SAE Aerospace
An SAE International Group

AEROSPACE STANDARD

SAE AS4710

Issued 1993-05
Reaffirmed 2006-07

PI-BUS

RATIONALE

This document has been reaffirmed to comply with the SAE 5-year Review policy.

FOREWORD

(This foreword is not part of the SAE requirements for 4710, SAE AS Pi-Bus Standard.)

As part of the VHSIC Phase 2 program, IBM, TRW, and Honeywell developed a specification for a 16 or 32-bit parallel backplane communication bus. This specification was the VHSIC Phase 2, Interoperability Standard PI-Bus Specification 2.2. The Joint Integrated Avionics Working Group (JIAWG) adopted the Pi-Bus as its "standard" backplane communications bus. However, by examining the various Pi-Bus implementations occurring during the initial phases of JIAWG, it was determined that the VHSIC Pi-Bus Specification needed some clarification to better support interoperability and some modifications to support JIAWG requirements. The JIAWG formed a Pi-Bus Working Group whose goal was to develop a JIAWG Pi-Bus specifications by enhancing/modifying the VHSIC Pi-Bus Specification. The membership of the JIAWG Pi-Bus Working Group happened to also be a subset of the SAE AS-2C1 Pi-Bus Working Group. Since the VHSIC Pi-Bus Specification was not being maintained and since the Pi-Bus had applicability beyond JIAWG, the two working groups met in concert with a goal of first creating a JIAWG Pi-Bus Specification and then transitioning this specification to become an SAE standard. This standard represents over 4 years of effort by a number of dedicated individuals to achieve this goal.

This standard was prepared by the SAE AS-2C1 Pi-Bus Working Group. There were in excess of 25 individuals who supported this effort over the 4 years of its development. At the time the standard was approved, the active membership of the working group was as follows:

Chuck Roark, Chairman
Fred Jackson, Vice Chairman
Paul Bowen, Secretary
Dave Dzatko
John Libby
Dennis Kalajainen
Tracy Markie
Denise Kelley-Jones
Tom McDermott
Sheila Kline

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF ERROR TABLES

LIST OF FIGURES

1. SCOPE:

This document is a result of the desire for interoperability of modules on a Pi-Bus. This standard is a stand alone document that used the Very High Speed Integrated Circuit (VHSIC) Phase 2, Interoperability Standard PI-Bus Specification 2.2, as a starting point.

1.1 Purpose:

This standard defines a bus and module interface to the Pi-Bus. This standard specifies the requirements/functions for the Pi-Bus side of a Bus Interface Unit (BIU) needed to facilitate the interoperability of modules on a Pi-Bus for avionics systems. The corresponding software/device side interface requirements are not specified in this standard.

1.2 Field of Application:

The Pi-Bus is intended to provide a Master-Slave communications path for transferring digital messages between a set of up to 32 modules residing on a single backplane.

1.3 Classification:

Bus configurations and modules which conform to this standard are specified in Table 1:

TABLE 1 - Bus Configuration Types

Type	Class	Mode	Feature	Description
16	ED	--	SO	16 bit data transfer, error detecting, Slave only operation
16	ED	--	M&S	16 bit data transfer, error detecting, Master and Slave only operation
32	EC	Nonmixed	SO	Operate as EITHER a 32 bit data transfer, error correcting, OR 16 bit data transfer, error detecting, Slave only operation
32	EC	Nonmixed	M&S	Operate as EITHER a 32 bit data transfer, error correcting, OR 16 bit data transfer, error detecting, Master and Slave only operation
32	EC	Mixed	SO	Operate as BOTH a 32 bit data transfer, error correcting, AND 16 bit data transfer, error detecting, Slave only operation
32	EC	Mixed	M&S	Operate as BOTH a 32 bit data transfer, error correcting, AND 16 bit data transfer, error detecting, Master and Slave only operation

1.3 (Continued):

Buses and modules shall be classified according to their maximum capabilities. Bus sequences shall be classified according to the Type or Class of transfer actually used.

All modules and buses shall be capable of operating in Type 16, Class ED mode.

1.3.1 16 ED Mode: A 16 ED module shall be configurable to operate on one of two separate 16 ED buses. The module shall be capable of participating in 16 ED Vie and message sequences.

1.3.2 32 EC Nonmixed Mode: A 32 EC nonmixed mode module shall be configurable to operate in either a 16 ED configuration or a 32 EC nonmixed mode configuration. In a 16 ED configuration, it shall operate as a 16 ED module. In the 32 EC nonmixed mode configuration, it shall operate on a single error correction bus (which contains only modules operating in 32 EC nonmixed mode) using 32 EC Vie and message sequences.

1.3.3 32 EC Mixed Mode: A 32 EC mixed mode module shall be configurable to operate in one of two configurations. The first configuration is the 32 EC nonmixed mode configuration.

In the second configuration, it shall be capable of performing 16 ED Vie, 16 ED message and 32 EC message sequences, to allow interoperation of 16 ED modules and 32 EC mixed mode modules. In this configuration, the module shall be configurable to operate on one of two separate 16 ED buses as well as a 32 EC bus. A 32 EC mixed mode module operating in the second configuration will not be interoperable with a 32 EC nonmixed mode module operating in the 32 EC configuration. This configuration shall use the format bit, DF<0>, and DC<0> to determine which message Type (i.e., 16 ED or 32 EC) sequence to perform.

2. REFERENCES:

2.1 Applicable Documents:

2.1.1 Military Documents:

VHSIC Phase 2 Ver. 2.2 15 Mar 90	Interoperability Standard PI-Bus Specification

2.2 Definitions:

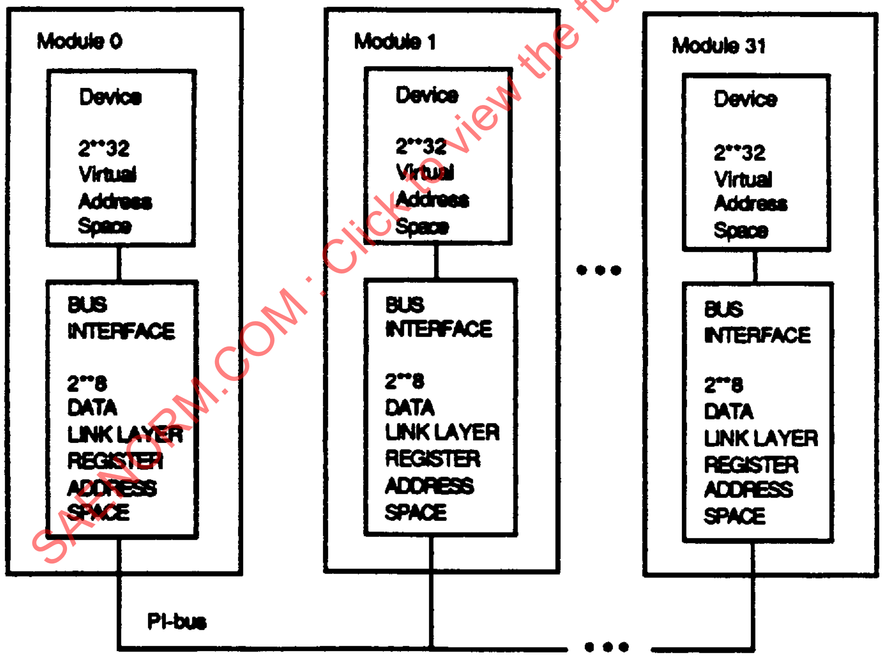
The definitions listed herein shall apply to the Pi-Bus and the Pi-Bus module.

2.2.1 Item Definitions: The Pi-Bus is a linear, multidrop communications medium which transfers datum serial, bit parallel information among up to 32 modules residing on a single backplane. The datum size may be a single word or a double word.

Pi-Bus modules are those modules which implement the Slave Only (SO) or Master and Slave portions of the Pi-Bus protocol as specified herein.

Figure 1 illustrates the Pi-Bus and Pi-Bus modules. Conceptually, each module consists of a device which performs the application specific function of the module and a Bus Interface which implements the Pi-Bus Master-Slave communications protocol.

The device portion of each module is modeled as a virtual memory space with a 32-bit address range. The Bus Interface is modeled as a separate memory space with an 8-bit Data Link register address range. A separate, 8-bit module virtual address called the Slave ID is used by the bus Master to select one or more modules to participate in a particular communications sequence as Slave(s).



MODULE VIRTUAL ADDRESS SPACE (SLAVE ID) = 2**8

- PHYSICAL SLAVE ID	0 - 31
- BROADCAST ADDRESS	32
- LOGICAL SLAVE ID	33 - 255

FIGURE 1 - Conceptual Model of Bus and Modules

2.2.2 Term Definitions: The definitions given below shall apply to the Pi-Bus and Pi-Bus modules.

//: The concatenation operator for groups of bits.

ACTIVE BUS INTERFACE: A Bus Interface that is connected to the bus media, and is currently capable of (and not inhibited from) participating in bus transactions.

ARBITRATION: The process by which a single bus Master is selected from competing potential bus Masters.

ASSERT (SIGNAL): The action of changing the state of a bus signal line from released, logic 0, to asserted, logic 1, or of ensuring that the line remains in the asserted state.

ASSERTED (SIGNAL): The logic 1 state of a bus signal line. The least positive of the two states of a bus signal line.

BACKPLANE: A motherboard comprising wiring for the bus and connectors to the modules attached to the bus.

BROADCAST: A mode of operation where the bus Master transmits data to all modules during a single transfer.

BUS ACQUISITION LATENCY: The time from the highest priority module’s request for bus Mastership to the time at which that module becomes bus Master.

BUS INTERFACE: This is a synonym for active Bus Interface.

BUS MASTER: The module currently in control of the bus.

BUS TENURE: The period of time between the time a bus Master gains control of the bus and the time at which control is released. Control is released when the bus is returned to Idle with no wait states asserted.

CONTENDER: A potential bus Master module which is actively vying for bus Mastership.

DATUM: A unit of information that matches the width of the Data Line Group.

DEVICE: The portion of a module, excluding the Bus Interface, which does the application dependent function of the module.

DOUBLE WORD: An ordered set of 32-bits operated on as a pair of words or as a single unit. The most significant bit of a double word is labeled bit 31 and the least significant bit is labeled bit 0.

LINEAR BUS: A bus with a single shared medium segment.

2.2.2 (Continued):

MESSAGE: A set of sequences starting with a header and terminating when all bus actions indicated by the header have been performed.

MODULE: An entity which is addressable via the bus and has a single connection to the bus.

MULTICAST: A mode of operation where the Master transmits data to more than one Slave during a single transfer.

NONTRANSFER CYCLE: A bus cycle that immediately follows a bus cycle in which a valid Wait is asserted or one of the VZ, HZ, or DZ cycles specified in the protocol. Information on the Data lines is not used during a nontransfer cycle.

PARTIAL MESSAGE: A sequence starting with a header and terminating prematurely due to a Suspend, Abort, or other exception indication prior to a normal completion.

POST (SYMBOL): The action taken to assert and/or release individual bus lines within one particular group of bus lines, such that either the associated symbol appears on the group or another symbol appears that is the result of individual line ORing of simultaneously posted symbols.

RELEASE (SIGNAL): The action of ceasing to assert a logic 1 on a bus signal line. The action of releasing a signal line produces a change in the state of the signal line only if no module is asserting that signal.

RELEASED (SIGNAL): The logic 0 state of a bus signal line produced when no module asserts the signal associated with that line. The more positive of the two states of a bus signal line relative to the 0 volt logic reference.

SEQUENCE: A transaction comprising a number of ordered transfers performing one intended function.

SLAVE: A module which is selected by the bus Master to participate in a message sequence.

SYMBOL: A unit of information on a particular group of bus lines, as represented by a particular binary encoding of bits. A valid symbol is one which conforms to the signal definitions herein: including correct parity, Hamming encoding or redundant coding, as applicable.

TRANSFER: A set of elemental operations on the bus which results in the communication of a bit parallel datum unit between the current bus Master and the selected Slave(s). The datum unit is either 16-bits (Type 16) or 32-bits (Type 32). See Sequence.

WORD: An ordered set of 16-bits operated on as a unit. The most significant bit is labeled bit 15 and the least significant bit is labeled bit 0.

2.3 Abbreviations and Acronyms:

A	- Acknowledge
AB	- Abort
ABn	- Last Abort Cycle n
ACK	- Acknowledge
AS	- Acknowledge Line Set
AT	- Access Type
AWM0	- Multiple Slave Status Word
AWM1..4	- Multiple Slave Acknowledge Words
AWS	- Single Slave Acknowledge Word
AWT	- Acknowledge Word Type
B	- Busy
BIT	- Built-in Test
BIU	- Bus Interface Unit
BR	- Bus Request Lines
CT	- Cycle Type
CTC	- Cycle Type Check
D	- Data Cycles or Data Lines
DA0	- Data Acknowledge Word O
DAn	- Last Data Acknowledge Cycle
DC	- Data Check (Lines)
DF	- Data Format Line
Dn	- Last Data Cycle
DZ	- Decision Cycle After Data
EC	- Error Correction
ED	- Error Detection
EH	- Extended Header
F	- Format
H	- Header
HAn	- Last Header Acknowledge Cycle
HA0	- Header Acknowledge Word O
Hn	- Last Header Cycle
H0	- Header O
HWA	- Header Word A
HWB	- Header Word B
HWC0	- Header Word C0
HWC1	- Header Word C1
HZ	- Decision Cycle After Header
ID	- Identification
Io1	- Low-level Output Sink Current
MID	- Module Identification
MIP	- Module Identification Parity
MS	- Master Slave
NAK	- Negative Acknowledgment
NS	- Not Selected
NT	- Nontransfer
RC	- Resume Control
RCG	- Recognize
RCW	- Resume Control Word
RCW0	- Resume Control Word 0
RCW1	- Resume Control Word 1
S	- Suspend

2.3 (Continued):

SH	- Short Header
SO	- Slave Only
SS	- Single Slave
SSR	- Single Slave Read
SSW	- Single Slave Write
Tf	- Fall Time
Tr	- Rise Time
Tpdlh	- Minimum and Maximum Propagation Delay
V	- Vie
VC0	- Aggregate Vie Code
VHSIC	- Very High Speed Integrated Circuit
Vih	- High Level Signal Line Input Voltage
Vil	- Low Level Signal Line Input Voltage
Vn	- Vie Cycle n
Vol	- Low Level Output Voltage
VP0..8	- Vie Priority Code Bits
VZn	- Decision Cycle for Vie Cycle n
W	- Wait (Lines)

3. PHYSICAL LAYER:

3.1 Introduction:

The physical layer of the Pi-Bus is specified herein. Signal lines required to implement the Pi-Bus are defined, including those used for signal line error detection and correction. The electrical characteristics of the module interfaces and backplane are specified and timing definitions are presented.

3.2 Line Definitions:

The Pi-Bus signal, clock, and Module Identification (MID) lines are defined in this section. In addition, the encoding used to achieve signal line error detection and correction is specified by definition of the valid symbols allowed for each signal line group.

3.2.1 Nomenclature: Lines shall be designated by name or by capital letter abbreviations, e.g., Data or D. Where a set of related lines are represented by the same name, the lines within the set shall be differentiated by number with the least significant bit numbered 0. The nomenclature for signal lines shall be the letter abbreviation for the line name followed by the bit number enclosed in < >, e.g., D<0>. The nomenclature X<m..n> shall be the abbreviation for the set of lines Xm to Xn, inclusive, where X is the letter abbreviation for the line name and m and n are the most and least significant bit numbers, respectively. Thus, D<7..0>, represents the least significant eight Data lines. In addition, X<i,j,...,k> shall be an abbreviation for the set of lines X<i>, X<j>,...,X<k>. Thus D<5,3,1,7> stands for the set of lines D<5>, D<3>, D<1>, and D<7>.

3.2.1 (Continued):

P<x> shall designate the parity (modulo 2 sum) of the set of signal lines defined by X. Thus P(D<15..0>, DC<0>) designates the parity of the 16 bits present on the Data lines plus the Data Check Line and P(D<15..0>, DC<0>)=0 represents even parity over the specified lines.

3.2.2 Bused Signal Lines: All Pi-Bus signal lines shall be implemented as wired- or lines bused between modules on a common backplane. Modules and buses shall implement those bused signal lines specified for their particular Type, Class, and Mode by Table 2. Any implemented bus signal lines which are not required during an operation of a particular Type, Class, and Mode shall be released during that operation.

Symbols posted onto signal lines shall be valid symbols as specified in this section, except that during diagnostic bus operations, invalid symbols are allowed as specified in 4.3.9.5.

As a required device function, Pi-Bus modules shall provide a command path independent of the Pi-Bus which provides a way to force all Pi-Bus signal lines to be released by the module. This capability may be used in bus diagnostics and fault isolation. In addition, no signal line shall be asserted from the time power is applied to the Pi-Bus module until the module has completed reset as defined in 4.3.8. Modules shall not assert any signal line in violation of this specification during power failure.

For a particular Type, Class, and Mode, only those lines required in Table 2, shall be sourced.

3.2.2.1 Data Line Group (D//DC): The Data Line Group shall consist of the Data (D) lines and the Data Check (DC) lines. The Data Line Group is a set of bidirectional lines which shall transfer header, data, and acknowledge information between the bus Master and the Slave(s). The Data Line Group shall also be used to resolve priority during a Vie sequence.

3.2.2.1.1 Data Lines - Type 16: Type 16 modules and buses shall provide 16 Data lines, D<15..0>. D<0> shall be the least significant and D<15> the most significant line. If data to be transferred exceeds 16-bits, the most significant portion shall be transferred first.

3.2.2.1.2 D lines - Type 32: Type 32 modules and buses shall provide 32 Data lines, D<31..0>. D<0> shall be the least significant and D<31> the most significant line. Type 16 data transfers shall always use D<15..0>. If data to be transferred exceeds 32-bits, the most significant portion shall be transferred first.

TABLE 2 - Pi-Bus Signal line Requirements By Type, Class and Mode

Name	Lines Required (Yes/No) Type 16 Class ED	Lines Required (Yes/No) Type 32 Class EC Nonmixed Mode	Lines Required (Yes/No) Type 32 Class EC Mixed Mode
Data (D)	No	Yes	Yes
D<31..16>	Yes	Yes	Yes
D<15..0>			
Data Check (DC)			
DC<8>	No	No	Yes
DC<7..1>	No	Yes	Yes
DC<0>	Yes	Yes	Yes
Cycle Type (CT)			
CT<2..0>	Yes	Yes	Yes
CT Check (CTC)			
CTC<2,1>	No	Yes	Yes
CTC<0>	Yes	Yes	Yes
Acknowledge Set (AS)			
AS<5,4>	No	Yes	Yes
AS<3..0>	Yes	Yes	Yes
Wait (W)			
W<2>	No	Yes	Yes
W<1,0>	Yes	Yes	Yes
Bus Request (BR)			
BR<2>	No	Yes	Yes
BR<1,0>	Yes	Yes	Yes
Data Format (DF<0>)	No	No	Yes
TOTAL Lines Required	29	58	60

3.2.2.1.3 Error Protection for the Data Line Group: The Data line Group shall use even parity for Class ED operation and an error correcting code for Class EC operation when there is a single source for the signals. When there may be multiple sources for the signals, as during Vie cycles and during multiple Slave Acknowledges, the Data line Group shall use duplication for Class ED operation and triplication for Class EC operation. All participating modules shall operate with the same Class.

3.2.2.1.3.1 Class ED Operation:

3.2.2.1.3.1.1 Single Source: During bus cycles, in which a single source is specified for the Data lines, valid symbols for the Data line Group shall have even parity.

3.2.2.1.3.1.1.1 Type 16: The module that sources the Data lines shall also source the Data Check line, such that the set of symbols on D<15..0>//DC<0> satisfies P(D<15..0>//DC<0>) = 0. A symbol on D<15..0>//DC<0> which does not satisfy P(D<15..0>//DC<0>) = 0 shall be defined as an uncorrectable line error and shall be reported as defined in 4.3.9.2.

3.2.2.1.3.1.1.2 Type 32: Type 32, Class ED operation is not permitted.

3.2.2.1.3.1.2 Multiple Sources: During Vie and multiple Slave Acknowledge cycles, the Data and Data Check lines may have multiple sources. For those operations, modules shall post duplicate copies of the required symbols using Data lines D<15..8> and D<7..0> as duplicate line sets.

Usage of the Data lines for Vie and multiple Slave Acknowledge cycles is specified in 4.3.3.1 and 4.2.3.3.2, respectively.

During multiple Slave Acknowledge cycles a mismatch between the symbol using the Data lines D<15..8> and the symbol using Data lines D<7..0> shall be defined as an uncorrectable line error and shall be interpreted according to Table 54. During multiple Slave Acknowledge cycles, parity shall not be asserted. However, parity shall be checked during multiple Slave Acknowledge cycles and an asserted Data Check line shall be reported as an error in accordance with the Error Tables.

The assertion of a Data Check line and the detection of a data parity error during Vie shall be ignored. The interpretation and handling of Data line errors detected during Vie are defined in 4.3.3.1.

3.2.2.1.3.2 Class EC Operation:

3.2.2.1.3.2.1 Single Source: During bus cycles, in which a single source is specified for the Data lines, valid symbols for the Data line Group shall use modified Hamming encoding as specified. The Data Check lines that are not specified in the subparagraphs below, shall not be asserted or checked.

3.2.2.1.3.2.1.1 Type 16: Type 16, Class EC operation is not permitted.

3.2.2.1.3.2.1.2 Type 32 Nonmixed Mode: The module that sources the Data lines shall also source the Data Check lines, such that the set of symbols on D<31..0>//DC<7..0> satisfies:

a. P[DC<7>, D<31, 30, 25, 24, 23, 19, 15, 14, 9, 8, 7, 3>] = 0
b. P[DC<6>, D<29, 28, 27, 26, 22, 18, 13, 12, 11, 10, 6, 2>] = 0
c. P[DC<5>, D<29, 28, 27, 26, 21, 17, 15, 14, 9, 8, 5, 1>] = 0
d. P[DC<4>, D<31, 30, 25, 24, 20, 16, 13, 12, 11, 10, 4, 0>] = 0
e. P[DC<3>, D<31, 27, 23, 22, 17, 16, 15, 11, 7, 6, 1, 0>] = 0
f. P[DC<2>, D<30, 26, 21, 20, 19, 18, 14, 10, 5, 4, 3, 2>] = 0
g. P[DC<1>, D<29, 25, 23, 22, 17, 16, 13, 9, 5, 4, 3, 2>] = 0
h. P[DC<0>, D<28, 24, 21, 20, 19, 18, 12, 8, 7, 6, 1, 0>] = 0

3.2.2.1.3.2.1.3 Type 32 Mixed Mode: The module that sources the Data lines shall also source the Data Check lines, such that the set of symbols on D<31..0>//DC<8..0> satisfies:

a.	P[	DC<8>,	D<28, 24, 21, 20, 19, 18, 12, 8, 7, 6, 1, 0>	]	= 0
b.	P[	DC<7>,	D<31, 30, 25, 24, 23, 19, 15, 14, 9, 8, 7, 3>	]	= 0
c.	P[	DC<6>,	D<29, 28, 27, 26, 22, 18, 13, 12, 11, 10, 6, 2>	]	= 0
d.	P[	DC<5>,	D<29, 28, 27, 26, 21, 17, 15, 14, 9, 8, 5, 1>	]	= 0
e.	P[	DC<4>,	D<31, 30, 25, 24, 20, 16, 13, 12, 11, 10, 4, 0>	]	= 0
f.	P[	DC<3>,	D<31, 27, 23, 22, 17, 16, 15, 11, 7, 6, 1, 0>	]	= 0
g.	P[	DC<2>,	D<30, 26, 21, 20, 19, 18, 14, 10, 5, 4, 3, 2>	]	= 0
h.	P[	DC<1>,	D<29, 25, 23, 22, 17, 16, 13, 9, 5, 4, 3, 2>	]	= 0
i.	P[	DC<0>,	D<15..0>		

3.2.2.1.3.2.2 Multiple Sources: During Vie, nontransfer cycles before multiple Slave Acknowledge cycles, and multiple Slave Acknowledge cycles, the Data and Data Check lines may have multiple sources. For those operations, modules shall post triplicate copies of the required symbols using the Data and Data Check lines D<15..8>, D<7..0> and DC<7..0> as triplicate line sets.

Usage of the Data and Data Check lines for Vie and multiple Slave Acknowledge cycles is specified in 4.3.3.1 and 4.2.3.3.2, respectively.

During Vie and multiple Slave Acknowledge cycles a mismatch between the symbols using the Data lines D<15..8>, Data lines D<7..0> and Data Check lines DC<7..0> shall be correctable. The corrected symbol shall be the symbol posted on two of the three triplicate copies. An error shall be reported as defined in 4.3.9.1.

