Digital's Storage Systems Interconnect

ECO Controlled Version 1.0.0

RELEASE VERSION

| 27 March 1990

Send inquiries and comments to SSAG::SSAG.

Abstract

This document defines the Digital Storage Systems Interconnect (DSSI). DSSI is based on the higher layer protocols of the CI (DEC STD 161) using a different data-link and physical interconnect.

CONTENTS

CHAPTER 5 THE DSSI PHYSICAL BUS PROTOCOL

FIGURES

TABLES

1 Preface To Version 1.0.0

This release of the specification is the final form version approved by the DSSI Review Committee on 27 March 1990. All further changes to the document shall be made by ECO.

CHAPTER 1

INTRODUCTION

This specification defines the architecture for the Digital Storage Systems Interconnect (DSSI). The DSSI, supporting the needs of low-end and mid-range systems, is one in a family of high-performance computer-to-computer interconnects (CI's) that combine a common host interface and port layer with an implementation-specific datalink and physical interconnect.

1.1 SCA Overview

CI-class interconnects provide the transmission services required by Digital's Systems Communication Architecture - a four-tiered set of protocols and interfaces as shown in figure 1-1.

```
                Node                              Node
          +----------------+                +----------------+
          |  System        |    SYSAP       | System         |
          |  Application   |<-------------->| Application    |
          |  ( SYSAP )     |    Protocol    | ( SYSAP )      |
          +----------------+                +----------------+
          | System         |    SCS         | System         |
          | Communications |<-------------->| Communications |
          | Services       |    Protocol    | Services       |
          | ( SCS )        |                | ( SCS )        |
          +----------------+    PPD         +----------------+
          | Port/Port      |<-------------->| Port/Port      |
          | Driver         |    Protocol    | Driver         |
          | Layer          |                | Layer          |
          +----------------+                +----------------+
                  |                                  |
                  -------------/......../------------
                          Physical Interconnect
```

Figure 1-1: The SCA Model

This architecture consists of the following layers:

- o System Application (SYSAP) - A process whose services and protocol are specific to a given application. The disk class driver is an example of such a process. It's protocol is the Mass Storage Control Protocol (MSCP).

- o Systems Communications Services (SCS) - Provides facilities for establishing and maintaining communications between two Sysaps.

- o Port/Port Driver Layer (PPD) - A combination of hardware and software that provides the host with a device-independant. packet-oriented interface for node-to-node data transfers.

- o Physical Interconnect - The physical media and signalling protocols used to transmit the raw bit stream.

The basic function of the PPD layer is the delivery of packets to the receiving node that are free of detectable errors. The PPD provides the following services:

- Datagram service - The delivery of independant packets to another node with high probability of success. Datagrams may be discarded if buffers to accommodate them are not available in the receiving node.

- Virtual Circuit Services - The transfer of data between nodes without duplication or loss. Virtual circuits are established between node pairs and support the following classes of service:

    1. Message Service - The delivery in correct sequence of one or more packets comprising a message.

    2. Buffered Data Service - The efficient transfer of large blocks of data directly to or from an application's memory. This service is often performed with the assistance of dedicated hardware.

A CI implementation consists of the physical interconnect, combined with hardware and software within the PPD layer, that implements SCA packet tranmission services. The following section presents an overview of this layer.

1.2 CI PPD Layer

Figure 1-2 illustrates the layers of a PPD implementation built on a CI-class interconnect. The specifications controlling each functional element are listed in section 1.5.

```
                                                      <-----+
                    +---------------------+                 |
                    | Port Driver         |                 |
                    | Layer               |                 |
                    | ( SCA Specification)|                 |
                    +---------------------+                 |
                              |                             |
                              |                             |
                    .......................                 |
                    . CI Port Adapter     .                 |
                    . ( eg: VAX CI Port ) .                 |
                    .......................                 |
                              |                             | P
  +------>                    |                             | P
  |                 +---------------------+                 | D
  |                 |     CI Port         |                 |
C |                 |     Layer           |                 | L
I |                 |   ( DEC Std 161 )   |                 | A
  |                 +---------------------+                 | Y
A |                    |               |                    | E
R |             +------+               +------+             | R
C |             |                             |             |
H |  +----------------------+      +---------------------+  |
I |  | DSSI Datalink        |      |    CI Datalink      |  |
T |  |    Layer             | .... |      Layer          |  |
E |  | ( DSSI Spec. )       |      |  ( Dec Std 161 )    |  |
C |  +----------------------+      +---------------------+  |
T |             |                             |       <-----+
U |             |                             |       <-----+ P L
R |  +----------------------+      +---------------------+  | H A
E |  |    DSSI Physical     |      |    CI Physical      |  | Y Y
  |  |    Interconnect      |      |    Interconnect     |  | S E
  |  |    ( DSSI Spec. )    |      |   ( DEC Std 161 )   |  | I R
  |  +----------------------+      +---------------------+  | C
  +------>                                            <-----+ A
                                                              L
```

Figure 1-2: CI PPD Architectural Layers

The topmost layer, the port driver, provides a device-independant interface to the data transfer services provided by lower layers. It is also responsible for managing the startup and orderly shutdown of virtual circuits.

The CI port adapter is the hardware interface that performs the physical transfer of packets between host memory and the CI port. The architecture of such hardware interfaces is outside the scope of SCA. In some implementations, this function is performed by host software.

The port implements the protocols that provide the datagram, block data and message services defined by the architecture. It is responsible for error recovery, the detection of lost packets within a message or block data transfer and the sequential delivery of message packets.

The Datalink layer is responsible for:

1. Channel control including channel access arbitration.

2. The transformation of outgoing packets into bit streams for transmission over the physical medium through the addition of addressing data, framing information and error-detection codes.

3. Recovery from transmission errors.

4. The conversion of incoming data streams into packets in memory free of detectable errors. This includes the detection and removal of framing information, address detection, error detection and receipt acknowledgment.

The Physical Interconnect layer consists of the line drivers, wires, bus arbitration and electrical signalling rules that control the physical transfer of data over the bus.

As indicated in the diagram, the CI architecture consists of a common port protocol, defined in DEC standard 161, coupled with unique datalink and physical channel layers.

1.3 Overview Of The DSSI Bus

This section presents an overview of the DSSI bus and its design goals in comparision to the CI.

The DSSI is designed to support low-end and mid-range systems, including VAX clusters, where an application can tolerate a reduction in availability and performance in exchange for reduced cost. The main features of DSSI are:

- Multidrop electrical topology

- o 8-bit parallel data bus.
- o DC coupling between nodes
- o Use of low-cost, single-ended drivers and receivers.
- o Restricted length (6 meters maximum) suitable for one-, two-, or three-box systems.
- o Single electrical path.

In contrast, the CI is designed to support the needs of high-end systems where availability and performance are the primary goals. The main features of the CI are:

- o "Star" electrical topology to isolate the bus from single-point node failures,
- o Passive star coupler for reliability
- o Active, redundant-element star coupler for large arrays of intelligent subsystems
- o Bit-serial signalling to simplify star coupler design and facilitate inter-system isolation.
- o Dual-path for redundancy
- o End-to-end length suitable for interconnecting large systems in a 'computer room' environment.

The following table summarizes the characteristics of each interconnect:

	CI	DSSI
Format of transmitted Data:	Bit-serial, embedded clock.	8-bit parallel, 'synchronous burst'
Data Bandwidth: (unformatted)	70Mbits/sec	4Mbytes/sec (~32Mbits/sec)
Maximum Number of Nodes per Datalink:	16 (Passive Hub) 224 (Active Exchange)	8
Data Format:	Packets, 7 - 4100 bytes (excluding bit synch, header and trailer)	Packets, 2 - 4114 bytes
Distance:	45 meters max from center of hub, 90 meters node-to-node.	6 meters end-to-end
Medium:	2 send-receive coaxial cable pairs	Single, 50-conductor cable with multiple taps.
Electrical Interface:	Transformer-coupled line drivers for high dc isolation.	DC-coupled, single-ended line drivers.

Figure 1-3: Comparison of CI and DSSI features

1.4 Scope Of This Document

This document is the functional specification of the DSSI bus. Its purpose is to specify the interconnect to the level of detail necessary to build a DSSI node and configure a series of nodes into an operational system.

The following elements are specified:

1. The DSSI Data Link Protocol.

2. The physical bus protocol including signal definitions and timing.

3. The bus electrical requirements including the specification for line drivers and receivers and rules for signal integrity.

4. The mechanical requirements, including the definition of connectors, cables and mounting rules.

5. The electrical and mechanical rules for configuring a system of DSSI nodes.

1.5 References

The following documents may be useful in fully understanding DSSI and its relationship with Digital's Storage Architecture (DSA) and other I/O architectures.

1. Mass Storage Control Protocol (MSCP); Revision 2.1.1 (13 March 1988)

2. Tape Mass Storage Control Protocol (TMSCP); Revision 2.0.2 (8 November 1987)

3. Diagnostics and Utilities Protocol (DUP); Revision 1.0 (May 15, 1984)

4. Systems Communications Architecture (SCA); Revision 7 (August 8, 1985)

5. Computer Interconnect Specification (DEC STD 161); Revision A (November 24, 1986)

6. VAX-11 CI Port Architecture (CIPORT); Revision 5.0 (August 17, 1987)

7. Storage Systems Port; Revision 3.1.0 (November 9, 1989)

CHAPTER 2

TERMINOLOGY AND NOTATIONAL CONVENTIONS

Active Node

During an atomic bus sequence, the node that controls the bus and drives bus-level commands. Either the initiator or target will assume the role of active node. See 'Passive Node'.

Asynchronous Mode

A mode of DSSI data transfer that requires an explicit send/receive handshake for each byte transferred.

Atomic Bus Sequence

The sequence of DSSI bus phases associated with the transmission of one data block from an initiator to a target device. The sequence is performed atomically with respect to other nodes on the bus in that bus ownership and control remain with the target-initiator pair until the sequence completes.

Bus Phase

The sequence of bus states comprising one step in an Atomic Bus Sequence. The ARBITRATION phase, through which a node bids for ownership of the bus, is one such phase.

Command Descriptor Block

Six bytes of control information (plus checksum) transmitted to the target during the DSSI COMMAND OUT phase. This block contains parameters used to control the transmission of information during the DATA OUT phase.

Data Block

The block of information (with checksum) transmitted from the initiator to the target during the DATA OUT phase of a DSSI bus transaction.

DSSI Node

The site of a DSSI datalink. The terms 'node' and 'DSSI node' are used synonymously in this specification.

DSSI Packet

The Command Out descriptor block and Data block transmitted by the initiator during an atomic bus sequence.

Hot-swap

The act of physically connecting or disconnectiong a node from the bus while the node is powered on.

Initiator

A DSSI node that has arbitrated for the bus with the intention of sending a data packet.

May

Denotes a lack of prohibition when used as a directive. It is used to indicate an option or options architecturally permissible for an implementation. No bias for or against the option is implied. "May not" denotes strict prohibition when used as a directive.

Node

See 'DSSI node'.

Nugatory

Of little or no consequence: Trifling, Inconsequential.

Passive Node

During a bus phase, the DSSI node that reacts to bus-level commands. For each phase of an atomic bus sequence either the initiator or target will assume the role of passive node. See 'Active Node'.

Physical Bus Protocol

The electrical signalling rules and timing required for a DSSI node to properly transmit control signals and data over the DSSI physical interconnect.

Shall

Denotes an architectural requirement. It is used as a directive to express what is mandatory for an implementation. "Shall not" denotes strict prohibition.

Should

Denotes an emphatic recommendation when used as a directive. It is used where "shall" might be too strong because of the possibility of technical, implementation-specific obstacles or other overriding considerations.

Synchronous Mode

Synonymous with "synchronous burst mode".

Synchronous Burst Mode

A mode of data transmission over the DSSI bus in which data bytes are transmitted in bursts, without waiting for an explicit send/receive "handshake" after each byte. This mode requires that each DSSI implementation have a dedicated hardware buffer (FIFO) that is tightly coupled to the bus.

Target

A DSSI node that has been signalled by an initiator, via the physical bus protocol, that it is to receive a packet.

Warm-swap

The act of physically disconnectiong or connecting a node to the bus while node power is off.

2.1 Notational Conventions

2.1.1 Numeric Notation

All numbers are decimal unless otherwise specified. Hexadecimal numbers are noted by a suffix of "h", for example, 1Ah.

2.1.2 Time Unit Abbreviations

The following time unit abbreviations are used:

1. ms - Milliseconds

2. us - Microseconds

3. ns - Nanoseconds

2.1.3 Bit Field Notation

A string of contiguous bits is described by the notation:

field_name< m:n >

where:

"field_name" is an ascii string describing the function or field

"m" is the bit position of the left-most (highest order) bit in the field.

"n" is the bit position of the right-most (lowest order) bit in the field.

2.1.4 Naming Conventions For Variables

Throughout this specification, variable names designating parameters and function values, are constructed in the form:

<xx>_<value_mnemonic>

Where:

"<xx>" is a two-character abbreviation for the layer or function that establishes the variable's value and "<value_mnemonic>" is a string suggestive of the parameter or condition represented.

Variables associated with the physical bus always have a null prefix string.

So, for example, the name "CC_NAK" represents a value returned by the datalink Channel Control layer, whereas "NAK" represents a value returned over the DSSI bus during the STATUS IN phase.

CHAPTER 3

PORT ADDITIONS AND/OR CLARIFICATIONS

3.1 Port Additions And/or Clarifications

This section clarifies or adds to the Port section of DEC STD 161. These changes are due to past experience with the CI, or due to different requirements of the underlying DSSI Physical Channel.

3.2 Unrecognized Packets

When an unrecognized packet (unrecognized port layer opcode, or invalid packet length) is received by a DSSI node, it should close the virtual circuit to the corresponding node.

3.3 Self Directed Commands

All DSSI adapters other than KFQSA should support the self directed commands (CI Port commands with Port Field the same as their own node number). Two SYSAP processes within a system may communicate with each other through self directed commands. They should be supported with all variations of the specific commands (such as R bit set or clear) like all other non-self directed commands. It is further recommended that they should be implemented to use the same code paths (to the extent possible) as non-self directed commands. Such an approach will provide a loop back mechanism and help diagnosis.

3.4 Loop Back Packets

CI Port Architecture and DEC STD 161 support loop back packets on the CI wires. The SNDLB command (Sec 7.8, Ref [6]) requires that the packet should be transmitted and received on the wires at the same time. CI data link supports such a feature. However DSSI (more specifically the SII) does not support such feature. However DSSI adapters other than KFQSA should treat the SNDLB commands similar to the self directed commands.

3.5 Multibit Sequence Numbers

CI port layer uses single bit sequenced numbers in all the sequenced message packets (refer to section 4.1 in DEC STD 161).

Single bit sequence numbers have the problem that if a single NAK status turns into an ACK, (i.e., the receiver sent a NAK and the sender received the status as ACK due to 8 bit errors in 9 bit DSSI status), two sequenced message packets will be discarded at the receiver and both nodes will be unaware of the lost packets. To overcome the above problem DSSI uses 3 bit sequence numbers. All references to NS and NR in DEC STD 161 and CI Port Architecture should be treated as 3 bit fields. The sequence number field, NS, is located in bits <3:1> of the FLAGS byte.

When a frame is received, a DSSI node should use the following rule for validating sequence numbers:

1. If NS = NR, the node should accept the frame and increment NR.

2. If NS = NR-1, the node should discard the frame, and should not increment NR.

3. With any other combination of NS and NR, the node should break VC, and report a sequence number mismatch error to higher layers.

3.6 CI ID Response Packet Fields

Sec 7.9.3 of CI Port Architecture (Ref[6]), describes ID Response packet format. The fields relevant to DSSI are in this section.

The only sub-field of PORT_FCN_EXT that is currently meaningful for DSSI devices is MAX_BODY_LEN, which is the low 13 bits of the high order word of the PORT_FCN_EXT longword. The contents of this field are the maximum number of bytes in a packet BODY supported by the sending node. The packet body comprises the data bytes passed during the DSSI data phase --- the packet opcode byte, flags byte, and other data bytes, but not the checksum.

It should be noted that the MAX_BODY_LEN value reported by a DSSI node must reflect the size of the buffers on the SII receive queue (target queue).

Nodes that do not have internal data buffers (SHAC) must always report the maximum size, (4114).

Block data transfer packets include 18 bytes of overhead in addition to data --- packet opcode (1), flags (1), transaction ID (8), buffer name (4), and offset (4). Thus transferring 4K bytes of data per packet requires that MAX_BODY_LEN be 4114 (4096+18) or larger.

DSSI nodes should return zero in all other currently defined sub-fields of PORT_FCN_EXT. XNR, AARB, XPRE, and AST report CI data link parameters that are inapplicable to DSSI. A CSZ (cluster size) value of zero means 16 nodes, which is the closest value available to the 8 nodes supported by DSSI.

DSSI nodes must ignore all the inapplicable fields of the PORTFCNTEXT field (as well as other fields of the ID response packet). They should ignore whatever values they receive in the XNR, AARB, XPRE, AST and MBZ (Must Be Zero) fields. They should also ignore the CSZ (cluster size) field.

DSSI nodes should use the contents of the MAX_BODY_LEN to determine the maximum size of a block data transfer. Specifically, DSSI disk and tape drives should do the following:

1. The drive has its own MAX_BODY_LEN that corresponds to its native or preferred data transfer size --- that is, to the size of its data buffers. While it is strongly recommended that this be 4114 (4K data bytes per packet), smaller values can be justified so long as their effect on system performance (not just device cost or performance) is considered and verified (through simulation, performance testing, etc.).

