural Model as they are not truly referenced by this document.
6. Corrected some errors in tailgate specific CSRs as regards bit read and write definitions. Also renamed u (ultradma) bit in ATA_TIMING_SELECT CSR to M bit to reduce conflict/confusion with the u standard definition for CSRs.
7. Expanded register delivered and packet delivered ORBs to permit renaming of pd bit to dma bit for clarity. Also changed referenced to pd bit throughout document. Renamed rdc to rd in figure and throughout document.
8. Corrected font problems in several figures
9. Updated access security state diagram and associated text to closely match 1394 Native Mass Storage Profile.
10. Deleted unreferenced and unnecessary definitions: “device selection”, “unit attention condition”, and “unrecoverable error”.

1248D Revision 5 - 28 October 1997

1. Made revisions to document to integrate accepted revisions based upon response to tailgate letter ballot and comments (T13/D97006R0 and D97009R0). The formal responses to the letter ballot comments may be found in document T13/D97151R1.
2. A few minor editorial changes not directly addressed by the letter ballot comments or the letter ballot response but related to them.
3. Add statement requested by October T13 Plenary on patent statement and 1394-1995 normative reference.

ANSI®
X3.**-199x**

American National Standard
for Information Systems—

1394 to AT Attachment – Tailgate

Secretariat
Information Technology Industry Council

Approved mm dd yy

American National Standards Institute, Inc.

Abstract

This standard specifies a bridge protocol between 1394 host systems and storage devices using the AT Attachment Interface. It provides a common attachment interface for systems manufacturers, system integrators, software suppliers, and suppliers of intelligent storage devices.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holder's of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made.

No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard. See clause 2.

Published by
American National Standards Institute
11 West 42nd Street, New York, New York 10036

Copyright 199n by American National Standards Institute
All rights reserved.

Contents Page

Foreword ... iv
Introduction ... vii
1 Scope ... 1
2 Definitions, abbreviations, and conventions ... 3
2.1 Definitions and abbreviations ... 3
2.2 Conventions ... 4
2.3 ... 6
CSR Specifications ... 7
2.4 State machines ... 9
2.5 Normative references ... 9
3 9
Tailgate model (Informative) ... 10
3.1 Single logical unit tailgate overview ... 10
3.2 Dual logical unit tailgate overview ... 11
3.3 Fetch engine/execution engine interface ... 12
3.4 Bus interface unit ... 13
3.5 Initiator processing ... 13
4 13
Protocol definitions ... 14
4.1 Access and security ... 14
4.2 Task management ... 17
4.3 Tailgate command level protocols ... 21
4.4 Dual logical units ... 28
5 28
Command and status registers (CSRs) ... 29
5.1 Core CSRs ... 29
5.2 Tailgate specific CSRs ... 29
6 32
Data structure definitions ... 33
6.1 Management ORBs ... 33
6.2 Tailgate command blocks ... 34
6.3 Tailgate Status Block Definition ... 36
7 37
Configuration ROM ... 38
7.1 Unit_Spec_ID entry ... 38
7.2 Unit_SW_Version entry ... 38
7.3 Command_Set_Spec_ID entry ... 38
7.4 Command_Set_Version entry ... 39
7.5 Logical_Unit_Characteristics entry ... 39
7.6 Management_Agent entry ... 40
7.7 Logical_Unit_Number entry ... 40

Tables Page

1 Register definition fields ... 7
2 Read field values ... 8
3 Write effect field values ... 8
4 Reset definitions for tailgate ... 17
5 Command Block registers summary (read) ... 24
6 Command Block registers summary (write) ... 25
7 ATA timing capability and selection CSRs ... 29
8 *tailgate_status* field values ... 37

Figures Page

1 CSR specification example 7
2 State machine example 9
3 Single logical unit tailgate block diagram 11
4 Dual logical unit tailgate block diagram 12
5 Access security state diagram 15
6 Abort Task Set flowchart 19
7 Register and packet delivered command processing 23
8 ATA_TIMING_CAPABILITIES CSR specification 30
9 ATA_TIMING_SELECT CSR specification 31
10 Login ORB 33
11 Set Password ORB 34
12 32-byte tailgate ATA command ORB 35
13 32-bytes tailgate ATAPI command ORB 36
14 Tailgate status block 37
15 Unit_Spec_ID entry format 38
16 Unit_SW_Version entry format 38
17 Command_Set_Spec_ID entry format 39
18 Command_Set_Version entry format 39
19 Logical_Unit_Characteristics entry format 39
20 Management_Agent entry format 40
21 Logical_Unit_Number entry format 40

Annexes Page

A Initialization Procedure (informative) 41
B Initialization procedure tailgate execution engine models (informative) 43
C Required core registers for tailgate (informative) 51
D Dual device support summary (informative) 53

Foreword

(This foreword is not part of American National Standard X3.***-199*.)

This 1394 to AT Attachment - Tailgate standard is designed to provide a bridge protocol between AT Attachment -4 Packet devices and host systems implementing 1394.

This standard was developed by the ATA ad hoc working group of T13 a Technical Committee of Accredited Standards Committee NCITS during 1996-97. The standards approval process started in 1997. This document includes annexes that are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the NCITS Secretariat, Information Technology Industry Council, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on Information Processing Systems, NCITS. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the NCITS Committee had the following members:

James D. Converse, Chair

Donald C. Loughry, Vice-Chair

Joanne M. Flanagan, Secretary

Organization Represented	Name of Representative
American Nuclear Society	Geraldine C. Main
	Sally Hartzell (Alt.)
AMP, Inc	Edward Kelly
	Charles Brill (Alt.)
Apple Computer	Karen Higginbottom
Association of the Institute for Certification of Professionals (AICCP)	Kennath Zemrowski
AT&T/NCR	Thomas W. Kern
	Thomas F. Frost (Alt.)
Boeing Company	Catherine Howells
	Andrea Vanosdoll (Alt.)
Bull HN Information Systems, Inc.	William George
Compaq Computer Corporation	James Barnes
Digital Equipment Corporation	Delbert Shoemaker
	Kevin Lewis (Alt.)
Eastman Kodak	James D. Converse
	Michael Nier (Alt.)
GUIDE International	Frank Kirshenbaum
	Harold Kuneke (Alt.)
Hewlett-Packard	Donald C. Loughry
Hitachi America, Ltd.	John Neumann
	Kei Yamashita (Alt.)
Hughes Aircraft Company	Harold L. Zebrack
IBM Corporation	Joel Urman
	Mary Anne Lawler (Alt.)
National Communication Systems	Dennis Bodson
National Institute of Standards and Technology	Robert E. Roundtree
	Michael Hogan (Alt.)
Northern Telecom, Inc.	Mel Woinsky
	Subhash Patel (Alt.)
Neville & Associates	Carlton Neville
Recognition Technology Users Association	Herbert P. Schantz
	G. Edwin Hale (Alt.)
Share, Inc	Gary Ainsworth
	David Thewlis (Alt.)

Sony Corporation	Michael Deese
Storage Technology Corporation	Joseph S. Zajaczkowski
	Samuel D. Cheatham (Alt.)
Sun Microsystems	Scott Jameson
	Gary Robinson (Alt.)
3M Company	Eddie T. Morioka
	Paul D. Jahnke (Alt.)
Unisys Corporation	John L. Hill
	Stephen P. Oksala (Alt.)
U.S. Department of Defense	William C. Rinehuls
	C. J. Pasquariello (Alt.)
U.S. Department of Energy	Alton Cox
	Lawrence A. Wasson (Alt.)
U.S. General Services Administration	Douglas Arai
	Larry L. Jackson (Alt.)
Wintergreen Information Services	Joun Wheeler
Xerox Corporation	Dwight McBain
	Roy Peirce (Alt.)

Technical Committee T13 on AT Attachment, that reviewed this standard, had the following members:

Gene Milligan, Chairman

Pete McLean, Vice-Chairman

Larry Lamers, Secretary

I. Dal Allan	Paul Raikunen	Richard Harcourt (Alt.)
Charles Brill	Yogi Schaffner	Pat LaVarre (Alt.)
Darrin Bulik	J. R. Sims	LeRoy Leach (Alt.)
Ben Chang	Victor Siu	Raymond Liang (Alt.)
Dan Colegrove	Ron Stephens	John Masiewicz (Alt.)
Greg Elkins	Curtis Stevens	Christopher Mayne (Alt.)
Mark Evans	Tokuyuki Totani	James McGrath (Alt.)
Lance Flake	Anthony Yang	Patrick Mercer (Alt.)
Tony Goodfellow	Carl Bonke (Alt.)	Marc Noblitt (Alt.)
Tom Hanan	Joe Chen (Alt.)	Ron Roberts (Alt.)
Richard Kalish	Mike Christensen (Alt.)	Yasuyuki Suemori (Alt.)
Kenichi Kojima	David Evans (Alt.)	Don Vohar (Alt.)
Hale Landis	Stephen Finch (Alt.)	Devon Worrell (Alt.)
Robert Liu	Robert Griffith (Alt.)	

1394 to ATA/ATAPI ad hoc working group of T13, that developed this standard, had the following additional participants:

Michael Alexenko	Scott Fierstein	Kevin Johnston
Nelson Arata	Steve Finch	David Jolley
T. R. Arvind	Bill Frank	Skip Jones
Geoffrey Barton	John Fuller	Marcus Kellerman
Bob Bellino	Anthony Fung	Rob Lash
Jake Berzon	Bill Galloway	Bryce Leach
Vilas Bhade	Carol Ann Garcia	Kenneth Lee
Rick Born	Edward A. Gardner	Tom Lenny
Gary Brandvold	Jim Gay	Arnold Limjoco
Richard Bronson	Peter Groz	Jerry Marazas
Mike Bryan	Jonathan Hanmann	Bill McFerrin
Ron Burns	Tracy Harmer	Masanori Mikatake
Frank Campbell	Greg Havenga	Danny Mitchell
Mike Chenery	Glenn Higa	Charles Monia
Tom Colligan	Randy Hines	Richard Mourn
Jay Cuccarese	Jack Hollins	Jim Msu
Hugh Curley	David James	Nedi Nadershahi
Tim Feldman	Peter Johansson	Kark Nakamura

Kazuo Nakashima
Tarl Neustaedter
Jon Newman
Michael Nguyen
Jes Nielsen
Ken Nordhauser
Tim Orsley
Bob Otis
Dennis Pak
Duncan Penman
Alan Perry
Ed Petrick
Alan Poeppelman
B. M. Ramesh
Mehran Ramezani
Darrell Redford
Wink Saville
Khorvash Sefidvash
Dave Skramsted
Scott Smyers
Robert Snively
Daryl Starr
Hiro Tahara
Wilson Tan
Steve Timm
Thinh Tran
Greg Urban
Mai Wang
Roger Wang
James Whitworth
Lee Wilson
Mike Winchell
Jeff Wolford
David Wooten
Charles Yeoh

Introduction

This standard encompasses the following:

Clause 1 describes the scope.

Clause 2 provides definitions, abbreviations, and conventions used within this document.

Clause 3 describes the model for tailgate attributes and operations. This clause is informative.

Clause 4 describes the access security model for the tailgate and the protocol for delivery of commands to devices attached to the tailgate.

Clause 5 contains descriptions of the registers addressable via the 1394 interface that are unique to this standard.

Clause 6 contains descriptions of the data structures for commands and status unique to this standard.

Clause 7 contains descriptions of the 1394 configuration ROM for this standard.

Annex A is an informative annex which describes the initialization sequence the initiator performs to identify and configure the tailgate and its attached devices.

Annex B is an informative annex which contains state diagrams and the associated descriptions, for single logical unit and dual logical unit with command overlap tailgate execution engines.

Annex C is an informative annex which contains a summary of the required registers addressable via the 1394 interface that are not unique to this standard. This is provided as reference information.

Annex D is an informative annex which contains a brief summary and list of major differences between single and dual logical unit tailgates.

AMERICAN NATIONAL STANDARD X3.****-199n

American National Standard
for Information Systems —

Information Technology—
1394 to AT Attachment – Tailgate

1 Scope

This standard specifies the protocol for passing ATA and ATAPI commands over the 1394 bus. It provides a common attachment interface for systems manufacturers, system integrators, software suppliers, and suppliers of intelligent storage devices.

The application environment for the 1394 to AT Attachment Interface is any system that interfaces AT Attachment storage devices via the 1394 Bus.

This standard defines the following attributes and features, required to interface AT Attachment devices via the standard 1394/SBP-2 mechanisms:

- Register addresses and bit definitions
- Request block and status block formats
- Command and status delivery
- Tailgate Protocol
- ATA and ATAPI command set restrictions

Restrictions have been applied to the ATA and ATAPI command sets and SBP-2 protocols to simplify the design of the tailgate. These restrictions are believed to have no material impact on tailgate performance or capabilities. Certain 1394 architectural characteristics, such as read and write data packet transfer size, have no bearing on this standard and are therefore not specified. These characteristics will be defined by the customer for the various tailgate implementations and target environments. The tailgate shall have the following limitations over those features and capabilities defined in the SBP-2 and ATA/ATAPI-4 standards:

- The tailgate shall not support 1394 isochronous capability. Only asynchronous data transfers are supported.
- The tailgate shall support a single login at a time to each logical unit.
- The Read Multiple ($C4_{16}$) and Write Multiple ($C5_{16}$) ATA commands shall not be supported by the tailgate.
- The Read Long (22_{16} and 23_{16}) and Write Long (32_{16} and 33_{16}) ATA commands shall not be supported by the tailgate.

- The Command Queuing and Overlap features defined by the ATA/ATAPI-4 standard shall not be supported by the tailgate.

2 Definitions, abbreviations, and conventions

2.1 Definitions and abbreviations

For the purposes of this American National Standard, the following definitions apply:

2.1.1 ATA (AT Attachment): ATA defines the physical, electrical, transport, and command protocols for the internal attachment of storage devices.

2.1.2 ATAPI (AT Attachment Packet Interface): A device implementing the Packet Command feature set.

2.1.3 command acceptance: A command is considered accepted whenever the tailgate writes to the Command Register and the device currently selected has its BSY bit equal to zero. An exception exists for the EXECUTE DIAGNOSTIC and DEVICE RESET command (see the ATA/ATAPI-4 standard for more detail on these commands).

2.1.4 Command Block registers: Interface registers used for delivering commands to the device or posting status from the device.

2.1.5 command packet: A command packet is a data structure transmitted to the device by a PACKET command that includes the command and command parameters.

2.1.6 Control Block registers: Interface registers used for device control and to post alternate status.

2.1.7 CSR (Control and Status Register): Control or status register addressable via the 1394 bus. The IEEE 1212 and IEEE 1394 standards define CSRs that compliant devices shall support. Those standards also define operations applicable to CSRs. The SBP-2 standard defines additional CSRs necessary for implementation of devices compliant to that standard. This standard defines additional CSRs necessary for implementation of devices compliant with this standard.

2.1.8 device: Device is a storage peripheral. Only devices compliant to the ATA or ATAPI standards are supported via tailgate. The term device throughout this document refers to peripherals compliant with the ATA or ATAPI standards.

2.1.9 DMA (direct memory access): A means of data transfer between device and host memory without processor intervention. When used in reference to ATA bus transfers this specifies the DMA transfer modes defined by the ATA/ATAPI-4 standard. In tailgate configurations the tailgate assumes the role of the host computer, as defined by the ATA/ATAPI-4 standard.

2.1.10 LBA (logical block address): This term defines the addressing of the device as being by the linear mapping of sectors.

2.1.11 ORB (Operation Request Block): This is the data structure containing information necessary to transport command information to a device via the 1394 bus.

2.1.12 packet delivered command: A command that is delivered to the device using the PACKET command via a command packet that contains the command and the command parameters.

2.1.13 PIO (programmed input/output): A means of accessing device registers. This term is used only in reference to ATA bus transfers and specifies the PIO transfer modes defined by the ATA/ATAPI-4 standard. In tailgate configurations the tailgate assumes the role of the host computer, as defined by the ATA/ATAPI-4 standard.

2.1.14 register: A term used to describe a read/write address on the tailgate that controls or configures device functionality. Within this standard, three possible register address spaces are possible. The first is the CSR space that is addressed via the 1394 bus. The other two are the ATA bus Command Block and Control Block register spaces. See definitions of those register address spaces for more detail. The term register shall, in the remainder of this document, refer to a register address in either the device Command Block or control block register space.

2.1.15 register delivered command: A command that is delivered to the device by placing the command and all of the parameters for the command in the device Command Block registers.

2.1.16 sector: A uniquely addressable set of 256 words (512 bytes).

2.1.17 signature: A unique set of values placed in the Command Block registers by the device to differentiate register delivered command devices and packet delivered command devices.

2.2 Conventions

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, then text.

2.2.1 Keywords

Several keywords are used to differentiate between different levels of requirements and optionality, as follows:

expected – A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

mandatory – A keyword indicating items to be implemented as defined by this standard.

may – A keyword that indicates flexibility of choice with no implied preference.

optional – This term describes features that are not required by this standard. However, if any optional feature defined by the standard is implemented, it shall be done in the way defined by the standard. Describing a feature as optional in the text is done to assist the reader.

reserved – reserved bits, bytes, words, fields, and code values are set aside for future standardization. Their use and interpretation may be specified by future extensions to this or other standards. A reserved bit, byte, word, or field shall be set to zero, or in accordance with a future extension to this standard. The recipient shall not check reserved bits, bytes, words, or fields. Receipt of reserved code values in defined fields shall be treated as an error.

shall – A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other conforming products.

should – A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “it is recommended”.

Lowercase is used for words having the normal English meaning. Certain words and terms used in this American National Standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 0 or in the text where they first appear.

The names of abbreviations, commands, fields, and acronyms used as signal names are in all uppercase (e.g., IDENTIFY DEVICE). Fields containing only one bit are usually referred to as the "name" bit instead of the "name" field. (See 0 for the naming convention used for naming bits.)

Names of device registers begin with a capital letter (e.g., Cylinder Low register).

2.2.2 Numbering

Numbers that are not immediately followed by a subscript are in decimal. Numbers may also be followed by a subscript, for example 0011000100110011_2 or 13_{16}, which indicate binary and hexadecimal numbers, respectively.

2.2.3 ATA/ATAPI-4 Register Bit conventions

Bit names are shown in all uppercase letters except where a lowercase n precedes a bit name. If there is no preceding n, then when BIT is set to one the meaning of the bit is true, and when BIT is cleared to zero the meaning of the bit is false. If there is a preceding n, then when nBIT is cleared to zero the meaning of the bit is true and when nBIT is set to one the meaning of the bit is false.

2.2.4 Ordering for data transfers

All addressing specified within this document shall be big endian ordering unless specifically identified as otherwise.

Bytes (8-bit values) shall be represented with the most significant bit (MSb) positioned to the left. Bytes are transmitted on the bus MSb first and LSb last.

7 (MSb)	6	5	4	3	2	1	0 (LSb)

Doublets (16-bit values) shall have the most significant byte (MSB) as the lower addressed. Doublets are transmitted on the bus MSB first and LSB last.

MSB	LSB
Addr n	Addr n+1

Quadlets (32-bit values) shall have the most significant doublet (MSD) as the lower addressed. Quadlets are transmitted on the bus MSD first and LSD last.

Most Significant Doublet (MSD)		Least Significant Doublet (LSD)	
MSB	LSB	MSB	LSB
Addr n	Addr n+1	Addr n+2	Addr n+3

Octlets (64-bit values) shall have the most significant quadlet (MSQ) as the lower addressed. Octlets are transmitted on the bus MSQ first and LSQ last.