NOTE: During Vie and multiple Slave Acknowledge cycles for Type 32 EC transfers, there are no detectable uncorrectable line errors for the Data Line Group.

3.2.2.2 Cycle Type Line Group (CT//CTC): The Cycle Type Line Group shall consist of the Cycle Type (CT) and the Cycle Type Check (CTC) lines. The Cycle Type Line Group is a set of lines onto which the bus Master shall post symbols to indicate the current bus Cycle Type. The bus Cycle Types shall be encoded as shown in Table 3.

TABLE 3 - Pi-Bus Cycle Types and Valid Symbols

Cycle Type	Abbr.	Class ED Symbol CT<2..0>	Class ED Symbol CTC<0>	Class EC Symbol CT<2..0>	Class EC Symbol CTC<2..0>
Abort	AB	111	1	111	001
Acknowledge	A	011	0	011	110
Data	D	001	1	001	011
Header 0	HO	101	0	101	100
Header	H	010	1	010	101
Idle	I	000	0	000	000
Suspend	S	110	0	110	010
Vie	V	100	1	100	111

3.2.2.2.1 Class ED: The module that sources the Cycle Type lines shall also source the Cycle Type Check line, such that the set of symbols on Cycle Type<2..0>//CTC<0> satisfies P(CT<2..0>//CTC<0>) = 0. A symbol on Cycle Type<2..0>//CTC<0> which does not satisfy P(CT<2..0>//CTC<0>) = 0 shall be defined as an uncorrectable line error, interpreted according to Table 54 and reported in accordance with the Error Tables.

3.2.2.2.2 Class EC: The module that sources the Cycle Type lines shall also source the Cycle Type Check lines. The set of symbols on Cycle Type<2..0>//CTC<2..0> shall produce a Cycle Type Syndrome (CTS<2..0>) defined by:

a. CTS<2> = P(CTC<2>, CT<2,1>)
b. CTS<1> = P(CTC<1>, CT<2,0>)
c. CTS<0> = P(CTC<0>, CT<2..0>)

The CT//CTC symbols shall be interpreted as follows in Table 4:

TABLE 4 - Interpretation of Cycle Type Symbols

CTS<2..0>	Error	CT Bit in Error
000	No Error	None
001	Correctable	CTC<0>
010	Correctable	CTC<1>
011	Correctable	CT<0>
100	Correctable	CTC<2>
101	Correctable	CT<1>
110	Uncorrectable	N/A
111	Correctable	CT<2>

When an uncorrectable line error is detected, it shall be interpreted in accordance with Table 54. All errors shall be reported in accordance with the Error Tables.

NOTE: Some double bit errors will be erroneously corrected.

3.2.2.3 Acknowledge Line Set (AS): The Acknowledge Line Set is a group of lines onto which the Slave(s) or contenders shall post symbols to indicate synchronization or to signal uncorrectable detected errors. Valid symbols for the Acknowledge Line Set shall be as defined in Table 5.

TABLE 5 - Acknowledge Line Set Valid Symbols

Response	Abbr.	Class ED Symbol AS<3,2>	Class ED Symbol AS<1,0>	Class EC Symbol AS<5,4>	Class EC Symbol AS<3,2>	Class EC Symbol AS<1,0>
Acknowledge	ACK	10	10	10	10	10
Negative ACK	NAK	11	11	11	11	11
Not Selected	NS	00	00	00	00	00
Recognize	RCG	01	01	01	01	01

3.2.2.3.1 Class ED: A mismatch between the symbol using the Acknowledge lines AS<3,2> and the symbol using Acknowledge lines AS<1,0> shall be defined as an uncorrectable line error, interpreted according to Table 54 and reported in accordance with the Error Tables.

3.2.2.3.2 Class EC: A mismatch of either the triplicated signal set AS <5,3,1> or AS <4,2,0> shall be correctable. The corrected signal shall be the signal posted on two of the three triplicated signals. An error shall be reported as defined in 4.3.9.1.

3.2.2.4 Wait (W) Lines: The Wait lines shall be a set of redundant lines which the current bus Master and Slave(s) may assert to obtain extra nontransfer bus cycles to supply information to the bus or to accept information from the bus.

3.2.2.4.1 Class ED: Class ED modules and buses shall provide two Wait lines, W<1,0>, that shall operate as redundant lines. Valid symbols for this case shall be W<1,0> = 00, no wait request, and W<1,0> = 11, wait requested. The symbol W<1,0> = 01 and the symbol W<1,0> = 10 shall be uncorrectable line errors. The Wait lines shall be interpreted according to Table 54 and reported in accordance with the Error Tables.

3.2.2.4.2 Class EC: Class EC modules and buses shall provide three Wait lines, W<2..0>, that shall operate as redundant lines. Valid symbols for this case shall be W<2..0> = 000, no wait request, and W<2..0> = 111, wait requested. A mismatch between the three redundant lines shall be correctable. The correct symbol on the triplicate lines shall be the symbol posted on two of the three lines. The error shall be reported in accordance with 4.3.9.1.

3.2.2.5 Bus Request (BR) lines: The Bus Request lines shall consist of a set of redundant lines which shall be asserted to request that the current bus Master release the bus.

3.2.2.5.1 Class ED: Class ED modules and buses shall provide two Bus Request lines, BR<1,0>, that shall operate as redundant lines. Valid symbols for this case shall be BR<1,0> = 00, no Bus Request, and BR<1,0> = 11, Bus Requested. The symbol BR<1,0> = 01 and the symbol BR<1,0> = 10 shall be uncorrectable line errors. The Bus Request lines shall be interpreted according to Table 54 and reported in accordance with the Error Tables.

3.2.2.5.2 Class EC: Class EC modules and buses shall provide three Bus Request lines, BR<2..0>, that shall operate as redundant lines. Valid symbols for this case shall be BR<2..0> = 000, no Bus Request, and BR<2..0> = 111, Bus Requested. A mismatch between the three redundant lines shall be correctable. The correct symbol on the triplicate lines shall be the symbol posted on two of the three lines. The error shall be reported in accordance with 4.3.9.1.

3.2.2.6 Data Format (DF) Line: The Data Format line is only used for mixed mode operation. The Data Format line shall consist of a single line, DF<0>, which shall indicate the message type. The Data Format line shall be asserted only during Cycle Type Header O (HO).

3.2.2.6.1 Class ED: Class ED modules shall not source the Data Format line, DF<0>.

3.2.2.6.2 Class EC Nonmixed Mode: Class EC nonmixed modules shall not source the Data Format line, DF<0>.

3.2.2.6.3 Class EC Mixed Mode: Class EC mixed mode modules shall provide a Data Format line, DF<0>. Valid symbols for this case shall be DF<0> = 0, Type 16 ED message transfer, and DF<0> = 1, Type 32 EC message transfer.

3.2.3 Bus Clock: Bus Clock shall be a single phase clock. All bus timing shall be referenced to the high-to-low transition of the Bus Clock. The generation and distribution of Bus Clock is beyond the scope of this specification. However, the period of Bus Clock shall be selected to guarantee that bus timing constraints are satisfied for the bus delays and clock skews resulting from the backplane design.

3.2.4 Module Identification (MID): Pi-Bus modules shall have inputs for a set of lines that shall be hardwired on the Pi-Bus backplane to provide a unique module identification.

3.2.4.1 Module Identification (MID) Lines: The set of 5 MID lines shall be hardwired on the backplane and shall be used by the module as the module's physical identification code. The identification codes shall consist of an unsigned binary number in the range of 0-31, inclusive, encoded in MID<4..0>.

3.2.4.2 Module Identification Parity (MIP) Line: A MIP line shall be hardwired on the backplane, such that MID<4..0>//MIP<0> satisfies P(MID<4..0>//MIP<0>) = 1.

3.2.5 32-Bit EC/Dual 16-Bit ED Interoperability: A 32-bit EC module shall be capable of operating in the 16 ED Class on either of two buses (primary or backup). The two 16-bit Pi-Buses and the 32-bit Pi-Bus signals shall be paired as shown in Table 6. Interpretation of the backpanel signals by 32 EC modules depends on whether the primary or backup 16 ED bus is selected. If the 16 ED primary bus is selected, the second and fifth columns (where column one is the far left column) of Table 6 define the mapping of 32 EC signals to the 16 ED primary and backup signals, respectively. If the 16 ED backup bus is selected, the third and sixth columns of Table 6 define the mapping of 32 EC signals to the 16 ED primary and backup signals, respectively.

3.2.6 Signal Driver Partitioning: The ability to detect a failure in a Pi-bus transceiver when all of the transceivers in a package fail depends upon which signals use the package. The partitioning of the 32-bit Pi-Bus signals shown in Table 7 shall be used for octal transceiver implementations. This partitioning insures that a failure of all bits within an octal transceiver will be detected.

Signal partitioning with other than octal driver packages should consider the error detection effects of completely failing transceiver packages. For example, signals cannot be partitioned in a 16-bit package such that a failing package can always be detected.

TABLE 6 - Pi-Bus Signal Pairing

Primary 16-Bit Signal	32-Bit Signal Primary Selected	32-Bit Signal Backup Selected	Backup 16-Bit Signal	32-Bit Signal Primary Selected	32-Bit Signal Backup Selected
P-D<15>[1]	D<15>	D<19>	B-D<15>[2]	D<19>	D<15>
P-D<14>	D<14>	D<18>	B-D<14>	D<18>	D<14>
P-D<13>	D<13>	D<17>	B-D<13>	D<17>	D<13>
P-D<12>	D<12>	D<16>	B-D<12>	D<16>	D<12>
P-D<11>	D<11>	D<31>	B-D<11>	D<31>	D<11>
P-D<10>	D<10>	D<30>	B-D<10>	D<30>	D<10>
P-D<09>	D<09>	D<29>	B-D<09>	D<29>	D<09>
P-D<08>	D<08>	D<28>	B-D<08>	D<28>	D<08>
P-D<07>	D<07>	D<27>	B-D<07>	D<27>	D<07>
P-D<06>	D<06>	D<26>	B-D<06>	D<26>	D<06>
P-D<05>	D<05>	D<25>	B-D<05>	D<25>	D<05>
P-D<04>	D<04>	D<24>	B-D<04>	D<24>	D<04>
P-D<03>	D<03>	D<23>	B-D<03>	D<23>	D<03>
P-D<02>	D<02>	D<22>	B-D<02>	D<22>	D<02>
P-D<01>	D<01>	D<21>	B-D<01>	D<21>	D<01>
P-D<00>	D<00>	D<20>	B-D<00>	D<20>	D<00>
P-DC<0>	DC<0>	DC<3>	B-DC<0>	DC<3>	DC<0>
P-CT<2>	CT<2>	DC<6>	B-CT<2>	DC<6>	CT<2>
P-CT<1>	CT<1>	DC<5>	B-CT<1>	DC<5>	CT<1>
P-CT<0>	CT<0>	DC<4>	B-CT<0>	DC<4>	CT<0>
P-CTC<0>	CTC<0>	BR<2>	B-CTC<0>	BR<2>	CTC<0>
P-AS<3>	AS<3>	CTC<1>	B-AS<3>	CTC<1>	AS<3>
P-AS<2>	AS<2>	DC<2>	B-AS<2>	DC<2>	AS<2>
P-AS<1>	AS<1>	DC<7>	B-AS<1>	DC<7>	AS<1>
P-AS<0>	AS<0>	CTC<2>	B-AS<0>	CTC<2>	AS<0>
P-W<1>	W<1>	AS<4>	B-W<1>	AS<4>	W<1>
P-W<0>	W<0>	DC<1>	B-W<0>	DC<1>	W<0>
P-BR<1>	BR<1>	W<2>	B-BR<1>	W<2>	BR<1>
P-BR<0>	BR<0>	AS<5>	B-BR<0>	AS<5>	BR<0>
---	DF<0>	DC<8>	---	DC<8>	DF<0>

[1] P - Primary Bus signal
[2] B - Backup Bus signal

TABLE 7 - Signal Partitioning

Xcvr 1	Xcvr 2	Xcvr 3	Xcvr 4	Xcvr 5	Xcvr 6	Xcvr 7	Xcvr 8
D<31>	D<27>	D<23>	D<19>	D<15>	D<11>	D<07>	D<03>
D<30>	D<26>	D<22>	D<18>	D<14>	D<10>	D<06>	D<02>
D<29>	D<25>	D<21>	D<17>	D<13>	D<09>	D<05>	D<01>
D<28>	D<24>	D<20>	D<16>	D<12>	D<08>	D<04>	D<00>
DC<08>	DC<07>	DC<04>	DC<03>	DC<00>	AS<02>	AS<01>	DF<00>
DC<06>	DC<05>	CTC<02>	CTC<01>	AS<03>	CT<02>	CT<01>	AS<00>
AS<05>	AS<04>	DC<02>	BR<02>	CTC<00>	BR<00>	W<01>	CT<00>
--	--	DC<01>	W<02>	BR<01>	--	--	W<00>

3.3 Electrical Requirements:

Electrical characteristics for the Pi-Bus backplane and modules shall be as specified herein.

3.3.1 Backplane Requirements:

3.3.1.1 Bus Signal line Characteristic Impedance: Pi-Bus signal lines shall have a characteristic impedance of not less than 20 Ω and not more than 50 Ω for all operating and module loading conditions. The characteristic impedance of less than 50 Ω shall apply to any segment of a bus signal line in a backplane which has all the module mating connectors attached but no modules installed. The characteristic impedance of not less than 20 Ω shall apply to any segment of a bus signal line in a backplane which has all modules installed and each module exhibits the maximum input capacitance as specified in 3.3.2.2.1.1.

The maximum round trip bus signal line propagation time measured at the module connector pin shall be less than or equal to 20.0 ns in a backplane which has all modules installed and each module exhibits the maximum input capacitance as specified in 3.3.2.2.1.1.

NOTE: This round trip signal propagation time allows any incident signal edge reflections to propagate back to the source by the end of the current bus cycle. Hence, correct signal operation is not required on incident signal edges. This also greatly reduces the effects of current bus signal reflections upon the next bus cycle.

3.3.1.1.1 Example Bus Analysis:

a. Definitions:

(1) CO = Capacitance/unit length of backplane stripline without connectors or mounting vias

(2) LO = Inductance/unit length of backplane stripline

(3) CV = Capacitance of connector mounting via

(4) CC = Capacitance of mating connector pin

(5) CA = Added loading capacitance/unit length

(6) S = Module spacing

(7) ZO = SQRT(LO/CO) = Characteristic impedance of backplane stripline

(8) PO = SQRT(LO X CO) = Propagation constant of backplane stripline

(9) ZA = ZO/(SQRT(1 + CA/CO)) = Characteristic impedance of loaded backplane

3.3.1.1.1 (Continued):

(10) PA = PO X (SQRT(1 + CA/CO)) = Propagation constant of loaded backplane

(11) ZU = Unloaded backplane characteristic impedance (backplane stripline plus vias and mating connectors)

(12) ZL = Fully loaded backplane characteristic impedance (unloaded backplane plus modules)

3.3.1.1.2 Analysis: Typical values for a stripline in an epoxy-glass backplane are:

a. CO = 2.46 picofarads/inch
b. ZO = 65 Ω
c. PO = 0.16 ns/in
d. CV = 0.4 picofarads

Module mating connectors are mounted on 0.6 in centers and the connector capacitance is 0.6 picofarads.

a. ZU = 65/SQRT(1 + 1.0/(0.6 X 2.46)) Ω = 50.2 Ω
b. ZL = 65/SQRT(1 + (1.0 + 13.2)/(0.6 X 2.46))) Ω = 19.9 Ω
c. PL = 0.16 X SQRT(1 + (1.0 + 13.2)/(0.6 X 2.46))) ns/in = 0.52 ns/in

This analysis shows that the bus impedance is approximately met, however, manufacturing tolerances have not been taken into account.

3.3.1.2 Bus Signal Line Termination: Signal lines shall be terminated at each electrical end of the bus to a circuit which is the Thevenin-equivalent of a terminating resistor in series with a voltage source of not less than +1.9 volts nor more than +2.1 volts. The value of the terminating resistance shall be between 30 and 40 Ω, inclusive.

3.3.1.3 Bus Signal Line Resistance: The series resistance for backplane signal lines shall be limited, such that the maximum voltage rise from any asserted module output to the terminating resistance at either end of the backplane is less than 100 mV.

3.3.1.4 Module Identification Line Resistance: The resistance of the grounded MID and MIP lines, with respect to the signal ground, shall be less than 10 Ω.

3.3.1.5 Bus Clock Requirements: The following characteristics are measured at the input to the using module, not at the source.

3.3.1.5.1 Voltage Levels: The low voltage for Bus Clock shall be less than or equal to +0.55 volts. The high voltage for Bus Clock shall be greater than or equal to +2.4 volts.

3.3.1.5.2 Rise and Fall Time: The rise time (Tr) of the Bus Clock from 0.8 to 2.0 volts shall be less than 5 ns. The fall time (Tf) of the Bus Clock from 2.0 to 0.8 volts shall be less than 5 ns.

3.3.1.5.3 Duty Cycle: The ratio of the Bus Clock high state duration to the Bus Clock period measured at 1.5 volts, shall not be less than 0.45, nor greater than 0.55.

3.3.2 Module Requirements: The following characteristics are measured at the pin of the module under test.

3.3.2.1 Bus Clock Requirements:

3.3.2.1.1 DC Requirements: All Data Check characteristics must be met whether the module is powered on or off.

3.3.2.1.1.1 Input Capacitance: Bus Clock capacitance to logic ground shall be less than 22 picofarads.

3.3.2.1.1.2 Input Inductance: Bus Clock series inductance from the module input to the receiver of the signal shall be less than 27 nanohenries.

NOTE: Inductance is specified to limit the overshoot of signals asserted on the Pi-Bus.

3.3.2.1.1.3 Bus Clock Current: The maximum current supplied by the module, when the clock input voltage is +0.55 volts, shall be 1.6 mA. The maximum current into the module, when the Bus Clock voltage is +2.4 volts, shall be less than 100 µA.

3.3.2.1.1.4 High-level Input Voltage: A Bus Clock input voltage of +2.0 volts or more shall be interpreted as a high level.

3.3.2.1.1.5 Low-level Input Voltage: A Bus Clock input voltage of +0.8 volts or less shall be interpreted as a low level.

3.3.2.1.2 AC Requirements: Modules shall operate correctly with the Bus Clock characteristics specified in 3.3.1.5.

The maximum Bus Clock frequency supported by the modules shall be 12.504 MHz and the minimum Bus Clock frequency shall be 0 Hz. The clock distribution shall provide a clock skew of less than or equal to 4.0 ns when measured at the clock input pins of any two modules operating on the bus. Correct module operation at 0 Hz Bus Clock frequency shall be demonstrated by testing with a Bus Clock frequency of not greater than 50 Hz.

All Pi-Bus timing shall be referenced to the high-to-low transition of Bus Clock through a voltage of 1.5 volts.

3.3.2.2 Signal Line Requirements:

3.3.2.2.1 DC Requirements: All Data Check characteristics must be met whether the module is powered on or off.

3.3.2.2.1.1 Input Capacitance: Signal line capacitance to logic ground shall be less than 22.0 picofarads for module spacing of 1 in or greater and shall be less than (22.0 picofarads/in) x (module spacing) for module spacing less than 1 in. The module spacing is the center to center backplane mounting distance specified for the module. (See 3.3.1.1 for detail analysis.)

The maximum round trip signal propagation delay, measured from the module pin to the transceiver pin and back to the module pin, shall be less than or equal to 1.0 ns. [This makes the maximum round trip propagation delay to be about (1/3) x (the minimum driver rise or fall time) as specified in 3.3.2.2.2.2.2].

3.3.2.2.1.2 Input Inductance: Signal line series inductance from the module input to the driver or receiver of the signal shall be less than 27 nH.

3.3.2.2.1.3 Leakage Current: Over the input voltage range of +0.3 to +2.1 volts, the absolute value of the output current, for any signal line which is not being asserted by the module, shall be less than 100 µA.

3.3.2.2.1.4 Low-level Sink Current: The low-level output sink current (Iol) drive capability for signal lines shall be greater than 95 mA at an output voltage of 1.15 volts.

3.3.2.2.1.5 High-level Output Voltage: The high-level output voltage shall be determined by the backplane signal line termination voltage which is +1.9 to +2.1 volts. The signal line outputs shall permit wired-OR operations on the bus.

3.3.2.2.1.6 Low-level Output Voltage: The low-level output voltage (Vol) for signal lines shall be less than 1.15 volts at an input current of 95 mA.

3.3.2.2.1.7 High-level Input Voltage: A signal line input voltage (Vih) of +1.6 volts or more shall be interpreted as a logic 0. A signal line input which is not electrically connected to the backplane (i.e., an open line) shall be interpreted as logic 0.

3.3.2.2.1.8 Low-level Input Voltage: A signal line input voltage (Vil) of +1.45 volts or less shall be interpreted as logic 1.

3.3.2.2.2 AC Requirements:

3.3.2.2.2.1 Signal Line Inputs: Figure 2 illustrates the timing relationships specified below.

3.3.2.2.2.1.1 Setup Time: The maximum time that each input signal is required to be uniquely above or below the input voltage threshold for a logic 0 or logic 1 prior to the high-to-low transition of the clock (setup time, Ts) shall be 15.0 ns.

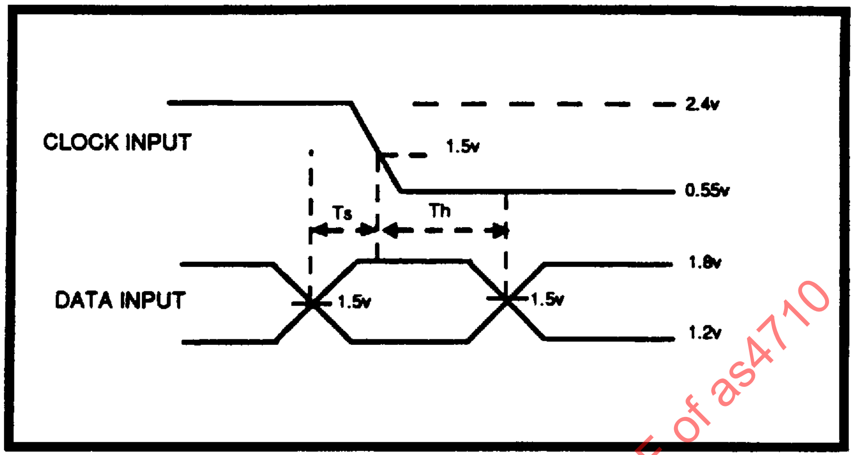


FIGURE 2 - Set-up and Hold Timing

3.3.2.2.2.1.2 Hold-time: The maximum time that each input signal is required to be uniquely above or below the input voltage threshold for a logic 0 or logic 1 following the high-to-low transition of the clock (hold time, Th) shall be 10.0 ns and shall not exceed the minimum propagation delay time of the module.

3.3.2.2.2.1.3 Noise Rejection: The input signal lines shall reject and the Bus Interface shall not respond to any signal pulse whose width, as measured between 1.5 volts on the low-to-high transition and 1.5 volts on the high-to-low transition, is less than 4 ns.

3.3.2.2.2.2 Signal Line Outputs: The following specifications shall apply when the signal line is connected to the test circuit of Figure 3. Unless specified elsewhere, all measurements shall be taken at the module connector pin.

3.3.2.2.2.2.1 Propagation Delay: Propagation delay shall be measured with respect to high-to-low transition of Bus Clock as illustrated in Figure 4. The reference clock voltage for timing shall be +1.5 volts. The reference signal voltage for timing shall be +1.5 volts.