2. The drive determines the host's MAX_BODY_LEN from the configuration polling process.

3. For each host, the drive determines an effective MAX_BODY_LEN equal to the minimum of the drive's MAX_BODY_LEN and the host's MAX_BODY_LEN.

4. DSSI drives may only initiate data byte transfers with base packet size of 512 bytes (P=0).

5. The drive determines the packet multiple (M value) to use for the block data transfers that the drive initiates on each host as follows:

```
  if (effective MAX_BODY_LEN < 530) then begin
```

```
        { host either does not implement the CI ID response ECO,   }
        { in which case MAX_BODY_LEN is zero, or the host returned }
        { a MAX_BODY_LEN value smaller than one 512 byte block.    }
        { In either case use M=0, or one 512 byte block per packet.}
        M := 0;
    end else begin
        M := (effective MAX_BODY_LEN - 18) div 512 - 1;
        { div means integer division with truncation towards zero. }
        if M < 0 then M := 0;
        if M > 7 then M := 7;
    end;
```

The drive supplies this M value in all DATREQn packets and uses it to limit the size of all SNTDAT packets. It should be noted that the drive determines a separate value of M for each host.

DSSI host nodes should do the following:

1. All native DSSI host nodes (all host ports that attach to a DSSI cable or bus) should return a MAX_BODY_LEN value of 4114 in ID response packets.

   DSSI host nodes should also support packets with body lengths up to 4114 bytes.

2. All DSSI host nodes shall support 512 byte block packets (P=0).

3. DSSI nodes are not required to support 576 byte block packets. Any such node that does not support 576 byte block packets should verify that P=0 in received DATREQn packets, and treat the packet as invalid (and close the virtual circuit) if P=1.

3.7 Only Single Path In DSSI.

Unlike the CI, DSSI has only a single `path'.

Section 4.3 of DEC STD 161 which specifies path selections is not applicable to DSSI. All senders in DSSI must set the path field(s) so that it appears that path zero (0) is being used for all communication. Receivers may ignore path fields.

CHAPTER 4

THE DSSI DATALINK

4.1 The DSSI Datalink

The datalink layer, the next to lowest tier of DSSI, is responsible for the reliable delivery of single frames of information over the physical channel. To meet the requirements of the port layer the datalink must:

1. deliver sequential, error free text from one DSSI node to another (duplication can occur), and

2. perform error recovery by retransmission, only notifying the Port Layer of failures.

Datalink functions are separated into two areas, higher-level mechanisms for error recovery and packet exchange with the Port layer and a low level Channel Control layer that manages the exchange of packets with the remote node over the physical interconnect.

An overview of the datalink layer and a detailed description of these higher level mechanisms is presented in this chapter. A complete description of the Channel Control layer is given in chapter 5.

4.2 Overview Of A DSSI Data Transfer

The transfer of data packets from one node to another is accomplished by the Channel Control layer through a series of bus phases defined as the DSSI atomic bus sequence. Since bus ownership and control remain with the sending and receiving node pair until the transfer completes, the sequence is atomic relative to other nodes on the bus.

A simplified diagram of this sequence, omitting exception paths, is shown below (figure 4-1).

```
                   +--------+
                   |Bus Free|<-----------+
                   +--------+            |
                       |                 |
                       v                 |
                 +-----------+           |
                 |Arbitration|           |
                 +-----------+           |
                       |                 |
                       v                 |
               +----------------+        |
               |   Selection    |        |
               +----------------+        |
                       |                 |
             +--->     v                 |
             |   +-----------+           |
             |   |  Cmd Out  |           |
             |   +-----------+           |
             |         |                 |
Information  |         v                 |
Transfer     |    +----------+           |
Phases       |    | Data Out |           |
             |    +----------+           |
             |         |                 |
             |         v                 |
             | +----------------+        |
             | |    Status In   |--------+
             | +----------------+
             +--->
```

Figure 4-1: Simplified DSSI Atomic Bus Sequence

After detecting the bus free phase, a node wishing to start a transfer must successfully arbitrate for exclusive bus access. The node winning arbitration, referred to as the initiator, then attempts to select a receiving node. When a response to selection is detected, the initiator passes bus ownership to the selected node, or target, who controls the remaining information transfer phases.

The transfer of a data packet requires a three-step exchange of information between initiator and target.

1. During the Command Out phase, the target solicits from the intitiator a 6-byte Command Descriptor Block, containing parameters required by the target's Channel

Control Layer for the Data Out phase

2. During the Data Out phase, the target receives and buffers a Data block containing the actual port layer data packet.

3. During the Status In phase, the target returns a single byte of status to the initiator. One of two values is returned:

   - ACK - Indicates that the data packet was received and buffered successfully,

   - NAK - Indicates a receive failure and requests retransmission.

All data is transmitted via an 8-bit, parity-protected data path. In addition, vertical checksums are appended to the Command Descriptor and Data blocks.

Should an error occur, the initiator's datalink will attempt to recover by resending the packet. If this fails, the initiator's port layer will be notified of an unrecoverable error.

4.3 DSSI Datalink Logical Description

Figure 4-2 is schematic of datalink layer functions. The purpose of this model is to facilitate the description of essential datalink functions. It is not meant to represent or suggest actual implementations.

```
                           +-----------------+
                           | Interface to    |
                           | Port Layer      |
                           +-----------------+
                             |      |      |
                +------------+      |      +------+
                |                   |             |
       +----------------+    +-------------+      |
       | Datalink Xmit  |    |   Receive   |      |
       |     and        |    |   Packet    |      |
       | Error Recovery |    |   Buffer    |      |
       +----------------+    +-------------+      |
                |                   |             |
                +---+           +---+             |
                    |           |                 |
                 +-----------------+     +-----------------+
                 | Channel Control |     |Internal Event   |
                 | Layer           |     |Logs and Counters|
                 +-----------------+     +-----------------+
                    |           |
            +-------------+     |
            |   Transmit/ |     |
            |   Receive   |     |
            |   FIFO      |     |
            +-------------+     |
                    |           |
                    |           |
               <------------------------------> DSSI Bus
```

Figure 4-2: DSSI Datalink Model

The Port Interface layer provides the Port with access to datalink channel control services and the ability to read and reset various datalink internal counters and event logs.

For outgoing data, this layer interacts with the Datalink Transmit and Error Recovery layer to forward the packet to the target and return the results of the transfer to the Port.

For incoming data, the Port Interface signals the Port that an incoming packet has been received and unloads the Receive Packet buffer on request.

The event logs and counters maintained by a DSSI datalink implementation are described in section 5.6. They are read by the Port Layer and returned to higher layers as described in that section.

The Receive Packet Buffer represents a generic delivery site for incoming data from the Channel Control Layer. The potential for congestion exists when the arrival rate for incoming packets exceeds the rate at which such packets are consumed by higher layers.

The function of the Datalink Transmit and Error Recovery Layer is the reliable delivery of outgoing packets to the target. It initiates error recovery by retransmission according to the policy described in section 4.5, notifying the port layer only when an unrecoverable transmission error occurs.

As viewed by higher layers, the Channel Control Layer provides the following services:

1. Transmits outbound packets over the physical interconnect and reports delivery status. Error-free delivery is not guaranteed.

2. Buffers incoming packets, returning delivery status to the remote node and notifying higher layers when an error-free packet has been received.

At the physical interconnect, this layer is responsible for the following

1. Bus arbitration,

2. Initiating and responding to target selection,

3. Receipt of incoming data, including the detection of checksum and parity errors

4. Packetization of outgoing data through the addition of parity and checksum information.

5. Detection of bus protocol errors and exceptions

6. Returning delivery status to the remote node.

The Channel Control layer provides two data transfer modes: asynchronous mode, characterized by the transfer of single bytes directly to or from the bus, and synchronous burst mode, which moves data by way of the Transmit/Receive FIFO. The less efficient asynchronous mode requires an explicit bus handshake for each byte transferred and thus has a throughput of approximately 1.5Mb/sec. This mode is used to transmit the Command Descriptor block and the Status In byte.

Synchronous burst mode is used to efficiently transfer large blocks of data during the DATA OUT phase by pipelining requests for data. Using this mode, a data transfer rate of 4Mb/sec can be achieved.

4.4 Channel Control Services

The Channel Control layer provides the following services to higher layers:

- o Transmit a data packet and return status to higher layers,
- o Receive and buffer a data packet.

After transmitting a packet, the initiator's Channel Control Layer shall return one of the following status values to higher datalink layers:

- o CC_ACK - The packet was successfully transmitted. A value of ACK was received from the target during the STATUS IN phase.
- o CC_NAK - The Channel Control layer has detected a 'short term' error, probably correctable through retransmission. Such errors usually result from congestion in the target node due to lack of receive buffers.
- o CC_NORSP - The Channel Control Layer has detected what is probably a persistent error. Such errors are frequently caused by polling traffic addressed to non-existent nodes.

Implementations may differ in the ability to discriminate between various error conditions. An implementation may, as a minimum, return CC_NAK whenever a transmit request terminates with NAK status from the target, reporting all other errors with status CC_NORSP.

When receiving a packet, the target's Channel Control Layer shall only notify higher layers, and return a STATUS IN value of ACK to the initiator, when the packet is received and buffered without error.

4.5 Data Transmit And Error Recovery Layer

This layer performs only one function: The reliable transmission of a single data packet. Any failure status returned to higher layers indicates a non-recoverable error.

In DSSI, the procedure for error recovery is to retry the failing transmission. The goals of an efficient retry procedure are recovery from errors in the shortest possible time using the fewest retransmissions. For that reason, DSSI transmission failures are divided into two types: short-term, transient failures - likely to last for only a few milliseconds - and persistent, long-term errors. For transient failures, caused by bus errors or short-term receiver congestion, the initiator recovers by generating a burst of retransmissions within a short time. For persistent failures, usually caused by polling a non-existent node, such retries are widely spaced and extend over a much longer time.

To reduce or prevent bus congestion caused by recovery traffic, DSSI nodes use a random coin-toss to control retries during long-term recovery.

The retry procedure consists of the following steps:

1. When a transmission fails, the recovery procedure uses the transmit status returned by the Channel Control layer to infer whether the condition is of short- or long-term duration.

2. If a transient problem is indicated, the procedure performs up to 8 retries within approximately 2.0ms as described in section 4.5.1.

3. If a persistent error occurs or immediate recovery fails, a series of 'delayed' retry events is scheduled at 10ms intervals. At each interval, the decision to retransmit is made by 'coin-toss'.

In the procedure described below, certain steps are contingent on the existence of the target node. For DSSI implementations, the probable existence of a node is determined at a protocol layer above the Datalink. To avoid knowlege of packet contents and higher layer protocols, it is assumed that the higher layer passes an appropriate 'existence' flag (see the PL_EXIST parameter in section 4.6). with each outgoing data packet. For the CI port protocol, this flag shall be set - unless the following conditions are true:

1. The virtual circuit to the addressed port is 'closed' and

2. The CI port data packet is 'IDREQ'

4.5.1 Error Recovery Procedure

1. After a failed transmission attempt with CC_NAK status, the sender shall perform immediate retries as described below. Otherwise the delayed retry procedure shall be executed starting at step 5.

2. Immediate retries shall consist of up to 8 evenly spaced retry attempts over an interval of 2.0ms - the immediate retry duration. This represents an average of 1 retry every 250us.

3. The initiator shall guarantee that immediate retries conform to the following constraints:

   a. The first retransmission shall be delayed by at least 250us from the initial attempt.

   b. The immediate retry duration shall be at least 2.0ms for 90% of all immediate retries that perform 8 attempts. The sender should attempt to achieve this minimum duration for all retry sequences. If the immediate retry limit is ever changed, the minimum immediate retry duration shall scale linearly with the retry limit.

   c. The sender may interleave immediate retries with transmissions to other nodes. A node having several immediate retries in progress shall limit the aggregate rate of retries to an average of no more than 1 attempt every 250us.

4. Immediate retries shall continue until the immediate retry limit is reached or a status other than NAK is received after a retransmission. If immediate retries fail, the sender shall perform delayed retries.

5. Delayed retry events are scheduled at 10ms intervals. When the interval elapses, the sender shall randomly determine, by means of a 'coin-toss', whether or not a retransmission is required. The sender shall retransmit the failed packet according to the coin-toss result or if 10 delayed retry events in a row have occurred with no retransmission attempt.

   The random coin-toss algorithm shall generate a sequence of 'heads' and 'tails' that is unique to each node on the bus. See section 4.5.2 for an example of such an algorithm.

6. If the 'existence' flag is asserted, as described above, delayed retries shall terminate when a CC_ACK is received from the Channel Control later or after 256 delayed retransmissions have occurred.

7. When the 'existence' flag is deasserted only a single delayed retransmission shall be performed.

NOTE

While delayed retries are in progress, the sender shall interleave such retries with transmissions to other nodes.

4.5.2 Example Of Random Number Generator

The following pseudo-code is an example of a coin tossing routine that generates a random number using a seed that is unique to each node on the bus. The value returned is TAILS if the result is less than zero and HEADS otherwise.

The procedure for random number generation is derived from VMS library routine MTH$RANDOM.

The initial seed is a 32-bit integer, derived by concantenating the first three characters of the SCS system I/D and the node number as shown in figure 4-3:

```
31    24|23  16|15   8|7     0
+-------+------+------+------+
| Node  | Char | Char | Char |
| Number|  2   |  1   |  0   |
+-------+------+------+------+
```

Figure 4-3: Format of Unique Initial Seed

```
procedure coin_toss()

{                                                             }
{ All variables are 32-bit integers.
{                                                             }
{ Multiply operations generate 64-bits. The low order         }
{ 32 bits are assigned to the specified variable. The         }
{ high-order portion is discarded.                            }
{                                                             }
BEGIN

{ Variables current_seed and cold_start are assigned to       }
{ static storage to preserve their values across calls.       }

o allocate current_seed in static storage.
o allocate cold_start in static storage with an initial value
  of 0

if ( cold_start is 0 ) then
    BEGIN
    o cold_start := 1
    o current_seed := initial_seed
    END

o current_seed := ( current_seed * 69069 ) + 1

if ( current_seed is less that 0 ) then

   return ( TAILS )
```

```
else

   return ( HEADS )

END
```

4.6 Interface To The CI Port Layer

The interface to the CI Port Layer is the highest Datalink layer. As noted earlier, this layer is responsible for delivering incoming data blocks to the CI Port Layer, forwarding outgoing packets to the Data Transmit layer and returning transmit status.

Parameters passed to or from the Port layer with each Port layer packet are:

1. PL_EXIST -- existence flag. For outgoing packets, set to a non-zero value by the Port layer if the target node is believed to exist. The rules for setting or clearing this flag are given in section 4.5.

2. DL_DST -- for outgoing packets, the DSSI I/D of the node to receive the packet; for incoming packets, the DSSI I/D of this node.

3. DL_SRC -- for outgoing packets, the DSSI I/D of this node; for incoming packets, the DSSI I/D of the node that sent the packet.

4. DL_LEN -- The length of the packet in bytes (excludes the checksum appended by the Channel Control layer).

5. DL_STATUS -- packet status passed with each packet received or transmitted.

Possible received packet status values are:

- DL_OK -- Packet successfully received in its entirety.

- DL_OVERSIZE -- Packet was successfully received into the Datalink Receve buffer but was too large to deliver in its entirety to the Port layer buffer. The packet is truncated to fit into the Port layer buffer.

Possible transmit packet status values (after attempted error recovery) are:

- DL_ACK -- Successful transmission. A value of ACK was received from the target during the STATUS IN phase.

- DL_NAK -- The target exists but was unable to succesfully receive the packet, most likely due to lack of a Receive Packet buffer.

- DL_NORSP -- transmission was attempted to a possibly non-existent target node.

CHAPTER 5

THE DSSI PHYSICAL BUS PROTOCOL

5.1 The DSSI Physical Bus Protocol

The following figure (figure 5-1) shows the datalink Channel Control layer, with its interfaces to the DSSI bus and higher datalink layers.

```
Transmit   Transmit  Received
 Data       Status    Data
   |          ^        ^
   v          |        |
 +-------------------------+
 |   Channel Control       |
 |   Layer                 |
 +-------------------------+
            ^
            | Physical Bus
            v Protocol
    <------------------>
         DSSI Bus
```

Figure 5-1: Datalink Channel Control Layer

As indicated in the diagram, the DSSI physical bus protocol defines the interface between the Channel Control Layer and the DSSI bus. This signalling protocol, used to exchange data packets between an initiator and target over the interconnect, supports the following Channel Control services:

1. Transmit a data packet, returning status to a higher layer,

2. Receive a packet, notifying higher layers only when the packet is delivered without error.

The following sections specify the logical behavior of the interconnect when servicing a transmit request, including data formats, bus signals, bus phases, timing, error detection and conditions reported to higher datalink layers.

The electrical and physical bus requirements are specified in chapter 6.

5.2 Logical Description Of The DSSI

The DSSI is a daisy-chained bus, connecting up to 8 nodes, that uses low-cost, open-collector drivers and receivers for data transfer and control. The physical bus contains 16 active signals, 9 of which comprise an 8-bit, parity protected data bus.

The bus protocol is designed for the transfer of data in packets between an initiator and target node. Other nodes are excluded from the bus while such a transfer is in progress. All nodes on the bus are capable of acting as either an initiator or target.

5.3 Signal Definitions

Signals on the bus are defined as TRUE (asserted) when low. With open-collector logic, signals are driven low by the driver and pulled high by the terminator.