Most Significant Quadlet (MSQ)				Least Significant Quadlet (LSQ)			
MSD		LSD		MSD		LSD	
MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB
Addr n	Addr n+1	Addr n+2	Addr n+3	Addr n+4	Addr n+5	Addr n+6	Addr n+7

The above ordering of bytes are as defined in IEEE1212, SBP-2 and within this document. The ATA-4 standard also defines byte ordering of data when data is transferred across the 16 bit ATA data bus. The ATA-4 standard defines that the low order 8 bits of its 16 bit bus represent the data of the lower byte address and the upper 8 bits are from the next higher address, as shown in the following illustration.

	DD 15	DD 14	DD 13	DD 12	DD 11	DD 10	DD 9	DD 8	DD 7	DD 6	DD 5	DD 4	DD 3	DD 2	DD 1	DD 0
First transfer	Byte (n+1)								Byte (n)							
Second transfer	Byte (n+3)								Byte (n+2)							
.....																
Last transfer	Byte (n+transfersize-1)								Byte (n+transfersize-2)							

To insure compatibility, all tailgate implementations must comply with the byte ordering requirements on both the 1394 interface and the ATA interface.

2.3

CSR Specifications

This document defines the format and function of CSRs. Some of these CSRs are read-only, some are read-write, and some have special side-effects on writes. CSR specifications include the definition, the initial value of the CSR, the value returned when the CSR is read, and the effect(s) when the CSR is written. The definition includes the name and size of each bit field. An example CSR is illustrated in Figure 1.

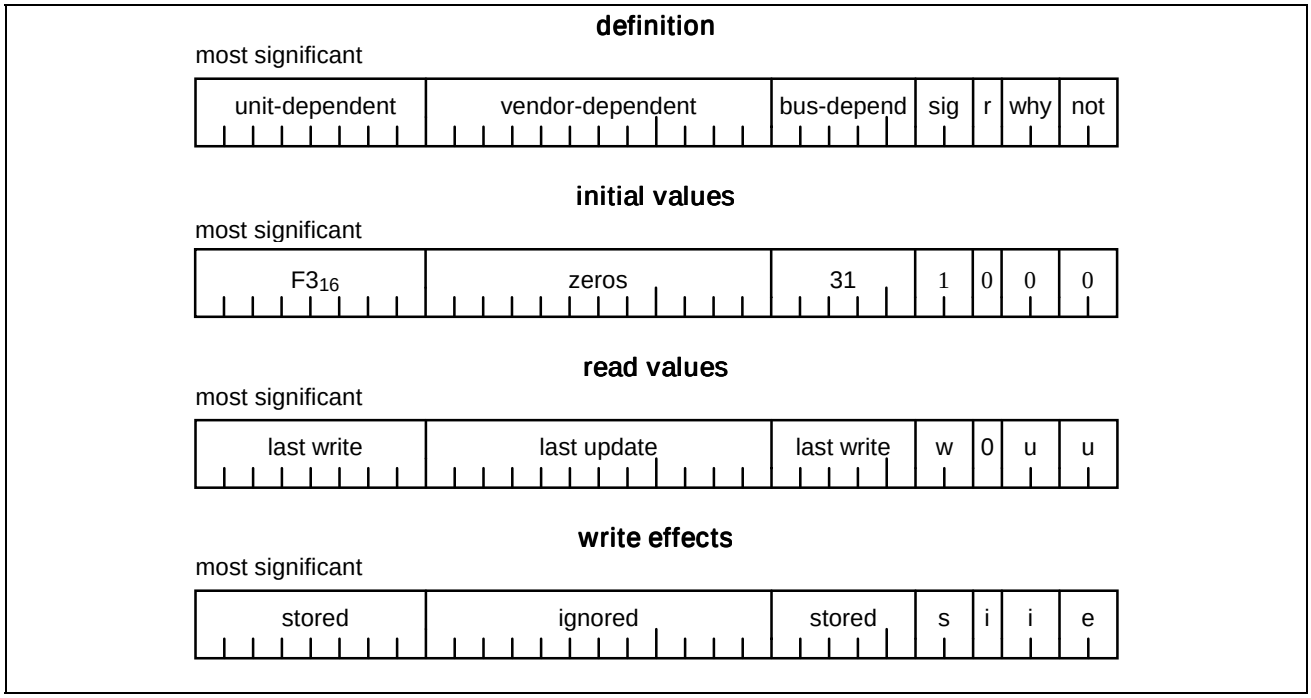

Figure 1 – CSR specification example

The CSR definition lists the names of fields. These names are descriptive, but the fields are defined in the text; their function should not be inferred from their names. However, the following register definition fields have standard meanings:

Table 1 – Register definition fields

Name	Abbreviation	Definition
unit dependent	unit_depend	The meaning of this field shall be defined by the node's unit architecture.
vendor dependent	vendor_depend	The meaning of this field shall be defined by the node's vendor. Within a unit architecture, the unit dependent fields may be defined to be vendor dependent.

The CSRs defined in this document shall be initialized when powered on or reset.

The following read field values have standard meanings:

Table 2 – Read field values

Name	Abbreviation	Definition
last write	w	The value of the data field shall be the value that was previously written to the register.
last update	u	The value of the data field shall be the last value that was updated by device hardware.
undefined	x	The value of the data field shall be undefined and no assumption shall be made of its contents.

The following write-effect fields have standard meanings:

Table 3 – Write-effect field values

Name	Abbreviation	Definition
stored	s	The value of the written data field shall be immediately visible to reads of the same register.
Ignored	I	The value of the written data field shall be ignored; it shall have no effect on the device's state.
effect	e	The value of the written data field shall have an effect on the device's state, but is not immediately visible to reads of the same register.

The register description specifies its bus transaction read/write characteristics, as well as whether it is a required register. A read-write register (RW) is expected to be read and written via bus transactions; a read-only register (RO) is expected to only be read; a write-only register (WO) is expected to only be written. Although reads of WO registers and writes of RO registers are not expected, the register definition still defines their results.

2.4 State machines

All state machines in this standard are defined in the style illustrated by Figure 2.

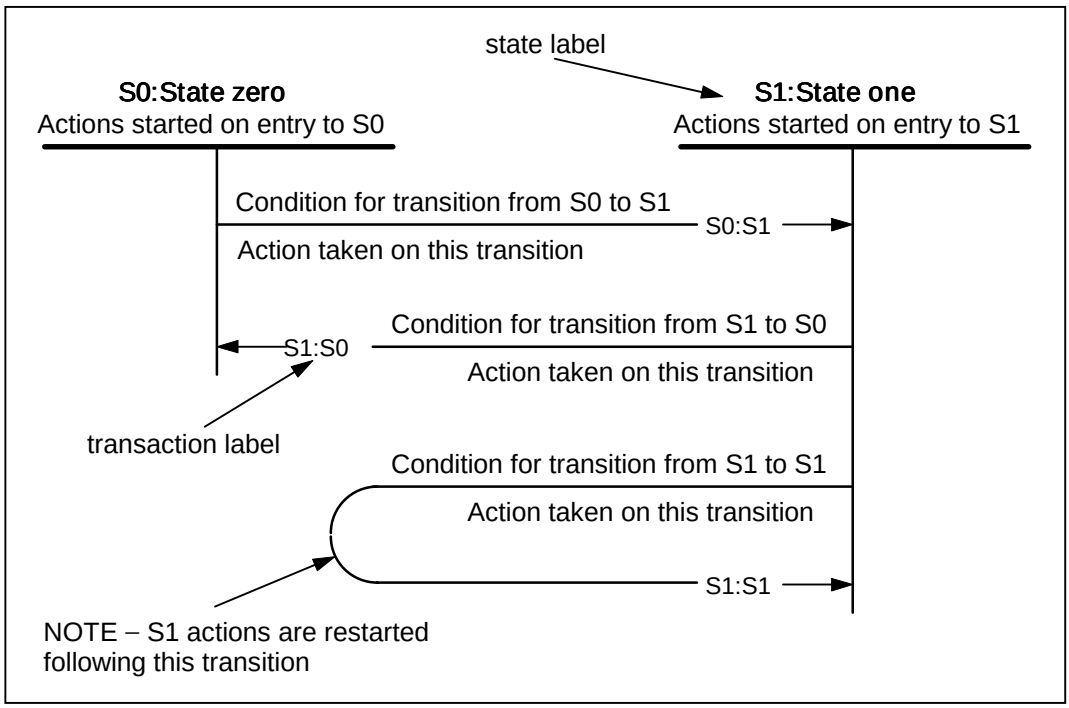

Figure 2 – State machine example

The state machines in this standard make three assumptions:

Time elapses only within the discrete state;

State transitions are conceptually instantaneous; the only actions taken during the transition are the setting of flags or variables and the sending of signals; and

Each time a state is entered (or reentered from itself), the actions of that state are performed.

Multiple transitions may connect two states. In this case, the transitions are uniquely labeled by appending a character to the transition label, *e.g.*, S0:S1a and S0:S1b.

2.5 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

ISO/IEC 13213:1994, Control and Status Register (CSR) Architecture for Microcomputer Buses

IEEE Std 1394-1995, Standard for a High Performance Serial Bus (see patent statement)

ANSI X3.301-1997, SCSI-3 Primary Commands (SPC)

ANSI T13 1153D, AT Attachment - 4 with Packet Interface Extension (ATA/ATAPI-4)

ANSI T10 1155D, Serial Bus Protocol 2 (SBP-2)

3

Tailgate model (Informative)

All tailgate devices implement a single management agent engine. The management agent processes commands associated with the management of, access to and control of the command agent engines. The initiator utilizes the management agent to gain access to a command agent. A command agent is utilized to access a device attached to the tailgate. The SBP-2 standard defines the operations of the management agent.

The SBP-2 standard also defines the several management functions. Some management functions are classified as task management functions (e.g., Abort Task Set) whose functionality within a Tailgate environment are further defined within this standard (see 0). This standard defines an extension to the management functions of SBP-2 that controls access to devices via password protection (see 0 and 0).

For simplicity of discussion within this clause, the operation of the management agent is not included. The remainder of this clause discusses the command agent.

Tailgate devices may be implemented as supporting either single or dual logical units. Tailgates supporting single logical unit configurations can only control a single ATA or ATAPI device. Tailgates that support dual logical unit configurations can control one or two ATA or ATAPI devices. Each logical unit equates to a single ATA or ATAPI device.

The tailgate uses a fetch engine and an execution engine to process device level commands. The fetch engine complies with the SBP-2 protocol for processing command chains. The execution engine is described in 0. This clause and the associated block diagrams are informative and should not be considered as the only possible, or even the suggested, implementation. This clause defines a model used in the remainder of the document to describe the requirements of tailgate implementations. Tailgates that do not fit the model defined in this clause but do meet the functional requirements defined in the subsequent normative clauses of this document are possible.

3.1 Single logical unit tailgate overview

Figure 3 illustrates an example block diagram of a single logical unit tailgate. The 1394 interface block handles all interaction with the 1394 bus. The interface block reads the configuration ROM to respond to requests for information for configuration ROM data by other nodes. The interface block also performs read and write functions to the CSRs of the tailgate.

The fetch engine maintains command chain context and fetches ORBs from the initiator's ORB chain.

The execution engine accepts the ORB from the fetch engine and performs the necessary operations on the device. The execution engine uses the throttle indication to control when the fetch engine can send another ORB.

The execution engine uses the data in and data/status out controls to transfer data to or /from the device and return completion status for an ORB.

The bus interface unit acts as a translator between the ATA/ATAPI data transfer protocol and the requirements of data transfer on the 1394 bus.

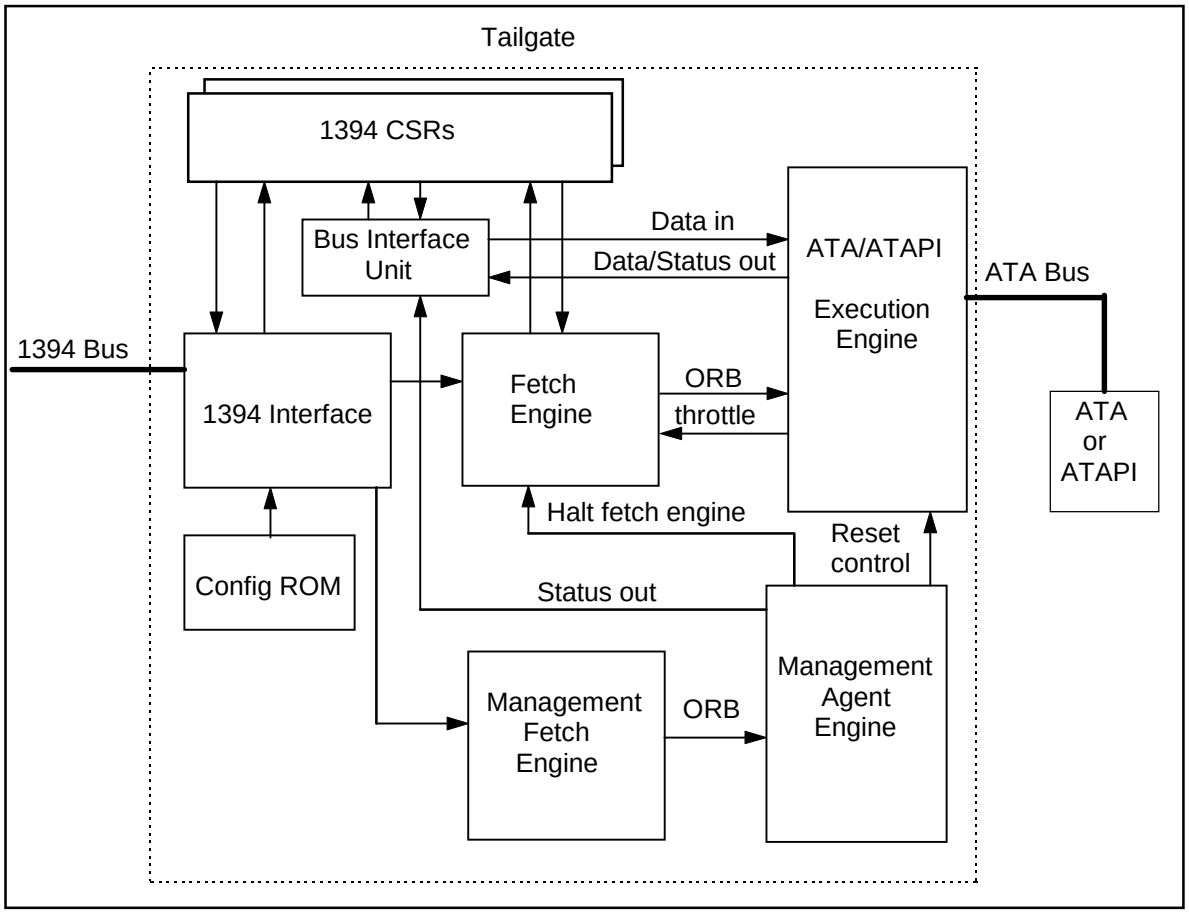

Figure 3 – Single logical unit tailgate block diagram

3.2 Dual logical unit tailgate overview

Figure 4 illustrates an example block diagram of a dual logical unit tailgate. The 1394 interface block handles all interaction with the 1394 bus. The interface block reads the configuration ROM to respond to requests for information for configuration ROM data by other nodes. The interface block also performs read and write functions to the CSRs of the tailgate.

The Fetch Engine for logical unit 0 and logical unit 1 contain the necessary contextual information to maintain separate command chain context and fetches ORBs from the initiator's ORB chain for each device. Only a single fetch engine can be active on the 1394 bus at once.

The execution engine accepts the ORB from the fetch engines and performs the necessary operations on the devices. The execution engine uses the throttle indications to control when the fetch engines can send another ORB.

The execution engine uses the data in and data/status out controls to transfer data to or /from the device and return completion status for an ORB.

The bus interface unit acts as a translator between the ATA/ATAPI data transfer protocol and the requirements of data transfer on the 1394 bus.

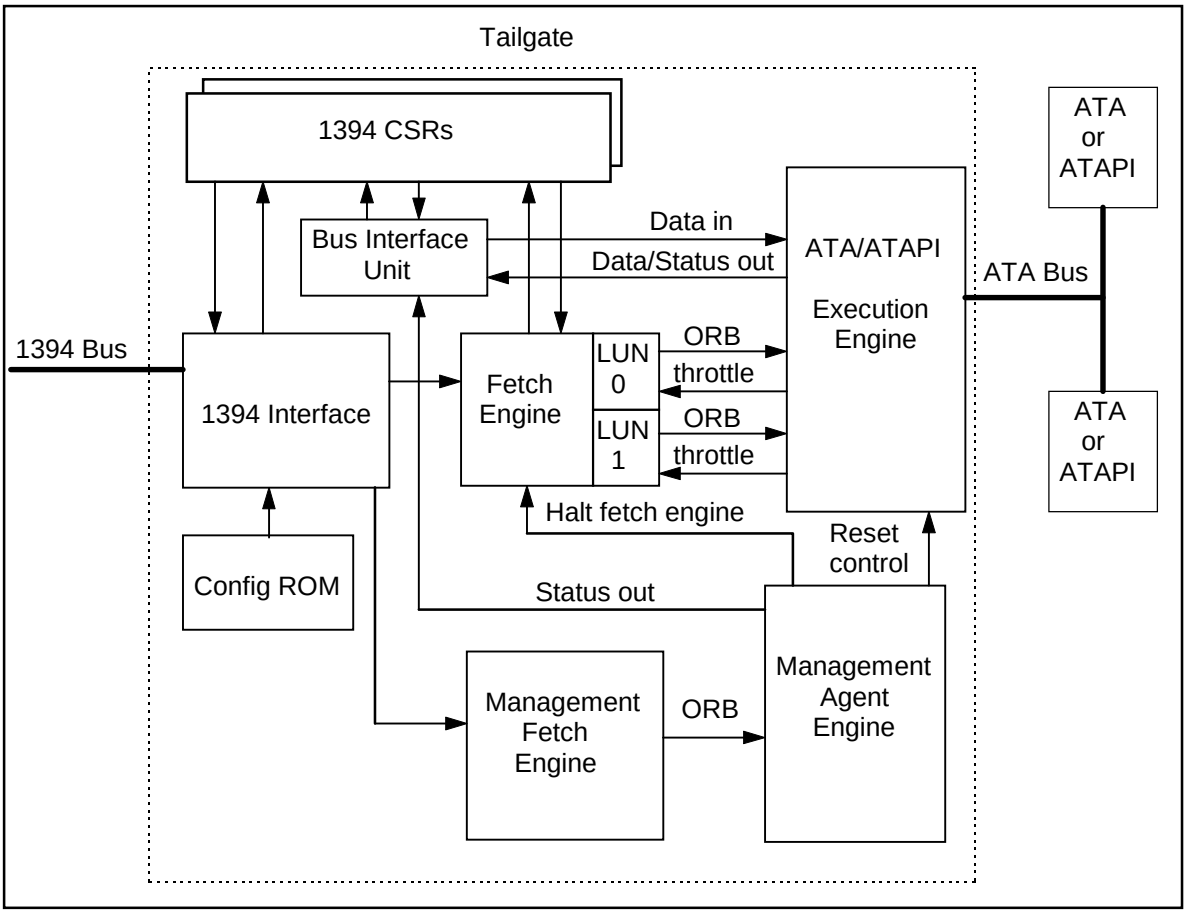

Figure 4 – Dual logical unit tailgate block diagram

3.3 Fetch engine/execution engine interface

The fetch engine interface to the execution engine provides for the passage of ORBs to the execution engine and a throttle to control the flow of ORBs. This throttle can be viewed as a ready signal to the fetch engine indicating when the execution engine is capable of accepting another ORB. The execution engine enables the throttle indication to permit the fetch engine to send an ORB and disables it to stop the fetch engine from transferring ORBs.

For single logical unit tailgates, the throttle is governed by the ability to accept commands of a single ATA or ATAPI device.

For dual logical unit tailgates, the throttle regulates the flow to two ATA or ATAPI devices.

For dual logical unit tailgate supporting overlapped operation, the throttle adds another stage of complexity. The execution engine must have the ability to individually control the throttles for each fetch engine. This is required since when a device releases the ATA bus, commands may be issued to the other device.