The minimum and the maximum propagation delay (Tpdlh) for an output signal changing from a logic 1 (low voltage) to a logic 0 (high voltage) shall meet the following limits:

a. 40 ns maximum
b. 14 ns minimum

The minimum and the maximum propagation delay (Tpdhl) for an output signal changing from a logic 0 (high voltage) to a logic 1 (low voltage) shall meet the following limits:

a. 40 ns maximum
b. 14 ns minimum

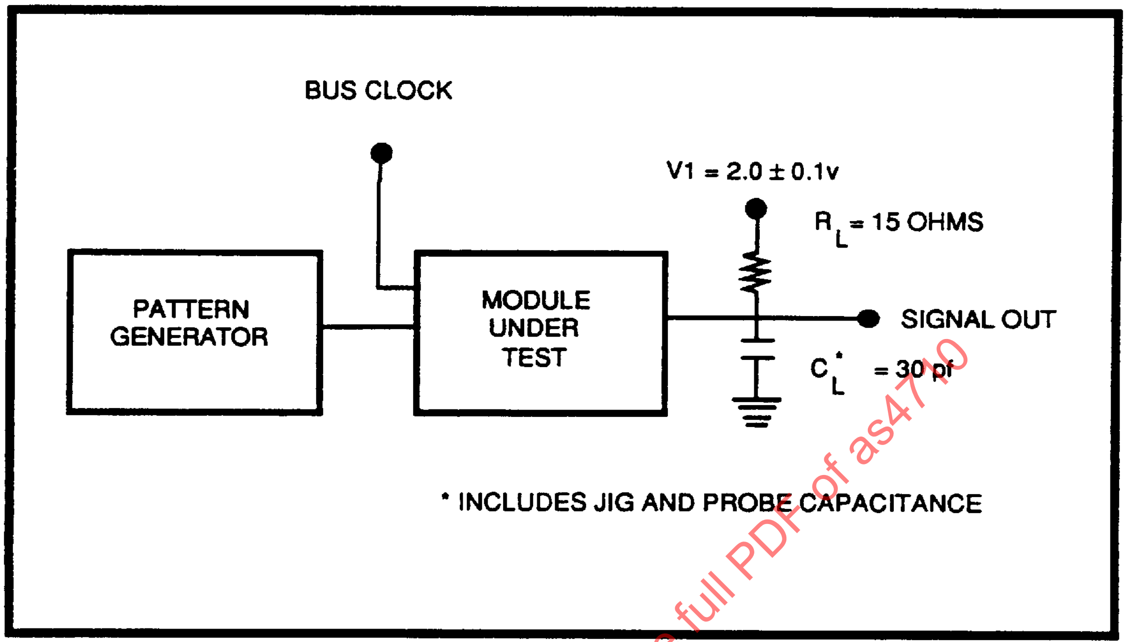


FIGURE 3 - Signal Output Test Circuit

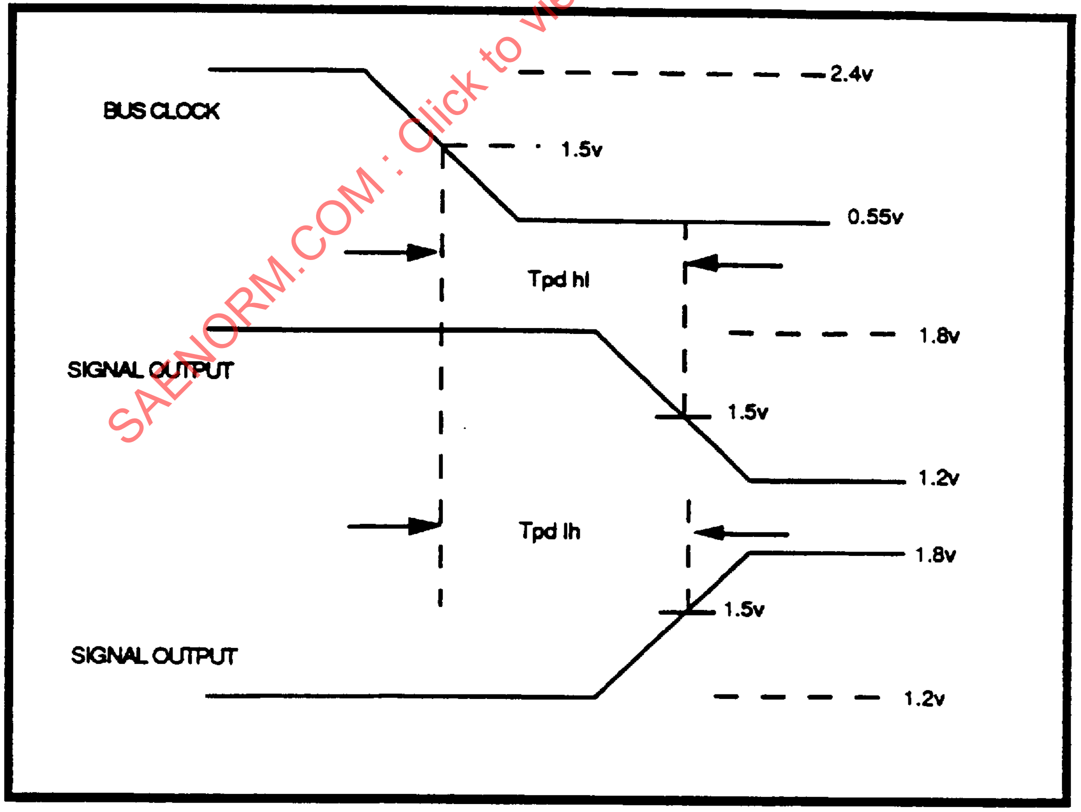


FIGURE 4 - Output Signal Timing

3.3.2.2.2.2.2 Rise and Fall Time: The rise time of an output signal from +1.2 to +1.8 volts shall be greater than 2.8 ns and less than 9 ns.

The fall time of an output signal from +1.8 to +1.2 volts shall be greater than 2.8 ns and less than 9 ns.

3.3.2.3 MID and MIP lines: A binary 1 shall be represented by a connection to signal ground and a binary 0 shall be represented by an open circuit. Modules shall incorporate any circuits they require to sense the MID and MIP lines. The absolute current into a grounded MID or MIP line shall be less than 1 mA. The maximum voltage that shall exist on an open MID or MIP line shall not exceed 5.5 volts.

4. DATA LINK LAYER:

4.1 Introduction:

The Data Link layer of the Pi-Bus is specified herein. The general protocol used by the Pi-Bus is defined through specification of the protocol state transitions and the generic message sequence. Detailed requirements for the protocol and communications sequences are specified by defining each sequence and the rule associated with the Pi-bus protocol. Responses to exception conditions are defined.

4.2 General Requirements:

4.2.1 Introduction: The Pi-Bus uses a Master-Slave protocol under which communications sequences are defined for the following:

a. Transferring messages between modules
b. Changing bus Mastership

The Pi-Bus communications sequences are listed in Table 8. The Vie sequence shall be performed only when there is no current bus Master. All other sequences shall be performed under the control of the current bus Master.

The Pi-Bus uses a set of protocol state transitions to define and control the communication sequences. Protocol state transitions shall be signaled on the Cycle Type (CT) lines and shall be controlled by the bus Master.

NOTE: Protocol states are not necessarily defined by a single unique Cycle Type symbol. For example, the Header protocol state is indicated by the HO, H and HZ Cycle Type symbols.

The Slave(s) shall operate in synchronization with the bus Master and shall signal compliance with protocol state transitions using the Acknowledge Set (AS) lines. Slave(s) shall also use the AS lines to notify the bus Master of any uncorrectable line errors that are detected.

TABLE 8 - Pi-Bus Communications Sequences

Sequence Type	Function
Mastership Sequences:	
Vie	Assigns bus Mastership to the highest priority module contending for Mastership through arbitration.
Message Sequences:	
Parameter Write	Transfer a parameter consisting of three words from the bus Master device to the Slave device(s).
Block Message	Transfer up to 65,536 datum units from Slave device to Master device or from Master device to Slave(s). Master sends a 32-bit address and may send 6 other Header words. May be used to continue a suspended message.
Datagram Message	Transfer up to 65,536 datum units from Master device to Slave(s). Master sends a 32-bit address and may send 6 other Header words. May be used to continue a suspended message. There is no Header Acknowledge and Data Acknowledge is optional.
Bus Interface Message	Transfers up to 256 words from Slave Bus Interface to the Master device or from Master device to Slave Bus Interface(s). Master provides an 8-bit address.
Exception Sequences:	
Suspend	Suspends a Block Message data sequence and transfers Resume Control Words as required from the Slave to the Master.
Abort	Abnormally terminates current sequence.

4.2.1 (Continued):

The eight protocol states defined for the Pi-Bus protocol are summarized in Table 9. Within each protocol state, bus states are defined to distinguish individual bus cycles. The specific sequences of bus states required to perform Pi-Bus communications are defined under 4.3. In this section, general requirements for the overall operation of the Pi-bus are specified by reference to the protocol states and the generic message protocol they support.

4.2.2 Protocol State Transitions: The protocol states which shall be used in Pi-Bus operations are illustrated in Figure 5. All state transitions shall occur on the high-to-low transition of Bus Clock. The allowable transitions between protocol states are specified in the sections below and in Figure 5.

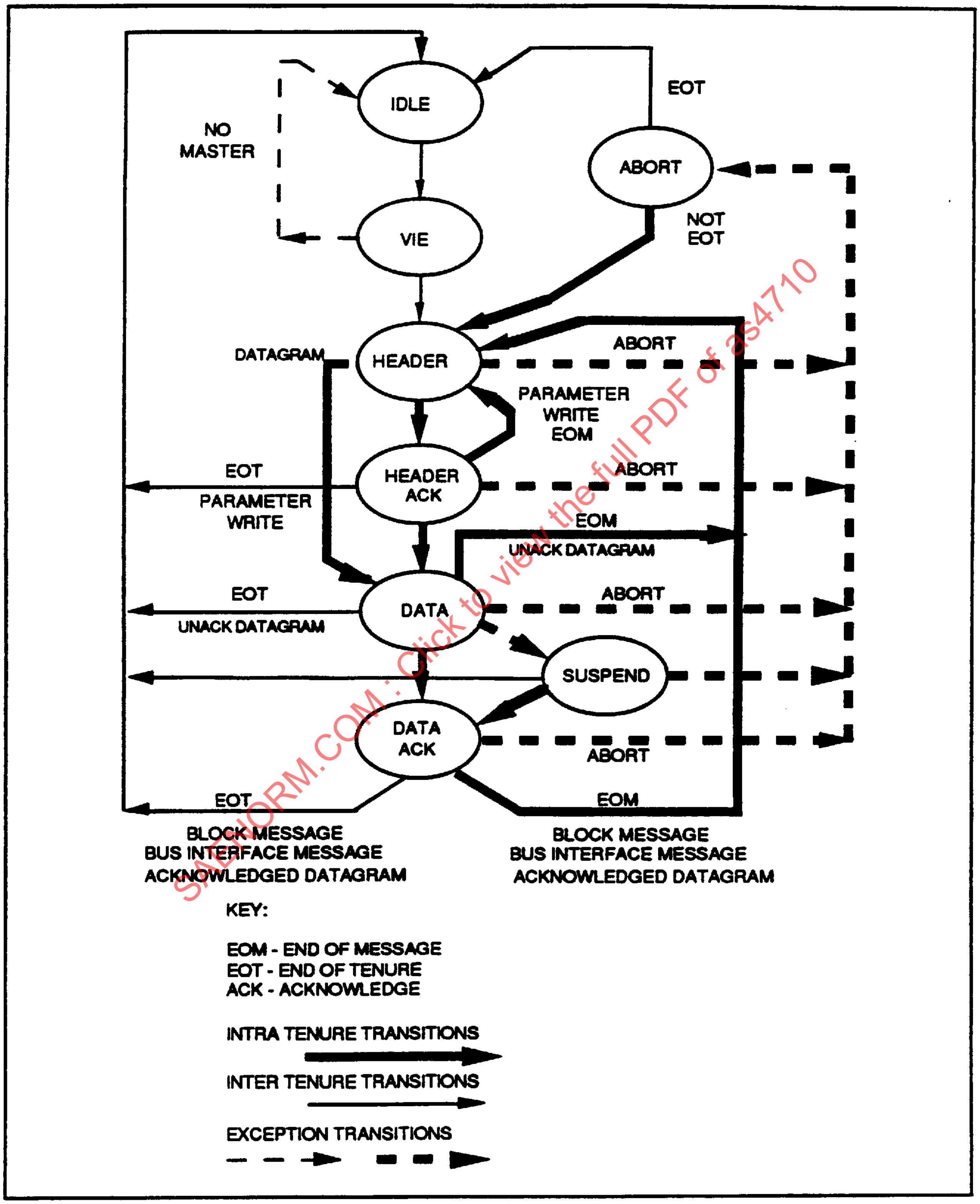


FIGURE 5 - Pi-Bus Protocol State Diagram

TABLE 9 - Pi-Bus Protocol States

Protocol State	Function
Idle	Bus not in use and no current bus Master defined.
Vie	State following Idle. Used to select the next bus Master from one or more contending modules. The module with the highest priority is selected as the next bus Master.
Header	State in which information is transmitted by the Master to specify the type of message sequence, identify modules to participate as Slaves and specify additional application dependent information.
Header Acknowledge (Header Ack)	State following Header during which Slave module(s) provide message status information to the Master. This state is not applicable to Datagram Messages.
Data	State during which data are transferred between Slave module(s) and the Master for Block Messages, Bus Interface Messages, and Datagram Messages.
Data Acknowledge (Data Ack)	State following Data during which Slave module(s) provide message status information to the Master. This state is optional for Datagram Messages.
Abort	State used to abnormally terminate another bus sequence.
Suspend	State following Data for which Block Message and Datagram Message Suspend sequences are performed. In this state, Data is transferred between Slave module(s) and the Master.

4.2.2.1 Idle: The bus shall enter the Idle state whenever all Cycle Type lines are released during a transfer cycle. There shall be no bus Master during Idle and the current bus Master priority code shall be undefined. Idle shall consist of two or more consecutive bus cycles in which the Cycle Type lines are released. No Pi-Bus operations shall be performed during Idle, except that the symbol NAK (Negative Acknowledgment) may be posted on the AS lines as specified in 4.2.3.1.2.2. Vie shall be the only valid successor state to Idle. The Idle state shall be terminated and the Vie state entered only when one or more modules post the symbol V on the Cycle Type lines.

4.2.2.2 Vie: The Vie state shall consist of eight bus cycles which shall be used to select the next bus Master from one or more contenders. The Vie state shall be succeeded by the Header state, except that if no bus Master is selected due to erroneous operation, the bus shall return to the Idle state.

4.2.2.3 Header: A bus Master’s tenure shall begin when the Header state is entered from the Vie state. The current bus Master’s tenure shall continue when the Header state is entered from the Header Acknowledge state of the Parameter Write sequence, from the Data Acknowledge state or from the Abort state. During the Header state, the bus Master shall transmit header information across the bus on two or more bus cycles.

The header shall specify the type of message sequence to be performed, identify the modules required to participate in the sequence as Slaves and define the number of data transfer cycles required for the sequence. The Header state shall be succeeded by the Header Acknowledge state, except that Abort may be entered to terminate the sequence and the Data state shall be entered for the Datagram Message type.

4.2.2.4 Header Acknowledge: The Header Acknowledge state shall be used to transmit message status from the Slave module(s) to the Master. The transitions out of the Header Acknowledge state shall be as specified below:

a. For a Parameter Write Message sequence, the successor states to Header Acknowledge shall be Header, Idle and Abort. A transition to Header shall initiate a new message and extend the current bus Master’s tenure. A transition to Idle shall terminate the current bus Master’s tenure. Abort may be entered to terminate the Parameter Write Message.

b. For Block Message and Bus Interface Message sequences, the successor states to Header Acknowledge shall be Data and Abort. A transition to Data continues the current bus Master’s tenure. Abort may be entered to terminate the message.

4.2.2.5 Data: The Data state shall consist of a sequence of Data transfer cycles performed as part of a Block Message, Bus Interface Message, or Datagram Message. Data may be transferred from the Master to the Slave(s), defined as a write sequence, or from the Slave to the Master, defined as a read sequence. For Block Message sequences and Datagram Message sequences, the Data sequence may be suspended by entry into the Suspend state. Unless a Data sequence is suspended or terminated by entering Abort, the successor state to data shall be Data Acknowledge, except for Non-Acknowledged Datagram Messages, which shall be Header or Idle.

4.2.2.6 Suspend: The Suspend state shall be used to interrupt a Block Message sequence and a Datagram sequence as specified in the detailed requirements (see 4.3.5.1). A suspended Block Message Data sequence can be resumed by another Block Message whose header contains the appropriate Resume Control Words. The normal successor state to Suspend shall be Data Acknowledge, or for Non-Acknowledged Datagram Messages the normal successor state shall be Idle.

4.2.2.7 Data Acknowledge: The Data Acknowledge state shall be used to transfer Acknowledge information from the Slave(s) to the Master during a Block Message, Bus Interface Message, or Acknowledged Datagram Message sequence. The successor states to Data Acknowledge are Header, Idle, and Abort. A transition to Header shall initiate a new message and extend the current bus Master’s tenure. A transition to Idle shall terminate the current bus Master’s tenure. Abort may be entered to terminate a message.

4.2.2.8 Abort: The Abort state shall consist of four consecutive bus cycles in which the Abort Cycle Type is posted on the CT lines. The successor states to Abort shall be Header and Idle. A transition to Header shall initiate a new message and extend the current bus Master’s tenure. A transition to Idle shall terminate the current bus Master’s tenure.

4.2.2.9 Tenure limitations:

4.2.2.9.1 Bus Request to Vie Interval: When Bus Request is asserted, the bus Master shall limit the number of bus cycles remaining in the current tenure to the sum of the bus cycles specified by the contents of the Vie Interval A Register plus the contents of the Vie Interval B Register plus six cycles (see 4.3.7.3.4 and 4.3.7.3.5). Paragraph 4.3.3.3 specifies procedures which the bus Master shall use to relinquish tenure and permit a Vie sequence in response to Bus Request. A Master's tenure is released when the bus is returned to the Idle state with no wait states asserted.

4.2.2.9.2 Absolute Tenure Limit: Pi-Bus modules shall internally limit each of their individual tenures as bus Master to a maximum of (2**24)+8 bus cycles. The cycle count shall begin with the first HO cycle of the Master's tenure and shall include all bus cycles (including nontransfer cycles). Each module shall provide a hardwired mechanism to automatically force all signal outputs from the module to end tenure such that this tenure limit is not exceeded. The Master shall generate an Abort sequence when the absolute tenure limit expires. Specifically, the fourth cycle of the Abort shall occur on or before 2**24 + 8 cycles. The module may resume normal operation, including vying for the bus, after allowing the bus to be in the Idle state for a minimum of two cycles.

4.2.3 Generic Message: The generic message sequence that forms a basis for Pi-Bus message sequences is described in this section. The Vie sequence is specified in 4.3.3 and the exception sequences are specified in 4.3.5. The details of each message sequence are specified in 4.3.4.

Table 10 illustrates the generic Pi-Bus message sequence which is composed of Header, Header Acknowledge, Data and-Data Acknowledge sequences that correspond to the protocol states described in the preceding section. Some message sequences do not require every generic sequence described below (Message sequence requirements are specified in 4.3.4).

TABLE 10 - Generic Pi-Bus Message Sequence

Signal lines	Protocol Bus State Header			Protocol Bus State Header Acknowledge	Protocol Bus State Data	Protocol Bus State Data Acknowledge
DATA Source =	Header Master			Acknowledge Slave(s)	Data Master (write) Slave (read)	Acknowledge Slave(s)
CYCLE TYPE Source = Master	HD	H		A	D (S)	A
Acknowledge Source =	NS	NS	RCG Slave	ACK Slave	RCG Slave	ACK Slave

4.2.3.1 Generic Message Sequence:

4.2.3.1.1 Normal Operation: The Data (D) lines transfer information between the Master and Slave modules to accomplish the following:

a. Signal the type of message sequence to be performed,

b. Establish a communications path to the Slave module(s).

c. Transfer data between the Master and the Slave(s).

d. Transfer Acknowledge information from the Slave(s) to the Master.

The bus Master uses Type 32 message sequences only when the Master and all modules selected as Slaves for that sequence are operating as Type 32 modules on a Type 32 bus. For a Type 32 mixed mode bus, Type 32/Type 16 message sequence selection can be made on a message-by-message basis and thus may vary during the bus Master's tenure.

The Cycle Type (CT) and Acknowledge Set (AS) lines provides message sequencing synchronization between the Master and Slave(s) to control the sequence of bus states. The AS lines shall also be used by the Slave(s) to report errors.

4.2.3.1.1.1 Header: The bus Master initiates a message sequence by transmitting header information on the Data lines. The bus Master posts the symbol HO on the Cycle Type lines during the first bus cycle of Header transfer and posts H for each of the remaining bus transfer cycles of Header transfer.

The header specifies the module(s) which are selected as Slave(s) for the message sequence. Active module(s) which are addressed by the Slave ID field of HWA become Slaves on the third cycle of Header transfer. Slaves signal their participation in the message by posting the symbol RCG (recognize) on the AS lines beginning with the third cycle of Header transfer and continuing until Header transfer is complete. Modules cease being Slaves only in response to conditions defined in this specification. All modules ensure that the AS lines are released during the first two cycles of Header except that NAK may be asserted as specified in 4.2.3.1.2.2 to report errors from the preceding sequence.

4.2.3.1.1.2 Header Acknowledge: The Header Acknowledge sequence follows the Header sequence and is used to transfer acknowledge information from the Slave(s) to the Master. A single Header Acknowledge transfer cycle is used for all single Slave sequences. The bus Master posts the Header Acknowledge cycle (A) symbol on the Cycle Type lines during the Header Acknowledge cycle in which the Slave is scheduled to post the Slave Acknowledge word. The Slave indicates synchronization with the bus Master by posting ACK on the AS lines during the Header Acknowledge cycle. Five Header Acknowledge transfer cycles are used for a multiple Slave sequence. During the first multiple Slave Header Acknowledge cycle, all

4.2.3.1.1.2 (Continued):

Slaves post a message status symbol on the Data lines and the ACK symbol on the AS lines. During each of the four remaining Acknowledge cycles, eight of the 32 modules are assigned a bit position on the Data lines upon which to post an Acknowledge bit and indicate synchronization with the bus Master by posting ACK on the AS lines.

4.2.3.1.1.3 Data: The bus Master posts the symbol Data during each cycle of the Data sequence. The Slave module(s) post RCG during each cycle of the Data sequence. The bus Master transmits data during write sequences and the Slave transmits data during the read sequences.

4.2.3.1.1.4 Data Acknowledge: The Data Acknowledge sequence is used to transmit status information from the Slaves(s) to the Master. The Data Acknowledge sequence is identical in form to the Header Acknowledge sequence.

4.2.3.1.2 Operations Under Exception Conditions:

4.2.3.1.2.1 Suspend: The Data sequence of a Block Message or Datagram Message may be suspended by the bus Master to permit higher priority communications. The Suspend sequence is performed as defined in 4.3.5.1.

4.2.3.1.2.2 Uncorrectable Line Errors: In general, modules which are Slaves signal uncorrectable detected errors by posting the symbol NAK on the AS lines and providing an error log in the Acknowledge words as specified herein. All modules post the symbol NAK in response to certain uncorrectable line errors which occur during a Vie.

NOTE: NAK itself is not an error. It is a notification of an error. Specific requirements regarding response to errors are found in 4.3.9.6.4.

A module that detects an uncorrectable line error which applies to the operation of bus cycle N posts the symbol NAK on bus cycle N+2. If the detected error occurred during the last two cycles of a message, the resultant NAK occurs during the first two cycles of the following message or Idle. Modules which are neither Slaves nor contenders in a particular message sequence do not post NAK during any sequence other than Vie.

For single cast messages, Slave modules record detected error conditions for the current message in the single Slave Acknowledge word. The Bus Interface should also notify the device of any detected errors. When an Acknowledge word is transmitted on the bus, the error field does not include errors which apply to the immediately preceding bus cycle.

4.2.3.1.2.2 (Continued):

NAK has no specified effect on the resulting operation of the Pi-Bus other than that when NAK occurs or the asserted state of an Acknowledge word error bit is detected, the Master Bus Interface should report that fact to the Master device. The protocol provides the Abort sequence as a means for the message to be terminated if required by the device.

4.2.3.2 Generic Header Information: The Pi-Bus protocol defines message headers consisting of two to ten words. The first header word, Header Word A (HWA), is used in all message sequences to define:

a. The type of message
b. The Slave modules for that message.

The number of scheduled cycles in a message header is defined by the message type. The second header word, Header Word B (HWB), is specified for each message type. Typically, HWB contains the datum transfer cycle count for the data sequence. The third and fourth header words, Header word CO (HWC0) and Header Word C1 (HWC1), are specified for each message type. Typically, HWC0 and HWC1 contain a 32-bit virtual address. Extended header sequences provide six additional header words, HWD0 through HWD5, for application dependent uses.

The generic format for HWA is defined in this section. Formats for the remaining header words are specific to each message sequence and are defined in 4.3.

4.2.3.2.1 Header Word A: The format illustrated in Figure 6 is used for Header Word A. The fields of HWA are described below.