The DSSI bus contains the following signals:

- DATA<7:0>, PARITY - Signals that comprise an 8-bit wide data path plus odd parity. Collectively, these signals are referred to as the DATA BUS.

- BSY (Busy) - When asserted, indicates that the DSSI bus is currently in use.

- SEL (Select) - When asserted, indicates that an initiator is attempting to address (Select) a target.

- C/D (Command/Data) - When asserted, indicates that the data lines are being used to pass control information to or from the target; when deasserted, indicates that data is being transferred.

- I/O (Input/Output) - When asserted, indicates that data movement is toward the initiator; when deasserted indicates that data movement is toward the target.

- REQ (Request) - Asserted by the target to request the transfer of one byte of data in the direction specified by the I/O signal.

- ACK (Acknowlege) - Asserted by the initiator in response to a REQ. When I/O is asserted, indicates that the initiator has read data placed on the bus by the target. When I/O is deasserted, indicates that the initiator has placed data on the bus to be read by the target.

- RST (Reset) - When asserted, indicates that a reset condition exists.

The 8-bit data path is also used to convey DSSI I/Ds during the arbitration and selection phases. In this case, each bit corresponds to the DSSI I/D of a node, where DATA<0> corresponds to I/D 0, DATA<1> to I/D 1, and so forth.

[1]

5.4 DSSI Bus Timing [1]

Unless otherwise indicated, the delay-time measurements for each DSSI device shown in table 5-1 shall be calculated from signal conditions existing at that DSSI device's own DSSI bus connection. Thus, these measurements (except for cable skew delay) can be made without considering delays in the cable. The timing characteristics of each signal are described in the following paragraphs.

----

[1] The bus signals, timing and the low level bus protocols (bus phases) are derived from Small Computer System Interface (SCSI, ANSI Standard, X3T9.2/82.2 - Rev17B)

Arbitration Delay	2.4 microseconds
Assertion Period	90 nanoseconds
Bus Clear Delay	800 nanoseconds
Bus Free Delay	800 nanoseconds
Bus Set Delay	1.8 microseconds
Bus Settle Delay	400 nanoseconds
Cable Skew Delay	10 nanoseconds
Data Release Delay	400 nanoseconds
Deskew Delay	45 nanoseconds
Hold Time	45 nanoseconds
Negation Period	90 nanoseconds
Reset Hold Time	25 microseconds

Table 5-1: DSSI Bus Timing Values

5.4.1 Arbitration Delay [2.4us]

The minimum time a DSSI device shall wait from asserting BSY for arbitration until the DATA BUS can be examined to see if arbitration has been won. There is no maximum time.

5.4.2 Assertion Period [90ns]

The minimum time that a target shall assert REQ while using synchronous data transfers. Also, the minimum time that an initiator shall assert ACK while using synchronous data transfers.

5.4.3 Bus Clear Delay [800ns]

The maximum time for a DSSI device to stop driving all bus signals after:

1. The BUS FREE phase is detected (BSY and SEL both false for a bus settle delay [400ns]),

2. SEL is received from another DSSI device during the arbitration phase,

3. The transition of RST to true.

5.4.4 Bus Free Delay [800ns]

The minimum time that a DSSI device shall wait from its detection of the BUS FREE phase (BSY and SEL both false for a bus settle delay), until its assertion of BSY when going to the ARBITRATION phase.

5.4.5 Bus Set Delay [1.8us]

The maximum time for a DSSI device to assert BSY and its DSSI I/D bit on the data line after it detects BUS FREE phase (BSY and SEL both false for a bus settle delay) for the purpose of entering the ARBITRATION phase.

5.4.6 Bus Settle Delay [400ns]

The time to wait for the bus to settle after changing certain control signals as called out in the protocol definitions.

5.4.7 Cable Skew Delay [10ns]

The maximum difference in propagation time allowed between any two DSSI bus signals when measured between any two DSSI devices.

5.4.8 Data Release Delay [400ns]

The maximum time for an initiator to release the Data bus signals following the transition of the I/O signal from false to true.

5.4.9 Deskew Delay [45ns]

The minimum time required for the deskew of certain signals.

5.4.10 Hold Time [45ns]

The minimum time added between the assertion of REQ and ACK and the changing of the data lines to provide hold time in the initiator or target, respectively while using synchronous data transfers.

5.4.11 Negation Period [90ns]

The minimum time that a target shall negate REQ while using synchronous data transfers. Also, the minimum time that an initiator shall negate ACK while using synchronous data transfers.

5.4.12 Reset Hold Time [25us]

The minimum time for which RST is asserted. There is no maximum time.

5.4.13 Transfer Period [180ns]

The Transfer Period specifies the minimum time allowed between the leading edges of successive REQ pulses and of successive ACK pulses while using synchronous data transfers.

5.5 Bus Protocol

The following sections specify the physical bus protocol and exception handling associated with packet transmission. Each section describes:

1. The associated bus activity,
2. The required initiator and target behavior,
3. The status returned to higher datalink layers in the initiator.

As discussed in section 4.4, that status shall reflect one of the following conditions:

- CC_ACK - The initiator's Channel Control layer has successfully transmitted the packet to the target.
- CC_NAK - The Channel Control Layer has detected a short term error.
- CC_NORSP - The Channel Control Layer has detected a persistent error.

The conditions under which each value should be returned are summarized in table 5-2 below.

Status	Condition	Section/Page
CC_ACK	Succesful packet transmission.	Section 5.5.8.4, page 5-37.
CC_NORSP	Bus Reset detected after winning arbitration but before selection phase.	Section 5.5.5.5, page 5-17
	No response to selection	Section 5.5.6.1, page 5-21
	Initiator timeout (if node does not support selection timeout).	Section 5.5.7.4, page 5-27.
CC_NAK	NAK Received during STATUS IN phase or parity error detected during STATUS IN.	Section 5.5.6.1, page 5-21.
	Premature BUS FREE Phase detected during any information transfer phase.	Section 5.5.7.4, page 5-27.
	Initiator timeout detected.	Section 5.5.7.4, page 5-27.
	Invalid information transfer phase.	Section 5.5.7.4, page 5-27.
	"Third party" bus reset detected.	Section 5.5.7.4, page 5-27.
	Phase other than BUS FREE following the STATUS IN phase.	Section 5.5.8.4, page 5-37.

Table 5-2: Summary of Channel Control Layer Status Values

5.5.1 General Requirements

5.5.1.1 Implicit Signal State

Unless otherwise noted in the following descriptions, signals that are not mentioned shall not be asserted.

5.5.1.2 Definition Of Initiator And Target Nodes

The following sections define requirements that apply to initiator or target nodes. A node becomes an initiator if and only if it decides it has won arbitration (see step 4 on page 5-15). A node shall continue as an initiator until it detects the BUS FREE phase (see section 5.5.3)

A node becomes a target if and only if it decides to assert BSY in response to SEL during the SELECTION phase (see section 5.5.6). A node shall continue as a target until it releases all signals to enter the BUS FREE phase.

5.5.1.3 Use Of And Reaction To Bus Reset (RST)

The bus reset signal (RST) allows a node not in control of the bus to force the BUS FREE phase in response to an exception condition. Any DSSI device can create a reset condition by asserting RST for a minimum of 25us.

The processing of Bus Reset shall take precedence over all other bus phases. All DSSI devices shall release all DSSI signals within 800ns of the transition of RST to true. Nodes shall not drive any other bus signals while RST is true.

A Bus Reset shall only be asserted:

- When explicitly required in the following sections,

- Whenever a node is powered up as described in section 6.3.9.3.3.

- Whenever an initiator detects an internal error condition (for example, a parity error in the local RAM while transferring data on the bus, or a bus parity error during the STATUS IN phase).

NOTES

1. The assertion of a DSSI Bus Reset shall not cause any node on the bus to suffer a loss of context (i.e., the receipt of reset by a node on the bus is not sufficient reason for breaking any virtual circuit).

2. The detection of Reset by an initiator or target shall cause the transfer in progress to be aborted. The initiator shall respond by invoking the error recovery procedure defined in section 4.5. The target shall consider a bus reset as equivalent to a data error and shall not forward the data packet to higher layers. In particular, a reset that occurs after the target has otherwise successfully completed the DATA OUT and STATUS IN phases should prevent such packets from being forwarded.

5.5.1.4 Watchdog Timers

The datalink shall maintain watchdog timers, which cause a transfer to be aborted in the event of a bus lockup. The following conditions shall be detected:

1. Selection Timeout [20us to 30us] - failure of a node to respond to selection within a predetermined time limit. The initiator shall enforce Selection Timeout as described in section 5.5.6.

2. Initiator Timeout [2800us] - The maximum time allowed between consecutive occurrences of the BUS FREE phase. This interval shall be enforced by an initiator or any node monitoring bus state with the intention of entering into arbitration for the bus. A node detecting Initiator Timeout shall generate a Bus Reset.

3. Target Timeout [2400us] - The maximum time between the detection of selection and observation of the next BUS FREE condition by the target Node. The target shall respond by releasing all signals and entering the BUS FREE phase.

5.5.2 DSSI Bus Phases And Signal Sources

As discussed in chapter 4, a data transfer requires the following sequence of bus phases:

- o BUS FREE - The bus is not in use, nodes may arbitrate for bus access.
- o ARBITRATION - A prospective initiator competes for exclusive bus ownership.
- o SELECTION - The initiator selects the target node.
- o COMMAND OUT - The initiator sends a 6-byte Command Descriptor block to the target.
- o DATA OUT - The initiator transmits the data block.
- o STATUS IN - The target returns the results of the transfer to the initiator.

The table below (table 5-3) summarizes the signals present on the bus during each phase and the nodes driving each signal.

Bus Phase	BSY	SEL	<C/D,I/O,REQ>	ACK	<DB<0-7>, PARITY>	
BUS FREE	None	None	None	None	None	None
ARBITRATION	All	Winner	None	None	DSSI ID	N/A
SELECTION	I&T	Init	None	None	Init	Init
COMMAND	Targ	None	Targ	Init	Init	Init
DATA OUT	Targ	None	Targ	Init	Init	Init
STATUS IN	Targ	None	Targ	Init	Targ	Targ

All: The signal shall be driven by all DSSI devices that are actively arbitrating.

DSSI I/D: A unique data bit shall be driven by each DSSI device that is actively arbitrating; the other seven data bits shall be released (ie: not driven) by the DSSI device. The parity bit may be released or driven to the true state.

I&T: The signal shall be driven by the initiator, target, or both, as specified in the SELECTION phase (see section 5.5.6).

Init If driven, this signal shall be driven only by the active initiator

None: This signal shall be released; that is, not be driven by any DSSI device. The bias circuitry of the bus terminators pulls the signal to the false state.

Win: The signal shall be driven by the one DSSI device that wins arbitration.

Targ: If the signal is driven, it shall be driven only by the active target.

Table 5-3: DSSI Signal Sources

The following sections describe the signal timing and sequencing requirements associated with each phase.

5.5.3 BUS FREE Phase

This phase is present when SEL and BSY have been simultaneously and continuously false for a Bus Settle delay [400ns]. A node may not begin arbitration for the bus until this phase is detected.

DSSI devices shall be prepared to detect BUS FREE at any time and shall respond by releasing all DSSI bus signals within a Bus Clear delay [800ns].

A BUS FREE phase, caused by the target's release of BSY, might occur because of:

1. Completion of the STATUS IN phase (target successfully delivers the status byte to the initiator).

2. After a Bus Reset is detected (see section 5.5.1.3).

3. After certain target-detected errors.

An initiator shall be prepared to detect the drop of BSY by the target at any other time. When this occurs, the target is indicating an error condition to the initiator. The initiator shall treat this as an unsuccessful packet transmission to be handled by its error recovery mechanism (see section 4.5).

The BUS FREE phase may also be entered after an unsuccessful selection (see section 5.5.6). In that case, it is the initiator's drop of SEL, after selection timeout elapses, that establishes the BUS FREE phase.

5.5.4 ARBITRATION Phase

This phase allows a device to compete for control of the DSSI bus for the purpose of selecting another device.

The mechanism uses the round-robin or 'fair' arbitration scheme, described below, to distribute bus access equally to all DSSI nodes.

5.5.5 Fair (Round-Robin) Arbitration

Round-robin arbitration insures that all devices have equal access to the bus. Except for the products listed in appendix H, which use the fixed priority arbitration scheme described in section H.2.1, all DSSI nodes shall implement the round robin arbitration algorithm described in this section. As described below, both types of bus arbitration can be intermixed on a single bus.

The basic concept of Round Robin arbitration is that DSSI nodes can be enabled or disabled. An enabled node may participate in arbitration if it has a message to send, a disabled node may not. The normal sequence of events starts with all nodes enabled. Each node in turn wins arbitration, sends a message, and disables itself. Eventually all nodes are disabled, resulting in the DSSI bus becoming idle. All the nodes recognize that the bus is idle (BSY and SEL both false for a long enough time) and re-enable themselves.

When fixed priority and round robin implementations share the same bus, nodes that use priority arbitration behave as though they are permanently enabled. So long as such nodes use less than the total bus bandwidth, the bus will periodically go idle and the fair arbitration nodes will re-enable themselves. This criteria for correct operation --- that nodes using fixed priority arbitration use less than the DSSI bus bandwidth --- is exactly the same as the requirement for correct operation of a DSSI bus that only contains nodes supporting fixed priority arbitration.

5.5.5.1 Specification Of DSSI Arbitration Algorithm

This section is the exact specification of the DSSI round robin arbitration algorithnm.

The following timing parameters are defined:

DSSI Arbitration Skew Delay [600ns]

> The maximum allowable skew for a DSSI node to assert BSY and its DSSI ID during arbitration.

DSSI Idle Delay [2200ns]

> The minimum time a DSSI node must wait before concluding that no enabled nodes are arbitrating for the bus.

The rationale for using the above two parameters and the derivation of their values are described in Appendix A.

Signal definitions and all other timing parameters are the same as in sections 5.3, Signal Definitions, and 5.4 DSSI Bus Timing. All timing measurements shall be calculated from the signal conditions existing at each DSSI node's own DSSI connection (that is, the DSSI bus side of drivers and receivers).

Each DSSI node implements an internal flag indicating whether it is enabled or disabled. This arbitration algorithm specification is divided into three sections:

1. Rules for enabled nodes. Enabled nodes that wish to send a message participate in arbitration. Upon winning arbitration, the node disables itself and attempts to send the message.

2. Rules for disabled nodes. Disabled nodes do not participate in arbitration. However, disabled nodes monitor the DSSI bus, and enable themselves upon detecting that the bus is idle.

3. Rules for manipulating the enabled/disabled flag.

5.5.5.2 Rules For Enabled Nodes

Enabled nodes that do not have a message to send should normally remain enabled. However, node resets or other conditions may cause a node to spontaneously become disabled.

An enabled DSSI node that has a message to send shall arbitrate for the DSSI bus as follows:

1. The enabled DSSI node with a message to send shall first wait for the BUS FREE phase to occur. The BUS FREE phase is detected whenever both BSY and SEL are simultaneously and continuously false for a minimum of a bus settle delay [400ns]

2. The DSSI node shall wait a minimum of a bus free delay [800ns] after detection of the BUS FREE phase (i.e. after BSY and SEL are both false for a bus settle delay [400ns]) before driving any signal.

3. Following the bus free delay [800ns] in step 2, the DSSI node shall assert both BSY and its own DSSI ID within an Arbitration Skew Delay [600ns]. The node shall not arbitrate (i.e. assert BSY and its DSSI ID) if more than a bus free delay [800ns] plus a DSSI Arbitration Skew Delay [600ns] have passed since the BUS FREE phase was last observed. However, it should be noted that whenever BSY and SEL are simultaneously, and continuously false for more than 400ns, a node may detect the BUS FREE phase at any time during that period and thus start its reference point (i.e., start at Step 1).

4. After waiting at least an arbitration delay [2200ns] (measured from its assertion of BSY) the DSSI node shall examine the DATA BUS. If a higher priority DSSI ID bit is true on the DATA BUS (DATA<7> is the highest), then the DSSI node has lost the arbitration and the DSSI node shall release its signals and return to step 1. If no higher priority DSSI ID bit is true on the DATA BUS, then the DSSI node has won the arbitration and it shall assert SEL. Any other DSSI node that is participating in the ARBITRATION phase has lost the arbitration and shall release BSY and its DSSI ID bit within a bus clear delay [800ns] after SEL becomes true. A DSSI node that loses arbitration may return to step 1.

5. The DSSI node that wins arbitration shall wait at least a bus clear delay [800ns] plus a bus settle delay [400ns] after asserting SEL before changing any signals.

6. The DSSI node that wins arbitration shall disable itself prior to releasing both BSY and SEL. It shall disable itself regardless of the outcome of the subsequent SELECTION phase or message transmission attempt.

5.5.5.3 Rules For Disabled Nodes

1. Disabled DSSI nodes that have a message to send shall monitor the DSSI bus for an idle condition as described below. Disabled DSSI nodes that do not have a message to send may or may not monitor the DSSI bus for an idle condition. DSSI nodes that always monitor the DSSI bus for an idle condition will exhibit improved performance over DSSI nodes that do not. A DSSI node that is

disabled shall not become enabled for any reason other than detection of a DSSI bus idle condition as described below.

2. DSSI nodes monitoring the bus shall detect DSSI bus idle condition whenever BSY and SEL are simultaneously and continuously false for a minimum of DSSI Idle Delay [2200ns]. DSSI nodes intending to gain bus access which are not previously monitoring the bus and are currently disabled shall monitor the bus and detect DSSI bus idle condition when both BSY and SEL are simultaneously false for a minimum of DSSI Idle Delay [2200ns]. A DSSI node shall become enabled upon detecting a DSSI bus idle condition.