3.3.1 Dual logical unit tailgate fetch agents

System software should not assume any overlap limitations between the two logical units. The tailgate performs all necessary command flow control described in prior clauses. Thus the system drivers can treat a dual-device tailgate the same in all cases and need not be concerned about whether the tailgate supports overlap or not. Therefore, the initiator drivers never need be concerned about the type of tailgate or devices

connected. A single software driver model works for either dual-device tailgate mode (overlap or non-overlap capable).

3.4 Bus interface unit

The Bus Interface Unit, or BIU, provides the mechanism to move data between the 1394 and ATA buses. Since the 1394 bus is memory mapped the BIU provides 1394 bus memory addresses for 1394 data transactions. The BIU also handles processing of the page table, if one was provided. The BIU may utilize a buffer to optimize the flow of data between the 1394 and ATA buses. This buffer may also permit improved 1394 bus utilization.

For reads from the device, the BIU accepts the data that was read from the ATA bus. The BIU packages the data into one or more 1394 data packets, adds the appropriate information such as bus address and other control information, and transmits the packet to the initiator.

For writes to the device, the BIU receives read data requests from the execution engine. The BIU then requests the data from the initiator via a 1394 read data packet. Once the data is received from the initiator, the BIU removes the 1394 specific control information and passes the data to the execution engine that sends the data to the appropriate device.

For completion status, the BIU receives a data packet containing the current Command Block register contents. The BIU adds the 1394 control information and writes the 16 byte data packet to the status posting address. The format of this status packet is found in 0.

3.5 Initiator processing

This clause provides a summary of the initiator's responsibilities and functionality in operation of tailgates. This clause addresses tailgate and device configuration and normal command processing. For more details on these topics, including register and Command Block definitions, see clauses 0 and 0. These provide details on the CSR and ORB definitions, respectively.

The ATA bus provides several timing modes and speeds. The initiator identifies timing capabilities for the tailgate and the tailgate's attached device(s). The initiator uses that capability information to configure the tailgate and device(s) so that data transfers performed occur with the correct speed and protocol between the tailgate and device(s).

The initiator places the desired command within an ORB structure. Depending upon the function to be performed, the initiator sets bits and fields within the ORB. The initiator then signals availability of the ORB to the tailgate. Once the ORB has been constructed and placed in the ORB chain for execution, the initiator waits for the operation to complete and return status.

The tailgate fetches the ORB from the initiator. The tailgate utilizes the information in the ORB to perform the requested action. The tailgate processes the ORB, issues the command to the device, transfers data, reads Command Block registers at command completion, and returns their contents via the status block.

3.5.1 Time-outs

The initiator provides for recovery of command requests to the tailgate that do not complete. The tailgate does not provide any detection of non-responsive (hung) devices. Therefore, the initiator should provide suitable time-out mechanisms for requests. A time-out period is not defined by the tailgate standard. Any time-out period would be device or command specific, therefore no fixed standard time-out period applies. The initiator should provide reasonable time-out periods for commands based upon device performance. If a time-out period should be reached by the initiator, it should use the Abort Task, Abort Task Set, or Target Reset management ORB functions to clear the non-responsive (hung) device condition in the tailgate. These functions are described in 0. Processing following a time-out and subsequent recovery mechanism are dependent upon the operation being performed and are not covered by this standard.

Protocol definitions

4.1 Access and security

The tailgate shall provide security against intrusion via the 1394 interface by unauthorized users/initiators. This security mechanism is not intended to guard against theft or prevent access to the device via a system that has valid access to the device. Every device attached via a tailgate shall have, at a minimum, access protection against unauthorized users or initiator as defined in this standard.

To access a device, the initiator shall first complete a successful login. Access to a device is restricted to the initiator which is logged in. Only the logged in initiator has access to the set of command agent registers used to perform device level commands (e.g., reads and writes).

Security is implemented by restricting access to the device to only authorized users/initiators. Authorization is granted by use of a password (or lack of one) in the Login ORB used by the initiator to request access. This standard provides support for the setting of an eight byte password.

4.1.1 Access security model

An initiator, at any particular time, is either logged in (and has access to the device) or logged out. When no initiator is logged in the device is in the off-line state. Otherwise, the device is in the online state. The device shall enter the off-line state after a power on.

The logical unit's password value shall be preserved through a power off/on cycle and shall be changeable via the Set Password function.

A successful login will transition the logical unit from off-line state to the online state. The tailgate shall only support a single login per logical unit. Any attempt to login to a logical unit which already has an existing login shall be rejected by the tailgate. Dual logical unit tailgates shall support the ability for logins from different initiators. A successful logout will transition the logical unit from the online state to the off-line state. After a bus reset the initiator has one second to reconnect to the tailgate before it loses its reconnection privilege.

The logical unit's password is set via the Set Password function. This command can only be issued by an initiator which currently has access to the logical unit (i.e., is online).

Login, Logout and Set Password functions are processed by the management agent (defined in SBP-2). Any initiator may issue commands to the management agent. It is the responsibility of the management agent to implement the security functions described here.

Access to the logical unit is limited to the logged in initiator.

Figure 5 contains a state diagram for this security model followed by more detailed descriptions of the states and transitions.

Tailgates may be shipped with the password(s) programmed to all zeros. This is a default password that initiators should attempt to use to login to newly detected logical units. If a logical unit is shipped with a password programmed other than the default password the provider shall include the password to enable the user to successfully install and use the tailgate.

SBP-2 defines a Reconnect ORB whose purpose is to allow the initiator which was logged in to a logical unit to re-secure (Reconnect) its ownership of the device after a 1394 bus device reset. If a valid Reconnect is received within 1 second after a bus device reset from an initiator whose is requesting to reconnect to the logical unit to which it was logged into before the bus device reset, the tailgate shall re-establish access permissions for that initiator.

A separate access state and password value shall be maintained for each logical unit supported by the tailgate.

A login retry counter is implemented by the tailgate that counts the number of successive, invalid login attempts. If the value of the retry counter exceeds a maximum value (*max*), then all login attempts whether valid or not shall be treated as an invalid login request. The purpose of the retry count is to increase the level of difficulty for an intruder to circumvent the security provided by the password field. The retry count shall be maintained on a per logical unit basis. The retry count shall not be maintained on a per initiator or any other basis. The retry count shall have a maximum value of three and implementation of the retry counter shall be mandatory.

NOTE – Implementation of the retry counter must prevent possible roll over of the retry counter from re-enabling login attempts after *max* unsuccessful login attempts.

The Query Login command has no affect on the access security state and is processed regardless of whether the requesting node is logged in or not.

Other management ORBs (task management and undefined/unsupported command values) have no effect on the access security state. However, the tailgate shall not respond to task management functions from an initiator other than the logged in initiator.

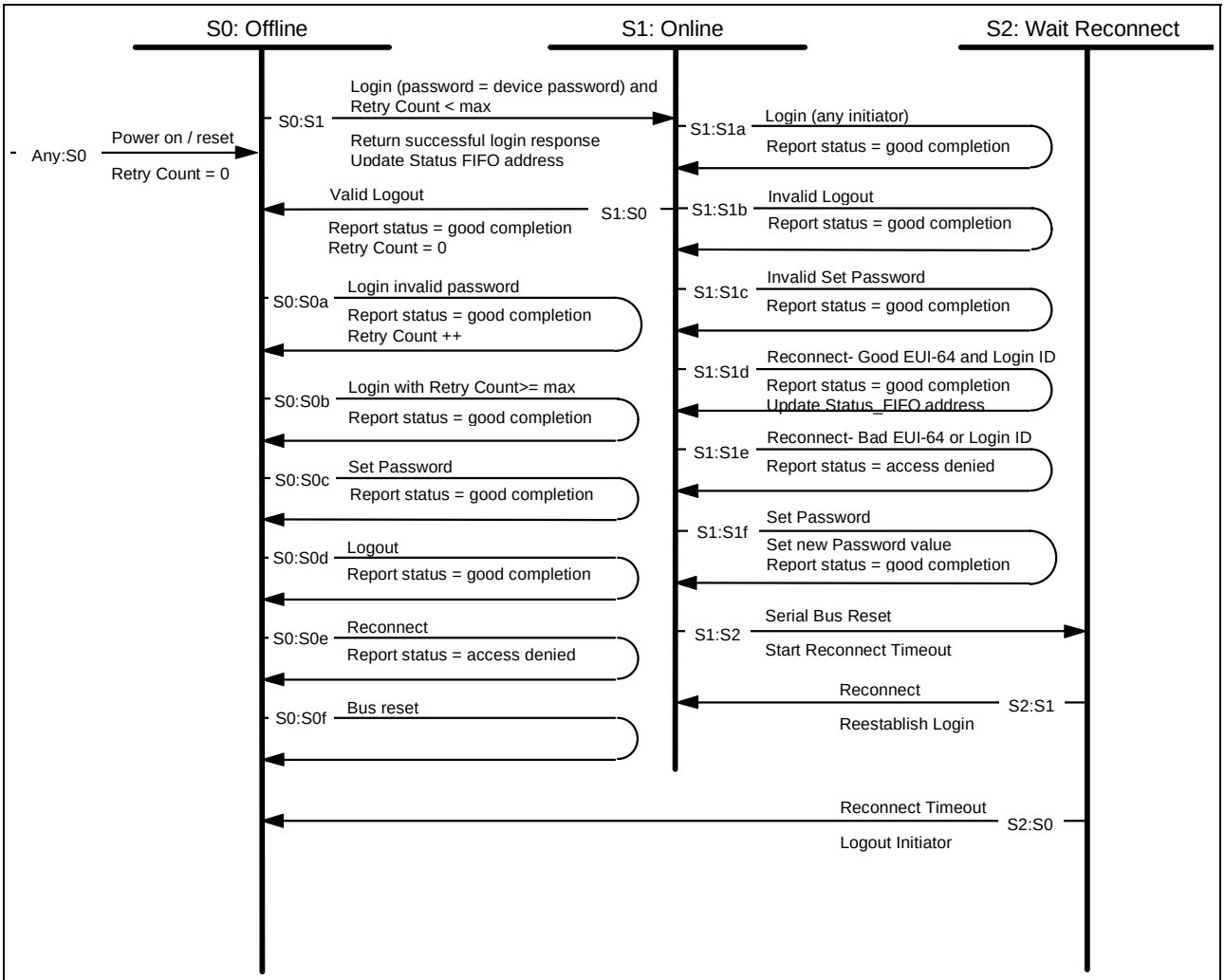

Figure 5 – Access security state diagram

Transition Any:S0. At power-on the logical unit shall transition to the S0 state and set the retry count to zero.

State S0. In this state the drive is in the Off-line state. There is no logged in initiator for the logical unit.

Transition S0:S1. When a Management ORB containing a Login function with a password that matches the logical unit's programmed password is fetched, the tailgate shall log that initiator in and grant it sole access to the device.

Transition S0:S0a. A login function shall not be accepted if the supplied password does not match the logical unit's programmed password. Rejected login requests shall increment the retry count by one. Status indicating normal (good) completion shall be returned even no action has been taken.

Transition S0:S0b. A Login function shall not be accepted if the retry count has exceeded its maximum allowed value. Status indicating normal (good) completion shall be returned even though no action has been taken.

Transition S0:S0c. A Set Password function shall be ignored. Status indicating normal (good) completion shall be returned even though no action has been taken.

Transition S0:S0d. A Logout function shall be ignored. Status indicating normal (good) completion shall be returned even though no action has been taken.

Transition S0:S0e. A Reconnect function shall be ignored. Status indicating access denied shall be returned to indicate that the function failed.

Transition S0:S0f. A Bus.Reset shall cause no state change.

State S1. In this state, the logical unit is in the Online state. The tailgate shall fetch Command Block ORBs and Management Agent ORBs at the addresses specified by the initiator.

Transition S1:S0. When a Management ORB containing a valid Logout function is fetched the hard disk drive shall log the initiator out and transition to the S0 state. A valid logout can only occur if: it is issued by the logged in initiator; the fetch agent for the logical unit is in the suspended state; and, the task set for the logical unit is empty. Upon completion of a successful logout function the retry count shall be set to zero.

Transition S1:S1a. A Login function shall be ignored. Status indicating normal (good) completion shall be returned even though no action has been taken.

Transition S1:S1b. An invalid Logout function shall be ignored. Status indicating normal (good) completion shall be returned even though no action has been taken.

Transition S1:S1c. An invalid Set Password function shall be ignored. Status indicating normal (good) completion shall be returned even though no action has been taken.

Transition S1:S1d. A Reconnect function with a good EUI-64 and Login ID shall force the tailgate to update the command block status_FIFO address. Status indicating normal (good) completion shall be returned.

Transition S1:S1e. A Reconnect function with an incorrect EUI-64 or Login ID shall be ignored. Status indicating access denied shall be returned.

Transition S1:S1f. When a Management ORB containing a valid Set Password function is fetched the new password value shall be accepted and saved. Status indicating normal (good) completion shall be returned.

Transition S1:S2. When the tailgate receives a Serial Bus Reset it shall start the Reconnect Time-out timer and wait until either the timer reaches one second or a Management ORB containing the Reconnect function is fetched from the initiator logged in prior to the bus reset

State S2. In this state the logical is in the Wait for Reconnect state. No commands can be fetched by either the Command Agent or the Management Agent.

Transition S2:S1. When a Management ORB containing a Reconnect function with a valid Login_ID is fetched the tailgate shall log that initiator in and grant it sole access to the associated logical unit. The logical unit shall transition to the Online state, S1, and return status indicating normal (good) command completion.

Transition S2:S0. When the Reconnect Time-out timer reaches one second, the tailgate shall log out the initiator logged in prior to the bus reset and transition to the Off-line state, S0.

4.2 Task management

SBP-2 specifies a set of functions required for basic task management of 1394 devices. This clause details the specific tailgate response or actions after receiving one of these task management functions. Definition of the terminology and states used in subsequent clauses may be found in the SBP-2 document. Table 4 lists the behavior of the tailgate for each initiator action or abnormal command termination.

Table 4 – Reset definitions for tailgate

Reset Type	ATA Reset	ATA SRST	Device Reset	Fetch Engine	Cmd Exec.	Mgmt.	Login	Timing Modes
Power on	B	N	N	BR	B	B	B	B
Bus	B	N	N	BR	B	B	B[1]	B
Command	B	N	N	BR	B	B	B	B
ATA/ATAPI Error	N	N	N	TD	N	N	N	N
Abort Task	N	N	N	N	N	N	N	N
Abort Task Set	N	B	T	BD	B	N	N	N
Target Reset	B	N	N	BD	B	N	N	B
Logout[2]	N	N	N	TR	T	N	T	N

T = unique to a logical unit, B = affects both logical units, N = no effect on either logical unit, D = fetch engine goes to dead state, R = reset

NOTE 1 – For each logical unit reconnect is accepted within one second following Bus Reset. If reconnect does not occur within one second of Bus Reset the Fetch Engine for that logical unit transitions to the reset state.

NOTE 2 – May be rejected if commands are outstanding for that logical unit.

4.2.1 ATA/ATAPI resets

There are three types of reset defined by the ATA/ATAP-4 standard: ATA Hard Reset, ATA Soft Reset, and Device Reset. There are several types of resets defined by SBP-2 standard. The SBP-2 reset functions

cannot be mapped to the ATA resets. The following paragraphs define how SBP-2 defined resets are to be implemented on a tailgate device.

The tailgate execution engine shall accept all reset functions regardless of the state of the execution engine, or the state of the fetch engine. One of the primary reasons for issuing reset functions is to clear non-responsive (hung) device conditions in either the device or the tailgate. If resets were issued via the command agent (i.e., the command is appended to the linked list of outstanding commands), then reset would either never be executed (due to the hang condition) or, when it is processed, the desired effect may no longer be necessary. For this reason SBP-2 reset functions are issued via the management agent mechanism rather than the command agent mechanism which is used to process normal register delivered or packet delivered command ORBs.

4.2.1.1 Soft resets

The Abort Task Set function maps into the two soft reset functions performed on the ATA interface. Specific functions performed by the Abort Task Set function are defined in 0. The Abort Task Set utilizes the Device Reset (08_{16}) command, a register delivered command, and the ATA soft reset, a bit in the ATA Device Control Block register .

4.2.1.2 Hard reset

The ATA Hard Reset may be issued via the Target Reset ORB function. The tailgate implements the hard reset function by activating the RESET- line on the ATA bus. This causes a hardware reset function to all devices on the ATA bus. After performing a Target Reset function the initiator shall be responsible for reconfiguration of the devices. Specific functions performed by the Target Reset function are defined in 0.

4.2.2 Abort Task

SBP-2 defines the Abort Task function as being implemented both as a management ORB and a command ORB. If an initiator is attempting to abort a command (task), that task was previously linked into a task set from which the fetch agent is obtaining commands. When the need to abort the task arises, the initiator has no means of knowing whether or not the task to be aborted has been fetched, if it had been fetched whether or not it has begun execution, and if it has begun execution how far into the execution process the command has proceeded. To abort the task the initiator first changes rq_fmt from a value of zero, the normal value for commands to be processed, to a value of three, indicating a Dummy ORB. After the initiator makes this change, it may issue an Abort Task with the appropriate identification parameters to indicate which ORB is to be aborted. Because of the delay in issuing the Abort Task within the initiator and the delay of fetching and processing of the Abort Task within the device, there is no guarantee that the Abort Task will ever “catch up” to the command that is to be aborted.

Tailgates shall recognize Dummy ORBs placed on the command agent’s command chain. When the tailgate fetches a Dummy ORB it shall return REQUEST ABORTED status and perform no other processing of the Dummy ORB.

It is expected that tailgates will not abort tasks in response to Abort Task functions. The SBP-2 standard permits implementations to ignore Abort Task functions and return no error status. Since devices provide no mechanism to abort commands, other than the reset functions, it is expected that tailgates shall not perform any actual processing of Abort Task functions other than returning no error status.

4.2.3 Abort Task Set

Tailgates shall implement the Abort Task Set function. The Abort Task Set function performs an Device Reset and ATA Soft Reset function, in addition to the standard SBP-2 function of clearing any active commands. The Abort Task Set function can therefore be used when the initiator wishes to cause an ATA Soft Reset or Device Reset to the selected logical unit. Figure 6 below illustrates the processing the tailgate performs when an Abort Task Set function is received.