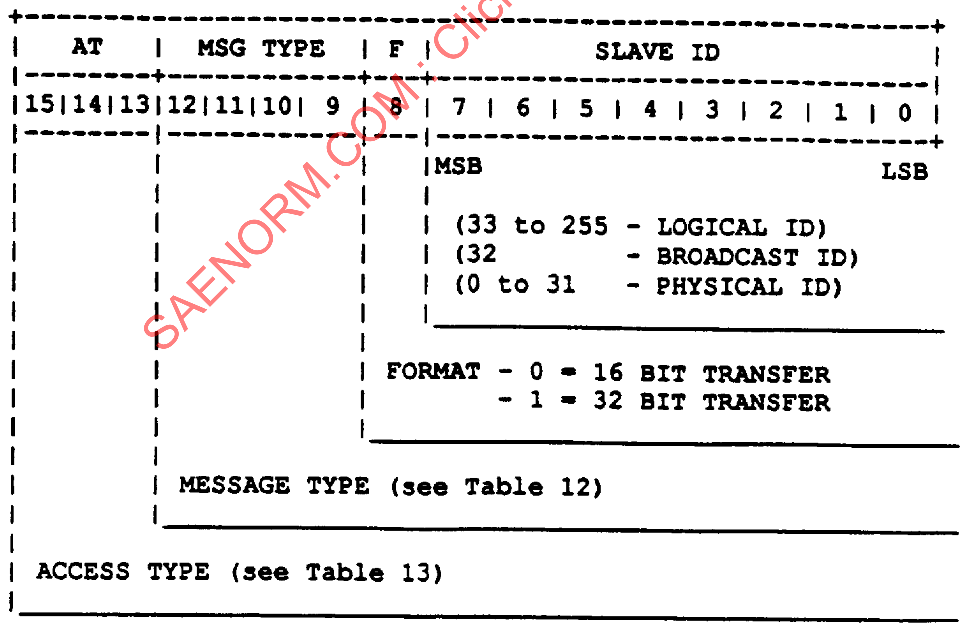


FIGURE 6 - Header Word A Format - Data Lines

4.2.3.2.1.1 Slave ID Field: The Slave ID field of HWA specifies the modules required to participate in the message sequence as Slaves. The Slave ID field provides an eight bit virtual Slave address. The virtual Slave address (Slave ID) range is partitioned into 32 single Slave physical addresses, a Slave broadcast address and 223 logical Slave addresses. The broadcast Slave ID selects all active modules as Slaves, including the bus Master module.

The logical Slave ID's shall be used only to define aliases for individual Slave physical addresses or sets of Slave physical addresses. The use of a logical Slave ID rather than a physical Slave ID in addressing a module shall not elicit any Slave module response other than that which would have been produced by using the module's physical Slave ID. A Slave ID value of 0 to 31 shall specify the single module whose Module Identification matches the Slave ID field. A Slave ID value of 32 shall select all active modules as Slaves. The Slave ID values of 33 to 255 shall select any active modules with the given Slave ID enabled. The number of modules which respond to each of the logical Slave ID values 33 to 255 is system dependent. This means that Slave ID values 33 to 255 can be used for single Slave messages or multiple Slave messages depending on the application.

While in on-line mode, a feature MS Pi-Bus shall be capable of simultaneous operation as both a Master and a Slave. During message sequences where the current bus Master module is selected as a Slave, both data and control must be stored properly and shall be required to be taken from the bus by the module.

4.2.3.2.1.2 Format (F) Field: The Format field specifies whether the message sequence is a 16-bit or 32-bit transfer.

4.2.3.2.1.2.1 Type 16 ED: Type 16 ED modules can only participate in 16-bit transfers. The format bit shall be set to zero for 16-bit transfers.

4.2.3.2.1.2.2 Type 32 EC Nonmixed Mode: Type 32 EC nonmixed mode modules can be configured to participate in 32-bit transfers or 16-bit transfers, but not both. For modules configured to participate in 16-bit transfers, the format bit shall be set to zero for all messages. For modules configured to participate in 32-bit transfers, the format bit shall be set to one for all messages, except the format bit shall be set to zero for Bus Interface Messages. A module shall correctly receive those messages for which the format bit matches its configuration.

4.2.3.2.1.2.3 Type 32 EC Mixed Mode: For Type 32 EC mixed mode modules, the Format field, the Data Format (DF<0>) and the value of P(DC<0>,D<15..0>) shall specify whether the message sequence is to be performed using Type 16 ED transfers or Type 32 EC transfers. For 16-bit ED transfers, the format bit shall be set to zero. For 32-bit EC transfers, the format bit shall be set to one. The Master device must insure the 32-bit transfers are used only when no Type 16 module is selected as a Slave.

4.2.3.2.1.2.3 (Continued):

Type 32 EC mixed mode modules shall select the message transfer type as specified in Table 11. Since no Type 16 ED and Type 32 EC nonmixed mode modules will be connected to DF<0>, DF<0> shall be unasserted when a Type 16 ED is bus Master.

TABLE 11 - Mixed Mode Data Format Matrix

F1	DF<0>	P(DC<0>, D<15..0>)	Message Transfer
0	0	0	Type 16 ED
0	0	1	Type 16 ED
0	1	0	Type 16 ED
0	1	1	Type 32 EC
1	0	0	type 32 EC
1	0	1	Type 16 ED
1	1	0	Type 32 EC
1	1	1	Type 32 EC

1 Data as received before correction.

NOTE: When both Type 16 ED and Type 32 EC mixed mode modules are present on the bus, the Vie sequence will be performed using error detection. See 4.3.3.1 for details. However, Type 16 ED and Type 32 EC mixed mode messages can be mixed in the same tenure. The Type 32 EC mixed mode selected Slaves must determine the Message Transfer Type by using Table 11 before any other error checking or correcting can occur.

RATIONALE for Table 11: The above table provides error correction for the choice of error detection or error correction. That choice cannot be made simply on the state of a bit in a field that must be either detected or corrected.

4.2.3.2.1.3 Message Type (MSG TYPE) Field: The Message Type field specifies the type of message sequence to be performed according to the values in Table 12. A Master shall not transfer any message with an undefined message type. The disallowed message types are 0, 2, and 10. The Master device must ensure that only the multiple Slave Message Types are used when more than one module is selected by the Slave ID field.

4.2.3.2.1.4 Access Type (AT) Field: The AT field is passed from the Master device to the Slave module for use as defined herein and listed in Table 13.

For Block Messages and Datagram Messages, bit 13 is used to signal the device that the current message is a new message (bit 13 = 0) or a resumption of a previously suspended message (bit 13 = 1).

TABLE 12 - Message Type Codes

Message Type	Single or Multiple Slaves	Read or Write	Message Type Code HWA <12..>
Parameter Write	Single	Write	0 0 0 1
	Multiple	Write	0 0 1 1
Block			
- Short Header	Single	Write	0 1 0 1
	Single	Read	0 1 0 0
	Multiple	Write	0 1 1 1
- Extended Header	Single	Write	1 1 0 1
	Single	Read	1 1 0 0
	Multiple	Write	1 1 1 1
Bus Interface	Single	Write	1 0 0 1
	Single	Read	1 0 0 0
	Multiple	Write	1 0 1 1
Datagram			
- Short Header	Multiple	Write	0 1 1 0
- Extended Header	Multiple	Write	1 1 1 0

NOTE: Codes not listed above are reserved.

TABLE 13 - AT (Access Type) Code Assignments

Message Type	At Code	Operation/Slave Response
Block (Short or Extended)	000	Label Address-New[1]
	001	Label Address-Resume[1]
	010	Direct Address-New[1]
	011	Direct Address-Resume[1]
	100	Resource Not Present
	101	Resource Not Present
	110	Resource Not Present
	111	Resource Not Present
Parameter Write	000	Event Filter[1]
	001	Logical
	010-110	Resource Not Present
	111	Reserved - Command Error
Bus Interface	000	Data Link Access
	001-011	Reserved - Command Error
	100	System Time[1]
	101-110	Resource Not Present
	111	Reserved - Command Error
Datagram (Short or Extended)	000	Non-Acknowledged-New[2]
	001	Non-Acknowledged-Resume[2]
	010	NAK and cease participation
	011	NAK and cease participation
	100	Acknowledged-New[1]
	101	Acknowledged-Resume[1]
	110	Resource Not Present
	111	Resource Not Present

1 If this optional Message Type/AT code is not implemented on a module, then the module shall respond Resource Not Present if so accessed.

2 If this optional Message Type/AT code is not implemented on a module, then the module shall NAK and cease participation if so accessed.

4.2.3.2.1.4 (Continued):

For Bus Interface Messages, the code 000 is used to access the Data link address space and the code 100 is used to transfer the Pi-Bus system time (see 4.3.7.4). The Bus Interface physical and Data link layers do not utilize any AT field information except that contained in a Bus Interface Message. Codes 001 through 011 are reserved for future use by higher level protocols. Higher level protocols, such as those which provide communications between modules on different backplanes, are beyond the scope of this specification.

Reserved AT codes can be defined only by future versions of this specification.

Paragraph 4.3 includes a specification of the protocol for each message type and related AT codes.

4.2.3.3 Header and Data Sequence Acknowledgment: Header and Data Acknowledge sequences have the same form and use the same word formats. There are two basic formats for the Acknowledgment, single Slave and multiple Slave. The single Slave Acknowledge sequence is used when the message type field in HWA specifies a single Slave sequence and the multiple Slave Acknowledge sequence is used whenever the Message Type and AT fields in HWA specifies a multiple Slave sequence. The single Slave Acknowledge sequence transfers one word of message status information from the Slave to the Master. The multiple Slave Acknowledge sequence transfers:

a. Eight bits of aggregate message status information from the Slaves to the Master.

b. An individual bit of message status information from each of the 32 possible Slave devices to the Master.

4.2.3.3.1 Single Slave Acknowledge: The Slave module shall perform error checking and logging during the message sequence. The single Slave Acknowledge Word (AWS) defined herein provides the Master with a record of the logged errors and the Module Identification of the Slave.

The single Slave Acknowledge Word shall be transmitted from the Slave to the Master during the Header and Data Acknowledge cycles of each single Slave sequence. The Master Bus Interface should pass the single Slave Acknowledge Word to the Master device.

The single Slave Acknowledge Word format shall be as shown in Figure 7 and specified below.

4.2.3.3.1.1 Slave Module Identification (MID) Field: The Slave Module Identification field shall contain the MID for the Slave.

4.2.3.3.1.2 Acknowledge Word Type (AWT) Field: The AWT field shall contain a two bit binary code as specified in Figure 7. An AWT code of 00 shall specify that the Slave has closed the message sequence with this Header or Data Acknowledge, as appropriate. For exception situations where the Slave will cease to participate in the bus sequence after Header Acknowledge completion, the AWT field shall be set to 00. In conjunction with an S field value of 1, an AWT of 00 shall specify message complete. In conjunction with an S field value of 0, an AWT of 00 shall specify that the Acknowledge Word completes the Slave's response to a Suspend sequence as specified in 4.3.5.1. AWT codes 01, 10, and 11 shall specify that the Slave is acknowledging the completion of the Header sequence. The codes 01 and 10 further specify that the Slave will respond to a Data sequence Suspend with two or eight Resume Control Words, respectively. The S field code shall be 0 for an AWT code of 01 or 10. An AWT code of 11 shall further specify that the Slave cannot perform a Suspend sequence during the current message. The S field code shall be 1 for an AWT code of 11. The Slave shall use an AWT code of 11 and an S code of 1 for any Bus Interface Message Header Acknowledge, since these messages are not suspendable.

4.2.3.3.1.3 Errors Field: The Slave module shall specify detected errors in bits 7 through 13 of the Acknowledge word. Errors reported in the Header Acknowledge Word shall be those errors that originate in the current message from the start of the Header through the second cycle prior to the Header Acknowledge. Errors reported in the Data Acknowledge Word shall be those errors that originate in the current message from one cycle prior to the Header Acknowledge through the second cycle prior to the Data Acknowledge. In addition to logging current error indications, the Bus Interface should report detected errors to the device at the time of detection.

The definition of each error type in the single Slave Acknowledge word shall be as listed below:

a. Correctable line Error - A signal line error has been detected and corrected.

b. Uncorrectable line Error - A signal line error that cannot be corrected has been detected.

c. Sequence Error - The Cycle Type, Acknowledge Set, Wait, and Bus Request line sequence of states is not in agreement with defined protocol sequences or rules.

d. Protect Error - A Bus Interface Message write operation has been attempted which is write protected.

4.2.3.3.1.3 (Continued):

e. Command Error - A Header Word A has been received which is not in agreement with the defined protocol or the Slave is unable to perform the commanded operation because the current bus Master's priority code is unknown, or a message has been received which is not in agreement with the defined format for the Slave device. The use of the Command Error bit is explicitly defined in 4.3.9.6.4.

f. Resource Not Present Error - A resource or capability that is not implemented has been addressed in this module.

g. Device Error - Module device has detected an error attempting to perform a bus related operation.

4.2.3.3.1.4 Busy (B) Field: The Slave module shall specify in bit 14 of the Acknowledge word whether the Slave device is Busy or not Busy. The device shall be recorded as Busy only when the device is unable to accept an otherwise valid message because of other operations in progress. If a Slave is busy during the Header sequence, the Slave shall cease to participate in the bus sequence after Header Acknowledge completion and the AWT field shall be set as defined in 4.2.3.3.1.2.

4.2.3.3.1.5 Suspend (S) Field: The Slave shall specify in bit 15 of the Header Acknowledge Word whether the message is suspendable or not suspendable. Only Block Messages and Datagram Messages may be specified as suspendable. The Slave shall specify in bit 15 of the Data Acknowledge Word whether the Data Acknowledge is in response to a Suspend sequence or the completion of the message.

During a Header sequence in which a Busy or an error, other than a correctable line error, is detected by a Slave and the Slave will cease to participate after Header Acknowledge, the "S" bit shall be set to one in the Header Acknowledge Word.

4.2.3.3.2 Multiple Slave Acknowledge:

4.2.3.3.2.1 Non-Datagram: The requirements for multiple Slave Acknowledge for Non-Datagram Messages are specified in this paragraph. A multiple Slave Acknowledge sequence consists of one bus transfer cycle to transmit aggregate message status and four bus transfer cycles to transmit individual Acknowledge bits. The first transfer cycle is used by each Slave to transmit a message status word to the Master. The individual Message Status Word is wire-ORed on the bus to form an aggregate Message Status Word. The remaining four transfer cycles are used to transmit multiple Slave Acknowledge symbols from the Slave(s) to the Master. The five transfer cycles are labeled HA0, HA1, HA2, HA3, and HA4 for a multiple Slave Header Acknowledge sequence or DA0, DA1, DA2, DA3, and DA4 for a multiple Slave Data Acknowledge sequence.

```
+-----------------------------------------------------------------+
| S | B |          ERRORS          | AWT |       SLAVE MID         |
|---+---+--------------------------+-----+-------------------------|
| 15| 14|13|12|11|10| 9| 8| 7| 6| 5|  4  |  3  |  2  |  1  |  0   |
|---+---+--+--+--+--+--+--+--+-----+-------------------------------+
|   |   |  |  |  |  |  |  |  |     | MSB                       LSB
|   |   |<-- 1 = ERROR ----->|     |
|   |   |<-- 0 = NO ERROR -->|     | SLAVE MODULE IDENTIFICATION
|   |   |  |  |  |  |  |  |  |     |____________________________
|   |   |  |  |  |  |  |  |  |
|   |   |  |  |  |  |  |  |  | ACKNOWLEDGE WORD TYPE
|   |   |  |  |  |  |  |  |  | 00 (S=1) - MESSAGE COMPLETE
|   |   |  |  |  |  |  |  |  | 00 (S=0) - MESSAGE SUSPENDED
|   |   |  |  |  |  |  |  |  |
|   |   |  |  |  |  |  |  |  | 01 - HEADER COMPLETE, DATA EXPECTED
|   |   |  |  |  |  |  |  |  | (S=0) SUSPENDABLE, 2 RC WORDS
|   |   |  |  |  |  |  |  |  |
|   |   |  |  |  |  |  |  |  | 10 - HEADER COMPLETE, DATA EXPECTED
|   |   |  |  |  |  |  |  |  | (S=0) SUSPENDABLE, 8 RC WORDS
|   |   |  |  |  |  |  |  |  |
|   |   |  |  |  |  |  |  |  | 11 - HEADER COMPLETE, DATA EXPECTED
|   |   |  |  |  |  |  |  |  | (S=1) SLAVE NOT SUSPENDABLE
|   |   |  |  |  |  |  |  |  |________________________________________
|   |   |  |  |  |  |  |  |
|   |   |  |  |  |  |  |  | CORRECTABLE LINE ERROR
|   |   |  |  |  |  |  |  |_____________________________
|   |   |  |  |  |  |  |
|   |   |  |  |  |  |  | UNCORRECTABLE LINE ERROR
|   |   |  |  |  |  |  |________________________________
|   |   |  |  |  |  |
|   |   |  |  |  |  | SEQUENCE ERROR
|   |   |  |  |  |  |___________________________________
|   |   |  |  |  |
|   |   |  |  |  | PROTECT ERROR
|   |   |  |  |  |______________________________________
|   |   |  |  |
|   |   |  |  | COMMAND ERROR
|   |   |  |  |_________________________________________
|   |   |  | RESOURCE NOT PRESENT ERROR
|   |   |  |____________________________________________
|   |   | DEVICE ERROR
|   |   |______________________________________________
|   |
|   | DEVICE BUSY  0 - NOT BUSY
|   |              1 - BUSY
|   |__________________________________________________
|
| HEADER ACKNOWLEDGE:  0 - SUSPENDABLE
|                      1 - NOT SUSPENDABLE
|   DATA ACKNOWLEDGE:  0 - MESSAGE SUSPENDED
|                      1 - MESSAGE COMPLETE
|______________________________________________________
```

FIGURE 7 - Single Slave Acknowledge Word Format - Data Lines

4.2.3.3.2.1 (Continued):

During cycles HA0 and DA0 Slave modules shall transmit a Message Status Word to the Master using Data lines <15...0>. The format of the Message Status Word shall be as defined in Figure 8 and Figure 9. Data lines <15...8> shall have the same meaning as bits <15...8> of the single Slave Acknowledge Word and these bits shall be replicated on the Data lines <7... 0>, respectively, to permit error checking. If there is an uncorrectable line error in the Data Line Group, the module shall assume that both lines in any affected pair of redundant lines are asserted. For Class EC messages, bits <15...8> shall also be replicated on Data Check lines <7..0>, respectively, to permit error correction. Modules shall not assert any Data Group line on cycle HA0 or DA0 other than the lines defined above. The Master Bus Interface should pass the aggregate Message Status to the Master device.

Errors reported in the multiple Slave Header Acknowledge Word cycle HA0 shall include those errors that originate in the current message from cycle H0 through the second cycle prior to cycle HA0. During multiple Slave Bus Interface Message and Block Message sequences, errors reported in the multiple Slave Data Acknowledge word during cycle DA0 shall include those errors that originate from one cycle before HA0 through the second cycle prior to cycle DA0.

Header Acknowledge cycles HA1 through HA4 shall be used to transfer "Acknowledge" multiple Slave Acknowledge symbols from the Slave(s) to the Master. Slave modules with MID values 0 through 7 shall post the "Acknowledge" symbols shown in Table 14 and Table 15 on the Data Line Group during cycle HA1. Slave modules with MID values 8 through 15 shall post their "Acknowledge" symbol during cycle HA2 of the sequence. Slave modules with MID values 16 through 23 shall post their "Acknowledge" symbol during cycle HA3 of the sequence and Slave modules with MID values 24 through 31 shall post their "Acknowledge" symbol during cycle HA4 of the sequence. Modules shall not assert any Data Line Group line other than the lines included in their symbol on their assigned Acknowledge cycle.

Data Acknowledge cycles DA1 through DA4 shall be used to transfer the "Acknowledged" or "Non-Acknowledged" multiple Slave Acknowledge symbols defined in Table 14 and Table 15 from the Slave(s) to the Master. The "Acknowledge" multiple Slave Acknowledge symbol shall be posted on the Data Line Group during the assigned cycle of a Data Acknowledge sequence when the module is a Slave and has detected no uncorrectable line errors in the current sequence. Otherwise, the Slave shall post "Non-Acknowledged" during the assigned Acknowledge cycle. Slave modules with MID values 0 through 7 shall post their multiple Slave Acknowledge symbol on the Data Line Group during cycle DA1. Slave modules with MID values 8 through 15 shall post their multiple Slave Acknowledge symbol during cycle DA2 of the sequence. Slave modules with MID values 16 through 23 shall post their multiple Slave Acknowledge symbol during cycle DA3 of the sequence and Slave modules with MID values 24 through 31 shall post their multiple Slave Acknowledge symbol during cycle DA4 of the sequence.

```
+-----------------------------------------------------------------+
| S  | B  |       ERRORS             | S| B|        ERRORS         |
|----+----+--------------------------+--+--+----------------------|
| 15| 14|13|12|11|10| 9| 8 | 7| 6| 5| 4| 3| 2| 1| 0|
|---+---+---+---+---+---+---+----+---+---+-----------------------|
|    |    |   |   |   |   |   |    |   |   |   |   |   |   |   |  |
|    |    |<-- 1 = ERROR --->|<--- BITS <15...8> --->|
|    |    |<-- 0 = NO ERROR->|        REPEATED
|    |    |   |   |   |   |   |    |_________________________
|    |    |   |   |   |   |   |
|    |    |   |   |   |   |   | UNCORRECTABLE LINE ERROR
|    |    |   |   |   |   |   |______________________________
|    |    |   |   |   |   |
|    |    |   |   |   |   | SEQUENCE ERROR
|    |    |   |   |   |   |__________________________________
|    |    |   |   |   |
|    |    |   |   |   | PROTECT ERROR
|    |    |   |   |   |______________________________________
|    |    |   |   |
|    |    |   |   | COMMAND ERROR
|    |    |   |   |__________________________________________
|    |    |   |
|    |    |   | RESOURCE NOT PRESENT ERROR
|    |    |   |______________________________________________
|    |    |
|    |    | DEVICE ERROR
|    |    |__________________________________________________
|    |
|    | DEVICE BUSY  0 - NOT BUSY
|    |              1 - BUSY
|    |_______________________________________________________
|
| HEADER ACKNOWLEDGE:     0 - SUSPENDABLE
|                         1 - NOT SUSPENDABLE
|
|   DATA ACKNOWLEDGE:     0 - MESSAGE SUSPENDED
|                         1 - MESSAGE COMPLETE
|____________________________________________________________
```

FIGURE 8 - Multiple Slave Status Word Format (AWMO) - Data Line

```
+--------------------------+
| S| B|      ERRORS         |
|--+--+--------------------|
| 7| 6| 5| 4| 3| 2| 1 | 0 |
|--+--+--+--+--+--+---+---|
|  |  |  |  |  |  |   |   |
|  |  |<-- 1 = ERROR   --->|
|  |  |<-- 0 = NO ERROR-->|
|  |  |  |  |  |  |   |   |______________________
|  |  |  |  |  |  |   |
|  |  |  |  |  |  |   | UNCORRECTABLE LINE ERROR
|  |  |  |  |  |  |   |__________________________
|  |  |  |  |  |  |
|  |  |  |  |  |  | SEQUENCE ERROR
|  |  |  |  |  |  |___________________________
|  |  |  |  |  |
|  |  |  |  |  | PROTECT ERROR
|  |  |  |  |  |____________________________
|  |  |  |  |
|  |  |  |  | COMMAND ERROR
|  |  |  |  |______________________________
|  |  |  |
|  |  |  | RESOURCE NOT PRESENT ERROR
|  |  |  |_______________________________
|  |  |
|  |  | DEVICE ERROR
|  |  |________________________________
|  |
|  | DEVICE BUSY   0 - NOT BUSY
|  |               1 - BUSY
|  |______________________________
|
| HEADER ACKNOWLEDGE:   0 - SUSPENDABLE
|                       1 - NOT SUSPENDABLE
|
|   DATA ACKNOWLEDGE:   0 - MESSAGE SUSPENDED
|                       1 - MESSAGE COMPLETE
|____________________________________________________
```

FIGURE 9 - Multiple Slave Status Word Format - Data Check Lines
(Used only for Class EC)

TABLE 14 - Multiple Slave Acknowledge Symbol Formats (Four Words)

Module ID Assignment AWM1	Module ID Assignment AWM2	Module ID Assignment AWM3	Module ID Assignment AWM4	Data Line 15	Data Line 14	Data Line 13	Data Line 12	Data Line 11	Data Line 10	Data Line 9	Data Line 8	Data Line 7	Data Line 6	Data Line 5	Data Line 4	Data Line 3	Data Line 2	Data Line 1	Data Line 0
0	8	16	24	0	0	0	0	0	0	0	A[1]	0	0	0	0	0	0	0	A
1	9	17	25	0	0	0	0	0	0	A	0	0	0	0	0	0	0	A	0
2	10	18	26	0	0	0	0	0	A	0	0	0	0	0	0	0	A	0	0
3	11	19	27	0	0	0	0	A	0	0	0	0	0	0	0	A	0	0	0
4	12	20	28	0	0	0	A	0	0	0	0	0	0	0	A	0	0	0	0
5	13	21	29	0	0	A	0	0	0	0	0	0	0	A	0	0	0	0	0
6	14	22	30	0	A	0	0	0	0	0	0	0	A	0	0	0	0	0	0
7	15	23	31	A	0	0	0	0	0	0	0	A	0	0	0	0	0	0	0

1 A) = Non-Acknowledged /1 = Acknowledge

TABLE 15 - Multiple Slave Acknowledge Symbols (Four Words)
Data Check Line Formats (Used only for Class EC)

Module ID Assignment AWM1	Module ID Assignment AWM2	Module ID Assignment AWM3	Module ID Assignment AWM4	Data Check Line 7	Data Check Line 6	Data Check Line 5	Data Check Line 4	Data Check Line 3	Data Check Line 2	Data Check Line 1	Data Check Line 0
0	8	16	24	0	0	0	0	0	0	0	A[1]
1	9	17	25	0	0	0	0	0	0	A	0
2	10	18	26	0	0	0	0	0	A	0	0
3	11	19	27	0	0	0	0	A	0	0	0
4	12	20	28	0	0	0	A	0	0	0	0
5	13	21	29	0	0	A	0	0	0	0	0
6	14	22	30	0	A	0	0	0	0	0	0
7	15	23	31	A	0	0	0	0	0	0	0

[1] A) 0 = Non-Acknowledged /1 = Acknowledge

4.2.3.3.2.1 (Continued):

Modules shall not assert any Data Group line other than the lines included in their symbol on their assigned Acknowledge cycle.