3. A disabled DSSI node that has a message to send shall arbitrate for the DSSI bus only after detecting a DSSI bus idle condition and becoming enabled. The node may omit arbitration step 1, since detecting a DSSI bus idle condition implies that the node has also detected the BUS FREE phase. After detecting a DSSI bus idle condition, a disabled DSSI node that has a message to send shall wait the bus free delay [800ns] of arbitration step 2, then assert BSY and its DSSI ID for arbitration step 3.

5.5.5.4 Rules For Manipulating The Enabled/Disabled Flag

This section states all the rules for manipulating the enabled/disabled flag, repeating the rules stated in the previous two sections. The rules are:

1. A newly initialized DSSI node or a DSSI node that has otherwise lost track of DSSI bus context shall be initially disabled.

2. An enabled DSSI node may spontaneously become disabled for any reason. However, doing so will impair that DSSI node's performance.

3. A disabled DSSI node shall not become enabled for any reason other than detection of a DSSI bus idle condition.

4. Upon winning arbitration, a DSSI node shall disable itself prior to releasing BSY or SEL (whichever it releases last).

5. A DSSI node shall enable itself upon detecting a DSSI bus idle condition.

5.5.5.5 ARBITRATION Phase Transitions

ARBITRATION phase transitions are shown in the following figure.

```
                         +------------+
                         |  BUS FREE  |<---------+
                         +------------+          |
                                .                |
                                .                ^
                       +---------------+         |
                       |  ARBITRATION  |--->----+
                       +---------------+ (2)(3)
                                |(1)      (4)
                                v
                       +---------------+
                       |  SELECTION    |
                       +---------------+
                                .
                                .
```

Figure 5-2: ARBITRATION Phase Transitions

All DSSI implementations shall detect the following events and respond as described.

1. The node successfully arbitrates for the bus and enters the SELECTION phase.

2. The node loses arbitration. Upon losing arbitration, the node shall release all signals, wait for the BUS FREE phase and restart arbitration without notifying higher layers.

3. The node detected a Bus Reset condition. If the node has not yet become an initiator, it should treat this condition as if the node had lost arbitration. Otherwise, the node should terminate the transmit request with a status of CC_NORSP.

4. The node detected an Initiator Timeout. The node shall generate a Bus Reset and proceed as in event 3 above.

5.5.6 SELECTION Phase

This section describes the bus signals and timing associated with the SELECTION phase, the phase used by the initiator to select a target device to receive information.

Upon winning arbitration, the initiator is asserting SEL, BSY, and the DATA BUS line corresponding to its DSSI ID. To begin the transition from arbitration to selection, the initiator shall:

1. Wait at least 1200 ns after asserting SEL then,

2. Change the data bus to the inclusive-or, with correct (asserted) parity, of the initiator's and target's DSSI IDs.

The initiator may "glitch" the data bus by first releasing it then asserting the DSSI IDs, provided that the data bus does not change until at least 1200 ns after the initiator asserted SEL.

After asserting the DSSI IDs onto the data bus, the initiator shall wait at least 90ns, then release BSY. The elapsed time from assertion of SEL to the release of BSY shall not exceed 2000ns. The initiator shall wait a minimum of 400 ns after releasing BSY before testing if BSY has been asserted by the target.

A device determines that an initiator has selected it as a target by observing that:

1. SEL is asserted,

2. BSY negated and,

3. the data bus has the target device's own and exactly one other DSSI ID asserted, and parity is correct (asserted).

Devices shall determine that they are selected whenever they are not the initiator and such a pattern is stable on the bus as described below.

The device shall determine that it is selected no sooner than after SEL, BSY, and the target device's own DSSI ID have been stable for a minimum of 400 ns, and the remaining data lines plus parity are correct at the end of the stable period. The device shall determine that it is selected no later than 20us after all signals have been stable.

A device that determines it has been selected as a target:

1. Shall respond to selection by asserting BSY no later than 20 us after BSY became stable and negated,

   and preferably as quickly as possible.

2. Shall not assert BSY, and not consider itself selected, if any bus signals change,

3. Shall save the initiator's DSSI ID, determined from the data bus.

After asserting BSY, the target device may assert or change C/D and/or I/O to prepare for an information transfer phase.

NOTE

> As described in section 5.5.7, an information transfer phase does not actually begin until the assertion of REQ by the target.

After releasing BSY, the initiator shall wait a minimum of 400 ns, then begin observing BSY for the target's response. If the the initiator sees BSY asserted, it shall wait a minimum of 90 ns, then it shall release SEL and may release or change the data bus. The bus is entering an information phase after the initiator releases SEL, and the initiator should interpret bus signals as described in note 1 on page 5-23. The C/D and I/O signals control what the initiator should do with the data bus.

If the initiator does not see BSY asserted within a Selection Timeout interval from the time it released BSY, then a selection timeout has occurred. The initiator shall release the data bus following the Selection Timeout interval, then wait a Selection Abort interval. If the initiator sees BSY asserted during the Selection Abort interval, the initiator shall treat it as a normal target response to selection. If BSY is not asserted before the end of the Selection Abort interval, the initiator shall release SEL and report a selection timeout error to higher layers. Note that, upon releasing SEL, the initiator has released all bus signals.

The minimum Selection Timeout interval shall be 20 us. The maximum Selection Timeout interval shall be 30us or less. The Selection Abort interval shall be 20us minimum and 30us maximum.

An implementation conforms with this selection timeout requirement if it initiates selection abort within the maximum timeout interval for at least 95 percent of all selection failures that occur over a period of 10 minutes or more in any operational configuration. In no case shall selection abort be initiated in less than the minimum selection timeout interval.

"Operational configuration" means any configuration that does not contain an identifiable failure or some component that requires replacement.

NOTE

> Certain datalink implementations, listed in appendix H, do not provide a mechanism for detecting selection timeout.

5.5.6.1 SELECTION Phase Transitions

The figure below(figure 5-3) shows the transitions that terminate the SELECTION phase.

```
       +--------+
       |BUS FREE|<-----------------+
       +--------+                  |
           .                       ^
           .                       |
   +----------------+     (2)(5)   |
   |    SELECTION   |------------>-+
   +----------------+              |
    |      |(1)    |(3)            ^
    |      |       V               |
    |      | +-----------+         |
    |      | |Unexpected |         |
    |      | |Information|------->-+
    |      | |Phase      |
    |(4)   | +-----------+
    |      V
    |   +---------+
    |   | Cmd Out |
    |   +---------+
    |        .
    V        .
  +-----------+
  | Status In |
  +-----------+
        .
        .
```

Figure 5-3: SELECTION Phase Transitions

All DSSI implementations shall detect the following events and respond as described.

1. The target has detected a valid selection and has responded by asserting BSY and driving the COMMAND OUT phase.

2. The initiator has failed to receive a selection response within the selection timeout period. The initiator shall release all signals, enter the BUS FREE phase and terminate the transmit request with a an error status. A status equivalent to CC_NORSP shall be returned to higher datalink layers.

   Selection response failures may result from:

a. An attempt to select a non-existent node, or

b. Detection of an invalid selection by the remote node (see section 5.5.6).

3. The target has responded to selection by asserting BSY but has entered an invalid information phase (a phase other that STATUS IN or COMMAND OUT). See section 5.5.7.4.

4. The target has detected a valid selection but has no Packet Receive buffer free. The target shall drive STATUS IN and return a value of NAK to the initiator.

   The initiator shall return a status of CC_NAK to higher layers.

5. The initiator has detected Bus Reset while waiting for a target response. The initiator shall terminate the transmit request and return an error status to higher layers. A value equivalent to CC_NORSP should be returned.

5.5.7 Information Transfer Phases

The COMMAND OUT, DATA OUT and STATUS IN phases are collectively referred to as the Information Transfer phases because they are all used to transfer data or control information via the DATA BUS.

The information phases are determined by the state of the C/D and I/O signals as shown in the following table.

Signal			
C/D	I/O	Phase Name	Transfer Direction
0	0	DATA OUT	Initiator to target
0	1	**	**
1	0	COMMAND OUT	Initiator to target
1	1	STATUS IN	Target to initiator

Key: 0 = False, 1 = True, ** = Reserved

Table 5-4: Information Transfer Phases

The target, by driving the C/D and I/O signals, controls all changes from one phase to another.

The information transfer phases use one or more REQ/ACK handshakes to control the information transfer. Each REQ/ACK handshake allows the transfer of one byte of information.

During the information transfer phases the BSY signal shall remain true and the SEL signal shall remain false. Additionally, during the information transfer phases, the target shall continuously envelope the REQ/ACK handshake(s) with the C/D and I/O signals in such a manner that these control signals are valid for a bus settle delay [400ns] before assertion of the REQ signal of the first handshake and remain valid until after the negation of the ACK signal at the end of the handshake of the last transfer of the phase.

NOTES

1. After the negation of the ACK signal of the last transfer of the phase, the target may prepare for a new phase by asserting or negating the C/D and I/O signals. These signals may be changed together or individually. They may be changed in any order and may be changed more than once. It is desirable that each line change only once. A new information phase does not begin until the REQ signal is asserted for the first byte of the new phase.

2. An information phase is defined as ending when the C/D or I/O signals change after the assertion of the ACK signal. The time between the end of a phase and the assertion of the REQ signal beginning a new phase is undefined (except for the ultimate limits imposed by the initiator and target timeouts). An initiator is allowed to anticipate a new phase based on the previous phase, the expected new phase, and early information provided by changes in the C/D and I/O signals. However, the anticipated phase is not valid until the REQ signal is asserted at the beginning of the next phase.

5.5.7.1 Asynchronous Information Transfer

The target shall control the direction of transfer by means of the I/O signal. When the I/O signal is true, information shall be transferred from the target to the initiator. When the I/O signal is false, information shall be transferred from the initiator to the target.

If the I/O signal is true (transfer to the initiator), the target shall first drive the DATA BUS signals to their desired values, delay at least one deskew delay [45ns] plus a cable skew delay [10ns], then assert the REQ signal. The DATA BUS signals shall remain valid until the ACK signal is true at the target. The initiator shall read the DATA BUS signals after the REQ signal is true then indicate its acceptance of the data by asserting the ACK signal.

When the ACK signal becomes true at the target, the target may change or release the DATA BUS signals and shall negate the REQ signal. After the REQ signal is false, the initiator shall then negate the ACK signal. After the ACK signal is false, the target may continue the transfer by driving the DATA<7:0> and PARITY signals and asserting the REQ signal as described above.

If the I/O signal is false (transfer to the target) the target shall request information by asserting the REQ signal. The initiator shall drive the DATA<7:0 > and PARITY signals to their desired values, delay at least one deskew delay [45ns] plus a cable skew delay [10ns] and assert the ACK signal. The initiator shall coninue to drive the DATA<7:0> and PARITY signals until the REQ signal is false.

When the ACK signal becomes true at the target, the target shall read the DATA<7:0> and PARITY signals then negate the REQ signal. When the REQ signal becomes false at the initiator, the initiator may change or release the DATA BUS signals and shall negate the ACK signal. The target may continue the transfer by asserting the REQ signal as described above.

5.5.7.2 Synchronous Data Transfer

Support for Synchronous Data Transfers (Synchronous Burst Mode) is mandatory for all implementations.

Synchronous Data transfers require the use of a REQ/ACK offset parameter that specifies the maximum number of REQ pulses that can be sent by the target in advance of the number of ACK pulses received from the initiator. As discussed in section 5.5.7.5, the target determines the maximum allowable offset by comparing the REQ/ACK offset sent by the initiator during the COMMAND OUT phase with the target's value for this parameter, using the smaller of the two. For the DSSI, the only legal values for this parameter are three (3), seven (7) or fifteen (15). All DSSI nodes should implement the maximum value.

If the number of REQ pulses exceeds the number of ACK pulses by the REQ/ACK offset, the target shall not assert the REQ signal until after the leading edge of the next ACK pulse is received.

The target shall assert the REQ signal for a minimum of an assertion period [90ns].The target shall then wait at least the transfer period [180ns] from the last transition of the REQ signal to true before again asserting the REQ signal.

The initiator shall send one pulse on the ACK signal for each REQ pulse received. The ACK signal may be asserted as soon as the leading edge of the corresponding REQ pulse has been received. The initiator shall assert the ACK signal for a minimum of an assertion period [90ns]. The initiator shall wait at least the transfer period [180ns] from the last transition of the ACK signal to true before again asserting the ACK signal.

During the DATA OUT phase (C/D false and I/O false), the initiator shall transfer one byte for each REQ pulse received. After receiving the leading edge of a REQ pulse, the initiator shall first drive the DATA BUS signals to their desired values, delay at least one deskew delay [45ns] plus one cable skew delay [10ns] then assert the ACK signal. The initiator shall hold the DATA BUS signals valid for at least one deskew delay [45ns] plus one cable skew delay [10ns] plus one hold time [45ns] after the assertion of the ACK signal. The initiator shall assert the ACK asignal for a minimum of an assertion period [90ns]. The initiator may then negate the ACK signal and may change or release the DATA BUS signals. The target shall read the value of the DATA BUS signals within one hold time [45ns] of the transition of the ACK signal to true.

The target shall not assert a new information phase (by changing C/D and I/O) until its counts of REQ and ACK pulses are equal.

A target shall determine that a synchronous transfer has completed successfully when:

1. The target's counts of REQ and ACK pulses are equal,

2. The number of bytes received equals the byte count parameter supplied by the initiator during the COMMAND OUT phase plus one byte for the appended checksum.

3. No parity or checksum errors are detected.

The target shall detect and respond to errors during a synchronous transfer as follows:

- The target detects more REQs than the number of ACKs sent. The target shall abort the transfer by entering the BUS FREE phase.

- The target detects a parity or checksum error. The target shall drive the STATUS IN phase (observing the above constraints regarding the assertion of a new information phase) and return a value of NAK to the initiator.

The target shall respond to all other errors by entering the BUS FREE phase.

5.5.7.3 Signal Restrictions Between Information Phases

When the DSSI bus is between two information transfer phases, the following restrictions shall apply to the DSSI bus signals:

1. The BSY, SEL, REQ and ACK signals shall not change.

2. The C/D, I/O, DATA and PARITY signals may change. When switching the DATA BUS direction from out (initiator driving) to in (target driving), the target shall delay driving the DATA BUS by at least a data release delay [400ns] plus a bus settle delay [400ns] after asserting the I/O signal. In addition, the initiator shall release the DATA BUS no later than a data release delay [400ns] after the transition of the I/O signal to true. When switching the DATA BUS direction from in (target driving) to out (initiator driving), the target shall release the DATA BUS no later than a deskew delay [45ns] after negating the I/O signal.

3. The RST signal may change at any time as described in section 5.5.1.3.

5.5.7.4 Information Phase Error Conditions

The following error conditions are common to all information transfer phases and shall be handled as described below. In all cases, the initiator shall terminate the transmit request with an error status. A value of CC_NAK should be returned to higher layers.

1. Target Timeout - The target shall release all signals and return to the BUS FREE phase as described in section 5.5.1.4.

2. Premature BUS FREE phase detected by initiator - This condition occurs as the result of certain target-detected errors (eg: synchronous data transfer errors (see section 5.5.7.2)).

3. Initiator Timeout - The initiator shall generate a Bus Reset and abort the request as described in section 5.5.1.3.

NOTE

> Implementations listed in appendix H that do not support selection timeout shall report an initiator timeout condition with status CC_NORSP.

4. Invalid information transfer phase - The target has asserted an unexpected information transfer phase. The initiator shall generate a Bus Reset.

5. "Third-party" Bus Reset -- A bus reset, asserted by a node other than the initiator, during any information transfer phase shall be handled as described in section 5.5.1.3.

5.5.7.5 The COMMAND OUT Phase

In the COMMAND OUT phase, the target solicits from the initiator a 6-byte Command Descriptor block (plus checksum) that contains control information required by the target during the DATA OUT phase.

The target shall assert C/D and transfer the information using the rules for asynchronous information transfer described in section 5.5.7.1.

The format and rules for validating the Command Descriptor are given below.

```
  7     6     5     4     3     2      1     0
+-----+-----+-----+-----+-----+-----+-----+-----+
|        Data Link Operation Code ( E0h )       |  command+0
+-----------------------+-----------------------+
|    REQ/ACK Offset     |   Protocol Identifier |  command+1
+-----------------------+-----+-----------------+
|         Reserved            | Dest. Node I/D  |  command+2
+-----------------------------+-----------------+
|         Reserved            | Source Node I/D |  command+3
+-----------------------------+-----------------+
|            Data Block Length<7:0>             |  command+4
+-----------------+-----------------------------+
|    Reserved     |    Data Block Length<12:8>  |  command+5
+-----------------+-----------------------------+
|                 Checksum                      |  command+6
+-----------------------------------------------+
```

Figure 5-4: Format of DSSI Command Out Descriptor Block

The fields of the DSSI Command Out descriptor block are as follows:

Command + 0 - Datalink Operation Code

Defines the Target Data Link function to be performed. Only one function is defined: "Receive Data Block" having a value of E0h. An illegal value shall cause the Target to return NAK status.

Command + 1 <3:0> - Protocol Identifier

A nibble that identifies a protocol layer, serviced by the Data Link, to receive the data block. A value of 0 specifies delivery to the CI Port Layer. For codes that are unsupported, the Target Data Link shall process such errors as described in section 5.5.7.6.1.

Command + 1 <7:4> - REQ/ACK Offset

A nibble that contains the initiator's maximum REQ/ACK offset. The only valid values for this field are three (3), seven (7), or fifteen (15). An illegal REQ/ACK offset shall cause the Target to return NAK status.