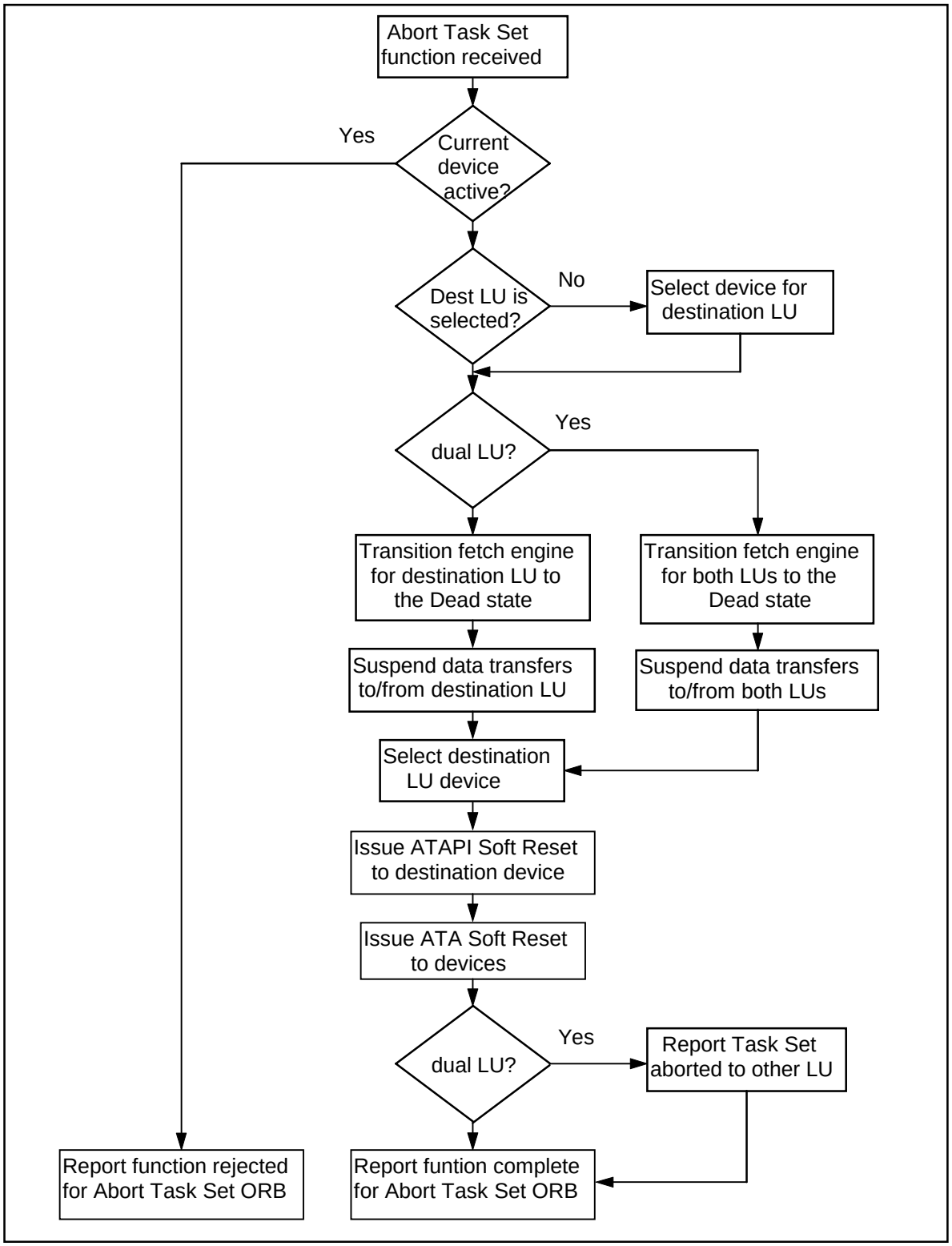

Figure 6 – Abort Task Set flowchart

In addition to Figure 6 above, the text that follows provides additional detail on the processing of Abort Task Set functions.

1. While a device controlled by the other logical unit is selected and active, receipt of an Abort Task Set shall result in function rejected status.

   The initiator shall be responsible for recovery mechanisms such as retrying the Abort Task Set function. The initiator may choose to use the Target Reset function as the final step of its recovery mechanism. The Target Reset function will, however, abort any background tasks devices may be performing such as an audio play operation on an ATAPI CD-ROM. Therefore Target Reset should be the solution of last resort for the initiator.

2. The tailgate shall cause the fetch engine for the logical unit to transition to the Dead state. If the tailgate supports dual logical units then the fetch engine for both logical units shall transition to the Dead state.

   A fetch engine may have an outstanding request pending. The fetch engine shall not wait for the response to be received prior to entering the Dead state. It is the initiators responsibility to either prevent any outstanding responses from being transmitted or delaying the re-enabling of the fetch engines until all such responses have been sent. The tailgate shall ignore responses to any fetch requests while it is in the dead state.

3. The tailgate shall not issue any further data transfer requests on the 1394 bus for logical unit 0. The tailgate shall not wait for responses for pending data transfers on the 1394 bus to complete. If the tailgate supports dual logical units then the same processing shall apply for both logical units.

   An execution engine or BUI may have outstanding data requests pending. These engines shall not wait for the response to be received prior to executing the remaining functions of the Target Reset. It is the initiator's responsibility to either prevent any outstanding responses from transmitted or delaying the re-enabling of the fetch engines and issuing of any new commands until all such responses have been sent. The tailgate shall ignore responses to any data transfer requests until a new command is received.

4. The tailgate shall issue a Device Reset to the destination logical unit. For both single and dual logical unit configurations the Device Reset shall only be issued to the logical unit specified by the Abort Task Set management ORB.

5. The tailgate shall perform an ATA Soft Reset function. The Abort Task Set function shall not return status until the reset function has completed.

   If the tailgate implements the dual logical unit model as two separate ATA channels (cables) then the ATA Soft Reset shall be issued to both ATA buses.

   Tailgate configurations that use other mechanisms to permit overlapped processing, other than ATAPI Overlap, or packet command devices that are reset by ATA Soft Reset may have commands in progress terminated by the ATA Soft Reset. These configurations shall still be considered compliant tailgate configurations. The modification is that the overlapped device's command is terminated and thus never completes. The initiator detects this occurrence via the unsolicited status .

6. If the tailgate supports dual logical units then an unsolicited status shall be posted for that logical unit of Task Set Aborted to notify that initiator that the associated fetch engine was transitioned to the Dead state and that outstanding data transfers and commands may have been aborted.

   NOTE − The posting of the unsolicited status does not occur unless the Unsolicited Status Enable semaphore is set. Otherwise, the unsolicited status is not posted and processing continues with step 7.

7. The tailgate shall store a completion status for the Abort Task Set function indicating Function Complete.

4.2.4 Target Reset

Tailgates shall implement the Target Reset function. The Target Reset function performs an ATA Hard Reset in addition to the standard SBP-2 processing. This function can therefore be used by the initiator to cause an ATA Hard Reset on the tailgate. The text that follows provides additional detail on the processing of Target Reset functions.

1. The tailgate shall process the Target Reset regardless of the state of the ATA bus, any associated execution engine condition, any data transfer state or any fetch engine state.

2. The tailgate shall cause the fetch engine for the logical unit to transition to the Dead state. If the tailgate supports dual logical units then the fetch engine for both logical units shall transition to the Dead state.

   A fetch engine may have an outstanding request pending. The fetch engine shall not wait for the response to be received prior to entering the Dead state. It is the initiators responsibility to either prevent any outstanding responses from being transmitted or delaying the re-enabling of the fetch engines until all such responses have been sent. The tailgate shall ignore responses to any fetch requests while it is in the dead state.

3. The tailgate shall not issue any further data transfer requests on the 1394 bus for logical unit 0. The tailgate shall not wait for responses for pending data transfers on the 1394 bus to complete. If the tailgate supports dual logical units then the same processing shall apply for both logical units.

   An execution engine or BUI may have outstanding data requests pending. These engines shall not wait for the response to be received prior to executing the remaining functions of the Target Reset. It is the initiator's responsibility to either prevent any outstanding responses from transmitted or delaying the re-enabling of the fetch engines and issuing of any new commands until all such responses have been sent. The tailgate shall ignore responses to any data transfer requests until a new command is received.

4. The tailgate shall perform an ATA Hard Reset function. The Target Reset function shall not return status until the ATA Hard Reset function has completed.

   If the tailgate implements the dual logical unit model as two separate ATA channels (cables) then the ATA Hard Reset shall be issued to both ATA buses.

   The ATA standard requires that the RESET signal be asserted for no less than 25µs. The tailgate shall assert and then deassert the RESET signal in compliance with this requirement.

5. The tailgate shall store completion status for the Target Reset function indicating Function Complete. If the tailgate supports dual logical units the tailgate shall post an unsolicited status to the other logical unit.

4.3 Tailgate command level protocols

The tailgate performs all operations related to execution of ATA or ATAPI commands on the ATA bus. The tailgate determines what operations need to be performed based on information supplied in the ORB. The initiator places the necessary parameters into the ORB and interprets the values returned by the tailgate in the Status Block, but does not perform any portion of the device bus operation.

The tailgate uses two logical engines, the fetch engine and the execution engine. These blocks may not be separate in the actual implementation, but are considered as separate functional blocks for the purpose of description of functionality and for clarity.

Figure 7 illustrates the command processing performed by the tailgate to handle device commands. Upon receiving a command the execution engine determines whether the command contains a register delivered or packet delivered command. The use of *rd* in Figure 7 indicates the command is a register delivered command. In either command case, the tailgate shall also identify whether data transfer is required. It examines the *data_size* field in the ORB to determine this. If *data_size* is zero then no data transfer is required for this command request. Otherwise the tailgate must examine the *p* bit to determine whether direct (address specified in ORB) or indirect buffer addressing (specified via a page table) is being used. If the *p* bit is zero, direct buffer addressing is being used and the *data_descriptor* field provides the address of

the data buffer while the *data_size* field specifies the number of bytes within the buffer. If the *p* bit is one, indirect buffer address is being used and the *data_descriptor* field provides the address of a page table, while the *data_size* field specifies the number of elements in the page table. The *d* bit indicates the direction of the data transfer. Complete details on the definition and processing of these fields may be found in the SBP-2 standard. By interpreting these bits and fields, the tailgate can determine the length, address, and direction of the data transfer to be performed on the 1394 bus. Figure 7 provides a flowchart of the basic processing for device commands. This clause builds upon the foundation provided by that flowchart.

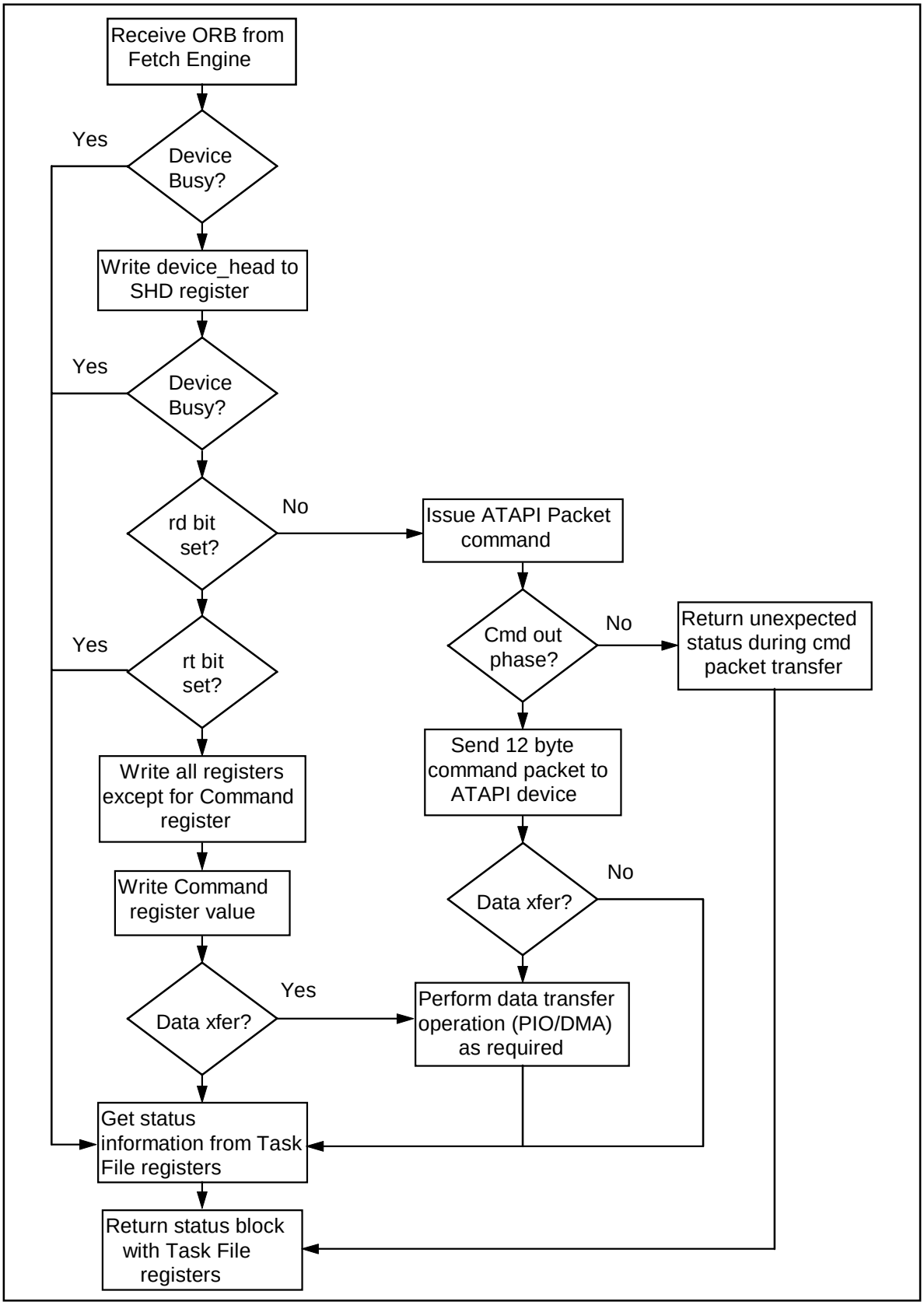

Figure 7 – Register and packet delivered command processing

Table 5 – Command Block registers summary (read)

Offset[2]	Register name for register delivered command devices	Register name for packet delivered command devices	Description
0	Data	Data	The tailgate provides no direct access to the data register. The tailgate automates all data transfer to or from the ATA/ATAPI device(s) performed via the Data register including delivery of ATAPI command packets.
1	Error	Error	The tailgate provides the value of the Error register via the status block returned at the completion of commands. This value provides information on error occurrences when bit 0 of the Status register is set.
2	Sector_Count	Interrupt rRason	The tailgate provides the value of the Sector_Count (called x for ATAPI devices) register via the status block returned at the completion of commands. The ATAPI Interrupt Reason register provides detail on the cause of an ATAPI device generated interrupt.
3	Sector_Number LBA (7:0) [1]	reserved	The tailgate provides the value of the Sector_Number (called x for ATAPI devices) register via the status block returned at the completion of commands.
4	Cylinder_High LBA (15:8) [1]	Byte Count (high)	The tailgate provides the value of the Cylinder_High (called x for ATAPI devices) register via the status block returned at the completion of commands. ATAPI devices redefine this register as the Byte Count (high) register, used in conjunction with the Byte Count (low) register, as a doublet byte counter. It indicates the number of bytes to be transferred at each data request.
5	Cylinder_Low LBA (23:16) [1]	Byte Count (low)	The tailgate provides the value of the Cylinder_Low (called x for ATAPI devices) register via the status block returned at the completion of commands. ATAPI devices redefine this register as the Byte Count (low) register, used in conjunction with the Byte Count (high) register, as a doublet byte counter. It indicates the number of bytes to be transferred at each data request.
6	Device_Head LBA (27:24)[1]	Drive_Head	The tailgate provides the value of the Device_Head register via the status block returned at the completion of commands.
7	Status	Status	The tailgate provides the value of the Status (called x for ATAPI devices) register via the status block returned at the completion of commands.

NOTE 1 – Register delivered command devices support Logical Block Addressing (LBA) and the specified registers are redefined to contain the indicated bits of a 28-bit LBA when bit 6 of the Device_Head register is set to 1.

NOTE 2 – The offset is in bytes from the base address of the Command Block registers.

Table 6 – Command Block registers summary (write)

Offset[2]	Register name for register delivered command devices	Register name for packet delivered command devices	Description
0	Data	Data	The tailgate provides no direct access to the data register. The tailgate automates all data transfer to or from the ATA/ATAPI device(s) performed via the Data register including delivery of ATAPI command packets.
1	Features	Features	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.
2	Sector_Count	reserved	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.
3	Sector_Number LBA (7:0) [1]	reserved	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.
4	Cylinder_High LBA (15:8) [1]	Byte Count (high)	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.
5	Cylinder_Low LBA (23:16) [1]	Byte Count (low)	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.
6	Device_Head LBA (27:24) [1]	Drive Select	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.
7	Command	Command	The initiator writes this register via the ATA device ORB. The tailgate provides no other direct write access to this register.

NOTE 1 – Register delivered command devices support Logical Block Addressing (LBA) and the specified registers are redefined to contain the indicated bits of a 28-bit LBA when bit 6 of the Device_Head register is set to 1.

NOTE 2 – The offset is in bytes from the base address of the Command Block registers.

NOTE – The Alternate Status and Device Control registers are not addressable by the initiator via the tailgate. The tailgate standard defines other mechanisms for generating the ATA SRST function provided in the Device Control register. Control of the interrupt enable is left to the tailgate which intercepts and processes device interrupts. There is no mechanism to poll the Alternate Status register provided via a tailgate.

The packet command is sent using the packet interface protocol defined in ATA/ATAPI-4. The packet delivered command protocol redefines the meaning of the Command Block registers used during the processing of the packet command. Packet delivered command devices also accept register delivered commands, including several new commands unique to packet delivered command devices. These register delivered commands are issued to the device using the register delivered command protocol.

The PACKET command, defined by ATA/ATAPI-4, provides the means of issuing a command to a packet delivered command device. Various standards or specifications define these packet commands for devices such as CD-ROM, CD-R, CD-E, floppy, PD, tape, and others.

The packet delivered command protocol requires that the Command Block registers be setup prior to sending the command packet including writing the command register with $A0_{16}$, the PACKET command. The tailgate shall set up the Command Block registers, issue the PACKET command, and send the command packet to the device.

4.3.1 Register delivered command protocol

For a register delivered command sent to the tailgate, the seven Command Block registers, excluding the Data register, will be used as the payload in the command block portion of the ORB. Table 5 and Table 6 list the definition of the Command Block registers. Full details on the meaning and usage of these registers may be found in the ATA standard.

The tailgate shall write the seven Command Block register bytes to the device. The Device/Head register shall be the first written and the Command register shall be the last written by the tailgate. Order of transfer of all other registers is not defined.

The initiator shall assume responsibility for detecting non-responsive (hung) devices via time-outs or other mechanisms. The tailgate provides no built-in time-out mechanisms to detect conditions that result in commands that never reach completion. The initiator shall detect non-responsive (hung) devices and perform recovery actions such as Abort Task Set or Target Reset, as defined in 0.

4.3.1.1 Initiating commands

The execution engine shall not accept commands for a device, from the fetch engine, while that device's BSY bit is set in the Status register. As each command completes, and the BSY bit is reset, the execution engine accepts another command from the fetch engine and processes it. Therefore, the initiator may build a list of commands while the fetch engine will read and pass the commands to the execution engine as the execution engine, and device, become available to process another command.

The tailgate's register delivered command request provides a mechanism to read the current contents of the Command Block registers without issuing a command to the connected device(s). When the tailgate receives such a command it shall return the current Command Block register contents without modifying the existing Command Block registers, other than the Device/Head register. The tailgate may need to write the Device/Head register to select the correct device prior to reading the Command Block registers. The tailgate shall not interrupt operation of any command currently being processed by the execution engine to respond to a request to read the Command Block registers.

4.3.1.2 Data transfers

The execution engine performs the data transfer processing on the device bus. For reads from the device, the data is sent to the Bus Interface Unit for transmission on the 1394 bus. The Bus Interface Unit handles the 1394 bus addressing required to complete the data transfer operation. For writes to the device, the data is received from the Bus Interface Unit and sent to the device using either PIO or DMA transfer mode, depending upon the transfer mode selected in the command. The processing required for data transfer on the 1394 bus and device bus are therefore logically separate, although no assumption is made about the actual implementation of these two mechanisms.

When performing PIO, the tailgate waits for the DRQ to be set in the device's Status register. Once it sees the DRQ it may transfer up to 512 bytes of data. The size of the each burst depends upon the buffering capacity of the tailgate and is not defined by this standard. After transferring 512 bytes the tailgate will check the data transfer size specified in the command. If it has reached zero, the data transfer part of the request is completed.

When performing DMA, the tailgate performs the transfer in accordance with the protocol appropriate for the DMA mode until the amount of data specified by the command has been transferred. Once this has occurred, the data transfer part of the request is completed.