The multiple Slave Acknowledge symbols posted by the Slave(s) during a particular Acknowledge cycle are logically OR'ed on the bus to produce one of the four multiple Slave Acknowledge Words (AWM1, AWM2, AWM3, or AWM4) which are used during each multiple Slave Acknowledge sequence. The Master Bus Interface should pass the multiple Slave Acknowledge words to the Master device.

In addition to posting their assigned Acknowledge symbol, Slave modules shall post the symbol ACK (or NAK if required in response to an error) on the AS lines during their assigned Acknowledge cycle. Slave modules shall post the symbol NS (or NAK if required in response to an error) on the AS lines during the Acknowledge cycles in which they are not assigned to post their Acknowledge symbol.

4.2.3.3.2.2 Datagram: The requirements for multiple Slave Acknowledge for Datagram Messages are specified in this paragraph. Since Non-Acknowledge Datagram Messages do not have Acknowledgments, multiple Slave Acknowledge sequences are not applicable to Non-Acknowledge Datagram Messages. A multiple Slave Acknowledge sequence consists of one bus transfer cycle (DA0) to transmit aggregate message status. The transfer cycle is used by each Slave to transmit a Message Status Word to the Master. The individual Message Status Word is wire-ORed on the bus to form an aggregate Message Status Word.

During cycle DA0, Slave modules shall transmit a Message Status Word to the Master using Data lines <15...0>. The format of the Message Status Word shall be as defined in Figure 8 and Figure 9. Data lines <15...8> shall have the same meaning as bits <15... 8> of the single Slave Acknowledge Word and these bits shall be replicated on the Data lines <7...0>, respectively, to permit error checking. If there is an uncorrectable line error in the Data line Group, the module shall assume that both lines in any affected pair of redundant lines are asserted. For Class EEC messages, bits <15.. 8> shall also be replicated on Data Check lines <7..0>, respectively, to permit error correction. Modules shall not assert any Data Group line on cycle DAO other than the lines defined above. The Master Bus Interface should pass the aggregate Message Status to the Master device.

Errors reported in the multiple Slave Data Acknowledge word during cycle DA0 shall include those errors that originate from H0 through the second cycle prior to cycle DA0.

4.3 Detailed Requirements:

4.3.1 Introduction: Detailed requirements for the Pi-Bus Data link protocol are specified in this section. The bus states which govern the cycle-by-cycle operation of the bus are defined and their relationships to the protocol states are given. The bus Mastership protocol is specified, including requirements for the use of Bus Request. All Pi-Bus communications

4.3.1 (Continued):

sequences are specified by sequence diagrams which show the scheduled sequence of bus states and corresponding module operations. Any sequence not specified herein shall be considered invalid.

The protocol for using Wait to insert non-transfer cycles into a message sequence is specified. The Data link facilities which are required to be accessible over the bus are defined. Finally, bus response to error conditions is specified and bus diagnostics techniques are defined.

4.3.2 Bus State Definitions: The protocol states defined in 4.2 consist of sequences of bus states. The bus states shall define the information content of each bus cycle. Table 16 lists the bus states along with their corresponding Cycle Type and the protocol states in which they appear.

TABLE 16 - Bus State Definitions

Bus States	CT	Protocol State	Comment
I	I	Idle	Bus Idle Cycle
V0..V3	V	Vie	Vie Priority bits resolved, 3 per step
VZ0..VZ3	V	Vie	Nontransfer cycle for vie decision time
H0	H0	Header	First cycle of header transfer
H1..H9	H	Header	Additional cycles of header transfer
HZ	H	Header	Nontransfer cycle for decision time
HA0	A	Header Ack	Single Slave or first multicast Header Acknowledge
HA1..HA4	A	Header Ack	Additional multicast Header Acknowledge
D	D	Data	Datum transfer cycles
DZ	D	Data	Nontransfer cycle for decision time
DA0	A	Data Ack	Single Slave or first multicast Data Acknowledge
DA1..DA4	A	Data Ack	Additional multicast Acknowledge cycles
S0..S2	S	Suspend	Cycles announcing message being suspended
D	D	Suspend	Slave transmitting Resume Control Word
DZ	D	Suspend	Nontransfer cycle for decision time
AB0..AB3	AB	Abort	Cycles announcing sequence being aborted

Each of the Pi-Bus communication sequences defined herein has a corresponding sequence diagram which shows the required schedule of bus states and the corresponding bus operations for that sequence.

The sequence diagrams contain the following information:

a. Bus State

(1) Bus State: Unique bus state is shown for each scheduled bus cycle.

b. Data

(1) D<31..16>: Data format type or state for the most significant 16-bits of a 32-bit bus.

4.3.2 (Continued):

(2) D<15..0>: Data format type or state for the 16-bit bus or the least significant 16-bits of a 32-bit bus.

(3) Source: Bus Interface driving the Data lines; Master (M) or Slave (S).

(4) Read Source: Shows cycles where the Slave (S) sources the Data during a read sequence.

(5) Write Source: Shows cycles where the Master (M) sources Data during a write sequence.

If no source (Read source or Write source) is shown, Data lines are released.

b. Cycle Type

(1) Cycle Type: Defines Cycle Type symbol for each scheduled bus cycle.

(2) Source: Shows the source of the Cycle Type symbol as the Master (M) or a contender (C).

c. Acknowledge Set

(1) Acknowledge: Defines the state of the Acknowledge Set lines for each bus cycle. The AS lines may be driven by a single Slave, by multiple Slave, or by contenders in Vie. For the multiple Slave case, a Not Selected (NS) beneath ACK means that an NS symbol may appear for that bus cycle if there are no sources for ACK during that cycle out of the group of potential Slave (S) sources.

(2) Source: The Acknowledge source is shown as Slave(s) (S) or contender(s) (C) or, if not shown, the lines are released.

(3) Source <7-0>: For multiple Slave case, the source may be any Slave(s) (S) with an MID value of 0 through 7.

(4) Source <15-8>: For multiple Slave case, the source may be any Slave(s) (S) with an MID value of 8 through 15.

(5) Source <23-16>: For multiple Slave case, the source may be any Slave(s) (S) with an MID value of 16 through 23.

(6) Source <31-24>: For multiple Slave case, the source may be any Slave(s) (S) with an MID value of 24 through 31.

d. Wait

(1) Allowed: Defines bus cycles where Wait may be asserted. A source for Wait is not shown in these sequences since the scheduled sequence of bus states assumes that Wait is not asserted.

A set of three (:::) or four (::::) colons in a sequence diagram indicates that a number of bus states occur in the sequence at that point.

4.3.3 Bus Mastership: The protocol governing bus Mastership is specified herein. The Vie sequence which assigns bus Mastership to a particular module is defined. The protocol which allows a module with higher priority than the current bus Master to request a Vie sequence by asserting Bus Request is specified.

4.3.3.1 Vie Sequence:

4.3.3.1.1 16-bit and 32-bit Nonmixed Mode Vie: The Vie sequence shall be used to determine a single bus Master for the first header sequence which occurs after the bus enters the Idle state. The bus Master shall be selected on the basis of the Vie Priority code stored in each contender's Vie Priority Register (see 4.3.7.3.6).

Any module that requires bus Mastership may initiate Vie after two or more cycles of Idle. Modules shall be capable of participating in a Vie sequence which begins on the third or any later cycle of Idle. All active modules shall monitor each step of the Vie process and store the Vie Priority level of the winning module.

Due to pipeline delays, a module may attempt to initiate Vie one cycle after Vie is initiated by another module. In this case, the module which attempted to initiate the late Vie sequence shall cease to contend within two cycles and shall complete the original Vie sequence as a noncontender. Modules shall not attempt to initiate Vie more than one cycle after the Vie Cycle Type has been posted on the Cycle Type lines.

The Vie sequence shall be a four step sequence as defined in Table 17. Each Vie step shall use two bus cycles to resolve three of the twelve bits of Vie Priority code. Modules which are contending for bus Mastership shall decode the three most significant bits (VP<11..9>) of their Vie Priority code into a one-of-eight Module Vie Code as shown in Table 18 and Table 19. Modules shall initiate Vie by posting the Module Vie Code on the Data line Group and the Vie cycle type on the Cycle Type lines. On the following bus cycle, contenders shall release the Data line Group.

Modules shall read the logical-OR of the Module Vie Codes posted by the contenders from the Data line Group at the end of the first cycle of Vie (bus state VO) as an Aggregate Vie Code (VCO). During the second cycle of each step, each contender (C) shall compare the posted Module Vie Code to the Aggregate Vie Code read from the bus. If the Aggregate Vie Code read from the bus has a bit asserted in a more significant bit position than the Module Vie Code posted by the module, then the module has lost contention and shall cease being a contender and shall continue as an active module. If the contender's posted Module Vie Code has the same bit asserted as the most significant bit asserted in the Aggregate Vie Code, the module has not lost the Vie step and shall proceed to the next Vie step as a contender. For a Class ED bus, if there is a mismatch in a bit pair for the Data line Group, the module shall assume that both lines in any affected pair of redundant lines are asserted.

4.3.3.1.1 (Continued):

This process shall be repeated in the second, third, and fourth Vie steps using Vie Priority code bits VP<8..6>, VP<5..3> and VP<2..0>, respectively. At the conclusion of the fourth Vie step, at most one module remains as a contender and that module shall become the new bus Master. If a bus Master is selected, uniqueness is guaranteed by the five MID bits within the Vie Priority code. The bus Master must post H0 on the bus cycle immediately following the VZ3 cycle. If no bus Master is selected due to error conditions, the bus shall return to the Idle state.

Table 17 defines the detailed sequence of bus states and module actions during the Vie sequence. The Data lines are shown as two groups of 16 lines each. Lines D<31..16>, if implemented, shall remain released throughout the Vie sequence. Contending modules shall post their Module Vie Codes on D<15..0> and, for EC Vies, on DC<7..0>. For an ED Vie, the Module Vie Codes posted by the contending modules shall be logically OR'ed during the first cycle of each Vie step to form the Aggregate Vie Code for that step. For EC Vies, each bit of the Aggregate Vie Code shall be determined by the symbol posted on two of the three triplicated copies of the Module Vie Code. Contenders shall post the symbol Vie on the Cycle Type lines during each cycle of the Vie as illustrated. Contenders shall also post the symbol RCG on the AS lines during each cycle of the second, third, and fourth Vie steps. All active modules shall post NAK on the AS lines on cycle N+2 each time an uncorrectable Data group or Cycle Type group error which applies to the operation of bus cycle N is detected.

Modules shall not assert Wait nor Bus Request during the Vie sequence. Noncontenders shall not assert any line or post any symbol during the Vie sequence other than posting NAK on the AS lines as specified above. All active modules shall monitor the Vie sequence and record the winning bus Master's priority code. Uncorrectable Data or Cycle Type group line errors detected during the Vie sequence shall cause the modules which do not win the Vie sequence to store "unknown" as the current bus Master's priority code. If errors occur during the Vie sequence, refer to Error Table 9 for error processing. During subcycle 1 of each Vie step, Data line errors for bit pairs other than the most significant pair that has a line asserted shall be considered correctable for an ED Vie since the winning priority is not affected. A module which has "unknown" for the current bus Master's priority code shall signal "Command Error" in the Header Acknowledge Word for Non Datagram Message types to the bus Master if the module becomes a Slave during the bus Master's tenure see 4.3.9). See 3.2.2.1.3.1.2 for Data Check line assertion and data parity detection requirements during Vie for an ED Vie. See 3.2.2.1.3.2.2 for Data Check line assertion and error correction requirements during Vie for an EC Vie.

TABLE 17 - Vie Sequence

Signal Lines	Bus State V0	Bus State VZ0	Bus State V1	Bus State VZ1	Bus State V2	Bus State VZ2	Bus State V3	Bus State VZ3
DATA[1]								
D<31..16>	0	0	0	0	0	0	0	0
D<15..8>	VC0	0	VC1	0	VC2	0	VC3	0
D<7..0>	VC0	0	VC1	0	VC2	0	VC3	0
DC<7..0>[2]	VC0	0	VC1	0	VC2	0	VC3	0
Source =	C		C		C		C	
CYCLE TYPE	V	V	V	V	V	V	V	V
Source = C								
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	RCG	RCG
Source =			C	C	C	C	C	C
WAIT	0	0	0	0	0	0	0	0
Allowed	NO	NO	NO	NO	NO	NO	NO	NO
STEP SUB-CYCLE	1	2	1	2	1	2	1	2
VIE PRIORITY	11	11	8	8	5	5	2	2
CODE BITS	10	10	7	7	4	4	1	1
	9	9	6	6	3	3	0	0
VIE STEP	0	0	1	1	2	2	3	3

[1] VCx is the vie code formed by the inclusive 'OR' of the vie codes asserted by all contending modules (C).

[2] DC<7..0> are only valid for error correction vie.

TABLE 18 - Module Vie Code Format - Data lines

Bit Pattern	Bit Pattern	Bit Pattern	Data Line 15	Data Line 14	Data Line 13	Data Line 12	Data Line 11	Data Line 10	Data Line 9	Data Line 8	Data Line 7	Data Line 6	Data Line 5	Data Line 4	Data Line 3	Data Line 2	Data Line 1	Data Line 0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
11	10	9	Vie Priority register bits for first vie step															
8	7	6	Vie Priority register bits for second vie step															
5	4	3	Vie Priority register bits for third vie step															
2	1	0	Vie Priority register bits for fourth vie step															

TABLE 19 - Module Vie Code Format - Data Check Lines

Bit Pattern	Bit Pattern	Bit Pattern	Data Check Line 7	Data Check Line 6	Data Check Line 5	Data Check Line 4	Data Check Line 3	Data Check Line 2	Data Check Line 1	Data Check Line 0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0
11	10	9	Vie Priority register bits for first vie step							
8	7	6	Vie Priority register bits for second vie step							
5	4	3	Vie Priority register bits for third vie step							
2	1	0	Vie Priority register bits for fourth vie step							

4.3.3.1.2 Mixed Mode Vie: For each BIU capable of becoming a Type 32 EC mixed mode bus Master, there shall be a programmable bit called Vie_Mode_EC. This bit specifies whether the Vie sequence shall operate in error detection mode (Vie_Mode_EC false) or error correction mode (Vie_Mode_EC true).

[Rationale: The Vie_Mode_EC bit was specified to determine what type of Vie sequence to perform. Basically, it tells the BIU whether there are any Type 16 ED modules on the bus. If there are, the Vie must be performed using error detection only. If some modules used detection while others used correction, multiple Masters could result when errors occurred during the Vie.]

4.3.3.2 Tenure Pass Message: This message type is not permitted.

4.3.3.3 Bus Request: The priority used during a message Vie sequence shall be defined as the concatenation of the module's logical priority with its MID. However, the priority used in determining if Bus Request may be asserted shall be the module logical priority only. If current Master wishes to change its priority, it shall assert Bus Request and follow the rules specified in 4.3.5.1.

The Bus Request line shall be asserted by a module to signal the current bus Master that the module has ahigher priority requirement for bus Mastership. The module asserting Bus Request shall ensure that this condition is met by only asserting Bus Request when the module's logical priority (contained in the Vie Priority register bits 5 through 11) contains a higher priority code than that of the current bus Master's logical priority. A module shall not assert Bus Request when the current bus Master's priority code is unknown. The current bus Master shall honor Bus Request by relinquishing tenure and releasing all bus signal lines by the end of the Vie Interval defined by the sum of the bus cycles specified by the contents of the Vie Interval A Register plus the contents of the Vie Interval B Register plus six cycles (see 4.3.7.3.4 and 4.3.7.3.5). The module that asserted Bus Request shall start its Vie sequence on the third cycle after the previous Master relinquished

4.3.3.3 (Continued):

tenure, if the module that asserted the Bus Request still requires bus Mastership.

The assertion of Bus Request shall be allowed on any bus cycle, except for the cycles starting with the third cycle of Idle and continuing through the last cycle of Vie.

The bus Master shall count all bus cycles, including non-transfer cycles, whenever Bus Request is asserted.The first cycle counted shall be the first cycle after the cycle containing the initial assertion for Bus Request. Any subsequent cycle in which Bus Request is released shall not be counted and shall cause the accumulated count to be discarded.

A Bus Request shall cause no activity in the Master, until Vie Timer A expiration. The allowable ways to release tenure include:

a. Release all bus lines to place the bus in the Idle state, rather than post an H0 cycle.

b. If a Block Message data sequence is in progress and the Slave(s) have indicated to the associated Header Acknowledge that suspension is allowed, the bus Master may perform a Suspend sequence (see 4.3.5.1) and release all bus lines before the total Vie Interval time elapses.

c. Datagram Messages are suspendable, the bus Master may perform a Suspend sequence (see 4.3.5.1) and release all bus lines before the total Vie Interval time elapses.

d. If the cycle count reaches the sum of the value specified in the Vie Interval A Register, plus the value specified in the Vie Interval B Register, the bus Master shall perform an Abort sequence, such that the first cycle of the Abort sequence occurs on the second bus cycle immediately following the cycle that exceeds the Vie Interval limit. After completing the Abort sequence, the Master shall immediately relinquish tenure and release all bus lines (see 4.3.5.2).

4.3.4 Message Sequences:

4.3.4.1 Parameter Write Message Sequence: The Parameter Write Message shall be used to transfer three parameter words from the Master device to a Slave or multiple Slave devices. The Parameter Write sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Single Slave: Table 20
b. Type 16 ED Multiple Slave: Table 21
c. Type 32 EC Single Slave: Table 22
d. Type 32 EC Multiple Slave: Table 23

TABLE 20 - Parameter Write Sequence - Type 16 ED Single Slave

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State HZ	Bus State HA0
DATA						
D<31..16>	0	0	0	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	0	AWS
Source =	M	M	M	M		S
CYCLE TYPE	H0	H	H	H	H	A
Source = M						
Acknowledge	NS	NS	RCG	RCG	RCG	ACK
Source =			S	S	S	S
WAIT	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES

TABLE 21 - Parameter Write Sequence - Type 16 ED Multiple Slave

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State HZ	Bus State HA0	Bus State HA1	Bus State HA2	Bus State HA3	Bus State HA4
DATA										
D<31..16>	0	0	0	0	0	0	0	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M	M		S	S	S	S	S
CYCLE TYPE	H0	H	H	H	H	A	A	A	A	A
Source = M										
Acknowledge	NS	NS	RCG	RCG	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =			S	S	S	S	S			
Source (15..8) =			S	S	S	S		S		
Source (23..16) =			S	S	S	S			S	
Source (31..24) =			S	S	S	S				S
WAIT	0	0	0	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 22 - Parameter Write Sequence - Type 32 EC Single Slave

Signal Lines	Bus State H0	Bus State H1	Bus State HZ	Bus State HA0
DATA				
D<31..16>	HWB	HWC1	0	0
D<15..9>	HWA	HWC0	0	AWS
Source =	M	M		S
CYCLE TYPE	H0	H	H	A
Source = M				
Acknowledge	NS	NS	RCG	ACK
source =			S	S
WAIT	0	0	0	0
Allowed	NO	NO	YES	YES

TABLE 23 - Parameter Write Sequence - Type 32 EC Multiple Slave

Signal Lines	Bus State H0	Bus State H1	Bus State HZ	Bus State HA0	Bus State HA1	Bus State HA2	Bus State HA3	Bus State HA4
DATA								
D<31..16>	HWB	HWC1	0	0	0	0	0	0
D<15..0>	HWA	HWC0	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M		S	S	S	S	S
CYCLE TYPE	H0	H	H	A	A	A	A	A
Source = M								
Acknowledge	NS	NS	RCG	ACK	ACK	ACK	ACK	ACK
					(NS)	(NS)	(NS)	(NS)
Source (7..0) =			S	S	S			
Source (15..8) =			S	S		S		
Source (23..16) =			S	S			S	
Source (31..24) =			S	S				S
WAIT	0	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES

4.3.4.1 (Continued):

Header word formats for the Parameter Write Message shall be as shown in Figure 10. Header Word A shall specify the Slave(s), Type 16 ED or Type 32 EC Format, Access and Message Types. The Access Type field shall be as defined in Table 13. Message Types shall be as defined for the single Slave (SS) and multiple Slave cases. Header Words B, C0, and C1 shall contain the parameter words to be passed to the device.

```
 HEADER WORD A (HWA)
+----------------------------------------------------------+
|   AT    |  MSG TYPE   | F |            SLAVE ID           |
|---------+-------------+---+-------------------------------|
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------+-------------+---+-------------------------------|



 HEADER WORD B (HWB)
+----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+----------------------------------------------------------+
MSB <--------------- Parameter Word 0 ------------------>LSB

 HEADER WORD C0 (HWC0)
+----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+----------------------------------------------------------+
MSB <--------------- Parameter Word 1 ------------------>LSB

 HEADER WORD C1 (HWC1)
+----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+----------------------------------------------------------+
MSB <--------------- Parameter Word 2 ------------------>LSB
```

FIGURE 10 - Parameter Write Header Word Formats

The Header Acknowledge Word (AWS) shall supply current message status information to the Master from the Slave for single Slave sequences. For multiple Slave sequences, the multiple Slave Status Word (AWM0) shall supply current message status information to the Master. The multiple Slave Header Acknowledge Words (AWM1, AWM2, AWM3, AWM4) convey the list of Slave participants to the Master.

AT codes 010-110 shall be illegal Parameter Write AT codes. If a Slave receives an AT code 010-110 for a Parameter Write, it shall respond Resource Not Present. AT code 111 shall be a Reserved Parameter Write AT code. A Slave shall respond with a Command Error if a Parameter Write with AT code of 111 is received.

4.3.4.1.1 Logical Type: The header words for Parameter Write Message Logical type shall be as shown in Figure 11. AT code 001 designates a logical Parameter Write Message in which HWB, HWC0, and HWC1 shall be transferred to the Slave device, the Slave device shall be notified (e.g., interrupted) upon completion of the transfer, and the Slave BIU shall provide a mechanism for the Slave device to access the transferred parameters.

```
 MSB                                             LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+------------------------------------------------+
|   AT    | MSG TYPE  | F|        SLAVE ID        | HWA
| 0  0  1|            |  |                        |
+------------------------------------------------+
|               PARAMETER WORD 0                  | HWB
+------------------------------------------------+
|               PARAMETER WORD 1                  | HWC0
+------------------------------------------------+
|               PARAMETER WORD 2                  | HWC1
+------------------------------------------------+
```

FIGURE 11 - Parameter Write Message, AT Code = 001

4.3.4.1.2 Event Filter Type: The header words for Parameter Write Message Event Filter type shall be as shown in Figure 12.

```
 MSB                                             LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+------------------------------------------------+
|   AT    | MSG TYPE  | F|        SLAVE ID        | HWA
| 0  0  0|            |  |                        |
+------------------------------------------------+
|                  EVENT FLAG                     | HWB
+------------------------------------------------+
| RESERVED |               LABEL                  | HWC0
+------------------------------------------------+
|                   RESERVED                      | HWC1
+------------------------------------------------+
```

FIGURE 12 - Parameter Write Message, AT Code = 000

AT code 000 designates an Event Filter Parameter Write Message. The LABEL field is a 12-bit right Justified label that is used by the Slave to determine:

a. Label Active: If the Slave will not accept messages for the specified label, then the Slave shall cease to participate as a Slave in the Pi-Bus sequence (prior to Header Acknowledge) and shall not assert any error conditions on the Pi-Bus due to the BIU not being able to accept a message.