Command + 2 <2:0> - Destination node I/D

The DSSI Bus ID of the node receiving the information (i.e., the Target). The Target shall return NAK status if this value is not equal to the Target's node I/D.

Command + 2 <7:3> - Reserved

See section 5.5.7.6.1.

Command + 3 <2:0> - Source node I/D

The DSSI Bus I/D of the node transmitting the information (i.e., the initiator). The target shall verify the initiator's Node I/D by comparing it against the Node I/D received during selection, returning NAK status if these values differ.

Command + 3 <7:3> - Reserved

See section 5.5.7.6.1.

Command + 4 <7:0> - Data Block Length <7:0>

Command + 5 <4:0> - Data Block Length <12:8>

The Data Block Length parameter is a 13-bit value that is returned in two consecutive bytes as shown above.

This value specifies the length of the data block in bytes (excluding the checksum). See section 5.5.8.1.

Command + 5 <7:5> - Reserved

See section 5.5.7.6.1.

Command + 6 - Checksum

This field contains the vertical checksum of the Command Descriptor and shall be computed as the XOR of the first 6 bytes. It is generated by the initiator and checked by the target. The checksum contents are calculated as:

```
checksum = command + 0 XOR Command +1 XOR ...
                XOR Command + 5
```

Detection of an invalid checksum by the Target shall cause the Target to return NAK status.

5.5.7.6 Command Out Descriptor Validation Precedence

Fields that result in NAK status when invalid shall be checked before all other fields. Reports of NAK status shall take precedence over all other validation error responses.

NOTE

A validation error that results in NAK status may be reported as soon as the erroneous parameter is received (validation "on the fly") or deferred until after the entire Command Descriptor block has been read in.

5.5.7.6.1 Reserved Field Validation

A field designated as a reserved field shall be set to zero (0) by the initiator.

Targets should validate reserved fields. An implementation that performs such validation shall:

1. Complete the transaction with the initiator, including error recovery retries, and return an ACK status to the initiator if the packet was received correctly.

2. Discard the packet if any reserved field in the Command Descriptor block is non-zero.

Higher protocol layers in the initiator or target are not notified when such a packet is discarded. A discarded packet is handled through provisions in higher level protocols for detecting lost packets.

5.5.7.6.2 COMMAND OUT Phase Transitions

The following figure (figure 5-5 shows the transitions that terminate the COMMAND OUT phase.

```
     +--------+
     |BUS FREE|<------------+
     +--------+             |
         .                  ^
         .                  |
 +----------------+  (2)    |
 |  COMMAND OUT   |------->-+
 +----------------+  (5)    |
  |      |(1)    |(3)       ^
  |      |       V          |
  |      | +-----------+    |
  |      | |Unexpected |-->-+
  |      | |Information|
  |      | |Phase      |
  |(4)   | +-----------+
  |      V
  |   +---------+
  |   | Data Out|
  |   +---------+
  V
+-----------+
| Status In |
+-----------+
```

Figure 5-5: COMMAND OUT Phase Transitions

All DSSI implementations shall detect the following events and respond as described.

1. A valid Command Descriptor block has been read by the target. The target shall assert the DATA OUT phase.

2. An initiator or target timeout has occurred. See section 5.5.7.4.

3. The initiator detects an unexpected information phase (a phase other than STATUS IN or DATA OUT). See section 5.5.7.4.

4. The target has detected a parity or checksum error in the Command Descriptor block or has discovered an invalid command parameter. (The Command Descriptor contents and validation rules are described in section 5.5.7.5.) The target shall enter the STATUS IN phase and return a status of NAK to the initiator.

   The initiator shall terminate the transmit request with an error status. A value of CC_NAK should be returned to higher layers.

5. A Bus Reset condition occurred. This conditon is processed as described in section 5.5.7.4.

5.5.8 The DATA OUT Phase

The target enters the DATA OUT phase by deasserting C/D and I/O. The target and initiator then interact to transfer a Data Block whose format is described below. The data block shall be transmitted using the rules for synchronous data transfer described in section 5.5.7.2.

As shown in figure 5-6, a DSSI data block consists of the number of bytes (n) specified in the length field of the Command Descriptor block plus one byte of vertical checksum.

```
   7     6     5     4     3     2     1     0
+-----+-----+-----+-----+-----+-----+-----+-----+
|              information byte 0              |  data text+0
+-----------------------------------------------+
|              information byte 1              |  data text+1
+-----------------------------------------------+
                      ...
+-----------------------------------------------+
|              information byte n - 2          |  data text+(n-2)
+-----------------------------------------------+
|              information byte n - 1          |  data text+(n-1)
+-----------------------------------------------+
|                 Checksum                      |  data text+(n)
+-----------------------------------------------+
```

Figure 5-6: DSSI Data Format

The checksum byte is computed as the XOR of all preceding data bytes. It is generated by the initiator and checked by the target. The checksum contents are calculated as:

Checksum = data text+0 XOR data text+1 XOR data text+2 ...
XOR data text+(n-2) XOR data text+(n-1).

5.5.8.1 Data Block Length

The minimum data block size transmitted by the initiator, excluding the checksum, shall not be less than 2 bytes. The maximum shall not exceed the largest value allowed by the higher level protocols serviced by the target's datalink layer. For the CI port protocol, the maximum value is specified in the MAX_BODY_LEN parameter transmitted in the ID packet.

The target datalink should be able to receive and buffer in its Receive Packet Buffer all data blocks, up to the maximum size that can be specified in the Command Descriptor block (8191 bytes).

Any packet received by the datalink layer that is outside the upper or lower limits accepted by higher layers should be passed to the higher layer. If an error-free data block is received that is too long, (greater than MAX_BODY_LEN for the CI port layer), the packet should be truncated and delivered with notification that an oversize data block was received.

NOTE

> The paragraphs above describe the preferred method for handling oversize packets. Certain DSSI datalink implementations, however, cannot receive such packets without buffer overflow or other adverse side effects. To prevent these conditions, such an implementation may abort packet transmission by checking the length parameter in the Command Descriptor block and issuing a NAK status whenever an invalid value is detected.
>
> Appendix D, "Implementation Functionality" describes how specific DSSI implementations handle such conditions.

5.5.8.2 DATA OUT Phase Transitions

The following transitions terminate the DATA OUT phase.

```
    +--------+
    |BUS FREE|<------------+
    +--------+             |
        .                  ^
        .                  |
+----------------+  (4)(5) |
|   DATA OUT     |------->-+
+----------------+         |
        |(1)    |(3)       ^
        |(2)    V          |
        | +-----------+    |
        | |Unexpected |-->-+
        | |Information|
        | |Phase      |
        | +-----------+
        V
   +-----------+
   | STATUS IN |
   +-----------+
```

Figure 5-7: DATA OUT Phase Transitions

All DSSI implementations shall detect the following events and respond as described.

1. The DATA OUT phase has completed succesfully as described in section 5.5.7.2. The target shall return a status value of ACK.

2. The target detected a parity or checksum error. The target shall return a status of NAK.

   The initiator shall terminate the transfer with an error status. A value of CC_NAK should be returned to higher layers.

3. The initiator has detected an unexpected information phase (a phase other than STATUS IN). See section 5.5.7.4.

4. The target detected one of the synchronous data transfer errors described in section 5.5.7.2.

5. An initiator timeout, target timeout or Bus Reset has occurred. See section 5.5.7.4.

5.5.8.3 The STATUS IN Phase

The target asserts the STATUS IN phase by setting C/D and I/O true. The target then transfers a single byte of status to the initiator using the rules for asynchronous transfers described in section 5.5.7.1.

Only two values are legal ACK (61h), indicating success, or NAK (1Eh), indicating failure. The associated bit patterns, including parity, are shown below (figure 5-8).

```
      P   7   6   5   4   3   2   1   0
    +---+---+---+---+---+---+---+---+---+
    | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | status text+0
    +---+---+---+---+---+---+---+---+---+
                 ACK Status ( 61h )


      P   7   6   5   4   3   2   1   0
    +---+---+---+---+---+---+---+---+---+
    | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | status text+0
    +---+---+---+---+---+---+---+---+---+
                 NAK Status ( 1Eh )
```

Figure 5-8: DSSI STATUS IN Data Formats

The above patterns are selected to reduce the likelihood of undetected errors by maximizing the code distance, including the odd parity bit, between the specified legal values.

5.5.8.4 STATUS IN Phase Transitions

The following figure shows the transitions that terminate the STATUS IN phase.

```
  +----------+
  | BUS FREE |<-------+
  +----------+        |
        .             |
        .             |
 +------------+ (1)(2)|
 |  STATUS IN |--->---+
 +------------+ (3)   |
        | (4)         |
        v             |
+-----------------+   |
| Any Other Phase |->-+
+-----------------+
```

Figure 5-9: STATUS IN Phase Transitions

All DSSI implementations shall detect the following events and respond as described.

1. The initiator has successfully completed the asynchronous transfer of status by asserting the ACK signal as described in section 5.5.7.1.

   The target shall enter the BUS FREE phase. If, during the STATUS IN phase just completed, the target has returned a value of ACK, signifying error-free completion of the DATA OUT phase, the target shall forward the received data packet to higher layers. The initiator shall return a status of CC_ACK to higher layers.

2. The STATUS IN value received by the initiator contained a parity error or value other than ACK. The initiator shall terminate the transfer and return a status of CC_NAK (or its equivalent) to higher datalink layers.

3. An initiator or target timeout has occurred. See section 5.5.7.4.

4. The initiator has detected a phase other than BUS FREE. The initiator shall reset the bus and should return a status of CC_NAK (or its equivalent) to higher datalink layers.

5.6 Performance And Diagnostic Counters

For purposes of monitoring performance and to aid diagnosis, all DSSI nodes shall maintain the counters described in this section. These counters shall be 32 bits in length, operated upon as an unsigned integer, and readable by higher layers such that their values may be collected via higher layer protocols. All nodes should maintain at least one set of these counters, and higher layer protocols may specify whether the events to a specified node or to all nodes should be counted.

In addition to the counters described in this section, a DSSI node may implement additional counters and include their values in the response returned to higher layers. The format and use of implementation-specific counters is outside the scope of this document.

The counters should be updated by a node only if a previous or current communication to the destination node is successful since the counters were last reset. A counter value of FFFFFFFFh is not valid and indicates an error or overflow condition.

All nodes shall provide an appropriate mechanism for reading and clearing these counters. Nodes that implement a VAX CI Port interface to the host should use the CI Port Read Count command (RDCNT).

5.6.1 Counter Descriptions

The following list describes the events to be counted. The two-letter symbolic preface indicates the layer in which the data is collected (ie: PL = Port Layer, DL = Datalink layer).

1. DL_TIME: Time (in 10 msec ticks) since all the diagnostic and performance counters were last zeroed.

2. DL_ACKS_REC: Total Number of frames transmitted by this node successfully (ie: total number of ACKs received).

3. DL_ACKS_SENT: Total number of frames received by this node successfully (i.e., ACKs sent).

4. DL_BUS_EXCEPTIONS: Number of transmission attempts from this node that terminated with error status due to:

   1. Premature target transition to BUS FREE.

   2. Local node asserting Bus Reset (eg: upon detecting an initiator timeout). (Also included in DL_RST parameter below.)

   3. Local node detecting Selection Timeout.

5. DL_NAKSREC: Number frames transmitted from this node for which a NAK status was received during the STATUS IN phase

6. DL_ILLPHASE: Number of frames transmitted from this node that resulted in the target entering an illegal phase.

7. DL_RST: Number of DSSI Bus Resets asserted by this node.

8. DL_RST_REC: Number of DSSI bus RESETS detected but not asserted by this node.

9. DL_CMDVAL_ERRS: Number of command validation errors in frames received by this node (ie: errors in the Command Descriptor block)). This count should include errors due to bits set in reserved fields.

10. DL_DATA_ERRORS: Number of frames received with redundancy errors (checksum or horizontal parity errors) by this node.

11. DL_BYTES_SENT: Total number of bytes transmitted by this node (sum of frame lengths)

12. DL_BYTES_REC: Total number of bytes received by this node (sum of frame lengths)

13. PL_DUP: Total number of duplicate packets discarded by this node's port layer.

14. PL_DG_DISCARD: Total number of Datagrams discarded: CI Datagrams discarded due to unavailable Datagram buffers.

VAX CI Port implementations shall return the specified parameters in the CNTRD response packet fields indicated below. All other counters in the fixed portion of the CNTRD packet shall be set to zero (0). The specification of data returned in other CNTRD counter fields is outside the scope of this document.

1. PATH0_ACK: DL_ACK (total Number of frames transmitted).

2. PATH0_NAK: DL_NAK (number of NAKs received).

3. PATH0_NORSP: DL_BUS_EXCEPTIONS (number of frames transmitted from this node for which the Channel Control layer detected an error other than CC_NAK).

4. DG_DISC: PL_DISC (total number of Datagrsms discarded).

CHAPTER 6

DSSI BUS ELECTRICAL AND PHYSICAL SPECIFICATION

6.1 Introduction

This chapter specifies the electrical and physical requirements for the DSSI bus.

6.2 Electrical Description

6.2.1 Required Signal Termination

The configuration of devices and terminators on the bus shall be as shown in figure 6-1.

All assigned signals shall be terminated with 120 Ohms +/- 2% to Vdd and 285 Ohms +/- 2% to ground at each end of the interconnect (see figure 6-2).

```
         V                                            V
          dd                                           dd
         |                                            |
         \                                            \
      R1 /                                            / R1
         \                                            \
         |                                            |
         +----+-~~~~--+--------+--------+-------+-----+
         |    |       |        |        |       |     |
         \  +---+   +---+    +---+    +---+   +---+   \
         /  |   |   |   |    |   |    |   |   |   |   /
      R2 \  |   |   |   |    |   |    |   |   |   |   \ R2
         /  +---+   +---+    +---+    +---+   +---+   /
         |  Device  Device   Device   Device  Device  |
        GND   7       3        2        1       0    GND
```

Figure 6-1: DSSI Bus Electrical Configuration

```
        V                                     V
         dd                                    dd
        |                                     |
        \                                     \
        /  120 Ohms                           /  120 Ohms
        \                                     \
        /                                     /
        |---        Interconnect           ---|
        |                                     |
        \                                     \
        /  285 Ohms                           /  285 Ohms
        \                                     \
        /                                     /
        |                                     |
       GND                                   GND
```

Figure 6-2: DSSI Bus Terminator Arrangement

6.2.2 Terminator Package

Acceptable terminator packages are listed in appendix \TBS\. Any system not using these terminator packages for both ends of the bus shall use a terminator arrangement identical to that shown in figure 6-3. This means that the decoupling capacitors and resistor network shall be as shown on both ends of the bus.   In addition, the tolerance on the resistors shall be +/- 2%.

```
            +-------------------------------------------------+
            |                                                 |
 TERMPWR ------+-------+------------+----------+------  ...   |
            |  |       |            |          |              |
            | ---     ---           \          \              |
            | --- 1uF --- 0.047uF   / 120      /              |
            |  |       |            \          \              |
            | GND     GND           /          /              |
            |                       +-----+    +-----+  ...   |
            |                       |     |    |     |        |
            |                       \     |    \     |        |
            |                       / 285 |    /     |        |
            |                       \     |    \     |        |
            |                       /     |    /     |        |
            |                       |     |    |     |        |
            |                      GND    |   GND    |        |
            +-----------------------------|----------|--------+
                                          |          |
```

Figure 6-3: Terminator Package Configuration

6.2.3 DSSI Driver

The QDXX standard cell or its equivalent shall be used as the transciever on the DSSI bus. For futher information on the charactoristics of this cell and a list of approved transceivers refer to appendix \TBS\

6.2.4 Electrical Characteristics

The electrical characteristics for the DSSI bus are described in five sections: the output characteristics, the input characteristics, the terminator power characteristics, the terminator power dissipation, and the noise margin.

6.2.4.1 Output Characteristics

The output characteristics are given in table 6-1. They specify the requirements which shall be met by any device driving a signal onto the DSSI bus.

Parameter	Min	Max	Units
Output Characteristics			
Signal Assertion	0.0	0.9	Volts
Signal Deassertion	3.15	3.51	Volts
Driver Output Capability		-87.5 (sinking)	mA
Rise Time	10		nS
Fall Time	10		nS

Table 6-1: DSSI Electrical Output Characteristics

All signals shall be supplied by open collector or tri-state drivers. The bus is pulled-up to the deasserted level by the terminators when the signal is in the high-impedance state. The minimum deasserted voltage level occurs at the minimum TERMPWR of 4.45V. The maximum deasserted voltage level occurs at the maximum TERMPWR of 5.25V.

The output of drivers on the bus shall be trapezoidal for loads in the range 30pF to 300pF and 60 Ohms to 240 Ohms.

6.2.4.2 Input Characteristics

The input characteristics are given in table 6-2. They specify the voltage levels which shall be recognized by any receiver on the DSSI bus.