A field in the command block for registered delivered command blocks specifies the number of 512 byte blocks to be transferred for this command. The execution engine shall transfer the specified number of blocks in the direction specified by the command. The tailgate may also use other mechanisms, such as monitoring the DRQ bit in the Status Block register, to determine command complete. However, the device shall not transfer more data than specified by the block count field. If the size of the data transfer specified by the block count field and the device command (*sector_count* for read and write operations) are not equal the tailgate may enter a non-responsive (hung) state. The tailgate may not detect and recover from this condition without

initiator intervention. The initiator shall have appropriate time-out mechanisms to detect and initiate the appropriate recovery mechanism from this non-responsive (hung) state.

If the size of the transfer specified by the transfer size field in the command and the block count do not define the same transfer length then the tailgate may enter a non-responsive (hung) state. The initiator must implement appropriate time-out mechanisms to detect this condition and use the appropriate task management function, see 0, to recover from the “hang” condition. The Abort Task Set or Target Reset functions will provide the necessary functionality to recover the tailgate from this condition. Use of the Target Reset function may affect operations in progress on packet delivered command devices connected to the same ATA channel as the non-responsive (hung) register delivered command device. It is therefore recommended that for dual logical unit configurations, in particular, the Abort Task Set function be used rather than Target Reset.

4.3.1.3 Command completion

If a command results in an error, detected by the tailgate by the setting of the ERR bit in the Status register, then the tailgate fetch engine is forced into the dead state. This is the normal processing specified for SBP-2 compliant devices which result in an error. If the command completed without error, ERR bit in Status register reset, then the execution engine indicates its readiness to accept another command from the fetch engine. Whether the command resulted in an error or not the Command Block registers are returned in the SBP-2 status block as defined in 0prior to accepting the next command from the fetch engine.

4.3.2 ATAPI command operation

The initiator builds a packet delivered command request in memory and submits it to the tailgate using the protocol defined in the SBP-2 standard. This clause specifies the special processing required for operation of the tailgate with packet delivered commands. Figure 7 provides a flowchart of the basic processing for register delivered and packet delivered commands. This clause builds upon the foundation provided by that flowchart.

4.3.2.1 Initiating commands

The execution engine will not accept commands from the fetch engine while the BSY bit is set in the Status register. As each command completes and the BSY bit in the Status register is reset, the execution engine will accept another command from the fetch engine and process it. Therefore the initiator may build a list of commands while the fetch engine will read and pass the commands to the execution engine as the execution engine becomes available to process another command.

The tailgate shall write the Features register prior to issuing the PACKET command by writing the appropriate command code ($A0_{16}$) to the Command register. This is to ensure that the correct transfer mode is selected prior to issuing the packet command to the device. The tailgate may detect that a device does not enter the command packet transfer phase within the appropriate period of time and return a transport error.

4.3.2.2 Data transfers

The execution engine performs the data transfer processing on the device bus. For reads from the device, the data is sent to the Bus Interface Unit for transmission on the 1394 bus. The Bus Interface Unit handles the 1394 bus addressing required to complete the data transfer operation. For writes to the device, the data is received from the Bus Interface Unit and sent to the device using either PIO or DMA transfer mode, depending upon the transfer mode specified by the command. The processing required for data transfer on the 1394 bus and device bus are therefore logically separate, although no assumption is made about the actual implementation of these two mechanisms.

When performing PIO, the tailgate waits for the DRQ to be set in the device’s Status register. Once the DRQ occurs, the tailgate reads the Byte Count registers to determine the size of the requested transfer. The size of the each burst depends upon the buffering capacity of the tailgate and is not defined by this standard. The tailgate shall use the transfer size specified in the command to determine the maximum transfer size. It shall not exceed this maximum transfer size.

When performing DMA mode transfers, the tailgate performs that transfer in accordance with the protocol appropriate for the DMA mode. The tailgate shall use the transfer size specified in the command to determine the maximum transfer size. It shall not exceed this maximum transfer size.

Since devices indicate command completion as a discrete event, the tailgate can detect the completion of device commands. Thus even though the size of the transfer specified by the command request block and the values in the command packet do not define the same transfer length, the tailgate shall not enter a non-responsive (hung) state. Rather, the tailgate should treat this as an error condition and transition the fetch engine to the Dead state. The status post should indicate the Transfer Length Mismatch transport error.

4.3.2.3 Command completion

If a command results in an error, detected by the tailgate by the setting of the CHK bit in the Status register, then the tailgate fetch engine is forced into the dead state. This is the normal processing specified for SBP-2 devices which result in an error. If the command completed without error, CHK bit in the Status register reset, then the execution engine indicates its readiness to accept another command from the fetch engine. Whether the command resulted in an error or not the Command Block registers are returned in the SBP-2 status block as defined in 0prior to accepting the next command from the fetch engine.

4.4 Dual logical units

Under normal operating conditions, the tailgate shall maintain separate command agent contexts for each logical unit of a dual logical unit tailgate. Fetching of ORBs from the ORB chain of one logical unit shall have no relationship to that of the other logical unit. Occurrences of errors and other unexpected conditions, such as resets, may affect the fetch processing of both logical units of a tailgate. The specific cases where actions or conditions that affect the fetch operation of both logical units exist the specific processing associated with that action or condition is detailed.

Although the tailgate shall maintain separate ORB chains for each logical unit the logical units will typically be on a shared ATA bus. Thus only a single device can be executing a command at once. Even though only one device can be executing a command at a time there should be a mechanism to regulate the flow to the two devices such that one device is not starved out by a very active device on the other logical unit. Some fairness algorithm should be provided to allow each device on the tailgate to receive commands regardless of activity on the other device.

5

Command and status registers (CSRs)

5.1 Core CSRs

A number of core CSRs are required by ISO/IEC 13213:1994, IEEE 1394-1995 and SBP-2 in any compliant implementation. The tailgate is a compliant implementation and shall implement those required CSRs. See annex C for additional information.

Optional registers in excess of this minimal set may be implemented within devices for any purpose; however, initiator systems and software shall not require these optional registers to provide support for primary functionality of compliant devices and shall provide support for the primary functionality of devices that do not support optional registers.

5.2 Tailgate specific CSRs

This standard defines two additional CSRs that shall be implemented by a tailgate compliant device. A separate set of these CSRs shall be implemented for each logical unit supported by the tailgate device. The following CSRs shall be implemented by a tailgate compliant device.

5.2.1 ATA timing and control registers

The two additional CSRs are ATA_TIMING_CAPABILITY and ATA_TIMING_SELECT. Each CSR is one quadlet in length. The ATA_TIMING_CAPABILITY CSR reports the ATA timing capabilities of the tailgate. The ATA_TIMING_SELECT CSR permits configuration of the ATA interface timing by the initiator. The location of these CSRs in the initial units space is relative to the base address of the command agent CSRs. The base address of the command agent CSRs is specified by the *command_agent* field from the login response. The relative offset value is specified in bytes.

Table 7 – ATA timing capability and selection CSRs

Relative offset	CSR name	Description
20_{16}	ATA_TIMING_CAPABILITY	A read-only CSR that reports the ATA mode timing capability for the logical unit.
24_{16}	ATA_TIMING_SELECT	A read/write CSR that controls the timing of the ATA interface for the logical unit.

If a tailgate implements dual logical units, a separate set of these CSRs (and the command agent CSRs) exist for each logical unit.

5.2.1.1 ATA_TIMING_CAPABILITY CSR

This CSR allows the initiator to determine the ATA interface transfer modes the tailgate supports. The initiator reads this CSR to determine which PIO and DMA modes the tailgate supports. The initiator issues an IDENTIFY DEVICE command to the device to determine the device capabilities. Based on these two sets of capabilities, the initiator may program the tailgate ATA_TIMING_SELECT CSR and issue the appropriate SET FEATURES command to the device. The initiator shall select timings supported by both the device and the tailgate to ensure proper operation. Figure 8 contains the specification for this CSR.

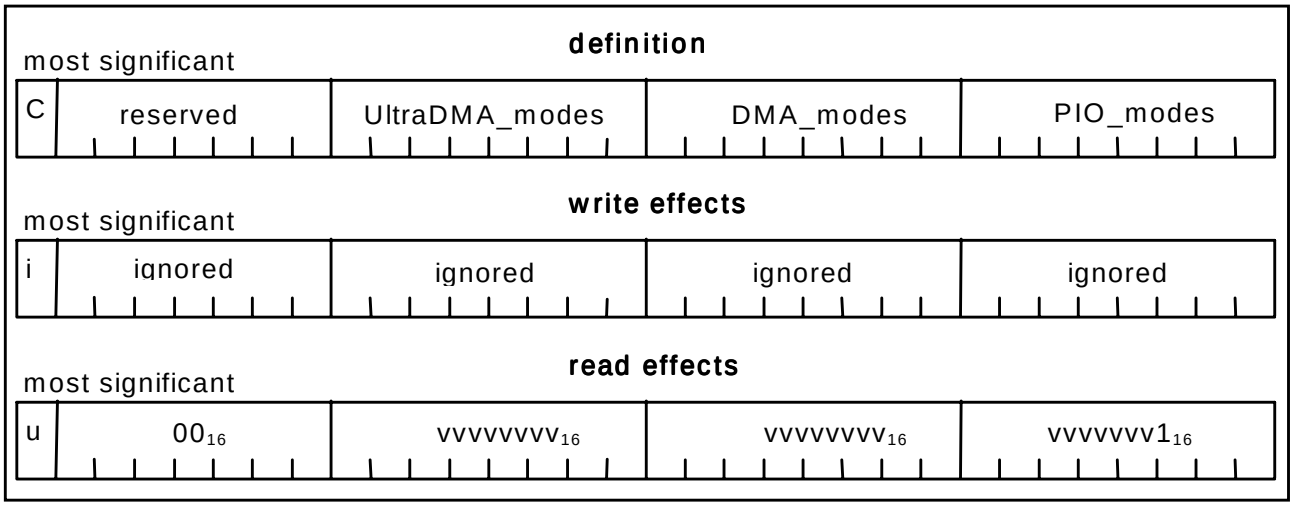

Figure 8 – ATA_TIMING_CAPABILITIES CSR specification

The *PIO_modes* field indicates the ATA bus PIO transfer modes the tailgate supports. More than a single bit may be set to indicate support of more than a single PIO transfer mode. A bit set to one indicates support for that respective transfer mode by the tailgate. A bit set to zero indicates the respective transfer mode is not supported by the tailgate. Support for PIO mode 0 is mandatory for all tailgates. The value of the remaining supported mode bits are dependent upon the transfer modes supported by the tailgate. The 'v' nomenclature indicates the implementation specific bits in the initial value definition for the ATA_TIMING_CAPABILITY CSR.

Bit	Definition
0	PIO Mode 0 capable
1	PIO Mode 1 capable
2	PIO Mode 2 capable
3	PIO Mode 3 capable
4	PIO Mode 4 capable
5-7	Reserved

The *DMA_modes* field indicates the ATA bus DMA transfer modes the tailgate supports. More than a single bit may be set to indicate support of more than a single DMA transfer mode. A bit set to one indicates support for that respective transfer mode by the tailgate. A bit set to zero indicates the respective transfer mode is not supported by the tailgate. The value of the remaining supported mode bits are dependent upon the transfers modes supported by the tailgate. The 'v' nomenclature indicates the implementation specific bits in the initial value definition for the ATA_TIMING_CAPABILITY CSR.

Bit	Definition
0	DMA Mode 0 capable
1	DMA Mode 1 capable
2	DMA Mode 2 capable
3-7	Reserved

The *Ultradma_modes* field indicates the ATA bus UltraDMA transfer modes the tailgate supports. More than a single bit may be set to indicate support of more than a single UltraDMA transfer mode. A bit set to one indicates support for that respective transfer mode by the tailgate. A bit set to zero indicates the respective transfer mode is not supported by the tailgate. The value of the remaining supported mode bits are dependent upon the transfers modes supported by the tailgate. The 'v' nomenclature indicates the implementation specific bits in the initial value definition for the ATA_TIMING_CAPABILITY CSR.

Bit	Definition
0	UltraDMA Mode 0 capable
1	UltraDMA Mode 1 capable
2	UltraDMA Mode 2 capable
3-7	Reserved

The *Power Controlt (C)* bit indicates whether the tailgate supports control of the power supplied to the device(s). When set to one it indicates power control is supported. This power control functionality is not related to the device's power management capabilities, such as that supported by many ATA hard disks.

5.2.1.2 ATA_TIMING_SELECT CSR

This CSR specifies the ATA timings that shall be used by the tailgate during data transfers. These timings are set on a bit basis with the LSB of each field representing Mode 0 timings. Only one bit shall be set in the *PIO_mode* field. If a tailgate supports more than one logical unit, a write to the ATA_TIMING_SELECT CSR of a logical unit shall have no effect on the ATA_TIMING_SELECT CSR of the other logical unit. Figure 9 contains the specification for this CSR.

The modes specified by the ATA_TIMING_SELECT CSR apply exclusively to accesses to the Task File Data register. All other registers of the device shall be accessed by the tailgate using PIO Mode 0 exclusively.

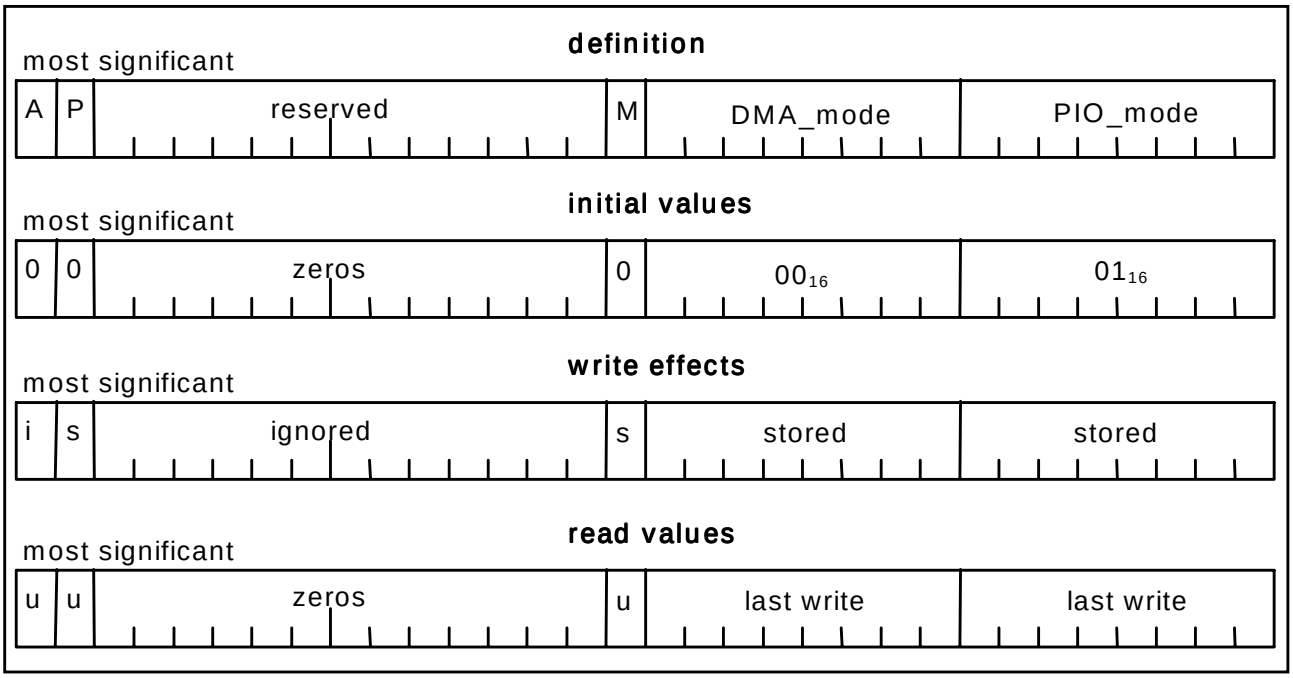

Figure 9 – ATA_TIMING_SELECT CSR specification

The *PIO_mode* field indicates the PIO transfer mode the tailgate shall use when performing subsequent ATA bus PIO transfer operations. The definition of the bits in this field match those for the *PIO_modes* field in the ATA_TIMING_CAPABILITY register given above. A single bit shall be set in this field.

The *DMA_mode* field indicates the DMA transfer mode the tailgate shall use when performing subsequent ATA bus DMA transfer operations. This field is used in conjunction with the *M* bit. The *M* bit indicates whether the DMA mode is a standard multi-word DMA mode or an UltraDMA mode. If the *M* bit is set to 0 then the DMA mode is a multi-word DMA mode and the definition of the bits in this field match those for the *DMA_modes* field in the ATA_TIMING_CAPABILITY register given above. If the *M* bit is set to 1 then the DMA mode is an UltraDMA mode and the definition of the bits in this field match those for the

Ultra_DMA_modes field in the ATA_TIMING_CAPABILITY register given above. Not more than a single bit shall be set in the *DMA_mode* field.

The *Power Control (P)* switches power on or off for each logical unit of the tailgate. Setting the *Power Control* bit to one switches power on to the device. The tailgate shall logically OR the *Power Control* bits for both logical units together to determine whether power should be supplied to the device(s), i.e., if the *Power Control* bit for either logical unit is set to one power shall be switched on for both devices.

The *Power Applied (A)* bit indicates whether power is currently being supplied to the device(s). When the *Power Applied* bit is set to one power is switched on to the device. This bit may be set to one to indicate power is switched on even though the *Power Control* bit for that logical unit is set to zero. This results since having the *Power Control* bit for either logical unit set to one switches power on for both devices. Therefore the initiator should use the value of the *Power Applied* bit to determine whether power is being supplied to a device rather than just using the current value of the *Power Control* bit.

6

Data structure definitions

6.1 Management ORBs

Tailgate devices shall implement the following management ORB functions as defined in SBP-2: Login, Query Login, Logout, Reconnect, Abort Task, Abort Task Set, and Target Reset. These management ORBs are described in the SBP-2 standard. This standard defines an additional ORB, the Set Pasword ORB.

The tailgate implements an access security model described in 0. The five access security functions defined by the access security model are provided by four management ORBs: the Login ORB, the Logout ORB, the Reconnect ORB, and the Set Password ORB. The Query Login ORB provides the related function of reporting the current logged in node, but this ORB has no affect on the access security model.

The tailgate shall implement Abort Task, Abort Task Set and Target Reset functions. The functionality of these reset functions are described in 0.

Most management ORBs required for tailgate implementation are described in the SBP-2 standard. Those ORBs that require additional definition or are not included by SBP-2 are defined in the following clauses: 0 expands upon the definition of the Login ORB provided by SBP-2; 0 provides the definition of the Set Password ORB. Refer to the SBP-2 standard for the definition of all other management ORBs and management ORB fields not defined within this standard.

6.1.1 Login

The format of the Login ORB is defined in Figure 10.