4.3.4.1.2 (Continued):

b. Event Mask Satisfied: The Event Flag in HBW shall be logically OR' ed with any previous Event Flag transferred to the active label. The resultant Event Flag shall be compared by the Slave BIU against a Mask word. The Slave device shall be notified (e.g., interrupted) upon a match between the resultant Event Flag and the Mask. The resultant Event Flag shall also be transferred to the Slave device, and the label shall be set inactive. The Slave BIU shall provide a mechanism for Slave device to supply the Mask and access the transferred Event Flag.

4.3.4.2 Block Message - Short Header (SH) Sequence: The Block Message - Short Header (SH) sequence shall be used to read data from a single Slave device to the Master device or to write data from the Master device to one or more Slave devices. The Block Message - Short Header (SH) sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Single Slave: Table 24
b. Type 16 ED Multiple Slave: Table 25
c. Type 32 EC Single Slave: Table 26
d. Type 32 EC Multiple Slave: Table 27

Block Messages (for both short and extended headers) may support both a direct memory access addressing mode and a label addressing access mode. Label addressing supports a "logical" addressing mode in which a 16 bit value is used by the module's BIU to determine the module's requirement to participate in the message sequence as a Slave and to determine the address of the buffer in the module to be used during the message transaction. Direct addressing supports an addressing mode in which the 32 bits specified in HWC0 and HWC1 specify a virtual address of the buffer in the Slave to be used during the message transaction. AT codes shall be used to determine whether label addressing or direct addressing is associated with a message.

The ability of a module type to accept a directly addressed or label addressed Block Message shall be controlled from the device-side interface. If direct addressing is used, the meaning of the direct address is as specified in the corresponding module specification.

NOTE: It is the intent of the Backplane Committee that the following philosophy be followed regarding Block Messages. Modules which act as both a Pi-Bus Master and Pi-Bus Slave should accept label addressed Block Messages for both short and extended headers. However, such modules should not accept Block Messages which use direct addressing, and, if such a module is so accessed, it should respond Resource Not Present. A module which acts only as a Pi-Bus Slave may accept Block Messages which use either direct addressing or label addressing.

TABLE 24 - Block Message - SH Sequence - Type 16 ED Single Slave

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State HZ	Bus State HA0	Bus State D0	Bus State ::::	Bus State Dn	Bus State DZ	Bus State DA0
DATA											
D<31..16>	0	0	0	0	0	0	0	::::	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	0	AWS	D0	::::	Dn	0	AWS
Source =	M	M	M	M		S					S
Read Source =							S	S	S		
Write Source =							M	M	M		
CYCLE TYPE	H0	H	H	H	H	A	D	::::	D	D	A
Source = M											
Acknowledge	NS	NS	RCG	RCG	RCG	ACK	RCG	::::	RCG	RCG	ACK
Source =			S	S	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	0	::::	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 25 - Block Message - SH Sequence - Type 16 ED Multiple Slave

TABLE 25A - Bus State H0 to Bus State HA4

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State HZ	Bus State HA0	Bus State HA1	Bus State HA2	Bus State HA3	Bus State HA4
DATA										
D<31..16>	0	0	0	0	0	0	0	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M	M		S	S	S	S	S
CYCLE TYPE	H0	H	H	H	H	A	A	A	A	A
Source = M										
Acknowledge	NS	NS	RCG	RCG	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =			S	S	S	S	S			
Source (15..8) =			S	S	S	S		S		
Source (23..16) =			S	S	S	S			S	
Source (31..24) =			S	S	S	S				S
WAIT	0	0	0	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 25B - Bus State D0 to Bus State DA4

Signal Lines	Bus State D0	Bus State ::::	Bus State DN	Bus State DZ	Bus State DA0	Bus State DA1	Bus State DA2	Bus State DA3	Bus State DA4
DATA									
D<31..16>	0	::::	0	0	0	0	0	0	0
D<15..0>	D0	::::	Dn	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M		S	S	S	S	S
CYCLE TYPE	D	::::	D	D	A	A	A	A	A
Source = M									
Acknowledge	RCG	::::	RCG	RCG	ACK	ACK	ACK	ACK	ACK
						(NS)	(NS)	(NS)	(NS)
Source (7.. 0) =	S	S	S	S	S	S			
Source (15..8) =	S	S	S	S	S		S		
Source (23..16) =	S	S	S	S	S			S	
Source (31..24) =	S	S	S	S	S				S
WAIT	0	::::	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 26 - Block Message - SH Sequence - Type 32 EC Single Slave

Signal Lines	Bus State H0	Bus State H1	Bus State HZ	Bus State HA0	Bus State D0	Bus State ::::	Bus State Dn	Bus State DZ	Bus State DA0
DATA									
D<31..16>	HWB	HWC1	0	0	D0H	::::	DnH	0	0
D<15..0>	HWA	HWC0	0	AWS	D0L	::::	DnL	0	AWS
Source =	M	M		S					S
Read Source =					MS	S	S		
Write Source =					M	M	M		
CYCLE TYPE	H0	H	H	A	D	::::	D	D	A
Source = M									
Acknowledge	NS	NS	RCG	ACK	RCG	::::	RCG	RCG	ACK
Source =			S	S	S	S	S	S	S
WAIT	0	0	0	0	0	::::	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES

TABLE 27 - Block Message - SH Sequence - Type 32 EC Multiple Slave

TABLE 27A - Bus State H0 to Bus State HA4

Signal Lines	Bus State H0	Bus State H1	Bus State HZ	Bus State HA0	Bus State HA1	Bus State HA2	Bus State HA3	Bus State HA4
DATA								
D<31..16>	HWB	HWC1	0	0	0	0	0	0
D<15..0>	HWA	HWC0	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M		S	S	S	S	S
CYCLE TYPE	H0	H	H	A	A	A	A	A
Source = M								
Acknowledge	NS	NS	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =			S	S	S			
Source (15..8) =			S	S		S		
Source (23..16) =			S	S			S	
Source (31..24) =			S	S				S
WAIT	0	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES

TABLE 27B - Bus State D0 to Bus State DA4

Signal Lines	Bus State D0	Bus State ::::	Bus State Dn	Bus State Dz	Bus State DA0	Bus State DA1	Bus State DA2	Bus State DA3	Bus State DA4
DATA									
D<31..16>	D0H	::::	DnH	0	0	0	0	0	0
D<15..0>	D0L	::::	DnL	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M		S	S	S	S	S
CYCLE TYPE	D	::::	D	D	A	A	A	A	A
Source = M									
Acknowledge	RCG	::::	RCG	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =	S	S	S	S	S	S			
Source (15..8) =	S	S	S	S	S		S		
Source (23..16) =	S	S	S	S	S			S	
Source (31..24) =	S	S	S	S	S				S
WAIT	0	::::	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES

4.3.4.2 (Continued):

The header word formats for the Block Message - Short Header sequence shall be as shown in Figure 13. Header Word A shall specify the Slave(s) ID, Type 16 ED or 32 EC Format, Access and Message Types. Bit 13 indicates whether the message is being used to resume a previously suspended Block Message (bit 13 = 1) or send a new message (bit 13 = 0). The Message Types shall be as defined for the Single Slave Write (SSW), Single Slave Read (SSR) and multiple Slave cases. Header Word B shall contain an unsigned binary datum count which specifies the number of datum units to be transferred between the Master and Slave(s), except that all zeros shall represent 65 536 datum units. The datum count shall be the number of 16-bit words for 16 bit transfers or the number of double words for 32-bit transfers. Header Word C0 and C1 shall contain 32-bits of virtual addressing information to be passed to the device.

The Header and Data Acknowledge words shall supply current message status information to the Master from the Slave.

Data word formats are application specific. The number of words or double words transferred for each Block Message - Short Header sequence shall be equal to the value in Header Word B. For the Type 32 EC sequences, data words are shown with L or H designations. The least significant half of a double word or a single word transmitted on Data lines D<15..0> contains the L designation. The most significant half of a double word or a single word transmitted on Data lines D<31..16> contains the H designation.

Except for the Message Type in HWA and the extended header words D0-D5 for Block Messages with extended header, the interpretation of the message headers (i.e., HWA, HWB, HWC0, and HWC1) are identical for both short and extended headers.

AT codes 100-111 shall be illegal Block Message AT codes. If a Slave receives an AT code 100-111 for a Block Message, it shall respond Resource Not Present.

4.3.4.2.1 Label Addressing: The header words for Block Messages - Short Headers using label addressing are shown in Figure 14.

AT codes 000 and 001 denote label addressing for Block Messages. When a message using label addressing is initially transferred to a Slave, the AT code shall be 000. When a suspended message using label addressing is resumed, the AT code shall be 001. The initial offset is added to the memory location designated by the label to designate the first location in memory where the message is to be transferred. For example, an initial offset of zero designates that the message will be transferred starting at the location designated by the label.

```
HEADER WORD A (HWA)
+---------------------------------------------------------+
|   AT    |  MSG TYPE   | F |           SLAVE ID           |
|---------+-------------+---+------------------------------|
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------+-------------+---+------------------------------|



HEADER WORD B (HWB)
+---------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---------------------------------------------------------+
MSB <----------------- Datum Count -------------------->LSB

HEADER WORD C0 (HWC0)
+---------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---------------------------------------------------------+
15 <--------- Least Significant Address Bits -----------> 0

HEADER WORD C1 (HWC1)
+---------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---------------------------------------------------------+
31 <--------- Most Significant Address Bits -----------> 16
```

FIGURE 13 - Block Message - Short Header Word Formats

```
MSB                                                LSB
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+-----------------------------------------------+
|   AT    | MSG TYPE  | F|       SLAVE ID        | HWA
+-----------------------------------------------+
|     INITIAL COUNT OR COUNT REMAINING **        | HWB
+-----------------------------------------------+
|                    LABEL                       | HWC0
+-----------------------------------------------+
|INITIAL OFFSET OR INIT OFFSET + # DATUM XFER **| HWC1
+-----------------------------------------------+

** For resume message headers, HWB is the
   number of datum remaining to be transferred
   and HWC1 is initial data offset + number of
   datum transferred.
```

FIGURE 14 - Block Message - Short Header - Label Addressing

4.3.4.2.1 (Continued):

For Block Message and Datagram Message types, the LABEL field is a 16-bit right justified label that is used by the Slave to determine:

a. Label in Range: If the label is not within a Slave's defined range, the Slave ceases to participate as a Slave in the Pi-Bus sequence and does not assert any error conditions on the Pi-Bus due to the label being out of range.

b. Label Active: If the Slave will not accept messages for the specified label, then the Slave shall cease to participate as a Slave in the Pi-Bus sequence and shall not assert any error conditions on the Pi-bus due to the BIU not being able to accept a message.

c. Label Busy: For Block Messages, if the message is an initial header (AT=000), and the label is busy, the BIU asserts Busy in the Header Acknowledge, and ceases to participate in the Pi-Bus sequence after Header Acknowledge is completed. For Datagram Messages the response to label busy is specified in 4.3.9.6.4.

Example conditions causing a label to be busy include:

a. A busy condition set because of a previous transaction to the specified label which caused an interrupt and for which the Slave device had not reset the busy condition.

b. The Slave setting the label to busy.

c. Label Suspended: For Block Messages, if the message is a resume header (AT=001), and the label is not suspended, the BIU shall assert Command Error in the Header Acknowledge and cease to participate in the Pi-Bus sequence after Header Acknowledge is completed. If the message is a resume header and the buffer is suspended, the BIU shall begin the Data Transfer sequence of the message. For Datagram Messages the response to Label Suspended is specified in 4.3.9.6.4.

d. Location: The location in the Slave device of where message data is to be placed or read.

e. Maximum Buffer Size Associated with Label: For Block Messages, if HWB + HWC1 > maximum buffer size associated with the label, then the BIU shall assert Resource Not Present in the Header Acknowledge and cease to participate in the Pi-Bus sequence after Header Acknowledge is completed. For Datagram Messages the response to maximum buffer size associated with label is specified in 4.3.9.6.4.

Both Block Messages and Datagram Messages can use the same label.

4.3.4.2.2 Direct Addressing: The header words for Block Messages - Short Headers using direct addressing are shown in Figure 15.

```
 MSB                                              LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+------------------------------------------------+
|   AT    | MSG TYPE  | F|        SLAVE ID        | HWA
|------------------------------------------------|
|      INITIAL COUNT OR COUNT REMAINING **        | HWB
|------------------------------------------------|
|               DIRECT ADDRESS LOW                | HWC0
|------------------------------------------------|
|               DIRECT ADDRESS HIGH               | HWC1
+------------------------------------------------+
** For resume message headers, HWB is the number
   of datum remaining to be transferred.
```

FIGURE 15 - Block Message - Short Header - Direct Addressing

AT codes 010 and 011 designate direct addressing for Block Messages. When a message using direct addressing is initially transferred to a Slave, the AT code shall be 010. When a suspended message is resumed, the AT code shall be 011.

4.3.4.3 Block Message - Extended Header Sequence: The Block Message - Extended Header (EH) sequence shall be used to read data from a single Slave device to the Master device or to write data from the Master device to one or more Slave devices. This sequence shall be used where more header information is required than that provided by the Block Message - Short Header sequence. The Block Message - Extended Header sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Single Slave: Table 28
b. Type 16 ED Multiple Slave: Table 29
c. Type 32 EC Single Slave: Table 30
d. Type 32 EC Multiple Slave: Table 31

Block Messages (for both short and extended headers) may support two addressing modes: label addressing and direct addressing. AT codes are used to determine whether the label addressing or direct addressing is associated with the message. Except for the Message Type in HWA and the extended header words D0-D5 for Block Messages with extended headers, the interpretation of the message headers (i.e., HWA, HWB, HWC0, and HWC1) are identical for both short and extended headers. Refer to 4.3.4.2 for a detailed discussion of the message sequence.

Header word formats for the Block Message - Extended Header sequence shall be as shown in Figure 16. Header Word A shall specify the Slave(s) ID, Type 16 ED or 32 EC Format, Access and Message Types. Bit 13 indicates whether the message is being used to resume a previously suspended Block Message (bit 13 = 1) or send a new message (bit 13 = 0). The Message Types shall be as defined for the single Slave Write, single Slave Read, and multiple Slave cases. Header Word B shall contain an unsigned binary datum count which specifies the number of datum units to

TABLE 28 - Block Message - EH Sequence - Type 16 ED Single Slave

TABLE 28A - Bus State H0 to Bus State HA0

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State ::::	Bus State H9	Bus State HZ	Bus State HA0
DATA									
D<31..16>	0	0	0	0	0	::::	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	HWD0	::::	HWD5	0	AWS
Source =	M	M	M	M	M	M	M		S
CYCLE TYPE	H0	H	H	H	H	::::	H	H	A
Source = M									
Acknowledge	NS	NS	RCG	RCG	RCG	::::	RCG	RCG	ACK
Source =			S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES

TABLE 28B - Bus State D0 to Bus State DA0

Signal Lines	Bus State D0	Bus State D1	Bus State D2	Bus State ::::	Bus State Dn	Bus State DZ	Bus State DA0
DATA							
D<31..16>	0	0	0	::::	0	0	0
D<15..0>	D0	D1	D2	::::	Dn	0	AWS
Source =							S
Read Source =	S	S	S	S	S		
Write Source =	M	M	M	M	M		
CYCLE TYPE	D	D	D	::::	D	D	A
Source = M							
Acknowledge	RCG	RCG	RCG	::::	RCG	RCG	ACK
Source =	S	S	S	S	S	S	S
WAIT	0	0	0	::::	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES

TABLE 29 - Block Message - EH Sequence - Type 16 ED Multiple Slave

TABLE 29A - Bus State H0 to Bus State HA2

	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus
	State	State	State	State	State	State	State	State	State	State	State
Signal Lines	H0	H1	H2	H3	H4	::::	H9	HZ	HA0	HA1	HA2
DATA											
D<31..16>	0	0	0	0	0	::::	0	0	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	HWD0	::::	HWD5	0	AWM0	AWM1	AWM2
Source =	M	M	M	M	M	M	M		S	S	S
CYCLE TYPE	H0	H	H	H	H	::::	H	H	A	A	A
Source = M											
Acknowledge	NS	NS	RCG	RCG	RCG	::::	RCG	RCG	ACK	ACK	ACK
										(NS)	(NS)
Source											
(7..0) =			S	S	S	S	S	S	S	S	
(15.8) =			S	S	S	S	S	S	S		S
(23.16) =			S	S	S	S	S	S	S		
(31.24) =			S	S	S	S	S	S	S		
WAIT	0	0	0	0	0	:::	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 29B - Bus State HA3 to Bus State DA4

	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus	Bus
	State	State	State	State	State	State	State	State	State	State	State
Signal Lines	HA3	HA4	D0	::::	Dn	DZ	DA0	DA1	DA2	DA3	DA4
DATA											
D<31..16>	0	0	0	::::	0	0	0	0	0	0	0
D<15..0>	AWM3	AWM4	D0	::::	Dn	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	S	S	M	M	M		S	S	S	S	S
CYCLE TYPE	A	A	D	::::	D	D	A	A	A	A	A
Source = M											
Acknowledge	ACK	ACK	RCG	::::	RCG	RCG	ACK	ACK	ACK	ACK	ACK
	(NS)	(NS)						(NS)	(NS)	(NS)	(NS)
Source											
(7..0) =			S	S	S	S	S	S			
(15.8) =			S	S	S	S	S		S		
(23.16) =	S	S	S	S	S	S	S			S	
(31.24) =			S	S	S	S	S				S
WAIT	0	0	0	:::	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 30 - Block Message - EH Sequence - Type 32 EC Single Slave

TABLE 30A - Bus State H0 to Bus State HA0

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State HZ	Bus State HA0
DATA							
D<31..16>	HWB	HWC1	HWD1	HWD3	HWD5	0	0
D<15..0>	HWA	HWC0	HWD0	HWD2	HWD4	0	AWS
Source =	M	M	M	M	M		S
CYCLE TYPE	H0	H	H	H	H	H	A
Source = M							
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	ACK
Source =			S	S	S	S	S
WAIT	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES

TABLE 30B - Bus State D0 to Bus State DA0

Signal Lines	Bus State D0	Bus State D1	Bus State D2	Bus State ::::	Bus State Dn	Bus State DZ	Bus State DA0
DATA							
D<31..16>	DOH	D1H	D2H	::::	DnH	0	0
D<15..0>	DOL	D1L	D2L	::::	DnL	0	AWS
Source =							S
Read Source =	S	S	S	S	S		
Write Source =	M	M	M	M	M		
CYCLE TYPE	D	D	D	::::	D	D	A
Source = M							
Acknowledge	RCG	RCG	RCG	::::	RCG	RCG	ACK
Source =	S	S	S	S	S	S	S
WAIT	0	0	0	::::	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES

TABLE 31 - Block Message - EH Sequence - Type 32 EC Multiple Slave

TABLE 31A - Bus State H0 to Bus State HA4

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State HZ	Bus State HA0	Bus State HA1	Bus State HA2	Bus State HA3	Bus State HA4
DATA											
D<31..16>	HWB	HWC1	HWD1	HWD3	HWD5	0	0	0	0	0	0
D<15..0>	HWA	HWC0	HWD0	HWD2	HWD4	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M	M	M		S	S	S	S	S
CYCLE TYPE Source = M	H0	H	H	H	H	H	A	A	A	A	A
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source											
(7..0) =			S	S	S	S	S	S			
(15.8) =			S	S	S	S	S		S		
(23.16) =			S	S	S	S	S			S	
(31.24) =			S	S	S	S	S				S
WAIT Allowed	0 NO	0 NO	0 YES	0 YES	0 YES	0 YES	0 YES	0 YES	0 YES	0 YES	0 YES

TABLE 31B - Bus State D0 to Bus State DA4

Signal Lines	Bus State D0	Bus State ::::	Bus State Dn	Bus State Dz	Bus State DA0	Bus State DA1	Bus State DA2	Bus State DA3	Bus State DA4
DATA									
D<31..16>	D0H	::::	DnH	0	0	0	0	0	0
D<15..0>	D0L	::::	DnL	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M		S	S	S	S	S
CYCLE TYPE Source = M	D	::::	D	D	A	A	A	A	A
Acknowledge	RCG	::::	RCG	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =	S	S	S	S	S	S			
Source (15..8) =	S	S	S	S	S		S		
Source (23..16) =	S	S	S	S	S			S	
Source (31..24) =	S	S	S	S	S				S
WAIT Allowed	0 YES	:::: YES	0 YES	0 YES	0 YES	0 YES	0 YES	0 YES	0 YES

```
 HEADER WORD A (HWA)
+-----------------------------------------------------------+
|   AT    |  MSG TYPE   | F |           SLAVE ID             |
|---------+-------------+---+--------------------------------|
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------+-------------+---+--------------------------------|



 HEADER WORD B (HWB)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB <----------------- Datum Count ---------------------->LSB

 HEADER WORD C0 (HWC0)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
15 <--------- Least Significant Address Bits -----------> 0

 HEADER WORD C1 (HWC1)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
31 <--------- Most Significant Address Bits -----------> 16

 HEADER WORD D0 (HWD0)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB                                                      LSB
               *               *             *
               *               *             *
 HEADER WORD D5 (HWD5)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB                                                      LSB
```

FIGURE 16 - Block Message - Extended Header Word Formats

4.3.4.3 (Continued):

be transferred between the Master and Slave(s), except that all zeros shall represent 65 536 datum units. The datum count shall be the number of 16-bit words for 16-bit transfers or the number of double words for 32-bit transfers. Header Word C0 and C1 shall contain 32-bits of virtual addressing information to be passed to the device.

4.3.4.3.1 Label Addressing: The header words for Block Messages - Extended Headers using label addressing are shown in Figure 17.

```
 MSB                                               LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+------------------------------------------------+
|   AT    | MSG TYPE  | F |       SLAVE ID        | HWA
+------------------------------------------------+
|      INITIAL COUNT OR COUNT REMAINING **        | HWB
+------------------------------------------------+
|                   LABEL                         | HWC0
+------------------------------------------------+
|INITIAL OFFSET OR INIT OFFSET + # DATUM XFER **| HWC1
+------------------------------------------------+
|             EXTENDED HEADER D0                  | HWD0
|                    |                            | HWD1
|                    |                            | HWD2
|                    |                            | HWD3
|                    V                            | HWD4
|             EXTENDED HEADER D5                  | HWD5
+------------------------------------------------+
** For resume message headers, HWB is the number
   of datum remaining to be transferred and HWC1
   is initial data offset + number of datum
   transferred.
```

FIGURE 17 - Block Message - Extended Header - Label Addressing

4.3.4.3.2 Direct Addressing: The header words for Block Messages - Extended Headers using direct addressing are shown in Figure 18.