Parameter	Min	Max	Units
Input Characteristics			
Signal Asserted		1.2	Volts
Signal Deasserted	2.0		Volts
Input Leakage Current	-50 (per signal)	50 (per signal)	uA

Table 6-2: DSSI Electrical Input Characteristics

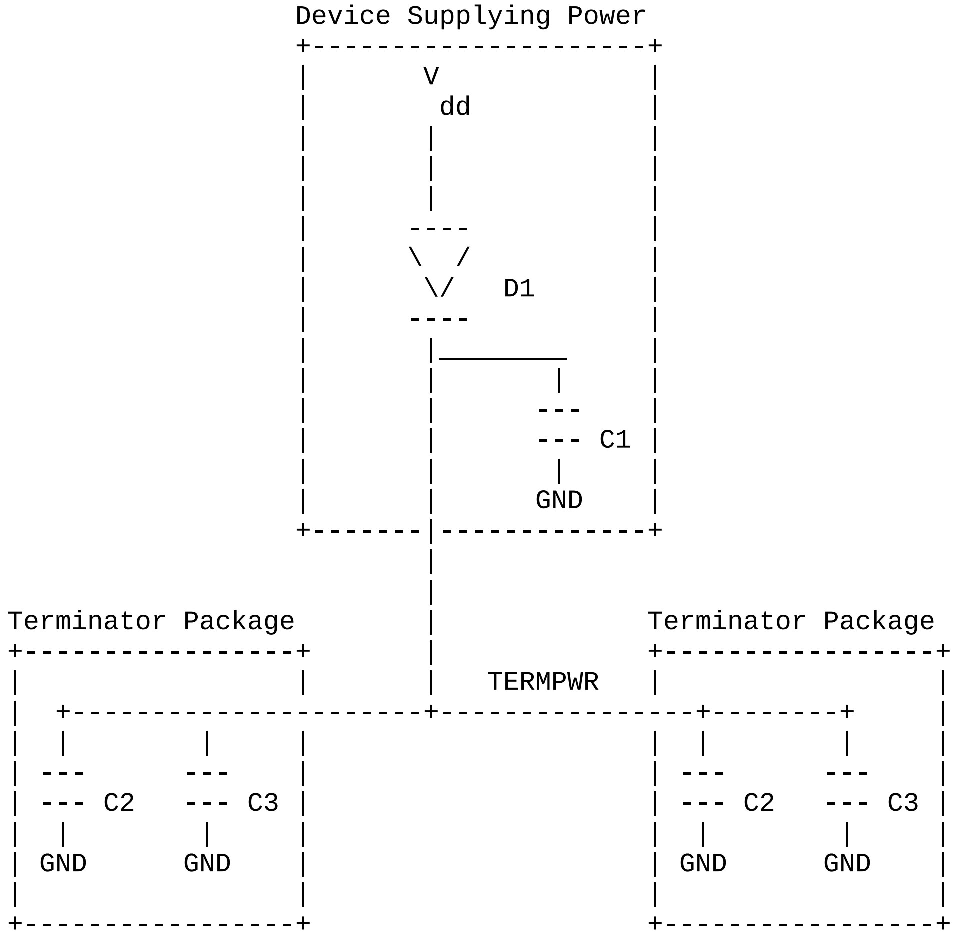
6.2.5 Terminator Power Characteristics

The terminator power characteristics describe the requirements which shall be met by the device supplying power to the terminators. They are given in table 6-3.

The physical configuration for terminator power is shown in figure 6-4. This setup shall be used by any device supplying power to the DSSI bus. The decoupling capacitors C2 and C3 are contained in the terminator package.

Vdd shall be current limited to no more than 2 Amps. This may be accomplished using either a fuse or active current limiting circuit.

TERMPWR shall be between 4.45 Volts and 5.25 Volts.



D1 - A diode or device to prevent the back flow of current from the bus to Vdd shall be used.

Decoupling Capacitors:
C1 - 0.47uF
C2 - 1uF
C3 - 0.047uF

Figure 6-4: Terminator Power Configuration

Parameter	Min	Max	Units
Terminator Power Characteristics			
Supply Voltage (Vdd)	4.75	5.25	Volts
TERMPWR	4.45	5.25	Volts
Source Current per signal	74.2	87.5	mA
Sink Current per signal		-50	uA

Table 6-3: Terminator Power Characteristics

6.2.5.1 Terminator Power Dissipation

This is the power dissipated in the terminators per signal. The power dissipated per signal in the terminator package is given in table 6-4. The total power dissipated in the terminator package depends on the number of signals asserted and deasserted.

Parameter	Min	Max	Units
Terminator Power Dissipation			
Power per asserted signal		213	mW
Power per deasserted signal		68	mW

Table 6-4: Terminator Power Dissipation

The power per asserted signal is calculated at the maximum current of -87.5 mA.

$$\text{Power} = V^2 / R = (5.25 \text{ Volts})^2 / 120 \text{ Ohms} = 230 \text{ mW}$$

The power per deasserted signal is calculated for the maximum voltage of 5.25 volts across the terminators.

$$\text{Power} = V^2 / R = (5.25 \text{ volts})^2 / (405 \text{ Ohms}) = 68 \text{ mW}$$

6.2.5.2 Noise Margin

As illustrated in figure 6-5, the noise margin is the voltage difference between the appropriate receiver threshold and output voltage level, under worst case load conditions, after taking transmission line effects into account. For the DSSI bus, the worst case noise margin is defined as the voltage difference between the deasserted output voltage level and the high-to-low receiver threshold voltage as shown in the figure. Appendix J discusses the factors that influence this parameter.

The worst case noise margin for the DSSI bus, as measured at the transceiver inputs under worst case loading conditions, shall not be less that the minimum value specified in table 6-5.

```
               |\  /\/^--------------.
        _______|_\/__                |
    Worst   ^  |                     |
    Case    |  |                     |
    Noise   |  |                     |
    Margin  v  |  High-to-low        |
       --------|---------------------|------------ 2.1 Volts
               |  Receiver threshold |
               |                     |
               |                     |
               |  Low-to-high        |
       --------|---------------------|------------- .5 Volts
               |  Receiver threshold | ^ Noise
               |                     | | Margin
               |                     | v
               |                     |--------
          _____|                     |/\/^-----------------
```

Figure 6-5: DSSI Noise Margin Definition

Parameter	Min	Max	Units
Noise margin			
Noise Margin	0.5	1.51	Volts

Table 6-5: DSSI Noise Margins

6.3 PHYSICAL CHARACTERISTICS

This section defines the physical characteristics of the DSSI bus, including such factors as bus length, connector and cable part numbers, configuration rules, design rules and rules for connecting and disconnecting nodes from an operating bus.

6.3.1 Maximum Length

The maximum length of the DSSI cable from terminator to terminator shall not exceed 6 meters.

6.3.2 STUB LENGTH

The module etch from the driver/receivers to the DSSI bus connector is referred to as the stub. The stub length shall not exceed 10 cm.

6.3.3 STUB SEPARATION

The stub separation is the mimimum distance between any two devices on the DSSI bus. The mimimum stub separation is 6".

6.3.4 MODULE ETCH REQUIREMENTS

The stub should have a characteristic impedance of no less than 70 ohms with capacitance per foot equal to 24pf +/- 10%.

6.3.5 INTERNAL CONNECTOR REQUIREMENTS

All internal connectors shall be non-shielded. The unshielded connector pin assignments are listed in table 6-6.

6.3.5.1 Device Connectors

Device connectors shall be 50 pin, center keyed, and latchable. They shall consist of two (2) rows of 25 male pins with adjacent pins 2.54mm (.10in) apart. Refer to the engineering specifications for DEC PN 12-16832-03 (3 wall) or 12-21545-03 (4 wall) for details.

The 4 wall connector (12-21545-03) shall be used when the connector is positioned in adherence with DEC STD 030. If the connector is positioned so that the male pins extend off the module edge then the 3 wall connector (12-16832-03) may be used. The device connectors shall be mounted on the component side of the module.

6.3.5.1.1 Female Cable Connector

The female cable connector shall have 50 pins, configured as 2x25 female contacts 2.54mm (.10in) apart. The connector shall be center keyed and strain relieved. Refer to the engineering specification for DEC PN 12-23995-05 for details.

6.3.5.1.2 Male Cable Connector

The male cable connector shall have 50 male pins 2x25 contacts 2.54mm (.10in) apart. Refer to DEC PN (TBD) Engineering spec for details.

6.3.6 INTERNAL CABLE REQUIREMENTS

To minimize discontinuities and signal reflections, cables of different impedances shall not be used on the same bus. The internal cable can be round or flat but shall have the following basic characteristics:

1. 28 AWG
2. 50 conductor (25 twisted pair for round cable)
3. Outside Diameter less than .3 inch for round cable
4. 100 +/- 10% ohm characteristic impedance
5. .065 ohms per foot resistance

For cable characteristics, refer to the engineering specifications for the following DEC part numbers:

1. Flat cable - 91-07738-02.
2. In-cabinet Round cable - 17-01901-01.

6.3.7 EXTERNAL CONNECTOR REQUIREMENTS

360 degree shielded connectors shall be used for external applications where electromagnetic compatibility (EMC) and electrostatic discharge (ESD) protection are required.Refer to DEC PN (TBD) Engineering spec for details.

6.3.8 EXTERNAL CABLE REQUIREMENTS

The external cable shall be 25 twisted pair, 28 AWG, shielded, with a characteristic impedance of 80 ohms. Refer to the engineeering specification for DEC PN 17-01901-01 for details.

Pin Number	Signal Name	Pin Number	Signal Name
1	DATA<0> L	26	TERMPWR
2	GND	27	TERMPWR
3	DATA<1> L	28	TERMPWR
4	GND	29	GND
5	DATA<2> L	30	Reserved
6	GND	31	GND
7	DATA<3> L	32	Reserved
8	GND	33	GND
9	DATA<4> L	34	Reserved
10	GND	35	GND
11	DATA<5> L	36	BSY L
12	GND	37	GND
13	DATA<6> L	38	ACK L
14	GND	39	GND
15	DATA<7> L	40	RST L
16	GND	41	GND
17	PARITY L	42	Reserved
18	GND	43	GND
19	Reserved	44	SEL L
20	GND	45	GND
21	Reserved	46	C/D L
22	GND	47	GND
23	TERMPWR	48	REQ L
24	TERMPWR	49	GND
25	TERMPWR	50	I/O L

Table 6-6: DSSI Signal Assignments

6.3.9 Connecting And Disconnecting A Device From The DSSI Bus

The following sections describe the design requirements for connecting and disconnecting a node to the bus without damage to components.

6.3.9.1 Hot Swap Prohibited

Hot-swap is the act of disconnecting or connecting a DSSI node to the bus while node power is on and the DSSI system is in operation. The ability to perform a hot-swap without harm to any component in a running system is not supported by the DSSI architecture. Hot-swap shall not be allowed on any DSSI system.

6.3.9.2 Power Supply Requirements

The power supplies for DSSI devices shall present an impedance of 1k Ohm or less during the power-down/power-up sequences and while powered off.

6.3.9.3 Mechanical Requirements

This section discusses the following mechanical requirements for safely connecting and disconnecting a DSSI device:

1. Printed circuit board wiring and layout rules,

2. Grounding of multi-cabinet systems,

3. Proper sequencing of DSSI pins when connecting or disconnectiong a device from the bus.

6.3.9.3.1 Printed Circuit Board Wiring And Layout Rules

All DSSI devices shall include a solid ground plane, which shall be directly connected to the logic ground provided by the power supply. All GROUND pins of the DSSI connector shall contact the ground plane directly.

When using the DXX transceivers (see engineering specification \TBS\ ) The VSS, IOVSS, QVSS1, and QVSS2 connections shall directly contact the ground plane.

6.3.9.3.2 Ground Connection

System configurations that use several power supplies within one cabinet or in an expansion cabinet should include ground straps connecting the chassis ground of each power supply to the chassis ground of each cabinet.

It is recognized that for systems that are far apart, grounding in this manner could be impractical. In this case, compliance with with the requirements of section 6.3.9.3.3 is essential.

6.3.9.3.3 Pin Sequencing

The design of cables, connnectors and other related hardware, when supplemented with the appropriate user documentation, shall implement the following rules for connecting or disconnectiong a device from the DSSI bus.

To connect a node to the bus, node power shall be off. The design of connectors and receptacles shall cause DSSI bus connections to be made in the following order:

1. The connector shield,

2. The DSSI GROUND pins,

3. DSSI signal-carrying pins.

Upon applying power to the device, the device shall issue a bus RESET.

To disconnect a node from the bus, node power shall be off. The design of cables, connectors and receptacles shall cause DSSI bus connections to be terminated in the following order when the connector is removed:

- DSSI signal-carrying pins,

- DSSI GROUND pins,

- The connector shield.

APPENDIX A

FAIR ARBITRATION ALGORITHM TIMING PARAMETERS

The basic concept used in the "Fair Arbitration" algorithm is that DSSI nodes can be enabled or disabled. An enabled node may participate in arbitration if it has a message to send, a disabled node may not. The normal sequence of events starts with all nodes enabled. Each node in turn wins arbitration, sends a message, and disables itself. Eventually all nodes are disabled, resulting in the DSSI bus becoming idle. All the nodes recognize that the bus is idle (BSY and SEL both false for a long enough time) and re-enable themselves.

In addition to the bus parameters specified in the fixed priority arbitration scheme (section H.2.1), it is necessary to specify a maximum delay for DSSI nodes to assert BSY and their DSSI ID. This delay is necessary to make the fair arbitration scheme work. "Fair arbitration" basically means round-robin among those nodes that are participating in arbitration.

All nodes participating in arbitration must assert BSY and their DSSI ID within a reasonable time period in order to ensure that they are visible to other nodes.

In order to derive the fair arbitration timings and show that they work, some assumptions or requirements about some timing parameters for the DSSI bus and nodes are required. The parameters needed and their assumed values are:

1. DSSI Cable Delay [200ns]. The maximum or worst-case time for a signal to propagate from the driver output of one DSSI node to the receiver input of any other DSSI node.

2. DSSI Arbitration Skew Delay [600ns]. The maximum allowable skew for a DSSI node to assert BSY and its DSSI ID during arbitration.

3. DSSI Idle Delay [2200ns]. The minimum time a DSSI node must wait before concluding that no enabled nodes are arbitrating for the bus.

4. The value for DSSI Bus Delay is based on Pete Mclean's SCSI timing analysis dated 4-Dec-1984, which gave a propagation delay for a DEC SCSI cable of 1.75ns/foot. This works out to a delay of 143.5ns for 25 meters, and rounding up to 200ns leaves a safety margin.

5. The value for DSSI Arbitration Skew Delay represents approximately 200ns skew for synchronizing a signal to the node's internal clock, 200ns skew for logic delays, and 200ns for logic decision time to arbitrate. Assuming a receiver delay of 65ns, a driver delay of 70ns (these values are quoted on a preliminary NCR data sheet for SCSI driver and receiver), and adding internal logic delays, gives somewhat less than 200ns total logic delays. If the node drives its arbitration state machine off a 200ns clock, a bit more than 200ns skew will result from clock synchronization (200ns skew, not total synchronizer delay). Thus this figure should be attainable for any node that uses a 200ns clock for its arbitration state machine, and trivial for a node that uses a faster clock.

6. DSSI Arbitration Skew Delay specifies the maximum delay for an enabled node to assert BSY and its DSSI ID during arbitration. This means that Step (3) of the fixed priority arbitration algorithm described in section H.2.1 gets replaced by the following:

(3) Following the bus free delay [800ns] in Step (2), an enabled DSSI device that wishes to arbitrate for the DSSI bus shall assert both BSY and its own DSSI ID within a DSSI Arbitration Skew Delay [600ns]. The DSSI device shall not arbitrate (i.e. assert BSY and its DSSI ID) if more than a bus free delay [800ns] plus a DSSI Arbitration Skew Delay [600ns] have passed since the BUS FREE phase was last observed.

From the above information, the DSSI Idle delay needs to be calculated. That is, if the DSSI bus is currently busy, and there is an enabled DSSI device that wishes to arbitrate for the DSSI bus, the maximum time that any other device on that bus may see BSY and SEL both false need to be calculated. The worst case occurs when the other device instantaneously sees BSY and/or SEL become false when they are released by the old device. The maximum time for the other device to see BSY asserted by the new device is the sum of:

[200ns] DSSI Cable Delay from old device to new device
[400ns] Bus Settle Delay for new device to have BUS FREE
[800ns] Bus Free Delay before new device may assert BSY
[600ns] DSSI Arbitration Skew Delay before new device must assert busy
[200ns] DSSI Cable Delay from new device to other device

or a total of 2200ns for DSSI Idle Delay. It should be noted that the DSSI Arbitration Skew Delay accounts for any skew in the new device recognizing BUS FREE.

This means that a device may conclude the DSSI bus is idle if it sees both BSY and SEL false for a minimum of the DSSI Idle Delay [2200ns]. In order for all devices to conclude that the bus is idle, BSY and SEL must both remain false for a minimum of the DSSI Idle Delay [2200ns] plus a DSSI Arbitration Skew Delay [600ns].

1 APPENDIX B

2 SII

3 [Deleted in version 1.0.0]

APPENDIX C

DSSI PARAMETER VALUES

This appendix is a compilation of numeric parameters defined elsewhere in this document.

Table C-1: DSSI General Parameters

Value			Description
Dec.	Oct.	Hex.	
Allowed REQ/ACK Offset Values			
3	03	03	REQ/ACK offset (minimum)
7	07	07	REQ/ACK offset
15	17	0F	REQ/ACK offset (maximum)
DSSI Command Out Format			
224	340	E0	Data Link Operation Code
DSSI Status In Format			
97	141	61	ACK
30	036	1E	NAK

Table C-2: Selected DSSI Timing Parameters

Description	Value
Initiator Timeout	2800us
Selection Abort Timeout (minimum)	20us
Selection Abort Timeout (maximum	30us
Selection Timeout (minimum)	20us
Selection Timeout (maximum)	30us
Target Timeout	2400us

APPENDIX D

IMPLEMENTATION FUNCTIONALITY

This appendix lists parameter values and describes the behavior of existing DSSI products in areas where the specification permits implementation-specific choices.

