LSI LOGIC®

4420 ArrowsWest Drive
Colorado Springs, CO 80907

June 29, 2000

To: T10 Technical Committee
From: John Lohmeyer, LSI Logic Principal Member of T10
Subj: Expander Communication Protocol

At the June 15, 2000 SDV working group meeting, I accepted an action item to document a method of expander communication and topology discovery that was developed during a discussion at the meeting. This method does not require any hardware changes to initiators or targets. It only depends on targets implementing the READ BUFFER and WRITE BUFFER commands, which have been documented since before SCSI-2. Furthermore, this method does not require collaboration between initiators in multi-initiator systems — all communication with the expanders is relative to the initiator, so host-to-host communication is not necessary.

Most of the burden is placed on the expanders, which are required to snoop and modify data transfers on the fly. This burden is significant, but achievable. An expander that supports the Expander Communication Protocol (ECP) keeps track of which SCSI addresses are on each of its ports. When an initiator on one of the ports addresses a target on another port, the expander is on the path between the initiator and the target. There may be zero or more expanders on the path between any initiator-target pair. This proposal supports a maximum of eight expanders on any path, but the choice of eight is arbitrary and could be changed. Each of these expanders may be individually controlled.

An initiator communicates to the expanders on the path to a target by sending a WRITE BUFFER command to the target with a specific signature data pattern in the first few bytes of the write data. This signature identifies the remainder of the write data as containing expander commands. Following the signature bytes are additional header bytes (described later), then up to eight command structures, one for each of the eight expanders allowed on the path to the selected target. The first expander between the initiator and target copies and modifies the first command structure to mark it as read. It passes the remaining seven command structures on unmodified. Each subsequent expander copies the first unmodified command structure it finds and modifies it to mark it as read. Thus each expander receives a different command structure and can be individually controlled.

An initiator receives information from expanders in a similar manner using the READ BUFFER command. First, using a WRITE BUFFER command, it sends a template to the target containing a header plus eight empty information structures. It then issues a READ BUFFER command and the expanders place their information in the first empty information structure they encounter.

One information structure that each ECP expander is required to support is a report of all target IDs it knows about on its far port (the one connected to the selected target). The enables each initiator to construct a complete topology map of the SCSI bus.

One potential flaw with this protocol is that the signature is not 100% fool proof. There is a small chance that an application could send a WRITE BUFFER command that just happens to contain the correct expander signature when it does not intend to communicate with expanders. I see no way to completely avoid this flaw if we desire to support both legacy targets and legacy initiators unless we add a further enabling mechanism of some sort. A new command or modification of an existing command could be used to enable/disable expanders to support ECP for that initiator. Non-legacy initiators would not send this new command so they would not accidentally invoke ECP. Is this enabling/disabling mechanism necessary?

Another issue with this protocol is that expanders are not capable of parsing SPI information units, so the ECP protocol must be run at a data rate below Fast-160 (since SPI information units are now required for Fast-160). To simplify expander implementation requirements, I've defined ECP to only operate with 16-bit asynchronous data transfers. This avoids the need for CRC generator circuits and for alternate error detection circuits (a.k.a., Asynchronous Information Protection circuits). This approach adds some complexity when doing margin testing, as it will be necessary to do PPR negotiations between each test.

I have documented the ECP in the form of an annex for SPI-4 on the following pages.

Annex X

(normative)

Expander Communication Protocol

X.1 Introduction

This annex describes a method of expander communication and topology discovery called Expander Communication Protocol (ECP). This protocol permits initiators to detect all expanders that support the protocol. It also permits the initiator to pass parameter settings to these expanders and permits the expanders to report settings and status information. No new hardware features are required of initiators or targets to implement this protocol.

ECP depends on the expander being able to monitor the data transfers associated with WRITE BUFFER and READ BUFFER commands (see SPC-2) and to alter specific portions of the data transferred as it passes through the expander. To simplify the expander implementation requirements, the ECP protocol is restricted to 16-bit asynchronous transfers.

X.2 Glossary and Definitions

X.2.1 Communicative expander: A simple expander (see Annex F) that has the additional capability to support the requirements of this annex and thus is capable of transmitting information beyond that received on its ports to specific other entities in the domain. In this annex, unless stated otherwise, the term expander means communicative expander.

X.2.2 Far port: For the current connection, an expander port that is not the near port.

X.2.3 First expander: In a series expander set, the expander that couples the bus segment containing the initiator to the next bus segment on the path to the target.

X.2.4 Last expander: In a series expander set, the expander that couples the bus segment containing the target to the next bus segment on the path to the initiator.

X.2.5 Near port: For the current connection, the expander port connected directly to the initiator through a bus segment or connected to the initiator through other expanders and bus segments.

X.2.6 Non-target port: A far port that is not a target port.

X.2.7 n^{th} expander: In a series expander set, the n^{th} expander on the path from the initiator to the target.