4.3.4.4 Bus Interface Message Sequence: The Bus Interface Message shall be used to read or write to the Bus Interface register address spaces. The Bus Interface Message sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Single Slave: Table 32
b. Type 32 EC Single Slave: Table 32
c. Type 16 ED Multiple Slave: Table 33
d. Type 32 EC Multiple Slave: Table 33

Header word formats for the Bus Interface Message sequence shall be as shown in Figure 19. Header Word A shall specify the participant Slave(s), Format, Access and Message Types. [NOTE: The format bit will be set to one for Bus Interface Messages between 32-bit EC mixed mode modules performing 32-bit EC transfers]. The message sequence shall be

```
MSB                                                    LSB
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1           0
+------------------------------------------------+
|   AT    | MSG TYPE  | F|        SLAVE ID       | HWA
+------------------------------------------------+
|     INITIAL COUNT OR COUNT REMAINING **        | HWB
+------------------------------------------------+
|             DIRECT ADDRESS LOW                 | HWC0
+------------------------------------------------+
|             DIRECT ADDRESS HIGH                | HWC1
+------------------------------------------------+
|             EXTENDED HEADER D0                 | HWD0
|                     |                          | HWD1
|                     |                          | HWD2
|                     |                          | HWD3
|                     V                          | HWD4
|             EXTENDED HEADER D5                 | HWD5
+------------------------------------------------+
** For resume message headers, HWB is the number
   of datum remaining to be transferred.
```

FIGURE 18 - Block Message - Extended Header - Direct Addressing

```
HEADER WORD A (HWA)
+-----------------------------------------------------------+
|   AT    |  MSG TYPE  | F |            SLAVE ID            |
+---------+------------+---+--------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---------+------------+---+--------------------------------+
```

```
HEADER WORD B (HWB)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------+-----------------------------+
|MSB                       LSB|MSB                       LSB
|                             |
|                             | Word Count
|                             |____________________________
|
| Register Address
|____________________________
```

FIGURE 19 - Bus Interface Message Header Word Formats

TABLE 32 - Bus Interface Message Sequence - Type 16 ED and Type 32 EC Single Slave

Signal Lines	Bus State H0	Bus State H1	Bus State HZ	Bus state HA0	Bus State D0	Bus State : : : :	Bus State Dn	Bus State DZ	Bus State DA0
DATA									
D<31..16>	0	0	0	0	0	: : : :	0	0	0
D<15..0>	HWA	HWB	0	AWS	D0	: : : :	Dn	0	AWS
Source =	M	M		S					S
Read Source =					S	S	S		
Write Source =					M	M	M		
CYCLE TYPE	H0	H	H	A	D	: : : :	D	D	A
Source = M									
Acknowledge	NS	NS	RCG	ACK	RCG	: : : :	RCG	RCG	ACK
Source =			S	S	S	S	S	S	S
WAIT	0	0	0	0	0	: : : :	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES

TABLE 33 - Bus Interface Message Sequence - Type 16 ED and Type 32 EC Multiple Slave

TABLE 33A - Bus State H0 to Bus State HA4

Signal Lines	Bus State H0	Bus State H1	Bus State HZ	Bus state HA0	Bus State HA1	Bus State HA2	Bus State HA3	Bus State HA4
DATA								
D<31...16>	0	0	0	0	0	0	0	0
D<15..0>	HWA	HWB	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M		S	S	S	S	S
CYCLE TYPE	H0	H	H	A	A	A	A	A
Source = M								
Acknowledge	NS	NS	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =			S	S	S			
Source (15..8) =			S	S		S		
Source (23..16) =			S	S			S	
Source (31..24) =			S	S				S
WAIT	0	0	0	0	0	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES

TABLE 33B - Bus State D0 to Bus State DA4

Signal lines	Bus State D0	Bus State : : : :	Bus State Dn	Bus State DZ	Bus State DA0	Bus State DA1	Bus State DA2	Bus State DA3	Bus State DA4
DATA									
D<31..16>	0	: : : :	0	0	0	0	0	0	0
D<15..0>	D0	: : : :	Dn	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M		S	S	S	S	S
CYCLE TYPE	D	: : : :	D	D	A	A	A	A	A
Source = M									
Acknowledge	RCG	: : : :	RCG	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =	S	S	S	S	S	S			
Source (15..8) =	S	S	S	S	S		S		
Source (23..16) =	S	S	S	S	S			S	
Source (31..24) =	S	S	S	S	S				S
WAIT	0	...	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES

4.3.4.4 (Continued):

identical for F=0 and F=1. Header and Data information shall be asserted on D<15..0>. For Type 32 EC, Data Check is generated and checked over D<31..0> and appropriate Data Check lines.

The Message Types shall be as defined for the single Slave Write, single Slave Read, and multiple Slave cases. Header Word B shall contain the address for the first Bus Interface register to be accessed in the sequence and an unsigned binary word count specifying the number of data words to be transferred between the Master and Slave(s), except that a word count of all zeros shall mean 256 words. Each data transfer after the first data transfer shall access a successive register address. The register address shall be incremented modulo 256 after each data word transfer.

Data word formats are as defined in 4.3.7. The number of data words transferred shall equal the value in Header Word B.

If a write is attempted and within the register set to be written there are reserved or write protected registers, the message sequence shall continue and the nonwrite protected and nonreserved registers in the register set to be written shall be modified by the write, with the appropriate error being reported in the Data Acknowledge word at completion of the message. In this case, in addition to the data associated with the nonreserved or nonwrite protected registers to be written, there shall be a Data word placed on the bus for each Reserved or write protected register included in the set, even though these data shall be discarded by the Slave.

NOTE: To avoid error conditions being reported on the bus, a Bus Interface Message should not attempt to write to a reserved or write-protected register. Multiple Bus Interface Messages should be used to avoid this problem.

4.3.4.4 (Continued):

AT codes 001-011 and 101-111 shall be illegal Bus Interface AT codes. If a Slave receives an AT code 101-110 for a Bus Interface, it shall respond Resource Not Present. AT codes 001-011 and 111 shall be a Reserved Bus Interface AT code. A Slave shall respond with a Command and Error if a Bus Interface Message with AT code of 001-011 or 111 is received.

4.3.4.4.1 Data Link Layer Registers: Access Type code 000 is defined for the Data Link register address space. The header words for a Bus Interface Message for Data Link Layer register access are shown in Figure 20. When a Slave receives a Bus Interface Message with AT code = 000, it causes the Slave's Pi-Bus Data Link registers to be accessed as defined above. The Data Link Layer registers are defined in 4.3.7.3.

```
 MSB                                               LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+-------------------------------------------------+
|   AT    | MSG TYPE  | F|        SLAVE ID        | HWA
| 0  0  0|            |  |                        |
|-------------------------------------------------|
|    REGISTER ADDRESS    |      WORD COUNT        | HWB
+-------------------------------------------------+
```

FIGURE 20 - Bus Interface Message - Data Link Register

4.3.4.4.2 System Timer: BIUs shall implement a 48-bit timer register with a 1 MHz clock. The header words for a Bus Interface Message for System Timer access are shown in Figure 21. When a Slave receives a Bus Interface Message with AT code = 100, Register Address = 0, and Word Count = 3, it shall cause data to be transferred directly between the system timer registers of the Master and Slave(s).

```
 MSB                                               LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+-------------------------------------------------+
|   AT    | MSG TYPE  | F|        SLAVE ID        | HWA
| 1  0  0|            |  |                        |
|-------------------------------------------------|
|    REGISTER ADDRESS    |      WORD COUNT        | HWB
| 0  0  0  0  0  0  0  0| 0  0  0  0  0  0  1  1|
+-------------------------------------------------+
```

FIGURE 21 - Bus Interface Message - System Timer

4.3.4.5 Datagram Message - Short Header Sequence: The Datagram - Short Header (SH) message sequence shall be used to transfer data from the Master device to multiple Slave devices. There shall be no Header Acknowledge states during this message. The use of Data Acknowledge shall be selectable using the Access Type.

Datagram Messages support a label addressing access mode. Header Word C0 shall contain a 16-bit label which is used by the module's BIU to determine the module's requirement to participate in the message sequence as a Slave and to determine the address of the buffer in the module to be used during the message transaction. HWC1 shall contain the initial label offset + the number of datum transferred (for resume operations).

Header word formats for the Datagram - Short Header sequence shall be as shown in Figure 22. Header Word A shall specify the participating Slave(s), Type 16 ED or 32 EC Format, Access and Message Types. Bit 15 shall indicate whether the Datagram Message has a Data Acknowledge cycle (bit 15 = 1) or completes with the last data cycle (Dn) with no Data Acknowledge (bit 15 = 0). Bit 13 indicates whether the message is being used to resume a previously suspended Datagram Message (bit 13 = 1) or send a new message (bit 13 = 0). Header Word B shall contain an unsigned binary datum count which specifies the number of datum units to be transferred between the Master and Slave(s), except that all zeros shall represent 65 536 datum units. The datum count shall be the number of 16-bit transfers or the number of double words for 32-bit transfers. Header Word C0 and C1 shall contain 32 bits of virtual addressing information to be passed to the Slave device.

```
 HEADER WORD A (HWA)
+-----------------------------------------------------------+
|   AT    |  MSG TYPE   | F |         SLAVE ID              |
+---------+-------------+---+-------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---------+-------------+---+-------------------------------+


 HEADER WORD B (HWB)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB <----------------- Datum Count -------------------->LSB

 HEADER WORD C0 (HWC0)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
15 <--------------------- LABEL ------------------------> 0

 HEADER WORD C1 (HWC1)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
31 <---------- NUMBER OF DATUM TRANSFERRED ------------> 16
```

FIGURE 22 - Datagram Message - Short Header Word Formats

4.3.4.5 (Continued):

For Datagram Messages with a Data Acknowledge the Data Acknowledge Word shall supply current message status information to the Master from the Slave.

Data word formats are application specific. The number of words or double words transferred for each Datagram Message - Short Header sequence shall be equal to the value in Header Word B. For the Type 32 EC sequences, data words are shown with L or H designations. The least significant half of a double word or a single word transmitted on Data lines D<15..0> contains the L designation. The most significant half of a double word or a single word transmitted on Data lines D<31..16> contains the H designation.

Except for the Message Type in HWA and the extended header words D0-D5 for Datagram Messages with extended headers, the interpretation of the message headers (i.e., HWA, HWB, HWC0, and HWC1) are identical for both short and extended headers.

AT codes 010, 011, 110, and 111 shall be illegal Datagram Message AT codes. If a Slave receives an AT code 010, 011, 110, and 111 for a Datagram Message, It shall respond in accordance with the Error Tables.

4.3.4.5.1 Non-Acknowledged: The header words for Datagram Messages - Short Header with Non-Acknowledged are shown in Figure 23.

```
 MSB                                            LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+-----------------------------------------------+
|   AT    | MSG TYPE   | F|      SLAVE ID        | HWA
+-----------------------------------------------+
|      INITIAL COUNT OR COUNT REMAINING **      | HWB
+-----------------------------------------------+
|                     LABEL                     | HWC0
+-----------------------------------------------+
|INITIAL OFFSET OR INIT OFFSET + # DATUM XFER **| HWC1
+-----------------------------------------------+

** For resume message headers, HWB is the number of
   datum remaining to be transferred and HWC1 is
   initial data offset + number of datum transferred.
```

FIGURE 23 - Datagram Message - Short Header (Non-Acknowledged)

At codes of 000 and 001 designate Non-Acknowledged with the Datagram Message. When a message with Non-Acknowledged is initially transferred to a Slave, the AT code shall be 000. When a suspended message is resumed, the AT code shall be 001. The initial offset in HWC1 is added to the memory location designated by the label to designate the first location in memory where the message is to be transferred. For example, an initial offset of zero designates that the message will be transferred starting at the location designated by the label.

4.3.4.5.1 (Continued):

The Datagram (Non-Acknowledged) - Short Header (SH) sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Multiple Slave: Table 34
b. Type 32 EC Multiple Slave: Table 35

TABLE 34 - Datagram Message (Non-Acknowledged) - SH Sequence Type 16 ED Multiple Slave

Signal lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn
DATA								
D<31..16>	0	0	0	0	0	0	....	0
D<15..0>	HWA	HWB	HWC0	HWC1	0	D0	: : : :	Dn
Source =	M	M	M	M		M	M	M
CYCLE TYPE	H0	H	H	H		D	: : : :	D
Source = M								
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	: : : :	RCG
Source =			S	S	S	S	S	S
WAIT	0	0	0	0	0	0	...	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES

TABLE 35 - Datagram Message (Non-Acknowledged) - SH Sequence Type 32 EC Multiple Slave

Signal lines	Bus State H0	Bus State H1	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn
DATA						
D<31..16>	HWB	HWC1	0	D0H	: : : :	DnH
D<15..0>	HWA	HWC0	0	D0L	: : : :	DnL
Source =	M	M		M	M	M
CYCLE TYPE	H0	H	H	D	: : : :	D
Source = M						
Acknowledge	NS	NS	RCG	RCG	: : : :	RCG
Source =			S	S	S	S
WAIT	0	0	0	0	: : : :	0
Allowed	NO	NO	YES	YES	YES	YES

4.3.4.5.2 Acknowledged: The header words for Datagram Messages - Short Headers with Data Acknowledge are shown in Figure 24.

```
MSB                                              LSB
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+-----------------------------------------------+
|   AT   | MSG TYPE  | F|        SLAVE ID       | HWA
+-----------------------------------------------+
|      INITIAL COUNT OR COUNT REMAINING **      | HWB
+-----------------------------------------------+
|                    LABEL                      | HWC0
+-----------------------------------------------+
|INITIAL OFFSET OR INIT OFFSET + # DATUM XFER **| HWC1
+-----------------------------------------------+

** For resume message headers, HWB is the number of
   datum remaining to be transferred and HWC1 is
   initial data offset + number of datum transferred.
```

FIGURE 24 - Acknowledged Datagram Message - Short Header

At codes of 100 and 101 designate a Data Acknowledge is to be transferred with the Datagram Message. When a message with an Acknowledge is initially transferred to a Slave, the AT code shall be 100. When a suspended message is resumed, the AT code shall be 101. The initial offset in HWC1 is added to the memory location designated by the label to designate the first location in memory where the message is to be transferred. For example, an initial offset of zero designates that the message will be transferred starting at the location designated by the label.

The Acknowledged Datagram - Short Header (SH) sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Multiple Slave: Table 36
b. Type 32 EC Multiple Slave: Table 37

TABLE 36 - Acknowledged Datagram Message - SH Sequence Type 16 ED Multiple Slave

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn	Bus State Dz	Bus State DA0
DATA										
D<31..16>	0	0	0	0	0	0	: : : :	0	0	0
D<15..0>	HWA	HWB	HWC0	HWC1	0	D0	: : : :	Dn	0	AWM0
Source =	M	M	M	M		M	M	M		S
CYCLE TYPE	H0	H	H	H		D	: : : :	D	D	A
Source = M										
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	: : : :	RCG	RCG	ACK
Source =			S	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	: : : :	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 37 - Acknowledged Datagram Message - SH Sequence Type 32 EC Multiple Slave

Signal lines	Bus State H0	Bus State H1	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn	Bus State Dz	Bus State DA0
DATA								
D<31..16>	HWB	HWC1	0	D0H	: : : :	DnH	0	0
D<15..0>	HWA	HWC0	0	D0L	: : : :	DnL	0	AWM0
Source =	M	M		M	M	M		S
CYCLE TYPE	H0	H	H	D	: : : :	D	D	A
Source = M								
Acknowledge	NS	NS	RCG	RCG	: : : :	RCG	RCG	ACK
Source =			S	S	S	S	S	S
WAIT	0	0	0	0	: : : :	0	0	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES

4.3.4.6 Datagram Message - Extended Header Message Sequence: The Datagram - Extended Header message sequence shall be used to transfer data from the Master device to multiple Slave devices. There shall be no Header Acknowledge states during this message. The use of Data Acknowledge shall be selectable using the Access Type. Datagram Messages support a label addressing access mode.

Header word formats for the Datagram Message - Extended Header sequence shall be as shown in Figure 25. Header Word A shall specify the participating Slave(s), type 16 ED or 32 EC Format, Access and Message Types. Bit 15 shall indicate whether the Datagram Message has a Data Acknowledge cycle (bit 15 = 1) or completes with the last data cycle with no Data Acknowledge (bit 15 = 0). Bit 13 indicates whether the message is being used to resume a previously suspended Datagram Message (bit 13 = 1) or send a new message (bit 13 = 0). Header Word B shall contain an unsigned binary datum count which specifies the number of datum units to be transferred between the Master and Slave(s), except that all zeros shall represent 65 536 datum units. The datum count shall be the number of 16-bit transfers or the number of double words for 32-bit transfers. Header Words C0 and C1 shall contain 32 bits of virtual addressing information to be passed to the Slave device.

For Datagram Messages with a Data Acknowledge, the Data Acknowledge Word shall supply current message status information to the Master from the Slave.

Data word formats are application specific. The number of words or double words transferred for each Datagram Message - Extended Header sequence shall be equal to the value in Header Word B. For the Type 32 EC sequences, data words are shown with L or H designations. The least significant half of a double word or a single word transmitted on Data lines D<15..0> contains the L designation. The most significant half of a double word or a single word transmitted on Data lines D<31..16> contains the H designation.

4.3.4.6 (Continued):

Except for the Message Type in HWA and the EH words D0-D5 for Datagram Messages with extended headers, the interpretation of the message headers (i.e., HWA, HWB, HWC0, and HWC1) are identical for both short and extended headers.

AT codes 010, 011, 110, and 111 shall be illegal Datagram Message AT codes. If a Slave receives an AT code 010, 011, 110, and 111 for a Datagram Message, it shall respond in accordance with the Error Tables.

```
 HEADER WORD A (HWA)
+-----------------------------------------------------------+
|    AT    |  MSG TYPE   | F |            SLAVE ID           |
+----------+-------------+---+-------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+----------+-------------+---+-------------------------------+



 HEADER WORD B (HWB)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB <----------------- Datum Count ------------------>LSB

 HEADER WORD C0 (HWC0)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
15 <--------------------- LABEL -----------------------> 0

 HEADER WORD C1 (HWC1)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
31 <---------- NUMBER OF DATUM TRANSFERRED ------------> 16

 HEADER WORD D0 (HWD0)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB                                                     LSB

                 *              *              *
                 *              *              *
                 *              *              *

 HEADER WORD D5 (HWD5)
+-----------------------------------------------------------+
|15|14|13|12|11|10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----------------------------------------------------------+
MSB                                                     LSB
```

FIGURE 25 - Datagram Message - Extended Header Word Formats

4.3.4.6.1 Non-Acknowledged: The header words for Datagram Messages - Extended Headers with Non-Acknowledged are shown in Figure 26.

```
 MSB                                              LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+------------------------------------------------+
|   AT    | MSG TYPE  | F|         SLAVE ID       | HWA
+------------------------------------------------+
|      INITIAL COUNT OR COUNT REMAINING **        | HWB
+------------------------------------------------+
|                     LABEL                      | HWC0
+------------------------------------------------+
|INITIAL OFFSET OR INIT OFFSET + # DATUM XFER **| HWC1
|------------------------------------------------|
|              EXTENDED HEADER D0                | HWD0
|                       |                        | HWD1
|                       |                        | HWD2
|                       |                        | HWD3
|                       V                        | HWD4
|              EXTENDED HEADER D5                | HWD5
+------------------------------------------------+

** For resume message headers, HWB is the
   number of datum remaining to be transferred
   and HWC1 is initial data offset + number of
   datum transferred.
```

FIGURE 26 - Datagram Message - Extended Header (Non-Acknowledged)

At codes of 000 and 001 designate Non-Acknowledged with the Datagram Message. When a message with Non-Acknowledged is initially transferred to a Slave, the AT code shall be 000. When a suspended message is resumed, the AT code shall be 001. The initial offset in HWC1 is added to the memory location designated by the label to designate the first location in memory where the message is to be transferred. For example, an initial offset of zero designates that the message will be transferred starting at the location designated by the label.

The Datagram (Non-Acknowledged) - Extended Header sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Multiple Slave: Table 38
b. Type 32 EC Multiple Slave: Table 39

TABLE 38 - Datagram Message (Non-Acknowledged) - EH Sequence Type 16 ED Multiple Slave

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State : : : :	Bus State H9	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn
DATA											
D<31..16>	0	0	0	0	0	: : : :	0	0	0	: : : :	0
D<15..0>	HWA	HWB	HWC0	HWC1	HWD0	: : : :	HWD5	0	D0	: : : :	Dn
Source =	M	M	M	M	M	M	M		M	M	M
CYCLE TYPE	H0	H	H	H	H	H	H		D	: : : :	D
Source = M											
Acknowledge	NS	NS	RCG	RCG	RCG	: : : :	RCG	RCG	RCG	: : : :	RCG
Source =			S	S	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	....	0	0	0	: : : :	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 39 - Datagram Message (Non-Acknowledged) - EH Sequence

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn
DATA									
D<31..16>	HWB	HWC1	HWD1	HWD3	HWD5	0	0	: : : :	0
D<15..0>	HWA	HWC0	HWD0	HWD2	HWD4	0	D0	: : : :	Dn
Source =	M	M	M	M	M		M	M	M
CYCLE TYPE	H0	H	H	H	H		D	: : : :	D
Source = M									
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	RCG	: : : :	RCG
Source =			S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	0	: : : :	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES

4.3.4.6.2 Acknowledged: The header words for Datagram Messages - Extended Headers with Data Acknowledge are shown in Figure 27.

AT codes of 100 and 101 designate a Data Acknowledge is to be transferred with the Datagram Message. When a message with an Acknowledge is initially transferred to a Slave, the AT code shall be 100. When a suspended message is resumed, the AT code shall be 101. The initial offset in HWC1 is added to the memory location designated by the label to designate the first location in memory where the message is to be transferred. For example, an initial offset of zero designates that the message will be transferred starting at the location designated by the label.

```
 MSB                                                LSB
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+------------------------------------------------+
|   AT    | MSG TYPE  | F |       SLAVE ID       | HWA
+------------------------------------------------+
|      INITIAL COUNT OR COUNT REMAINING **       | HWB
+------------------------------------------------+
|                     LABEL                      | HWC0
+------------------------------------------------+
|INITIAL OFFSET OR INIT OFFSET + # DATUM XFER ** | HWC1
+------------------------------------------------+
|              EXTENDED HEADER D0                | HWD0
|                      |                         | HWD1
|                      |                         | HWD2
|                      |                         | HWD3
|                      V                         | HWD4
|              EXTENDED HEADER D5                | HWD5
+------------------------------------------------+

** For resume message headers, HWB is the
   number of datum remaining to be transferred
   and HWC1 is initial data offset + number of
   datum transferred.
```

FIGURE 27 - Acknowledged Datagram Message - Extended Header

4.3.4.6.2 (Continued):

The Acknowledged Datagram - Extended Header (EH) sequence of bus states shall be as defined in the following tables:

a. Type 16 ED Multiple Slave: Table 40
b. Type 32 EC Multiple Slave: Table 41

TABLE 40 - Acknowledged Datagram Message - EH Sequence Type 16 ED Multiple Slave

TABLE 40A - Bus State H0 to Bus State Dn

Signal lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State ::::	Bus State H9	Bus State HZ	Bus State D0	Bus State ::::	Bus State Dn
DATA											
D<31..16>	0	0	0	0	0	::::	0	0	0	::::	0
D<15..0>	HWA	HWB	HWC0	HWC1	HWD0	::::	HWD5	0	D0	::::	Dn
Source =	M	M	M	M	M	M	M		M	M	M
CYCLE TYPE	H0	H	H	H	H	H	H		D	::::	D
Source = M											
Acknowledge	NS	NS	RCG	RCG	RCG	::::	RCG	RCG	RCG	::::	RCG
Source =			S	S	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	::::	0	0	0	::::	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 40B - Bus State Dz to Bus State DA0

Signal Lines	Bus State Dz	Bus State DA0
DATA		
D<31..16>	0	0
D<15..0>	0	AWM0
Source =		S
CYCLE TYPE	D	A
Source = M		
Acknowledge	RCG	ACK
Source (7..0) =	S	S
Source (15..8) =	S	S
Source (23..16) =	S	S
Source (31..24) =	S	S
WAIT	0	0
Allowed	YES	YES

TABLE 41 - Acknowledged Datagram Message - EH Sequence Type 32 EC Multiple Slave

TABLE 41A - Bus State H0 to Bus State Dn

Signal Lines	Bus State H0	Bus State H1	Bus State H2	Bus State H3	Bus State H4	Bus State HZ	Bus State D0	Bus State : : : :	Bus State Dn
DATA									
D<31..16>	HWB	HWC1	HWD1	HWD3	HWD5	0	0	: : : :	0
D<15..0>	HWA	HWC0	HWD0	HWD2	HWD4	0	D0	: : : :	Dn
Source =	M	M	M	M	M		M	M	M
CYCLE TYPE	H0	H	H	H	H		D	: : : :	D
Source = M									
Acknowledge	NS	NS	RCG	RCG	RCG	RCG	RCG	: : : :	RCG
Source =			S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	0	: : : :	0
Allowed	NO	NO	YES	YES	YES	YES	YES	YES	YES

TABLE 41B - Bus State Dz to Bus State DA0

Signal Lines	Bus State Dz	Bus State DA0
DATA		
D<31..16>	0	0
D<15..0>	0	AWM0
Source =		S
CYCLE TYPE	D	A
Source = M		
Acknowledge	RCG	ACK
Source (7..0) =	S	S
Source (15..8) =	S	S
Source (23..16) =	S	S
Source (31..24) =	S	S
WAIT	0	0
Allowed	YES	YES

4.3.5 Exception Sequences:

4.3.5.1 Suspend: BIUs shall be capable of supporting suspends as either a Master or Slave. Modules may elect to specify messages as not suspendable as either a Master or Slave. A Suspend sequence shall not be started prior to expiration of Timer A. If a module supports suspends, a Suspend sequence shall be initiated for Block Messages if:

a. The corresponding Header Acknowledge Word has an S of zero
b. For single Slave messages, an AWT of 01 or 10
c. The message data remaining to be asserted on the bus is greater than or equal to 8 data transfer cycles

or for Datagram Messages if:

a. The message data remaining to be asserted on the bus is greater than or equal to 8 data transfer cycles.