The behavioral descriptions are for information only. In cases where a choice is allowed and the specification expresses a preference, new implementations shall comply with the preferred behavior unless precluded from doing so by overriding considerations.

D.1 Section 5.5.8.1 "Data Block Length"

The referenced section specifies DSSI datalink requirements for receiving and buffering packets received during the DATA OUT phase.

As described in the referenced section, DSSI targets should be able to safely receive and buffer the maximum packet size that can be specified in the Command Descriptor block (8191 bytes). Packets longer than the upper limited expected by a higher layer are to be truncated and forwarded with notification that an oversize packet was received.

Due to implementation-specific constrains, DSSI products are not able to conform to this recommended behavior. The behavior of existing implementations is described in the following sections.

D.1.1 SWIFT And SII

SWIFT and SII are low-cost, single-chip implementations of DSSI datalink Channel Control functions. Both must be supplemented by an attached processor and memory to provide full CI Port functionality. SWIFT is a second generation replacement for SII that provides enhanced detection of memory errors and CI port layer assists to offload the processor.

The hardware will receive datalink packets up to 8190 bytes in length directly into hardware-managed buffer queues The maximum effective size, which determines the size of each queue entry, is negotiated between the sender and receiver through the CI port protocol and is less than the upper datalink limit. The hardware has no way to prevent buffer overflow due to a packet that is longer than this negotiated size. However, SWIFT's added capabilities facilitate the detection of such overflow occurrances.

The response to overflow is implementation-specific. DSSI devices may invoke crash recovery code when such an event is detected. A DSSI host, on the other hand, may attempt a graceful recovery by terminating all virtual circuits serviced through the host adapter.

D.1.2 SHAC And DASH

As described in section 5.5.8.1, "Data Block Length", SHAC and DASH check the length in the Command Descriptor block against the maximum allowed size of a CI port layer packet body (specified in the MAX_BODY_LEN field of a CI port layer ID packet). The transfer is terminated with a NAK STATUS IN value before the start of the DATA OUT phase if the length exceeds the above limit.

D.2 Table 5-2, "Summary Of Channel Control Layer Status Values"

The status values, returned by SHAC and DASH Channel Control layer implementations differ from the values summarized in this table. SHAC and DASH return CC_NORSP For the following events, instead of CC_NAK, as called out in the table, for the following cases:

Condition	Section/Page
Invalid information transer phase.	Section 5.5.7.4, page 5-27,
Inititiator Timeout.	Section 5.5.7.4, 5-27.
"Third Party" bus reset.	Section 5.5.7.4, page 5-27.
Phase other than BUS FREE following STATUS IN.	Section 5.5.8.4, page 5-37.

Table D-1: SHAC/DASH Differences in Channel Control Layer Status

D.3 CI Port Maintenance I/D's

The following is a tabulation of CI Port maintenance I/Ds for DSSI products. This list is provided for information only. The assignment of official maintenance I/Ds for products that are compatible with the CI port shall be done through the CI Architect.

Product	Maintance I/D
KFQSA	21h
SHAC	22h
DASH	24h
RF71	30h
RF30	31h
RF31	32h
TF70	40h
TF85	41h
MF2	20h

Table D-2: CI Port Maintenance I/Ds

APPENDIX E

USE OF CREDITS IN SCA/MSCP SPECIFICATIONS

This section is intended to clarify the use of credits in SCA, CI, and MSCP architecture specifications.

SCA provides credit based flow control for each connection. Whenever a connection exists, each side has some non-negative number of credits, termed a credit balance. Credits are initially established when the connection is formed, and all credits disappear when a connection terminates.

In any node or system, there will be a queue of things waiting for credits and a queue of things that have credits waiting to be transmitted. Depending on how the system is structured, the two queues can be anywhere.

About the only consistent rule is that the queue of things that have credits will be at a lower layer than the queue of things waiting for credits. In VMS the queue of things waiting for credits is maintained at the interface between the MSCP/TMSCP server (SYSAP) layer and the SCS layer. The important thing is that items in this queue are waiting for credits, not where the queue resides. Plus, from the perspective of MSCP or TMSCP, a command is completed when its end message is placed in this queue --- the server does not have to wait for a credit to arrive to consider the command complete.

In SII, the queue of things waiting at the SII transmit queue already have credits. The SII transmit queue represents deferral queue at the data link layer. However, deferral queues used to hold packets awaiting transmission retries, and port layer queues in the Port Queue Block are all equivalent. The important thing is that items on all of these queues already have whatever credits they need, and are waiting for access to the transmission media.

MSCP, TMSCP, and DUP class drivers (hosts) use credits to send commands to servers. This is described in the MSCP Spec and is not particularly unique or interesting in either SCA or DSSI.

MSCP, TMSCP, and DUP servers require credits to:

1. Send an end message.

2. Send an attention message.

3. Send a DATREQn packet.

4. Send a sequence of SNTDAT packets with a new transaction ID (XCTID). The Last Packet flag will be set in the last SNTDAT packet and will mark the end of a SNTDAT sequence that use the same XCTID. It should be noted that one credit is required for sending the first SNTDAT packet with a new XCTID. All the subsequent SNTDAT packets that use the same XCTID do not require additional credits. A credit is returned when the server receives CNF packet from remote node.

Servers may only send these messages when their credit balance is greater than zero. The act of sending one of these messages decreases the server's credit balance by one. In this context "send" means that the message has been queued to the Data Link layer for transmission. It does not mean that the message has been queued to the Transport (Port) or Session (SCA) layer, and will be transmitted once sufficient credits arrive (it should be noted that "send" is used with this latter meaning in some parts of the MSCP Spec).

MSCP, TMSCP, and DUP servers obtain credits by:

1. Receiving a Connect request and accepting the connection.

2. Receiving a Credit request.

3. Receiving an SCA Application Message.

4. Receiving a RETDAT packet with the Last Packet flag set.

5. Receiving a CNF packet.

The first three of these contain a credit field, which contains the number of credits granted. Strictly speaking, the server only obtains credits if it receives one of these messages with a non-zero positive credit field. The RETDAT and CNF packets grant exactly one credit. The server must supply a transaction ID field in DATREQn and SNTDAT packets that allows it to determine the corresponding connection when it receives a RETDAT or CNF packet.

The SCA Spec and the above description are written as if the server kept a single pool of credits for each connection. All credits are the same, and the server checks (and decrements) the credit balance immediately before sending each message or packet. Such an implementation maximizes flexibility, and is preferred in the absence of other criteria. However, it is not the only valid implementation.

VMS implementation follows the above "full" dynamic credit management. It maintains a single pool of credits per connection, and will not allow a SYSAP to send any sequence message packets (SNDMSG, REQDAT, SNDDAT) if there are no send credits available. Such messages will be held by the transport layer (SCS layer) until a send credit is available. While such an implementation is preferable due to it's inherent flexibility, and generality, there can be reasons for being less flexible and pre-allocating credits. For example, if the cost of checking credits at the time a packet is about to be sent is substantial, pre-allocating credits can eliminate the cost. HSC uses such an approach.

MSCP, TMSCP, and DUP servers may maintain independent pools of credits for different purposes on a single connection. That is, servers may pre-allocate credits to a specific purpose, presumably to optimize performance.

A good example is the HSC's use of credits.

The HSC maintains four distinct, separate pools of credits, one for each class of messages. A single credit is reserved for SNTDAT packets with the Last Packet flag. Similarly, a single credit is reserved for each outstanding command for that command's end message. A pool of DMA credits is used for DATREQn packets, and a pool of excess credits is used for attention messages.

The HSC requires a minimum of three initial credits with each Connect request. One is the SNTDAT credit, one is for the DMA credit (DATREQn) pool, and one is for the excess credit (attention message) pool. Additional initial credits are divided between the DMA and excess credit pools.

The HSC requires that each SCA Application Message (MSCP or TMSCP command) grant a minimum of one credit. This is the credit reserved for the command's end message. Any additional credits beyond one are added to the excess credit pool for attention messages. Similarly, any credits granted by Credit requests are also added to the excess credit pool.

In effect, the HSC only dynamically manages credits for attention messages. All other uses have their credits statically pre-allocated.

Now the HSC's example, while illustrative, may NOT be followed by DSSI devices. It will, of course, work perfectly well with ports such as SHAC and Mayfair II that provide a direct interface to the host class driver. However, the HSC's scheme will not work with the KFQSA.

The problem is that UQSSP does not require host class drivers to explicitly report credits to the port, controller, or server. Thus the KFQSA cannot determine how many credits the host has granted the server. In particular, the KFQSA cannot determine when the host grants excess credits for attention messages. Thus DSSI devices may statically pre-allocate the initial credits granted by the Connect request, in particular for data transfer operations, but must dynamically allocate credits for end messages and attention messages.

E.1 DSSI Device Credit Management

The following guidelines describe the DSSI devices' use of credits. It should be noted that these rules only apply to connections that use the MSCP and TMSCP protocols.

1. Class drivers shall provide a minimum of 3 initial credits with the Connect request for MSCP and TMSCP connections. Class drivers should specify 2 in the min_credit field of the Connect request.

2. Servers shall set the min_credit field of the Accept request to the minimum number of credits needed by the server.

3. Servers shall use a single pool of credits for end messages and attention messages. DATREQn packets and SNTDAT packets may share this credit pool (for full dynamic credit management), share a separate data transfer credit pool, or have separate credit pools for

each type of data transfer.

4. Servers shall allocate at least one initial credit to each of the end/attention message pool, the DATREQn pool, and the SNTDAT pool. If the server has fewer than three pools, due to sharing one pool for several purposes, then it shall allocate at least one initial credit to each pool. Additional initial credits, beyond the number of server credit pools, may be allocated in whatever manner the server desires.

5. Servers shall provide full functionality with the minimum 3 initial credits. However, they may require more credits to achieve full performance and in order to have multiple data transfer operations outstanding.

6. In general, DSSI servers should provide full performance with 5 initial credits. This is 3 credits for block data transfer operations, and 2 credits for messages. Any DSSI device that requires more that 5 initial credits to achieve full performance should prominently advertise and review their design with other DSSI implementors and host operating systems.

7. DSSI servers cannot assume that class drivers will follow the rules described in Chapter 3 of the MSCP specification.

   In particular, servers must operate normally even when they have fewer credits than outstanding commands. This means that servers must check for and properly handle the case where a command completes, so they have an end message to send, but do not have a credit available to send the end message. The server must queue the command and send it when the class driver provides sufficient credits. This queue effectively lies between the MSCP/TMSCP server and SCS layers. Queued messages must be sent in the same order that they were queued. In terms of "Command Categories and Execution Order", an MSCP or TMSCP command is completed when the end message is placed on this queue.

8. DSSI servers must ensure that attention messages do not lockout end messages and data transfer operations. That is, they must ensure that attention message are not continuously sent while end messages and/or data transfer operations are waiting for credits. One means

of achieving this, described in the MSCP Spec, is to limit the rate at which attention messages are generated to no more than two attention messages per second, averaged over the controller timeout interval. Another (preferable) means is to only generate attention messages when credits are available to immediately send them.

9. When messages of various types are waiting for credits, and the server receives some credits, the following is the recommended order in which the new credits be utilized:

   1. First, use the credits for any messages (primarily end messages) queued between the server and SCS layers.

   2. Next, use the credits to initiate any pending data transfer operations. In general, give SNTDAT operations priority over REQDATn operations.

   3. Finally, use the credits to generate any pending attention messages.

These are guidelines, not requirements --- they will tend to maximize performance. Only servers that implement full dynamic credit management will be able to follow all of them.

The guidelines for DUP connections are similar, but with two differences.

First, DUP does not have attention messages, so the initial credit for attention messages need not be provided. Second, it is reasonable to use DUP without data transfer operations. So, if the class driver knows that it will never issue a DUP command that contains a buffer descriptor, it may supply zero initial credits in the Connect request. Otherwise, the class driver shall provide at least two initial credits in the Connect request. If the class driver supplies zero initial credits, and subsequently issues a command that contains a buffer descriptor, the DUP server may either honor the command (if it implements full dynamic credit management) or break the VC.

SCS$DIRECTORY uses neither attention messages nor data transfer operations, and thus can be implemented with zero initial credits.

APPENDIX F

SCA AND MSCP CLARIFICATIONS

F.1 SCA Additions And/or Clarifications

This section attempts to clarify or amend sections of the various SCA documents.

F.2 Format Of Buffer Descriptors And Use In SNTDAT/DATREQn Packets

DSSI drives must support both the CI Port and SSP buffer descriptor formats. The reason for this is that the data transfer commands received by a drive may be from an SSP type host interface (KFQSA), or CI Port type host interface(SHAC, MAYFAIR Port, DASH) etc. Thus the buffer descriptors they receive as part of the transfer commands may be in either of the following two formats.

1. SSP Unmapped Buffer Descriptor Format - SSP Specification, Ref [7]. Host adapters with SSP type host interface (KFQSA) must ensure that the third longword of the buffer descriptor is either a valid Connection ID, or a zero.

2. CI Port Buffer Descriptor Format - VAX11-CI Port Architecture Specification, Ref [6].

F.3 Port And VC Concepts

Similar to a CI node, a DSSI node may be in any of the six global states: Uninitialized, Disabled, Enabled, with and without maintenance functions enabled.

The CI Specification (Pages 70-71, Ref [5]), describes the above states and their state transitions. It should be noted that all the six states are not required to be implemented in a node. The Uninitialized/Maintenance state must be implemented if the node RESET and START functions are supported.

The other /Maintenance states are optional.

It should be remembered that the normal operational state of a node is either ENABLED (if the node does not require external reset from another DSSI node) or ENABLED/MAINTENANCE (if the node supports external reset from another DSSI node); the virtual circuit services (message, datagram, and block data transfer services) are provided by a node only in these two states.

F.3.1 Example Port State Diagram In DSSI Drives

The state diagram of figure F-1 illustrates the minimum required port states that must be supported by RFxx and TFxx products. In addition, it is suggested that as many diagnostics as possible be run in response to a received RST.

```
                 `Power up'
                      |
                      V
              .---------------.
     .--------| Uninitialized |
     |        `---------------'
     |diags
     |passed  .------------------------.          .-------.
     |        |         STRT           |          |  RST  |
     V        V                        |          V       |
.---------------.      RST      .---------------------.   |
| Enabled/Maint |--------------->| Uninitialized/Maint |---'
`---------------'               `---------------------'
```

Figure F-1: Minimum Port State Diagram in DSSI Drives

F.3.1.1 Uninitialzed State

In this state the node does not send/receive any packets. The datalink interface will either return failure status (NAK or NO_RSP), or will not respond to selections on the DSSI Bus.

F.3.1.2 Uninitialized/Maint State

In this state the datalink will acknowledge any received packets. ID packets will be returned in response to IDREQ packets. This state is further explained in Sec 5.4, Page 72, DEC STD 161.

F.3.1.3 Enabled/Maint State

This is the normal operating state for DSSI drives, and communication services for the higher layers are supported only in this state.

F.4 Generation Of Unique Node Parameters

Two parameters types need to be differentiated in a node; critical parameters and non-critical parameters. All the parameters required to communicate with a DSSI node, even in a minimum configuration, are critical parameters and the remaining parameters are non-critical. Non-critical parameters that have to be saved across power failures are stored in non-volatile memory and altered via a suitable diagnostic program (for example via DUP in DSSI drives). Critical parameters are set to their default values on powerup (or on node reset), and changes made to them after powerup are volatile. The reason for this approach is that many DSSI drives do not have a management interface (separate console interface), and a guaranteed communication path is required to change any parameters stored in a semi-permanent medium such as NVRAM.

Critical parameters should not be stored in NVRAM, since any inadvertent, or erroneous changes to the NVRAM may prevent further communication with the drive. While most of the parameter definitions and their significance are node specific, the parameters described in this section are mandatory requirements in every DSSI node.

F.4.1 SCA And MSCP Parameters

The SCA and MSCP architectures require that certain parameters be unique across a shared bus. DSSI implementations that do not have a stand-alone maintenance interface (e.g.; most DSSI device nodes), shall automatically generate default values for these parameters as described below.

F.4.1.1 SCA System Name

DSSI nodes that automatically generate a default System Name shall use the following guidelines:

1. Ignore human readability

2. Make the name as unique as possible, for example, by deriving the value from serial number.

3. The name should be of the form, 'ppdddd', where:

   - pp: is the product name, abridged to two characters (eg.,R7)

   - dddd: is a base-36 string of characters (the consonants in the letters "A" -"Z" and the numbers "0" - "9") formed by the most unique part of the device's serial number taken six bits at a time, the overflow (above 35) is carried to the next six bit group.

4. The six characters in the name shall not include any of the five vowels.

5. The name should be alterable by a diagnostic program.

NOTE

Rules 1, 3 and 4 do not apply to manually generated System Names.

The KFQSA adapter should use a similar approach. In nodes with a management interface this parameter is an alterable system configuration parameter; such alterations shall be preserved across power failures.

F.4.1.2 SCS SYSTEMID

This 48 bit parameter should be derived from the following guidelines.

1. The default values for this parameter should be derived from drive serial number (or Ethernet Address).

2. The parameter must be changeable, and changes must be preserved across power failures.

3. It shall be unique across the domain of all Digital products.

Every DSSI node polls all other DSSI nodes by attempting to send IDREQ packets. If the transmission succeeds (remote node exists, powered up, and not in uninitialized state), the receiver will respond with ID packet indicating the Port State. After exchanging IDREQ/ID packets, a node will then attempt to establish a virtual circuit to this node if a VC is not already open. A VC is established between two nodes by the START/STACK/ACK handshake described in Sec 9.2 of SCA Specification, Ref [4].