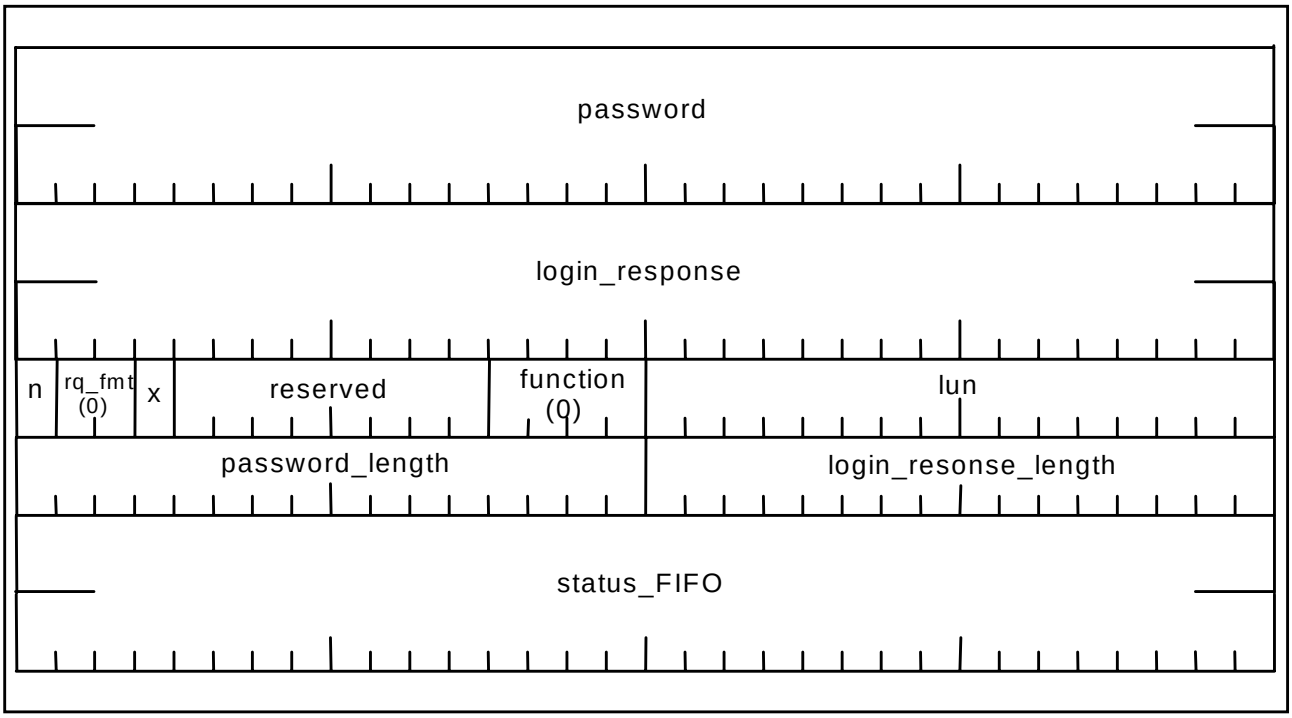

Figure 10 – Login ORB

When the *password* field is zeros no password is present. Otherwise, the *password* field contains eight bytes of password data. The *password_length field shall contain a value of zero.* All eight bytes of a programmed password are significant when performing compares during a login.

All other fields are described by the SBP-2 standard.

6.1.2 Set Password

This function is used to set a new password value for the tailgate.

In processing the Set Password, the tailgate shall validate the login_ID and the Source_ID from which the ORB was fetched. Only the currently logged in initiator is allowed to issue a Set Password ORB.

When the requested operation is completed, whether successful or not, the tailgate shall report a status of normal completion when posting status to the location indicated by the *status_FIFO* field.

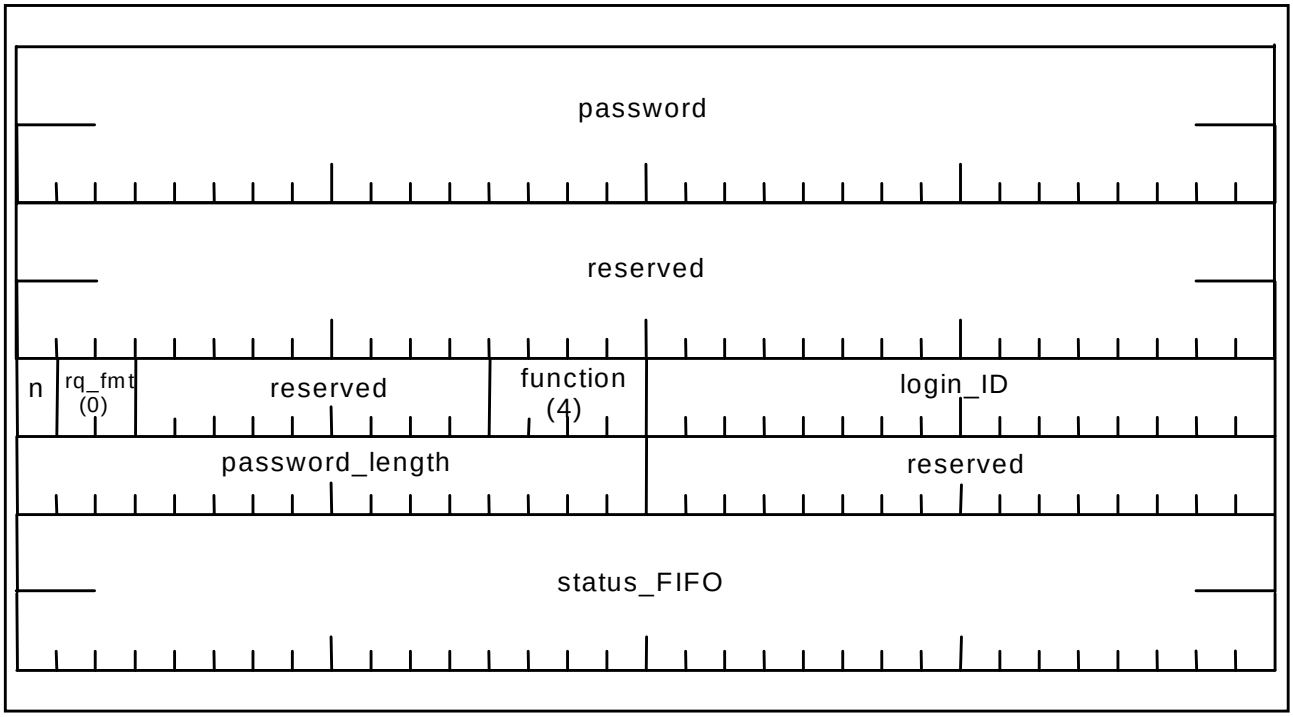

Figure 11 – Set Password ORB

The *password* field contains the password to be programmed into the tailgate. If a password value of zero is specified, the password is cleared. The *password_length* field shall be set to zero.

6.2 Tailgate command blocks

SBP-2 defines the first five quadlets of the command ORB. This standard defines two types of 32 byte command ORBs. The first type of command ORB provides transport of register delivered commands. The second type of command ORB provides transport of packet delivered commands.

6.2.1 Register command ORB

Register delivered commands are issued to the tailgate using the register command ORB. The register command ORB is shown in Figure 12.

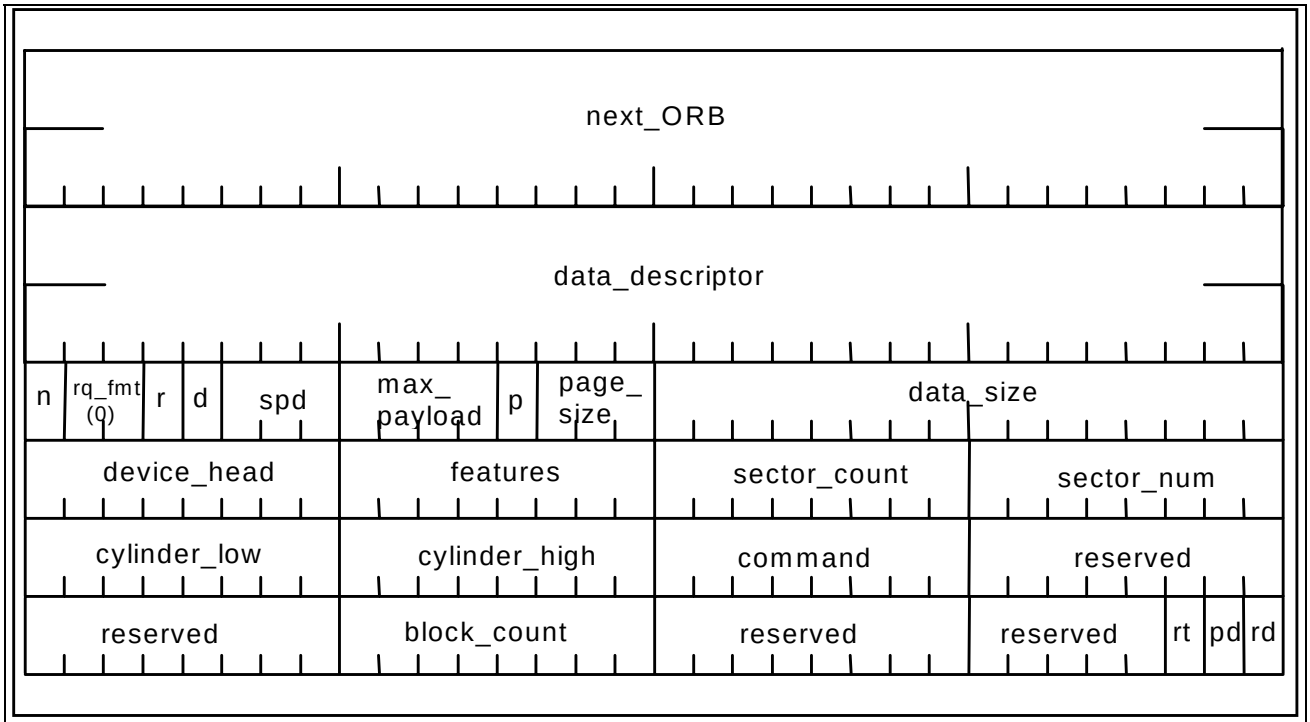

Figure 12 – 32-byte tailgate ATA command ORB

The *features*, *sector_count*, *sector_num*, *cylinder_low*, *cylinder_high*, *device_head*, and *command* fields correspond to the Command Block registers. These bytes contain the values that are written to the ATA device by the tailgate (see the *rt* bit description below).

When these values are written to the ATA task file registers, the tailgate device shall not modify the contents of these bytes prior to transferring them to the ATA device with one exception. Bit 4 of the *device_head* field is the DEV bit, that selects the device (Device 0 or Device 1), is ignored by the tailgate. The tailgate generates the value of the DEV bit based upon the logical unit the command was sent to and the implementation of the tailgate.

The *block_count* field is set by the initiator to indicate the number of 512 byte blocks to be transferred for this request. A value of zero in the *block_count* field indicates no data shall be transferred. The valid range for this field is 0 to 256. The result of commands with a *block_count* outside the specified range is indeterminate.

The *rt* bit directs the tailgate to return status, as described in 0. When the *rt* bit is set the tailgate may write to the Device/Head register in order to select the appropriate device for the logical unit the command was issued to, but shall not write to any other Command Block registers nor read from the Data register. If the *rt* bit is zero the tailgate shall write the appropriate values in the ORB to the Command Block registers of the device, perform any associated data transfers, and return status.

The *pd* bit specifies the data transfer mode. If the *pd* bit is zero then the tailgate shall transfer data using PIO mode. If the *pd* bit is one then the data shall be transmitted using DMA mode.

The initiator shall be responsible for selecting the appropriate command to accompany the mode selected via the *pd* bit. For example, with a *pd* bit of one the initiator should use the Read DMA or Write DMA commands to transfer disk data. Use of commands inappropriate to the setting of the *pd* bit may result in unpredictable device operation.

The *rd* bit specifies the type of command ORB. For the register command ORB, this bit shall be set to one by the initiator.

6.2.2 Packet command ORB

Packet delivered commands are issued to the tailgate using the packet command ORB. The packet command ORB is shown in Figure 13.

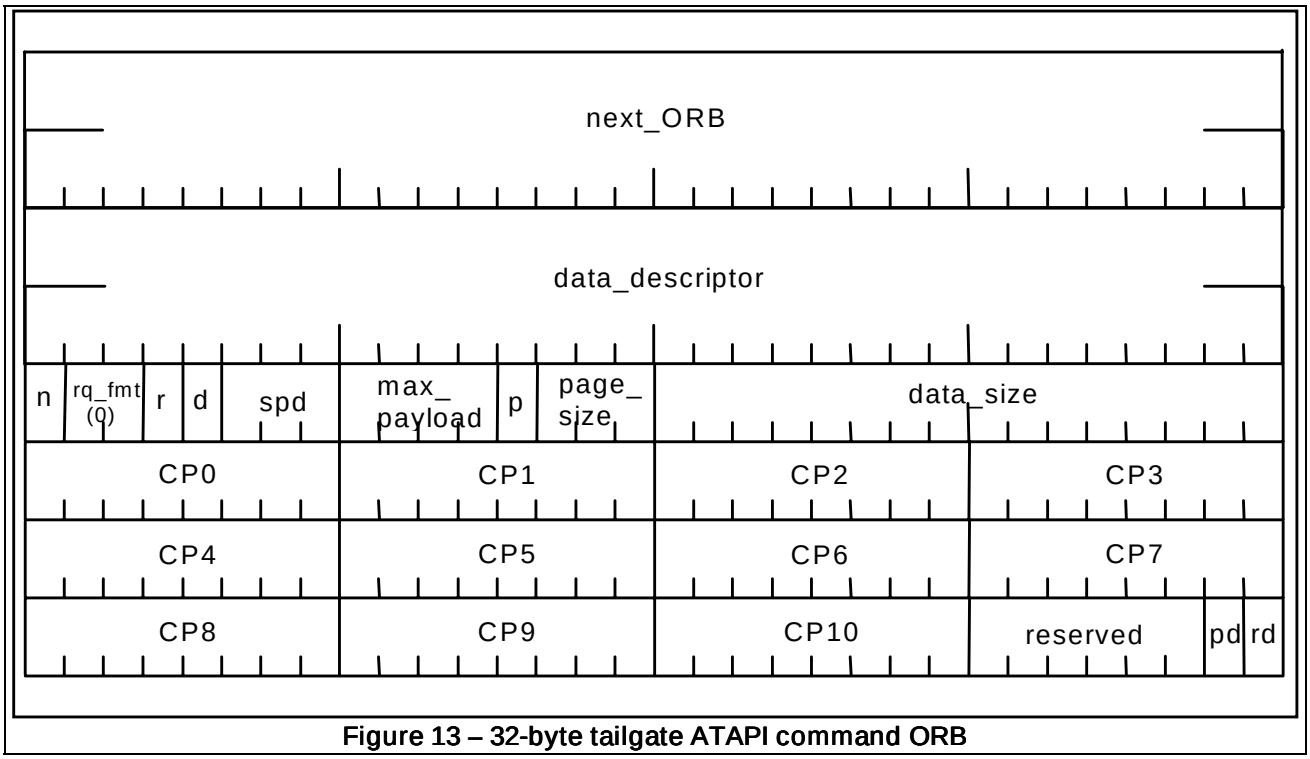

Figure 13 – 32-byte tailgate ATAPI command ORB

The *CP0* through *CPB10* fields are the first 11 bytes of the ATAPI command packet. The final byte, CP11, is generated by the tailgate during the command packet transfer. The value generated for CP11 shall be 0.

The *pd* field specifies the data transfer mode. If the *pd* bit is zero then the tailgate shall transfer data using PIO mode. If the *pd* bit is one then the data shall be transmitted using DMA mode.

The tailgate shall copy the value of the *pd* bit to bit 0 (DMA) of the ATAPI Feature register prior to issuing the packet command. This indicates to the ATAPI device whether the transfer mode is to be PIO or DMA. The sense of the bit is the same as defined for the *pd* bit above.

The *rd* field specifies the type of command ORB. For the packet command ORB, this bit shall be set to zero by the initiator.

6.3 Tailgate Status Block Definition

When the tailgate returns status, the status block shall be 4 quadlets in length. The format of the status block is shown in Figure 14. The status block is written to the initiator's status FIFO as specified at login.

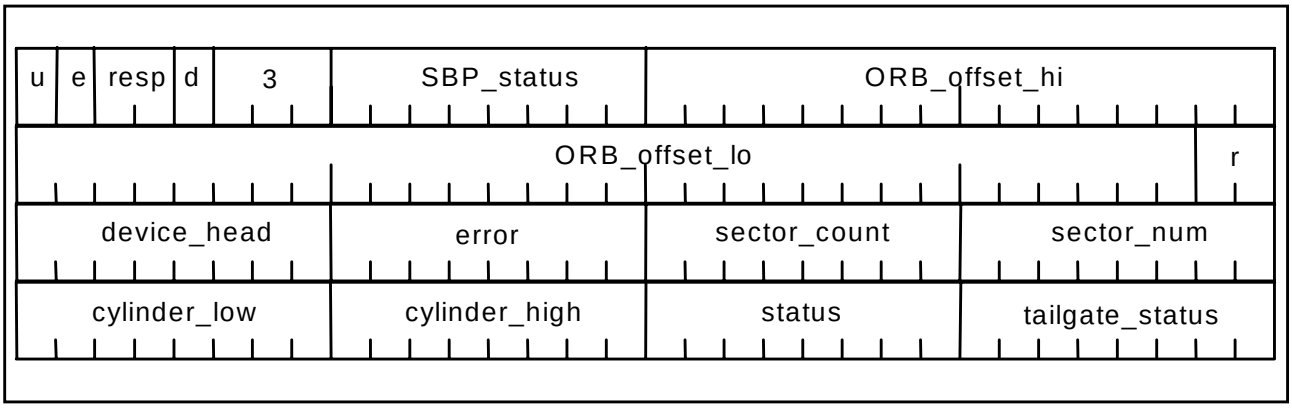

Figure 14 – Tailgate status block

The *device_head*, *error*, *sector_count*, *sector_num*, *cylinder_low*, *cylinder_high*, and *status* fields shall contain values as read from the device's respective Command Block registers at command completion. The initiator shall ignore these fields if the *unsolicited* bit is set to one or the *resp* field contains a value other than request complete. The initiator shall examine the status block fields to determine whether an error has occurred.

The *tailgate_status* field contains status specific to the tailgate's interaction with it's attached devices and the ATA bus. Table 8 contains a list of return values for the *tailgate_status* field.

Table 8 – *tailgate_status* field values

Value	Description
0	No error.
1	Data Size Not Exact (informative).
2	No ATAPI Command Phase. Upon received a packet command ORB the tailgate did not detect status from the device indicating the command phase after issuing the PACKET command.
3	Busy at start of command. Normally the tailgate waits until the device is not busy prior to attempting to issue a command to the device. This status code results when the execution engine begins execution of a command and determines that the device is currently busy.
4	Task Aborted
5	Task Set Aborted
6	Target Reset has completed
6 through FE_{16}	Reserved
FF_{16}	Other Protocol Errors

Fields other than those described shall be implemented as defined in SBP-2.

Configuration ROM

All nodes that implement tailgate units as a unit architecture shall implement general format configuration ROM in accordance with ISO/IEC 13213:1994, IEEE Std 1394-1995 and the SBP-2 standard. The only changes to the configuration ROM from the SBP-2 standard are in the Unit directory entry and are listed in the clauses below.

The entries below are duplicated from the SBP-2 and IEEE 1212 documents. Complete definitions of these entries may be found in those documents. The field entries below have specific values that are specific to tailgate implementations. Where values are indicated devices compliant to this standard shall use those values in their configuration ROM.

7.1 Unit_Spec_ID entry

The Unit_Spec_ID entry is an immediate entry in the tailgate unit directory that specifies the organization responsible for the architecture definition of the unit.

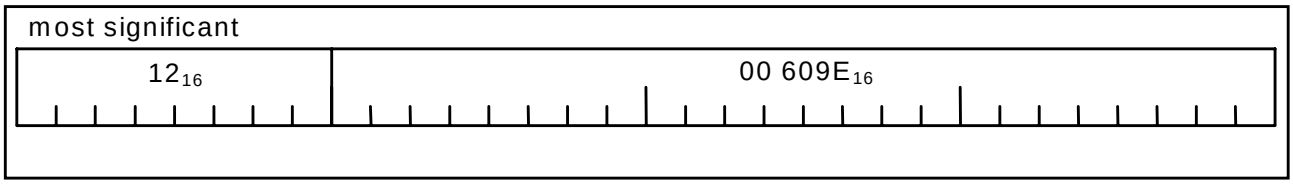

Figure 15 – Unit_Spec_ID entry format

12_{16} is the concatenation of *key_type* and *key_value* for the Unit_Spec_ID entry.