NOTE: It is optional to initiate a Suspend sequence if it is legal and there are between 3 and 7 data transfer cycles remaining to be asserted on the bus.

After completing the Suspend sequence, the current bus Master shall release the bus to initiate the Idle state. If Timer B expires, the message shall be aborted and at the end of the Abort sequence, the Master shall immediately release the bus to the Idle state.

The following applies to all but a Non-Acknowledged Datagram Message. If there has been an error, other than a correctable line error, during any part of the message sequence and if a Suspend sequence is attempted, or if an uncorrectable line error occurs during the Suspend sequence, the Master shall complete the Suspend sequence to obtain the Data Acknowledge Words and then the Master shall Abort the message. The message shall not be resumed.

A Non-Acknowledged Datagram Message with errors shall be allowed to be suspended and resumed. However, any Slave detecting an error shall report the error to its device and shall mark its label as nonsuspended in accordance with the Error Tables.

The Suspend sequence may be used in conjunction with the Vie Interval A and Vie Interval B Registers to optimize the tradeoff between short bus acquisition latency and large message sizes by allowing a Block Message or Datagram Message to be interrupted and resumed at a later time.

NOTE: Since suspends are time-consuming, a good systems philosophy is to minimize the use of them.

4.3.5.1.1 Single Slave Suspend: The Suspend sequence of bus states for single Slave Block Messages shall be as defined in the following tables:

a. Type 16 ED Single Slave, Short Data: Table 42
b. Type 32 EC Single Slave, Short Data: Table 43
c. Type 16 ED Single Slave, Extended Data: Table 44
d. Type 32 EC Single Slave, Extended Data: Table 45

The Short Data sequence shall be used if the AWT field of the single Slave Acknowledge Word from the suspended message was 01. The Extended Data sequence shall be used if the AWT field was 10.

In terms of the data transfer operation, the S cycles of these sequences shall be considered normal Data cycles and all Bus Interfaces shall respond accordingly. The Slave shall post ACK during the last of the three S cycles to indicate that the Suspend sequence has been recognized.

Beginning on the fourth cycle of the Suspend sequence, the Slave shall transmit to the bus Master implementation/device specific Resume Control Word (as defined below) that the Master shall store for later use in resuming the message. The Suspend - Short Data sequences transfer two Resume Control Words and the Suspend - Extended Data sequences transfer eight Resume Control Words. The Resume Control Words shall be followed by a Data Acknowledge Word.

For Block Messages this specification allows the following combinations of short and extended headers with short and extended resume sequences:

a. Short Header - Short Resume: RCW0 and RCW1 shall be supplied by the Slave. The Master shall save the two RC words received from Slave. For label addressing RCW0 shall be defined to be the label, and RCW1 shall be defined to be the sum of offset and number of datum previously transferred. For direct addressing the RC words shall be defined to be the next 32-bit address to be used. RCW0 shall contain the least significant 16-bits of the address and RCW1 shall contain the most significant 16-bits.

b. Extended Header - Extended Resume: RCW0-RCW7 shall be supplied by the Slave (8 words). The Master shall save the eight RC words received from Slave. For label addressing RCW0 shall be defined to be the label; RCW1 shall be defined to be the sum of offset and number of datum previously transferred; and RCW2-RCW7 shall be defined to be the same as HWD0-HWD5 in the original header. For direct addressing RCW0 and RCW1 shall be defined to be the next 32-bit address to be used. RCW0 shall contain the least significant 16-bits of the address and RCW1 shall contain the most significant 16-bits. RCW2-RCW7 shall be defined to be the same as HWD0-HWD5 in the original header.

c. Extended Header - Short Resume shall be treated as a nonsuspendable message.

d. Short Header - Extended Resume shall be treated as a nonsuspendable message.

TABLE 42 - Suspend - Short Data Sequence - Type 16 ED Single Slave

Signal Lines	Bus State Di	Bus State Di-1	Bus State Di-2	Bus State RD0	Bus State RD1	Bus State DZ	Bus State DA0
DATA							
D<31..16>	0	0	0	0	0	0	0
D<15..0>	Di	Di -1	Di -2	RCD0	RCD1	0	AWS
Source =				S	S		S
Read Source =	S	S	S				
Write Source =	M	M	M				
CYCLE TYPE	S	S	S	D	D	D	A
Source = M							
Acknowledge	RCG	RCG	ACK	RCG	RCG	RCG	ACK
Source =	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES
Source =							

TABLE 43 - Suspend - Short Data Sequence - Type 32 EC Single Slave

Signal Lines	Bus State Di	Bus State Di-1	Bus State Di-2	Bus State RD0	Bus State DZ	Bus State DA0
DATA						
D<31..16>	DiH	Di-1H	Di-2H	RCD1	0	0
D<15..0>	DiL	Di-1L	Di-2L	RCD0	0	AWS
Source =				S		S
Read Source =	S	S	S			
Write Source =	M	M	M			
CYCLE TYPE	S	S	S	D	D	A
Source = M						
Acknowledge	RCG	RCG	ACK	RCG	RCG	ACK
Source =	S	S	S	S	S	S
WAIT	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES
Source =						

TABLE 44 - Suspend - Extended Data Sequence - Type 16 ED Single Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2	Bus State RD0	Bus State RD1	Bus State : : : :	Bus State RD7	Bus State DZ	Bus State DA0
DATA									
D<31..16>	0	0	0	0	0	: : : :	0	0	0
D<15..0>	Di	Di+1	Di+2	RCD0	RCD1	: : : :	RDC7	0	AWS
Source =				S	S	S	S		S
Read Source =	S	S	S						
Write Source =	M	M	M						
CYCLE TYPE	S	S	S	D	D	: : : :	D	D	A
Source = M									
Acknowledge	RCG	RCG	ACK	RCG	RCG	: : : :	RCG	RCG	ACK
Source =	S	S	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	: : : :	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES
Source =									

TABLE 45 - Suspend - Extended Data Sequence - Type 32 EC Single Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2	Bus State RD0	Bus State RD1	Bus State RD2	Bus State RD3	Bus State DZ	Bus State DA0
DATA									
D<31..16>	DiH	Di+1H	Di+2H	RCD1	RCD3	RCD5	RCD7	0	0
D<15..0>	DiL	Di+1L	Di+2L	RCD0	RCD2	RCD4	RCD6	0	AWS
Source =				S	S	S	S		S
Read Source =	S	S	S						
Write Source =	M	M	M						
CYCLE TYPE	S	S	S	D	D	D	D	D	A
Source = M									
Acknowledge	RCG	RCG	ACK	RCG	RCG	RCG	RCG	RCG	ACK
Source =	S	S	S	S	S	S	S	S	S
WAIT	0	0	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES
Source =									

4.3.5.1.2 Multiple Slave Suspend: The Suspend sequence of bus states for multiple Slave Block Messages and Datagram Messages shall be as defined in the following tables:

a. Type 16 ED Block Message Multiple Slave: Table 46
b. Type 32 EC Block Message Multiple Slave: Table 47
c. Type 16 ED Non-Acknowledged Datagram Multiple Slave: Table 48
d. Type 32 EC Non-Acknowledged Datagram Multiple Slave: Table 49
e. Type 16 ED Acknowledged Datagram Multiple Slave: Table 50
f. Type 32 EC Acknowledged Datagram Multiple Slave: Table 51

In terms of the data transfer operation, the S cycles of these sequences respond accordingly. The Slaves shall post ACK during the last of the three S cycles to indicate that the Suspend sequence has been recognized. For Block Messages and Acknowledged Datagram Messages, a DZ cycle and a multiple Slave Data Acknowledge sequence shall follow the third S cycle. For Non-Acknowledged Datagram Messages, a Suspend sequence ends with the last of the three S cycles.

For Block Messages and Datagram Messages, this specification allows the following combinations of short and extended headers with short and extended resume sequences:

a. Short Header - Short Resume: The equivalent of RCW0 and RCW1 for single Slave Suspend shall be supplied by the Master. For label addressing RC words shall be defined to be label and the sum of offset and number of datum previously transferred. For direct addressing RC words shall be defined to be the next 32-bit address to be used.

b. Extended Header - Extended Resume: RCW0-RCW7 shall be supplied by the Master. For label addressing RCW0 shall be defined to be the label; RCW1 shall be defined to be the sum of offset and number of datum previously transferred; and RCW2-RCW7 shall be defined to be the same as HWD0-HWD5 in the original header. For direct addressing RCW0 and RCW1 shall be defined to be the next 32-bit address to be used and RCW2 RCW7 shall be defined to be the same as HWD0-HWD5 in the original header.

c. Extended Header - Short Resume: This is not possible for multiple Slave since no AWT code is returned by the Slaves.

d. Short Header - Extended Resume: This is not possible for multiple Slave since no AWT code is returned by the Slaves.

TABLE 46 - Suspend - Type 16 ED Block Message Multiple Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2	Bus State DZ	Bus State DA0	Bus State DA1	Bus State DA2	Bus State DA3	Bus State DA4
DATA									
D<31..16>	0	0	0	0	0	0	0	0	0
D<15..0>	Di	Di+1	Di+2	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M		S	S	S	S	S
CYCLE TYPE	S	S	S	D	A	A	A	A	A
Source = M									
Acknowledge	RCG	RCG	ACK	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =	S	S	S	S	S	S			
Source (15..8) =	S	S	S	S	S		S		
Source (23..16) =	S	S	S	S	S			S	
Source (31..24) =	S	S	S	S	S				S
WAIT	0	0	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 47 - Suspend - Type 32 EC Block Message Multiple Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2	Bus State DZ	Bus State DA0	Bus State DA1	Bus State DA2	Bus State DA3	Bus State DA4
DATA									
D<31..16>	DiH	Di+1H	Di+2H	0	0	0	0	0	0
D<15..0>	DiL	Di+1L	Di+2L	0	AWM0	AWM1	AWM2	AWM3	AWM4
Source =	M	M	M		S	S	S	S	S
CYCLE TYPE	S	S	S	D	A	A	A	A	A
Source = M									
Acknowledge	RCG	RCG	ACK	RCG	ACK	ACK (NS)	ACK (NS)	ACK (NS)	ACK (NS)
Source (7..0) =	S	S	S	S	S	S			
Source (15..8) =	S	S	S	S	S		S		
Source (23..16) =	S	S	S	S	S			S	
Source (31..24) =	S	S	S	S	S				S
WAIT	0	0	0	0	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES	YES	YES	YES	YES

TABLE 48 - Suspend - Type 16 ED Non-Acknowledged Datagram Multiple Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2
DATA			
D<31..16>	0	0	0
D<15..0>	Di	Di+1	Di+2
Source =	M	M	M
CYCLE TYPE	S	S	S
Source = M			
Acknowledge	RCG	RCG	ACK
Source (7..0) =	S	S	S
Source (15..8) =	S	S	S
Source (23..16) =	S	S	S
Source (31..24) =	S	S	S
WAIT	0	0	0
Allowed	YES	YES	YES

TABLE 49 - Suspend - Type 32 EC Non-Acknowledge Datagram Multiple Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2
DATA			
D<31..16>	DiH	Di+1H	Di+2H
D<15..0>	DiL	Di+1L	Di+2L
Source =	M	M	M
CYCLE TYPE	S	S	S
Source = M			
Acknowledge	RCG	RCG	ACK
Source (7..0) =	S	S	S
Source (15..8) =	S	S	S
Source (23..16) =	S	S	S
Source (31..24) =	S	S	S
WAIT	0	0	0
Allowed	YES	YES	YES

TABLE 50 - Suspend - Type 16 ED Acknowledged Datagram Multiple Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2	Bus State DZ	Bus State DA0
DATA					
D<31..16>	0	0	0	0	0
D<15..0>	Di	Di+1	Di+2	0	AWM0
Source =	M	M	M		S
CYCLE TYPE	S	S	S	D	A
Source = M					
Acknowledge	RCG	RCG	ACK	RCG	ACK
Source (7..0) =	S	S	S	S	S
Source (15..8) =	S	S	S	S	S
Source (23..16) =	S	S	S	S	S
Source (31..24) =	S	S	S	S	S
WAIT	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES

TABLE 51 - Suspend - Type 32 EC Acknowledged Datagram Multiple Slave

Signal Lines	Bus State Di	Bus State Di+1	Bus State Di+2	Bus State DZ	Bus State DA0
DATA					
D<31..16>	DiH	Di+1H	Di+2H	0	0
D<15..0>	DiL	Di+1L	Di+2L	0	AWM0
Source =	M	M	M		S
CYCLE TYPE	S	S	S	D	A
Source = M					
Acknowledge	RCG	RCG	ACK	RCG	ACK
Source (7..0) =	S	S	S	S	S
Source (15..8) =	S	S	S	S	S
Source (23..16) =	S	S	S	S	S
Source (31..24) =	S	S	S	S	S
WAIT	0	0	0	0	0
Allowed	YES	YES	YES	YES	YES

4.3.5.1.3 Resuming Suspended Messages: A Master shall resume a suspended Block Message or suspended Datagram Message by transmitting the remaining data to the Slave(s) with bit 13 of the HWA AT field set to 1. The Resume Control Words that the Slave(s) require to resume that message shall be sent to the Slave(s) during the appropriate header cycles as described in 4.3.4.2, 4.3.4.3, 4.3.4.5, and 4.3.4.6. For a single Slave message and a multiple Slave message, the Resume Control Words shall be as defined in 4.3.5.1.1 and 4.3.5.1.2.

A Block Message - Short Header shall be used to resume a suspended single Slave Block Message which had an AWT field of 01 (two Resume Control Words) and a Block Message - Extended Header shall be used to resume a suspended single Slave Block Message which had an AWT field of 10 (eight Resume Control Words). The type of Block Message or Datagram Message used to resume a multiple Slave message shall be determined as defined in 4.3.5.1.2.

If a BIU Master is to be suspended due to a Bus Request being asserted, then after the BIU Master completes a Suspend sequence, the BIU shall go to Idle. When a message, which was suspended, is to be resumed, the BIU shall revie for the bus.

4.3.5.1.4 Using Suspend With Vie Intervals A and B: The Pi-Bus Vie Interval A and Vie Interval B Registers shall be used as defined in 4.3.5.1.

4.3.5.2 Abort: The Abort sequence may be used to terminate a message prior to completion due to errors or other reasons. No Header or Acknowledge words are required for the Abort sequence. A Master may initiate the Abort sequence at any time during its tenure.

The Abort sequence shall be as shown in Table 52. The first three of the four bus cycles with the Abort Cycle Type are used to give the Slave time to recognize that an Abort has occurred. If no Slaves are present during cycle AB2, a not selected shall be present. For modules which have not ceased being Slaves (see 4.2.3.1.1.1), the Slave (or Slaves in the case of multicast or broadcast) shall respond by posting ACK on the AS lines during the third AB cycle to inform the Master that an Abort has been recognized. There are no Slaves on the fourth AB cycle and the Master is the only module that may assert Wait on that cycle. After the last AB cycle, the Master may continue with the HO of another message or relinquish control of the bus by releasing the bus signal lines.

TABLE 52 - Abort Sequence

Signal lines	Bus State AB0	Bus State AB1	Bus State AB2	Bus State AB3
DATA				
D<31..16>	X[3]	X	0	0
D<15..0>	X	X	0	0
Source =	M or S	M or S		
CYCLE TYPE	AB	AB	AB	AB
Source = M				
Acknowledge	X	X	ACK	NS
Source =	X	X	S	
WAIT	0/1	0/1	0	0/1
Allowed	a[1]	a	NO	b[2]
Source =	a	a		b

[1] a - Wait assertion allowed, but not honored
[2] b - Master Wait allowed, no Slaves exist
[3] x - not defined

4.3.6 Wait: A Master or Slave may assert Wait to control the data transfer rate on the Pi-Bus and thereby accommodate slow or temporarily busy devices. During a multiple Slave sequence, one or more of the Slaves may assert Wait. Each cycle on which Wait is asserted shall cause the insertion of a non-transfer cycle into a sequence on the following cycle.

4.3.6.1 Rules for Asserting Wait: A Bus Interface may assert Wait on one or more cycles subject to the following restrictions:

a. Wait may normally be asserted by a Bus Interface only if that module is scheduled to be a bus Master or Slave on the next transfer cycle. However, the current bus Master may assert Wait if the bus will be placed in the Idle state following the Wait induced non-transfer cycle(s) provided the Vie Interval requirement is not violated. A Slave may assert Wait on the last Acknowledge cycle to permit checking for a late Abort, updating the label table, or storing interrupt data.

b. A Bus Interface shall not assert Wait on H0 or H1 cycles.

c. A Bus Interface shall not assert Wait on a particular cycle (N), if on the second previous cycle (N-2) the module did not assert Wait but Wait was asserted on that cycle. If the Bus Interface does assert Wait on cycle N and Wait was not asserted by any module on cycle N-1, the Bus Interface shall assert Wait for an even number of contiguous cycles.

4.3.6.2 Effects of Wait: Other than during an Abort sequence, for each cycle that a Wait is asserted during a tenure, one non-transfer cycle (NT cycle) shall be inserted into the sequence immediately following the cycle in which Wait is asserted. The insertion of non-transfer cycles shall not change the sequence of scheduled bus states. However, if during the non-transfer cycle, the Master asserts an Abort sequence designation on the Cycle Type lines, the sequence shall be altered to Abort. During an Abort sequence, Wait shall not introduce non-transfer cycles.

4.3.6.2.1 Line Groups During Non-transfer Cycles: During non-transfer cycles produced by Wait, the signal line values shall be as specified below.

4.3.6.2.1.1 Data line Group During Non-transfer Cycles: The value on the Data lines shall be ignored except for line error checking. During a non-transfer cycle any value with correct parity may be posted on the Data line.

The Data Line Group shall not be posted by any module other than the module which would have posted the Data lines had that cycle been the scheduled transfer cycle. If a module asserts a value on the Data Line Group on such an NT cycle, the value shall be a Data symbol with correct parity.

4.3.6.2.1.2 Cycle Type Group During Non-transfer Cycles: The value on the Cycle Type lines shall be ignored except for line error checking and the occurrence of Abort. An Abort Cycle Type shall mark the beginning of an Abort sequence. The Master module responsible for posting the CT lines on the next scheduled transfer cycle shall source a valid symbol on the CT lines during the NT cycle(s). A valid symbol on the CT lines is any Cycle Type with correct parity.

4.3.6.2.1.3 AS Group During Non-transfer Cycles: If a module is responsible for posting the AS lines on the next transfer cycle, it shall post a valid symbol on the AS lines during the NT cycle(s). A symbol other than NAK shall be ignored during non-transfer cycles. NAK shall be posted on cycle N, to signal that an error, other than a correctable line error, associated with cycle N-2 was detected.

4.3.6.2.1.4 Wait Lines During Non-transfer Cycles: The rules for asserting Wait are specified in 4.3.6.1. The effects of asserting Wait on a non-transfer cycle shall be the same as specified herein for asserting Wait on a transfer cycle.

4.3.6.2.1.5 Bus Request Lines During Non-transfer Cycles: The rules for asserting Bus Request are specified in 4.3.3.3. Bus Request is completely independent of Wait.

4.3.7 Data Link and System Time Facilities: This section specifies the Data Link and system Time facilities which shall be accessible over the Pi-bus via the Bus Interface Message described in 4.3.4.4. A Master and Slave shall implement all Data Link facilities as specified in this specification.

4.3.7.1 Data link Register Address Space: The Data Link facilities specified herein shall be contained in the Data Link register address space which shall consist of 256 words assigned to consecutive addresses. This register space shall be allocated as shown in Table 53. Access to these facilities over the bus shall be allowed only via the Bus Interface Message with the Header Word A Access Type field set to zero (AT=000).

Reserved registers shall not be implemented and any attempt to access them shall cause a ’Resource Not Present’ error. For write operations to defined registers, the state of any bits in the word which corresponds to reserved bits shall be ignored.

TABLE 53 - Data link Address Space

Address	Register
255	Logical Slave
.	Identification
.	Registers
33	(33 - 255)
32	
.	Reserved Registers
.	(6 - 32)
6	
5	Vie Priority Register
4	Vie Interval B Register
3	Vie Interval A Register
2	Module Capabilities Register
1	Control Register
0	Reserved Register

4.3.7.2 Register Protection: Write access to the Data link registers via the Bus Interface Message shall be limited by write protection. Either permanent or programmable write protection shall be provided for each register as specified in the following sections. The current state of programmable write protect must be controlled from the device. On reset, all programmable protection must be placed in the nonprotect state. The BIU shall support programmable write protection.

4.3.7.3 Registers: Data received with an uncorrectable Data line error or Cycle Type sequence error shall not be transferred to a Data link layer register.

4.3.7.3.1 Reserved Register - Address 0: The use of the Reserved Register shall be defined only by future versions of this specification. Until then, this register shall not be implemented and shall cause a ’Resource Not Present’ error if an access is attempted.

4.3.7.3.2 Control Register - Address 1: The Control Register shall be a 2-bit register that shall contain a bit to initiate Bus Interface reset and a bit to initiate built-in test (BIT). The Data link Control Register - address 1, shall be permanently write protected.

The following requirements shall apply to the Control Register:

a. Word format: Figure 28
b. Write protect: Permanent
c. State after reset: All zeroes
d. Reserved bits: Read as zeros

```
+-----------------------------------------------------+
| 15| 14|13|12|11|10| 9| 8 | 7| 6| 5| 4| 3| 2| 1| 0|
+-------------------------------------------|--|--|
                RESERVED (ZEROS)            |  |  |
____________________________________________|  |  |
                                               |  |
           INITIATE BUILT-IN-TEST              |  |
             0 = NO INITIATE **                |  |
             1 = INITIATE                      |  |
_______________________________________________|  |
                                                  |
           BUS INTERFACE RESET                    |
             0 = NO RESET *                       |
             1 = RESET                            |
__________________________________________________|
 * - Placed in 0 state at completion of reset.
** - Placed in 0 state at completion of built-in-test.
```

FIGURE 28 - Control Register Word Format

4.3.7.3.3 Module Capabilities Register - Address 2: The Module Capabilities Register shall be a 4-bit register that shall specify the currently configured capabilities of the Bus Interface/Device Combination.

The following requirements shall apply to the Module Capabilities register:

a. Word format: Figure 29
b. Write protect: Permanent
c. State after reset: Appropriate capabilities code
d. Reserved bits: Read as zeros

```
+-------------------------------------------------+
| 15| 14|13|12|11|10| 9| 8 | 7| 6| 5| 4| 3| 2| 1| 0|
+-----------------------------------|--|--|--|--|
                                    |  |  |  |  |
  RESERVED (ZEROS)                  |  |  |  |  |
____________________________________|  |  |  |  |
                                       |  |  |  |
  MODE        0 = Non-Mixed Mode       |  |  |  |
              1 = Mixed Mode           |  |  |  |
_______________________________________|  |  |  |
                                          |  |  |
  FEATURE     0 = SLAVE ONLY              |  |  |
              1 = MASTER/SLAVE            |  |  |
__________________________________________|  |  |
                                             |  |
  CLASS       0 = ED (ERROR DETECT)          |  |
              1 = EC (ERROR CORRECT)         |  |
_____________________________________________|  |
                                                |
  TYPE        0 = TYPE 16 (16-BIT TRANSFERS)    |
              1 = TYPE 32 (32 OR 16-BIT TRANSFERS)  |
________________________________________________|
```

FIGURE 29 - Module Capabilities Register Word Format

4.3.7.3.4 Vie Interval A Register - Address 3. The Vie Interval A Register shall be a 16-bit register that shall contain the unsigned binary Vie Interval A timeout value expressed in bus cycles. The value in this register shall remain unchanged until a new timeout value is loaded. Reading this register shall return the last value written.

The following requirements shall apply to the Vie Interval A Register:

a. Word format: Figure 30
b. Write protect: Programmable
c. State after reset: All ones

```
+-------------------------------------------------+
| 15| 14|13|12|11|10| 9| 8 | 7| 6| 5| 4| 3| 2| 1| 0|
+-------------------------------------------------+
MSB <----------- VIE INTERVAL A VALUE------------> LSB
```

FIGURE 30 - Vie Interval A Register Word Format

4.3.7.3.5 Vie Interval B Register - Address 4: The Vie Interval B Register shall be an 8-bit register that shall contain the unsigned binary Vie Interval B timeout value expressed in bus cycles. The value in this register shall remain unchanged until a new timeout value is loaded. Reading this register shall return the last value written.