The default values for the polling interval, and the number of nodes to be polled per polling interval are established in one of the following ways.

In nodes that have a management interface, the default values are alterable (for example through SYSGEN parameters in host nodes that do not use the KFQSA).

In nodes that do not have a management interface (eg., DSSI drives, and KFQSA), the values are stored in a EEROM, and may be altered through a diagnostic program.

F.5 SCA Packet Format Clarification

The SCA Spec is somewhat confusing in its depiction of PPD packet formats. Specifically, the LENGTH field shown just before the PPD_TYPE field is *not* part of the packet body. The length field is derived from the frame length transmitted in the command out phase; it does not appear in the data out phase in any form.

For example, the format of a PPD START datagram is illustrated in section 10.1.1 of the SCA spec as:

```
3               1
1               5               0
+---------------+---------------+
|   PPD_TYPE    |    LENGTH     |
+---------------+---------------+
|                               |
=           START_DATA          =
|                               |
+-------------------------------+
```

Figure F-2: Format of PPD START Datagram

and illustrated in Appendix F of the SCA Spec with more detail as:

```
                 3
                 1                             0
                +-------------------------------+
                |   PPD MTYPE   |   LENGTH      |
                +-------------------------------+
                |           SEND_SYS            |
                +---------------+               |
                | RSV   | PRTVRS|               |
                +---------------+---------------+
                |    MAX_MSG    |    MAX_DG     |
                +---------------+---------------+
                |             SW_TYPE           |
                +-------------------------------+
                |           SW_VERSION          |
                +-------------------------------+
                |         SW_INCARNATION        |
                |                               |
                +-------------------------------+
                |             HW_TYPE           |
                +-------------------------------+
                |                               |
                =           HW_VERSION          =
                |                               |
                +-------------------------------+
                |           NODE_NAME           |
                |                               |
                +-------------------------------+
                |             CUR_TIME          |
                |                               |
                +-------------------------------+
```

Figure F-3: Detail of PPD START Datagram

What is actually sent during the DSSI data out phase for a START datagram is the following:

```
 31        24 23      16 15          8 7             0
+-----------+-----------+--------------+---------------+
| PPD_TYPE = START = 0  | CI flags = 0 | CI opcode = 1 |
+-----------+-----------+--------------+---------------+
|                   SEND_SYS                           |
+-----------+-----------+                              +
|    RSV    |   PRTVRS  |                              |
+-----------+-----------+--------------+---------------+
.                                                      .
.                                                      .
+-----------+-----------+--------------+---------------+
|                                                      |
+---                 CUR_TIME                       ---+
|                                                      |
+-----------+-----------+--------------+---------------+
```

Figure F-4: PPD Start Datagram Packet

and the frame length passed in the DSSI command out phase is START_LEN+2 or 64 (decimal).

The reason for this discrepancy is that the formats shown in the SCA Spec are the formats passed to and from the VAX CI port --- not the format that appears on the wire. The CI port interface has the packet length field in the position shown in the SCA Spec. The CI port adjusts the length field for the CI opcode and CI flags bytes, which is why the length that appears in the SCA Spec is 2 bytes less than the actual length on the wire.

Another example is a Disk MSCP transfer command. This is somewhat similar to Appendix G of the SCA Spec but with more detail. What is actually sent during the DSSI data phase is the following:

```
      31       24 23      16 15       8 7        0
      +----------+----------+----------+----------+<-------
      | PPD_TYPE =          |CI flags  | CI opcode| PPD Hdr
      |       SCS_MSG = 4   | = seqno  |     = 2  |
      +-----------+---------+----------+----------+<-------
      |       CREDIT = 1    |  SCS_TYPE           |
      |       ( usually )   |      = APPL_MSG = 10|
      +-----------+---------+----------+----------+
      |              REC_CONNECT_ID               | SCS Hdr
      +-----------+---------+----------+----------+
      |              SEND_CONNECT_ID              |
      +-----------+---------+----------+----------+<-------
      |        MSCP command reference number      |
      +-----------+---------+----------+----------+
      |       Reserved      |     Unit number     |
      +-----------+---------+----------+----------+
      |       Modifiers     |  Flags   |  MSCP    |
      |                     |          |  Opcode  |
      +-----------+---------+----------+----------+
      |                 Byte Count                |
      +-----------+---------+----------+----------+
      |                                           | MSCP
      +---                                     ---+ Cmd.
      |             Buffer Descriptor             | Packet
      +---                                     ---+
      |                                           |
      +-----------+---------+----------+----------+
      |           Logical Block Number            |
      +-----------+---------+----------+----------+
```

Figure F-5: SCA Message Packet containing MSCP Command

The frame length sent in the DSSI command out phase is:

(MSCP command packet length) + AP_MSG_HDR_LEN + 2

= 32 + 14 + 2 = 48

AP_MSG_HDR_LEN is defined in SCA Appendix D. The "+2" is for the CI opcode and flags, just as before.

Packet formats defined in DEC STD 161 are sent exactly as shown in that document. They include the CI opcode and flags fields but no other header information. The actual packet formats are described in section 4 of DEC STD 161 and the CI opcodes are summarized in Appendix A. These packet formats include:

SNTDAT	Sent Data
CNF	Confirm For Sent Data
DATREQ0/1/2	Data Request
RETDAT	Returned Data
IDREQ	ID Request
RST	Reset
SNTMDAT	Sent Maintenance Data
MDATREQ	Maintenance Data Request
STRT	Start
ID	ID Information
RETMDAT	Returned Maintenance Data
MCNF	Confirm For Sent Maintenance Data
LB	Loopback

The DG (datagram) and MSG (sequenced message) CI opcodes both imply that a PPD header is present. The PPD header, as sent across the wire, includes just the PPD_TYPE field. The following values of PPD_TYPE means that the packet has the format described in chapter 10 of the SCA Spec:

START
STACK
ACK
ERROR_LOG
NODE_STOP

If PPD_TYPE is SCS_DG or SCS_MSG, then an SCS header is present. The SCS header is the SCS_TYPE, CREDIT, REC_CONNECTION_ID, and SEND_CONNECTION_ID fields.

It should be noted that some of these fields are reserved in some of the SCS message types, as dictated by the SCS_TYPE. The following values of SCS_TYPE mean that the packet has the format and interpretation described in chapter 8 of the SCA Spec:

CONNECT_REQ
CONNECT_RSP
ACCEPT_REQ
ACCEPT_RSP
REJECT_REQ
REJECT_RSP
DISCON_REQ
DISCON_RSP
CREDIT_REQ
CREDIT_RSP

It should be noted that chapter 8 shows these messages without the PPD header and CI header.

If SCS_TYPE is APPL_MSG or APPL_DG, it means that an MSCP, TMSCP, DUP, or some other application protocol message immediately follows the SEND_CONNECT_ID field. This is illustrated in SCA sections 8.3 and 8.4 and in the previous paragraph. The PPD header and CI header are of course present.

The important point to remember is that the data structures that appear in the SCA document are completely oriented towards a VAX host system. That is, they are the data structures as they appear at the CI port interface, not as they appear on the wire. The format of a sequenced message at the CI port interface is the following (see, for example, section 7.2.1 of the VAX CI Port Spec) (diagram is 32 bits wide):

```
31                             0
+-------+-------+-------+-------+
|             FLINK             |
+-------+-------+-------+-------+
|             BLINK             |
+-------+-------+-------+-------+
|     Reserved for Software     |
+-------+-------+-------+-------+
| Flags |  OPC  |  MBZ  |  PORT |
+-------+-------+-------+-------+
|               |    MSG_LEN    |
+               +-------+-------+
|                               |
.                               .
.              MSG              .
.                               .
|                               |
+-------+-------+-------+-------+
```

Figure F-6: Format of VAX CI Port Packet

where PORT is the destination node number, OPC is the CI port opcode (*not* the CI packet opcode), and Flags is the flags field, somewhat similar in both CI port and CI packet. MSG_LEN is the length of MSG, which is the message text. This gets translated to the following CI packet format "on the wire" (diagram is 8 bits wide):

```
        .
|    preamble   |
+---------------+
|               |
+-    length   -+
|               |
+---------------+
|      dest     |
+---------------+
|   dest compl  |
+---------------+
|     source    |
+---------------+
|  opcode = 2   |
+---------------+
| flags = seqno |
+---------------+
|               |
.               .
.      text     .
.               .
|               |
+---------------+
|      CRC      |
        .
```

Figure F-7: Format of CI Packet

where "dest" is the destination node ("PORT" above), "dest compl" is its ones complement, "source" is the source node. "text" is the message text, copied from "MSG" above. "length" is the length from "length" through "text", which means it is MSG_LEN+7.

The same packet appears on DSSI as the following:

```
        Command Phase:

 7                    0
+----------------------+
|  DSSI opcode  ( E0h) |
+----------------------+
|  REQ/ACK |           |
+----------------------+
|        Dest          |
+----------------------+
|       Source         |
+----------------------+
|       Frame          |
+-      Length        -+
|                      |
+----------------------+
|      Checksum        |
+----------------------+

          Data Phase:

 7                    0
+----------------------+
|      Opcode = 2      |
+----------------------+
|    Flags = seqno     |
+----------------------+
|                      |
=        Text          =
|                      |
+----------------------+
|      Checksum        |
+----------------------+
```

Figure F-8: DSSI Packet Format

The "dest", "source", and "text" fields are the same as before. However, "frame length" is MSG_LEN+2.

For how this relates to the message formats shown in the SCA Spec, the Example CI Message shown in Appendix G should be considered.

That example message is:

```
+-------+-------+-------+-------+
|             FLINK             |
+-------+-------+-------+-------+
|             BLINK             |
+-------+-------+-------+-------+
|SWFLAG | TYPE  |     SIZE      |
+-------+-------+-------+-------+
| Flags |  OPC  | STATUS|  PORT |
+=======+=======+=======+=======+
|   PPD MTYPE   |    LENGTH     |
+=======+=======+=======+=======+
|    CREDIT     |   SCS MTYPE   |
+-------+-------+-------+-------+
|    DESTINATION CONNECT ID     |
+-------+-------+-------+-------+
|      SOURCE CONNECT ID        |
+=======+=======+=======+=======+
|                               |
=   Application Message Text    =
|    (e.g., MSCP command)       |
+-------+-------+-------+-------+
```

Figure F-9: SCS Application Message in VAX CI Port Packet

First, it should be noted that the "FLINK" through "LENGTH" fields are exactly the same as the CI port message format illustrated earlier. The "reserved for software" field has been interpreted, but everything else is the same.

Second, the PPD message descriptions that appear in SCA Chapter 10 start at the LENGTH and PPD MTYPE fields in this illustration; preceding fields are omitted from Chapter 10.

Third, the SCS message descriptions that appear in SCA Chapter 8 start at the SCS MTYPE and CREDIT fields in this illustration; preceding fields are omitted from Chapter 8.

Per the previous description, this message translates to the following sequence of DSSI Data Phase bytes:

```
+---------------+
|  Opcode = 2   |
+---------------+
| Flags = seqno |
+===============+
|               |
+-  PPD MTYPE  -+
|               |
+===============+
|               |
+-  SCS MTYPE  -+
|               |
+---------------+
|               |
+-   CREDIT    -+
|               |
+---------------+
|               |
+-             -+
|  DESTINATION  |
+-   CONNECT   -+
|      ID       |
+-             -+
|               |
+---------------+
|               |
+-             -+
|    SOURCE     |
+-   CONNECT   -+
|      ID       |
+-             -+
|               |
+===============+
|  Application  |
=    Message    =
|     Text      |
+---------------+
|    Checksum   |
+---------------+
```

Figure F-10: SCS Application Message in DSSI Data Block

There is no length field anywhere in the DSSI Data Phase transmission. The only length field is the one that appears in command phase. The length field that appears in the PPD message descriptions is derived from the command phase frame length field.

APPENDIX G

REVISION HISTORY

This appendix tracks the revision history of the document. Please note that the change bars present in the document only reflect changes from the previous version and are NOT cumulative.

Version 1.0.0

Date: 27 March 1990

General:

Version approved for ECO control.

APPENDIX H

EXCEPTIONS

H.1 Policy

An exception allows a product to differ from the specification in some way throughout the product's lifetime.

The exceptions described in the following sections have been granted to accomodate products in existence when the specification was prepared. The decision to grant such exceptions was made on the basis that the area of non-compliance, although resulting in less than optimum behavior, does not seriously compromise the long-term architectural goals.

All future exception requests shall be submitted as proposals for ECO's to this document.

H.2 Exceptions For Products Using The SII Bus Interface Chip

The SII chip is a first-generation bus interface chip that does not support the architecural features described below. Products that interface to the DSSI bus using this chip shall therefore adhere to the following requirements:

Section 5.5.6 SELECTION Phase -

> The SII chip has no selection timeout mechanism, therefore such products shall detect a selection response failure using the initiator timeout mechanism defined in section 5.5.1.4, Watchdog Timers.
>
> Note that these products shall, nevertheless, fully comply with the selection response requirements of section 5.5.6.

When such an initiator detects an initiator timeout or BUS RESET, it's Channel Control layer shall report such conditions with status CC_NORSP.

Section 5.5.5, Fair (Round-Robin) arbitration -

> The SII chip does not support round-robin arbitration. Instead, the fixed priority arbitration mechanism defined in

section H.2.1 shall be used.

The following is a list of products that use the SII bus interface:

- The KFQSA 6-Channel SSP controller
- The RF30 150mb winchester disk drive
- The RF71 400mb winchester disk drive
- The Mayfair II DSSI I/O Adapter

H.2.1 SII Fixed Priority Arbitration Scheme

This section documents the fixed priority arbitration scheme used on DSSI products implemented with the SII. This scheme is obsolete and shall not be implemented on any other DSSI products.

The timing for fixed priority arbitration is detailed below:

1. A DSSI device that needs bus access (Initiator) shall first wait for the BUS FREE phase to occur.

2. The device shall then wait 800 ns before driving any signal.

3. Following this delay, devices may arbitrate by asserting both BSY and their own DSSI ID. However, no device shall attempt arbitration if more than 1400 ns have passed since the BUS FREE phase was last detected. As long as the bus remains free, there is no maximum delay before arbitrating.

4. After waiting 2200 ns following assertion of BSY, the device shall sample the bus. If a higher DSSI ID is present on the bus or SEL is asserted, the device has lost arbitration and shall release its signals and return to step 1. If no higher DSSI ID is present and SEL is not asserted, the device has won arbitration and signals this by asserting SEL. All other devices must release their signals within 800 ns after SEL becomes true.

5. The device that has won arbitration shall wait at least 1200 ns after asserting SEL before changing any signals.

H.3 Exceptions For SHAC/DASH

SHAC and DASH are products based on a single-chip implementation of the VAX CI port layered on a DSSI datalink. SHAC (Single Host Adapter Chip) interfaces a single DSSI bus to a Microvax host while DASH (Dual Adapter - Single Host) interfaces two DSSI busses to an XMI-based host.

H.3.1 Target Timeout Enforcement

Section 5.5.6, SELECTION Phase, requires the target to begin the Target Timeout interval from the point at which the target detects valid selection. As a side-effect of host bus congestion, SHAC/DASH implementations may delay the start of the target timeout interval. In extreme cases this delay might result in an effective target timeout that exceeds the initiator timeout.

The arithmetic average of such delays, measured from the assertion of BSY by the target to the start of the timeout interval, shall not exceed 200us when observed over any ten-minute interval.

APPENDIX J

TERMINATOR SELECTION

In the selection of the terminator values, there were three main areas of interest: the signal integrity, particularly ringing and reflections due to impedance mismatches; the noise margin; and the low level output current.

The major factor in signal integrity on the interconnect is the relative impedances of the different components. The terminator values should be matched as closely as possible to the overall impedance of the interconnect. Impedance mismatches result in overshoot and ringing on the bus.

The effective impedance of the DSSI interconnect depends on the cable configuration and the number of devices present on the bus. Calculations show that the impedance can range from 65 to 86 Ohms over the variety of different cabling and device possibilities. As a result it was not possible to select a termination scheme which eliminated impedance mismatches. The selected terminators have an equivalent impedance of 81.5 Ohms. This value provides a good impedance match for most of the possible configurations, as was verified with extensive simulation.

The noise margin is the voltage difference between the high level output voltage, determined by the terminators, and the high-to-low receiver threshold voltage. This noise margin can be degraded by undershoot caused by ringing and reflections on the bus. The problem of impedance mismatches between the terminators and some interconnect configurations does cause undershoot in some cases. Therefore it was necessary to select terminators such that an adequate noise margin was maintained even for the worst case cable and loading conditions. A value of Voh=3.15 volts at a worst case TERMPWR of 4.45 volts was selected. Simulations of the worst case conditions (fully loaded system and attached expansion box) show that this value gives a minimum noise margin of 0.5 volts. This worst case occurs only for short periods of time (< 5nS) due to undershoot on the rising edge of the output waveform.

The values for the terminator resistors were calculated from the desired values for the impedance and the high level output voltage.

$$\frac{R1*R2}{R1 + R2} = 81.5 \text{ Ohms}$$

$$\frac{R2}{R1 + R2} * 4.45 \text{ Volts} = 3.15 \text{ Volts}$$

From these equations we get: R1 = 115 Ohms
R2 = 280 Ohms.

These values provide good signal integrity, noise margin, and low level output current, and were therefore the values selected. Extensive simulation using these and other values confirmed that this was the best choice.

```
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
###################################################################################
####################
```