00 609E$_{16}$ is the U*nit_Spec_ID* obtained from the IEEE/RAC. The value indicates that the NCITS Secretariat is responsible for the software interface definition.

7.2 Unit_SW_Version entry

The Unit_SW_Version entry is an immediate entry in the tailgate unit directory that, in combination with the *unit_spec_ID*, specifies the software interface of the unit.

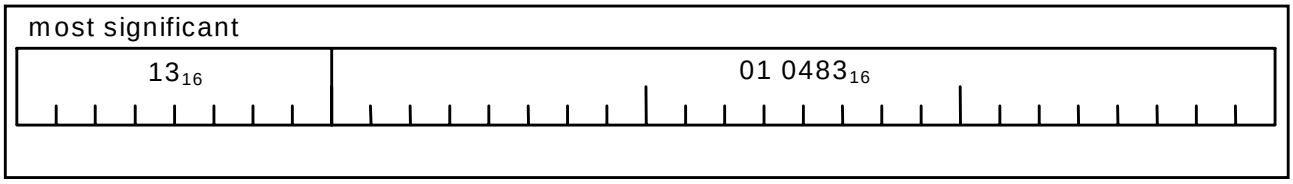

Figure 16 – Unit_SW_Version entry format

13_{16} is the concatenation of *key_type* and *key_value* for the Unit_SW_Version entry.

01 0483$_{16}$ is the U*nit_SW_Version* value that indicates that the unit conforms to this standard.

7.3 Command_Set_Spec_ID entry

The Command_Set_Spec_ID entry is an immediate entry in the tailgate unit directory that, when present in the unit directory, specifies the organization responsible for the command set definition for the target. Figure 15 shows the format of this entry.

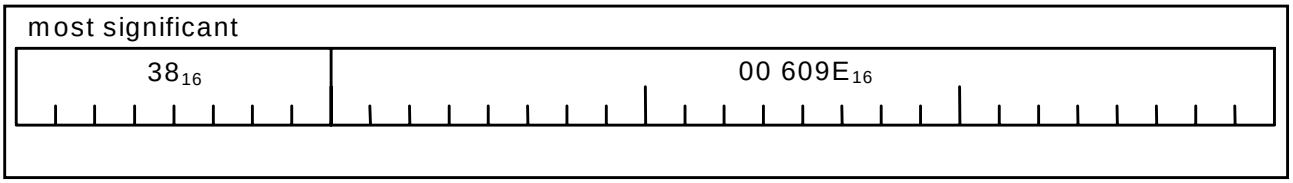

Figure 17 – Command_Set_Spec_ID entry format

38_{16} is the concatenation of *key_type* and *key_value* for the Command_Set_Spec_ID entry.

00 609E$_{16}$ is the 24-bit *command_set_spec_ID* value, an organizationally unique identifier, for the organization which has ownership of the tailgate standard.

7.4 Command_Set_Version entry

The Command_Set_Version entry is an immediate entry in the tailgate unit directory that, when present in the unit directory, in combination with the *command_set_spec_ID* identifies the target as a compliant to the tailgate standard. Figure 18 shows the format of this entry.

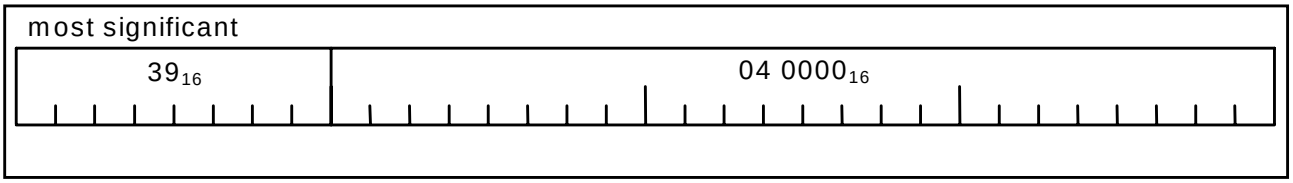

Figure 18 – Command_Set_Version entry format

39_{16} is the concatenation of *key_type* and *key_value* for the Command_Set_Version entry.

04 0000_{16} is the *command_set_version* value that indicates, in combination with the *command_set_spec_ID*, that the unit conforms to the tailgate standard.

7.5 Logical_Unit_Characteristics entry

The Logical_Unit_Characteristics entry is an immediate entry in the tailgate unit directory that, when present in the unit directory, specifies characteristics of the tailgate implementation. Figure 19 shows the format of this entry.

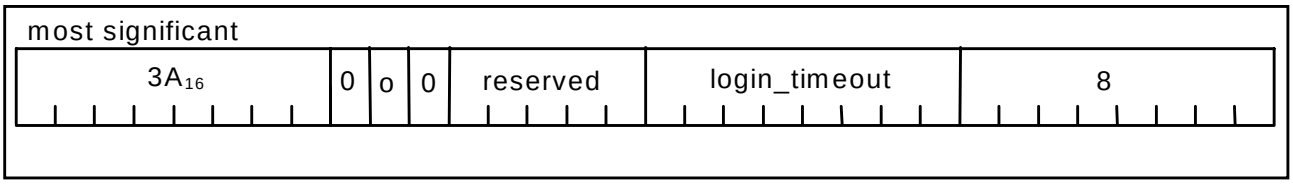

Figure 19 – Logical_Unit_Characteristics entry format

$3A_{16}$ is the concatenation of *key_type* and *key_value* for the Logical_Unit_Characteristics entry.

The other values indicate that the tailgate supports the basic task management model, does not support isochronous transfers, and transfers 32 byte ORBs, respectively.

The setting of the *login_timeout* field is implementation specific.

The setting of the *o* flag is implementation specific, however it shall typically be set to one to indicate the device performs no reordering.

7.6 Management_Agent entry

The Management_Agent entry is an immediate entry in the tailgate unit directory that specifies the base address of the tailgate's management agent CSR. Figure 20 shows the format of this entry.

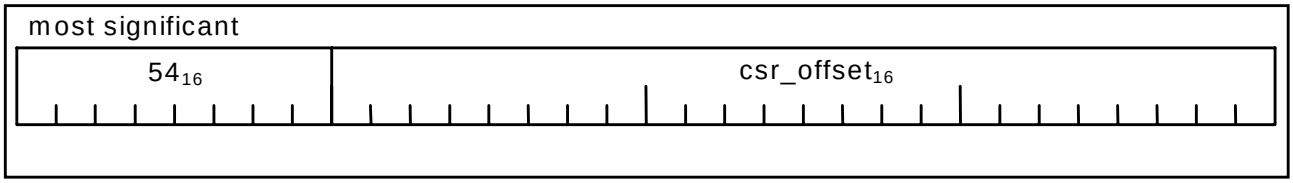

Figure 20 – Management_Agent entry format

54_{16} is the concatenation of *key_type* and *key_value* for the Management_Agent entry.

The *csr_offset* field shall contain the quadlet offset, from the base address of initial register space, FFFF F000 0000_{16}, to the base address of the MANAGEMENT_AGENT CSR for the target. All target CSR's shall be located at or above address FFFF F001 0000_{16}; therefore the value of *csr_offset* shall not be less than 4000_{16}.

7.7 Logical_Unit_Number entry

The Logical_Unit_Number entry is an immediate entry in the tailgate unit directory that, when present in the unit directory, specifies the peripheral device type and logical unit number of a logical unit of the tailgate. Figure 21 shows the format of this entry.

Tailgates that support single logical unit configurations shall implement a single Logical_Unit_Number entry. Dual logical unit tailgates shall implement two Logical_Unit_Number entries. The presence of two Logical_Unit_Number entries must not be assumed by the initiator to indicate the presence of two ATA or ATAPI devices. The initiator must use the initialization procedure defined in annex A to detect the number and type of ATA or ATAPI devices present.

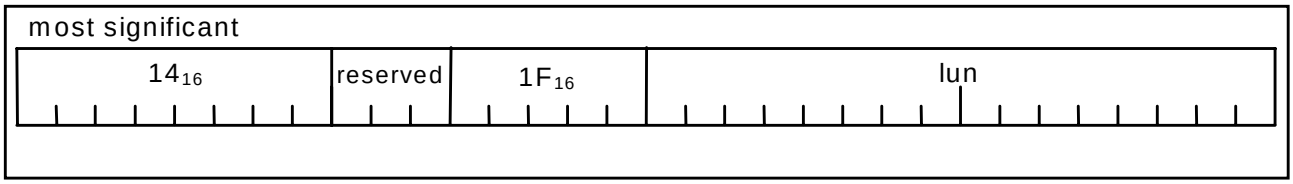

Figure 21 – Logical_Unit_Number entry format

14_{16} is the concatenation of *key_type* and *key_value* for the Logical_Unit_Number entry.

$1F_{16}$ is the device_type defined by SPC to indicate the device type is unknown. The device type for devices attached via a tailgate shall be determined using the initialization sequence outlined in annex A.

The *lun* field shall identify the logical unit to which the information in the Logical_Unit_Number entry applies.

Annex A
Initialization Procedure
(informative)

A.1 Initialization Procedure

The annex describes the steps required for the initiator to initialize the tailgate preceding ORB command operation. This annex is intended to be an example of a possible initialization and device detection sequence. Refer to the SBP-2 and 1394-1995 standards for the complete implementation details. Figure A.1contains a flowchart of the device detection process. The only indication the initiator receives via the 1394 device configuration ROM is that the device is a tailgate. It does not receive information on whether the attached device is a register delivered or packet delivered command device. Nor does it receive device class information indicating whether the attached device is a CD-ROM, 120 MB floppy, HDD, DVD, or any other type of device. The initiator must determine the device protocol (register or packet delivered commands) and class (CD-ROM, etc.) using the standard device detection mechanism for ATA/ATAPI-4 compliant devices. The 1394 configuration ROM does indicate whether the tailgate is a single or dual logical unit tailgate. However, all this indicates is support of single or dual logical units. It does not indicate the number of devices actually attached to the tailgate. This must be determined as indicated in the flowchart.

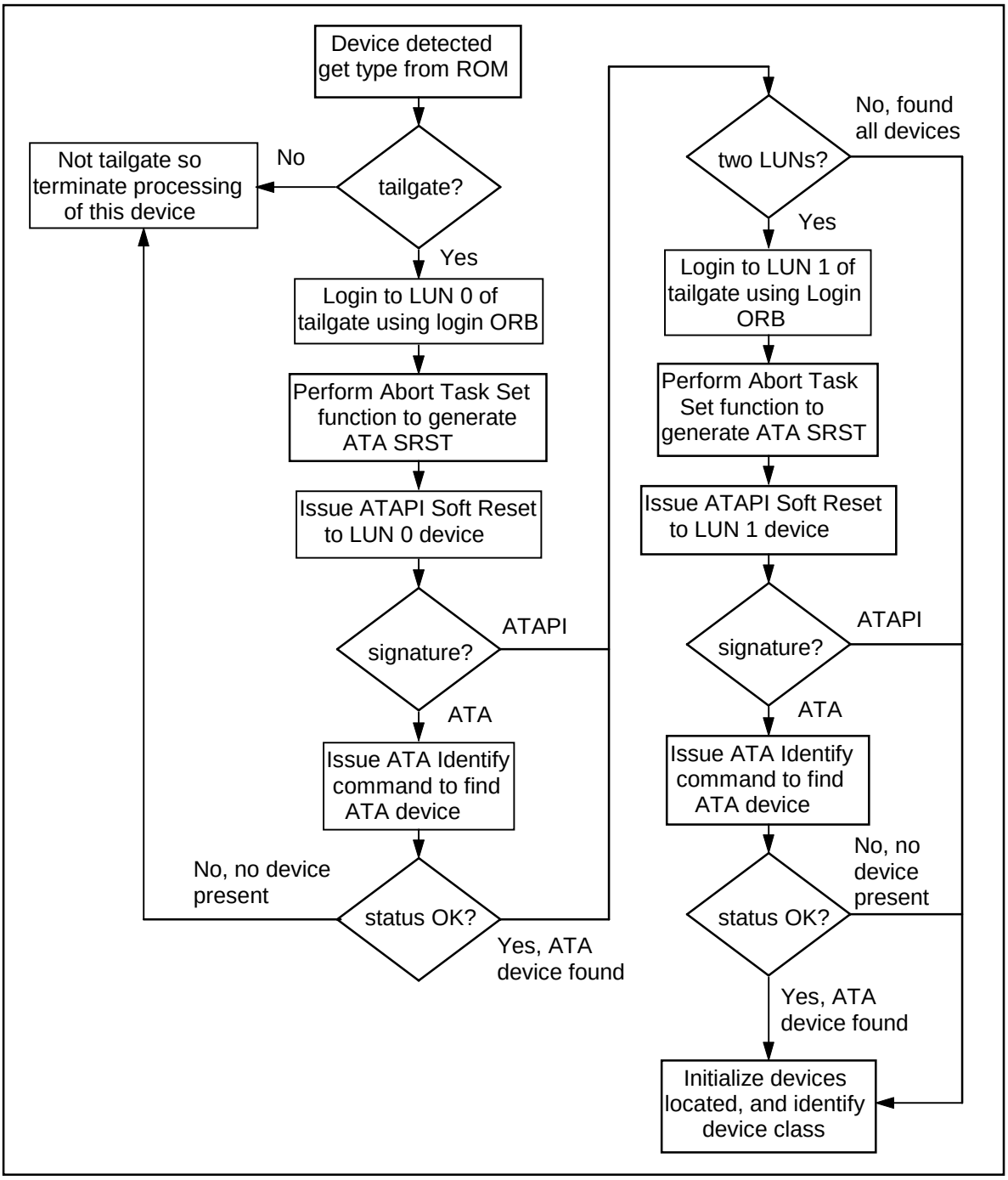

Figure A.1 – Initialization flowchart

Annex B
Initialization procedure tailgate execution engine models
(informative)

B.1 Execution engine processing

There are three primary types of execution engines: single device, dual device, and dual device with ATAPI overlap. Following are two state diagrams and accompanying text for these execution engine types. The execution engine for the single logical unit and dual logical unit (without ATAPI overlap) are identical. The differences are addressed via notes in the single logical unit state diagram description. The interface between the execution and fetch engine shall be defined by these state diagrams. The fetch engine must comply with the possible states of the execution engine. As long as the fetch engine does not violate these execution engine states any fetch engine implementation is valid. The primary points to be understood from these state diagrams is that the execution engine controls the reading of ORBs from the fetch engines via the throttle indications, as discussed in the clause 0.

B.2 Single/dual (without overlap) logical unit execution engine

The state diagram and text that follow describe the operation of single logical unit and dual logical unit (without overlapped command support) execution engines. The processing for the single logical unit and no overlap dual logical unit configurations is very similar. The no overlap dual logical unit configuration assumes that each logical unit is the sole owner of the execution engine for processing of a command. A mechanism implemented outside the execution state machine, as defined by the state diagram in figure B.1, performs an arbitration function to control which device gains ownership of the execution engine for each command sequence.

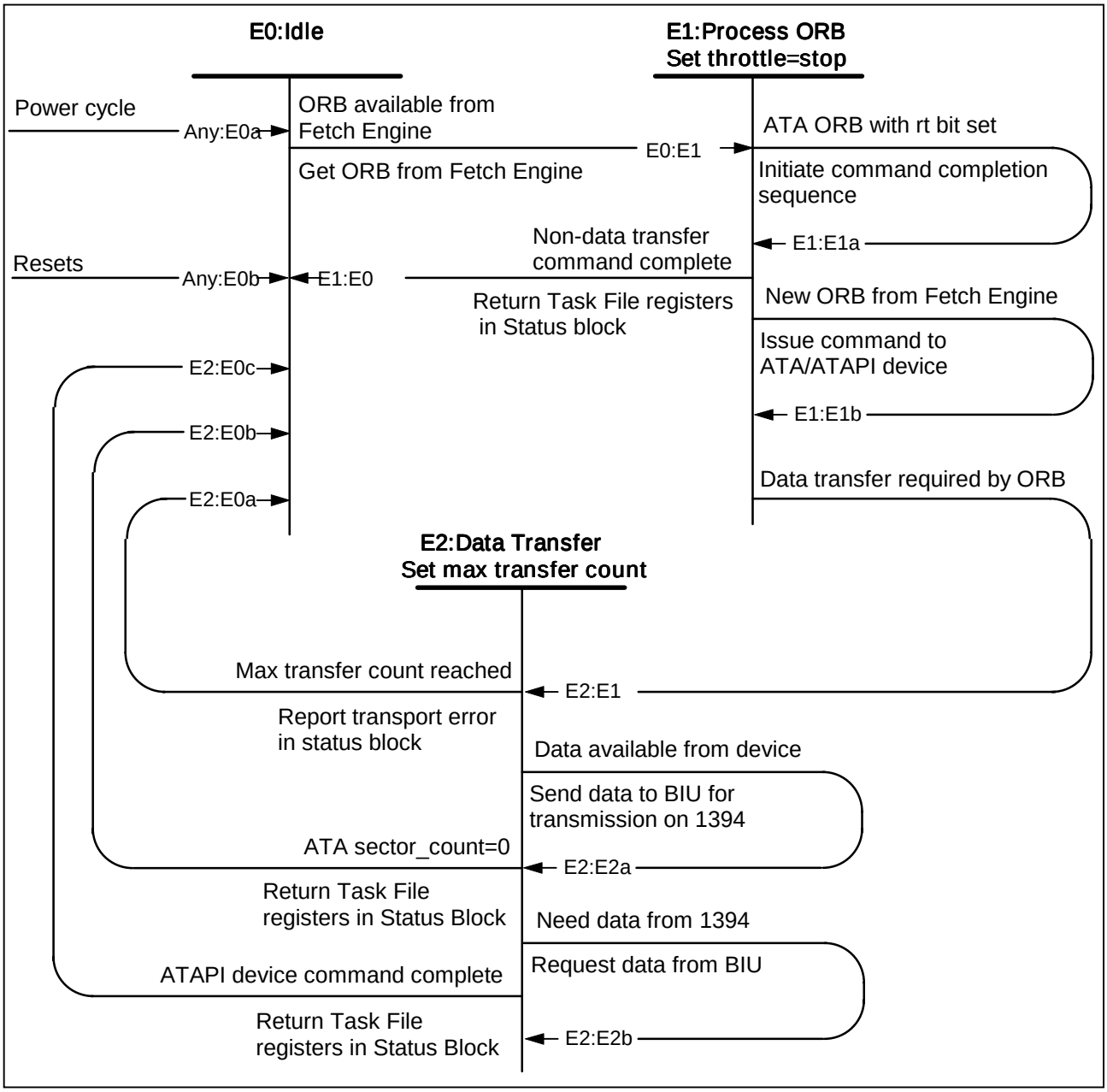

Figure B.1 – Single logical unit execution engine state diagram

Transition Any:E0a. Power cycles shall transition the execution engine to the idle state from any other state.

Transition Any:E0b. All resets (ATAPI Soft Reset, ATA Soft Reset, and ATA Hard Reset) shall transition the execution engine to the idle state from any other state.

State E0. While in the Idle state the execution engine shall set the throttle indication to the go state. This permits the fetch engine to send another ORB to the execution engine when available.

> NOTE – For dual logical unit execution engines the throttle indication applies to both fetch engines (logical unit 0 and logical unit 1) equally. Therefore when the throttle is set to the go state both fetch engines are enabled to send ORBs to the execution engine. The details of arbitration between the fetch engines is implementation specific.

Transition E0:E1. Once an ORB is sent to the execution engine, by the fetch engine, the execution engine accepts the ORB and transitions to the Process ORB state.

State E1. Upon entering the Process ORB state, the execution engine sets the throttle indication to the stop state. This prevents the fetch engine from transferring any more ORBs to the execution engine. The execution engine then begins processing the ORB.

> NOTE – For dual logical unit execution engines, the throttle indications apply to both fetch engines equally. Therefore once the throttle indication is set to stop neither fetch engine may send ORBs to the execution engine.

Transition E1:E1a. When an ORB with the *rt* bit set then no command shall be issued to the device and the command completion sequence (Transition E1:E0b) shall be initiated.

Transition E1:E1b. When an ATA ORB with the *rt* bit reset or an ATAPI ORB is received the execution engine shall issue the ATA or ATAPI command to the device.

Transition E1:E0. Once the execution engine completes execution of a request, including device completion of the command if required, the execution engine shall read the current contents of the Task File registers and transmit them to the initiator via the status block.

Transition E1:E2. If the execution engine receives a command requiring data transfer it transitions to the Data Transfer state. The throttle indications remain in the stop state preventing any further ORBs from the fetch engine.

State E2. While in the Data Transfer state, the execution engine moves data between the 1394 bus and the device. The direction of data is dependent upon the setting of the *d* bit in the ORB. The data transfer count is set to the maximum length of the data transfer based upon the ORB. If the *data_size* specifies a byte count then the maximum transfer count operates in units of bytes. If the *data_size* specifies a page table entry count then the maximum transfer count operates in units of pages. This places an upper limit on the transfer so that transfer boundaries chosen by the initiator are not exceeded.

Transition E2:E2a. When data becomes available from the device, the execution engine reads the data, using the appropriate data transfer protocol, and transfers the data to the Bus Interface Unit for transmission on the 1394 bus. The data transfer protocol used is dependent upon the device type (ATA or ATAPI) and the transfer mode selected in the ORB (PIO or DMA).

Transition E2:E2b. When the device requires data, the execution engine requests the data from the Bus Interface Unit and transfers, using the appropriate data transfer protocol, the data to the device. The data transfer protocol used is dependent upon the device type (ATA or ATAPI) and the transfer mode selected in the ORB (PIO or DMA).

Transition E2:E0a. When the maximum transfer count has been reached, the execution engine shall report a transport error in the status block and transition to the Idle state.

Transition E2:E0b. If the ORB contains an ATA command, the *block_count* field is used to determine the number of 512 byte blocks to be transferred by the execution engine. Once the programmed number of blocks has been transferred, the execution engine reads the current Task File registers and returns them to the initiator via the status block

Transition E2:E0c. If the ORB contains an ATAPI command then the ATAPI device indicates the command completion via an interrupt and specific values returned in the ATAPI Interrupt Reason register. Once the ATAPI device indicates command completion the execution engine reads the current Task File registers and returns them to the initiator via the status block.

B.3 Dual logical unit (with ATAPI overlap) execution engine

The state diagram and text that follow define the execution engine for dual logical unit tailgates that support overlapped command operation. Although the state diagram is oriented towards ATAPI overlap operation, any overlapped command model could operate within the constraints defined by the state diagram in figure B.2.

The terminology overlapped device and nested device are used in subsequent discussions of the dual logical unit (with ATAPI overlap) state diagram and the remainder of the document. The term overlapped device refers to an ATAPI device with ATAPI overlap capability that has released the bus. The term nested device refers to an ATA or ATAPI device that shares the ATA bus with the overlapped device and can therefore accept and process one or more commands until the ATAPI device is re-selected via the Service command. In the text accompanying the state diagram logical unit x refers to the overlapped device and logical unit y refers to the nested device.

Tailgates that implement dual logical unit support via separate ATA buses may choose to implement overlapped operation in conformance to the state machine as defined in figurs B.2 or may choose to implement two individual execution engines that conform to Figure . Either method shall be considered in conformance to the tailgate standard.

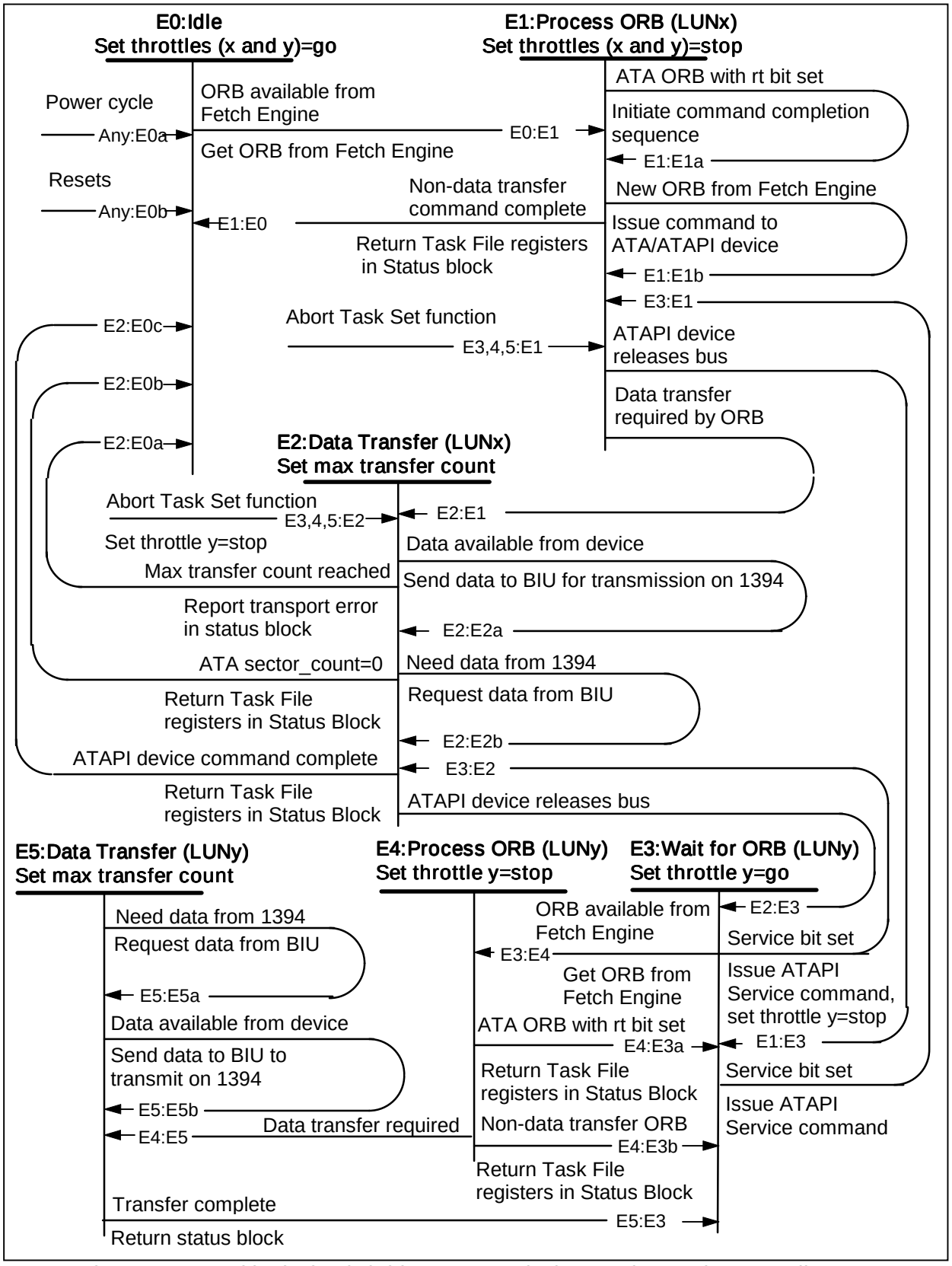

Figure B.2 – Dual logical unit (with ATAPI overlap) execution engine state diagram

Transition Any:E0a. Power cycles shall transition the execution engine to the idle state from any other state.

Transition Any:E0b. All resets (ATAPI Soft Reset, ATA Soft Reset, and ATA Hard Reset) shall transition the execution engine to the idle state from any other state.

State E0. While in the Idle state, the execution engine shall set the throttle indications for both fetch engines to the go state. This permits the fetch engine to send another ORB to the execution engine when available.

Transition E0:E1. Once an ORB is sent to the execution engine, by the fetch engine, the execution engine accepts the ORB and transitions to the Process ORB state.

State E1. Upon entering the Process ORB state, the execution engine sets the throttle indications for both logical units to the stop state. This prevents the fetch engine from transferring any more ORBs to the execution engine. The execution engine then begins processing the ORB. The logical unit which originated the ORB is designated as logical unit x. If the ORB is an ATAPI ORB the execution engine may set the Overlap enable bit in the ATAPI Task File Status register to enable the ATAPI overlap feature.

Transition E1:E1a. When an ORB with the *rt* bit set, no command shall be issued to the device and the command completion sequence (Transition E1:E0b) shall be initiated.

Transition E1:E1b. When an ATA ORB with the *rt* bit reset or an ATAPI ORB is received, the execution engine shall issue the ATA or ATAPI command to the device.

Transition E1:E0b. Once the device completes execution of a request, including device completion of a the command if required, the execution engine shall read the current contents of the Task File registers and transmit them to the initiator via the status block. If the device on logical unit x is an ATAPI device it may release the bus after receiving the ORB. Therefore the status and completion indicated by this transition may occur after an ATAPI device has released the bus and reconnected via the ATAPI Service command.

Transition E1:E2. If an ORB requiring data transfer was received by the device, it transitions to the Data Transfer state. The throttle indications remain in the stop state preventing any further ORBs from the fetch engines.

Transition E1:E3. If the ATAPI device releases the bus, the execution engine transitions to the Wait for ORB (logical unit y) state.

State E2. While in the Data Transfer state the execution engine moves data between the 1394 bus and the device. The direction of data is dependent upon the setting of the *d* bit in the ORB. The data transfer count is set to the maximum length of the data transfer based upon the ORB. If the *data_size* specifies a byte count then the maximum transfer count operates in units of bytes. If the *data_size* specifies a page table entry count then the maximum transfer count operates in units of pages. This places an upper limit on the transfer so that transfer boundaries chosen by the initiator are not exceeded.

Transition E2:E2a. When data becomes available from the device, the execution engine reads the data, using the appropriate data transfer protocol, and transfers the data to the Bus Interface Unit for transmission on the 1394 bus. The data transfer protocol used is dependent upon the device type (ATA or ATAPI) and the transfer mode selected in the ORB (PIO or DMA).

Transition E2:E2b. When the device requires data, the execution engine requests the data from the Bus Interface Unit and transfers, using the appropriate data transfer protocol, the data to the device. The data transfer protocol used is dependent upon the device type (ATA or ATAPI) and the transfer mode selected in the ORB (PIO or DMA).

Transition E2:E0a. When the maximum transfer count has been reached, the execution engine shall report a transport error in the status block and transition to the Idle state.

Transition E2:E0b. If the ORB contains an ATA command, the *block_count* field is used to determine the number of 512 byte blocks to be transferred by the execution engine. Once the programmed number of blocks has been transferred the execution engine reads the current Task File registers and returns them to the initiator via the status block.

Transition E2:E0c. If the ORB contains an ATAPI command, the ATAPI device indicates command completion via an interrupt and specific values returned in the ATAPI Interrupt Reason register. Once the ATAPI device indicates command completion, the execution engine reads the current Task File registers and returns them to the initiator via the status block.

Transition E2:E3. If the ATAPI device releases the bus, the execution engine transitions to the Wait for ORB (logical unit y) state.

State E3. While in the Wait for ORB (logical unit y) state, the execution engine waits for one of several events to occur. If first releases the throttle indication for logical unit y. While in this state, the execution engine shall monitor the interrupt or the Service bit in the ATAPI Task File Status register in the overlapped device. This is necessary to detect when the overlapped device requires service to continue processing the command.

Transition E3:E4. If the fetch engine has available ORBs for logical unit y, it shall accept those ORBs and transition to the Process ORB (logical unit y) state.

Transition E3:E1. If the execution engine detects, via the Service bit in the ATAPI Task File Status register, that the ATAPI device requires reconnection to continue processing the overlapped command it shall transition to the Process ORB (logical unit x) state. This transition represents the mirror of the E1:E3 transition. This transition occurs when a command that required no data transfer released the ATA bus.

Transition E3:E2. If the execution engine detects, via the Service bit in the ATAPI Task File Status register, that the ATAPI device requires reconnection to continue processing the overlapped command it shall transition to the Process ORB (logical unit x) state. This transition represents the mirror of the E2:E3 transition. This transition occurs when a command that required data transfer released the ATA bus.

State E4. While in the Process ORB (logical unit y) state, the throttle indication for logical unit y is set to the stop state. This prevents the fetch engine from sending additional ORBs to the execution engine. The execution engine then begins processing the ORB. If the ORB is an ATAPI ORB, the execution engine shall not set the Overlap enable bit in the ATAPI Task File Features register.

Transition E4:E3a. If an ATA ORB with the *rt* bit set was received, the device shall read the current contents of the Task File registers from the device and transmit them to the initiator via the status block.

Transition E4:E3b. If a non-data transfer ORB was received, the device shall send the command to the ATA/ATAPI device for execution. Once the device completes execution, the execution engine shall read the current contents of the Task File registers and transmit them to the initiator via the status block.

Transition E4:E5. If the ORB requires data transfer, the execution engine transitions to the Data Transfer (logical unit y) state.

State E5. While in the Data Transfer state, the execution engine moves data between the 1394 bus and the device. The direction of data is dependent upon the setting of the *d* bit in the ORB. The data transfer count is set to the maximum length of the data transfer based upon the ORB. If the *data_size* specifies a byte count then the maximum transfer count operates in units of bytes. If the *data_size* specifies a page table entry count then the maximum transfer count operates in units of pages. This places an upper limit on the transfer so that transfer boundaries chosen by the initiator are not exceeded.

Transition E5:E5a. When data becomes available from the device, the execution engine reads the data, using the appropriate data transfer protocol, and transfers the data to the Bus Interface Unit for transmission on the 1394 bus. The data transfer protocol used is dependent upon the device type (ATA or ATAPI) and the transfer mode selected in the ORB (PIO or DMA).

Transition E5:E5b. When the device requires data, the execution engine requests the data from the Bus Interface Unit and transfers, using the appropriate data transfer protocol, the data to the device. The data transfer protocol used is dependent upon the device type (ATA or ATAPI) and the transfer mode selected in the ORB (PIO or DMA).

Transition E5:E3. Once the transfer is completed, the execution engine transitions back to the Wait for ORB (logical unit y) state. The completion could be due to the maximum transfer count being reached, an ATA ORB sector count reaching zero, or an ATAPI device reporting command complete status. Under any of these conditions, the tailgate shall report the appropriate completion via the status block. Transitions E2:E0a, E2:E0b, and E2:E0c detail the appropriate processing and conditions relating to these three possible termination conditions that also apply to this transition. For brevity the complete text and transition break-down are not replicated here.

Transition E3,4,5:E1. If an Abort Task Set function is received by the execution engine during execution of a nested command, i.e., a command issued to the to an ATA or ATAPI device after an ATAPI device releases the bus in ATAPI Overlap mode, the execution engine shall transition from any of the states E3, E4, or E5 to the E1 state if the overlap command had not initiated any data transfer yet.

Transition E3,4,5:E2. If an Abort Task Set function is received by the execution engine during execution of a nested command, i.e., a command issued to the to an ATA or ATAPI device after an ATAPI device releases the bus in ATAPI Overlap mode, the execution engine shall transition from any of the states E3, E4, or E5 to the E2 state if the overlap command had already initiated data transfer.

Annex C
Required core registers for tailgate
(informative)

C.1 ISO/IEC 13213:1994 and IEEE 1394-1995 defined registers

A number of core registers are required by ISO/IEC 13213:1994 in any compliant implementation. IEEE 1394-1995 is a compliant implementation and implements these core registers. In addition, IEEE 1394-1994 defines a number of additional CSRs. Some of the CSRs defined are required for all 1394 compliant devices, while others are optional. Within some CSRs there may be fields or bits which may be optional.

Those CSRs which are required are listed below. If a field is optional for a tailgate device it will be explicitly labeled as optional, otherwise all fields are required.

The CSR architecture standardizes the locations and functions of core registers. The addresses of these registers are specified in terms of byte offsets within initial register space, where the base address of initial register space is FFFF F000 0000_{16} relative to initial node space. The table below summarizes which core registers are mandatory for tailgate devices.

Offset	Register name	Description
0	STATE_CLEAR	State and control information
4	STATE_SET	Sets STATE_CLEAR bits
8	NODE_IDS	Contains the 16-bit node_ID value used to address the node
$0C_{16}$	RESET_START	Resets the node's state
18_{16} – $1C_{16}$	SPLIT_TIMEOUT	Time limit for split transactions

The CSR architecture reserves a portion of initial register space for bus-dependent uses. Serial Bus defines registers within this address space, whose addresses are specified in terms of byte offsets within initial register space, where the base address of initial register space is FFFF F000 0000_{16} relative to initial node space. The table below summarizes which Serial Bus-dependent registers are mandatory for tailgate devices.

Offset	Register name	Description
210_{16}	BUSY_TIMEOUT	Controls transaction layer retry protocols

EEE Std 1394-1995 should be consulted for detailed descriptions of these core registers.

C.2 SBP-2 Defined Registers

SBP-2 defines a set of CSRs required to implement the Management Agent and Command Agent functions. All of the CSRs defined within the SBP-2 specification shall be implemented by a tailgate compliant device. Tailgates may, optionally, support dual ATA/ATAPI devices. A complete set of Command Agent CSRs shall be implemented for each logical unit supported. The following list of CSRs and/or fields shall be implemented by a tailgate compliant device:

C.2.1 MANAGEMENT_AGENT Register

IEEE Std 1394-1995 reserves a portion of initial units space for bus-dependent use; the MANAGEMENT_AGENT register shall be located at address FFFF F001 0000_{16} or above within the node's 48-bit address range. The address of the management agent is specified by the *csr_offset* field in the Management_Agent entry in configuration ROM.

Relative Offset	Name	Description
00_{16}	MANAGEMENT_AGENT	Address of ORB

C.2.2 COMMAND_AGENT Registers

Unlike the Management Agent, which services as the management function for the entire unit, a separate Command Agent exists for each logical unit supported. For each logical unit, one complete set of Command Agent CSRs shall be implemented. Each Command Agent's set of CSRs shall reside within the target's initial units and shall be located at address FFFF F001 0000_{16} or above within the node's 48-bit address range.

Although the location of each Command Agent's CSR's is not fixed, the relative relationship of the registers is fixed within a contiguous block of eight quadlets, as defined by the table below. The base address of the Command Agent CSRs is returned in the Login Response packet.

Relative Offset	Name	Description
00_{16}	AGENT_STATE	Reports fetch agent state
04_{16}	AGENT_RESET	Resets fetch agent
08_{16}	ORB_POINTER	Address of ORB
10_{16}	DOORBELL	Signals fetch agent to refetch an address pointer
14_{16}	STATUS_ACKNOWLEDGE	Acknowledges the initiator's receipt of unsolicited status
18_{16} – $1C_{16}$		Reserved for future standardization

Refer to the SBP-2 specification for definition of these registers.

Annex D
Dual device support summary
(informative)

D.1 Dual ATA/ATAPI device support

Tailgates may optionally support a Device 0 and Device 1 configuration in addition to the required device 0 only configuration. Tailgates may be designed to support either single or dual device configurations. The requirements for dual device configuration have no impact on cost or complexity of single device tailgate designs. Dual device tailgates shall support all functionality defined for single device tailgates. Dual device tailgates also have several additional requirements:

1. An additional ATA_TIMING_CAPABILITY register for Device 1.

2. An additional ATA_TIMING_SELECT register for Device 1.

3. Device 0 shall be accessed and addressed as logical unit 0 of the tailgate. Device 1 shall be accessed and addressed as logical unit 1 of the tailgate. The use of separate logical units for each device has several implications:

   – Each logical unit must have its own fetch engine and thus an individual ORB chain.

   – The configuration ROM must indicate that the tailgate supports two logical units.

The initiator must login to each logical unit individually and they must have unique login IDs. Each logical unit has a unique set of command agent registers. The login process will return a unique base address for the fetch agent registers for each logical unit.