X.2.8 Path: The set of all bus segments and expanders between an initiator and a target.

X.2.9 Series expander set: The set of all expanders that couple the bus segment containing an initiator to the bus segment containing a target.

X.2.10 Signature data pattern: A specific sequence of data bytes that identifies an ECP command in the data phase of a WRITE BUFFER or READ BUFFER command.

X.2.11 Target port: For the current connection, a far port that is connected directly to the target through a bus segment or connected to the target through other expanders and bus segments.

X.3 Communicative expander functions

Communicative expander functions consist of outbound and inbound functions. The outbound functions are contained in the data of a WRITE BUFFER command with either Echo buffer mode (1010b) or Write Data mode (0010b). The inbound functions return information in the data of a READ BUFFER command with either Echo buffer mode (1010b) or Data mode (0010b). In either case, the data transferred is a

144-byte data structure. The first 16 bytes of the data structure contain an expander function header as shown in table X.1.

Table X.1 — Expander function header

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2								
3				EXPANDER FUNCTION SIGNATURE				
4				(xxxxxxxxxxxxxxxxh)				
5								
6								
7								(LSB)
8	EXPANDER FUNCTION CODE							
9								
10								
11								
12				Reserved				
13								
14								
15								

The EXPANDER FUNCTION SIGNATURE contains a code of (xxxxxxxxxxxxxxxxh) that signifies this WRITE BUFFER or READ BUFFER data is an expander function. Any other code value is a normal WRITE BUFFER or READ BUFFER command and shall be repeated by communicative expanders but otherwise ignored.

The EXPANDER FUNCTION CODES are documented in X.4.

Following the expander function header are eight expander control/data structures as shown in table X.2

Table X.2 — Expander control/data structure

Bit / Byte	7	6	5	4	3	2	1	0
0	USED FLAG							
1								
2								
3								
4								
5								
6								
7				Function-specific fields				
8								
9								
10								
11								
12								
13								
14								
15								

The USED FLAG is initially set to 00h by the initiator in each of the eight expander control/data structures. The USED FLAG is changed to FFh as it passes through an expander to indicate that this expander control/data structure has been used per the rules in X.4.1 and X.4.2.

The remaining 15 bytes in the expander control/data structure are specific to the expander function and are documented in X.4.

X.4 Expander functions

The expander functions are shown in table X.3.

Table X.3 — Expander functions

Expander function code	Expander function	Direction
00h	Reserved	Outbound
01h	RESET ALL	
02h	RESET SPECIFIC	
03h	MARGIN CONTROL	
04h - 5Fh	Reserved	
60h - 7Fh	Vendor specific	
80h	Reserved	Inbound
81h	EXPANDER INQUIRY	
82h	Reserved	
83h	MARGIN REPORT	
84h - DFh	Reserved	
E0h - FFh	Vendor specific	

The outbound expander functions are documented in X.4.1 and the inbound expander functions are documented in X.4.2.

X.4.1 Outbound functions

Outbound expander functions are performed during a WRITE BUFFER command.

Each expander in the domain shall repeat the entire data structure without alteration to its non-target port or ports, if any. Each expander in the domain shall repeat the entire data structure without alteration to its target port, if any, except in the first expander control/data structure encountered with a 00h USED FLAG, it shall change the USED FLAG to FFh. The expander shall interpret the other fields of this altered expander control/data structure as described below for the EXPANDER FUNCTION CODE.

X.4.1.1 RESET ALL

The RESET ALL expander function returns all expanders in the domain to their default settings, the same as would be present at power up or following a reset condition. Eight expander control/data structures (table X.4) are transferred without alteration by all expanders in the domain.

Table X.4 — RESET ALL expander control/data structure

Bit / Byte	7	6	5	4	3	2	1	0
0	USED FLAG							
1	Reserved							
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

X.4.1.2 RESET SPECIFIC

The RESET SPECIFIC expander function returns specified expander functions to the default settings. The expander control/data structure for this expander function is shown in table X.5.

Table X.5 — RESET SPECIFIC expander control/data structure

Bit / Byte	7	6	5	4	3	2	1	0
0	USED FLAG							
1	Reserved							R_EXP
2								
3								
4								
5								
6								
7	Reserved							
8								
9								
10								
11								
12								
13								
14								
15								

A R_EXP bit of one indicates that the expander shall reset all of its settings to their default condition. A R_EXP bit of zero indicates that no settings shall be changed for this expander.

X.4.1.3 MARGIN CONTROL

The MARGIN CONTROL expander function sets various parameter settings in the expander for usage between the initiator-target pair on subsequent synchronous and paced transfers. These parameter settings shall remain in effect until changed by another expander function (RESET ALL, RESET SPECIFIC or MARGIN CONTROL) or by a reset condition.

The MARGIN CONTROL expander control/data structure is shown in table X.6

Table X.6 — MARGIN CONTROL expander control/data structure

Bit / Byte	7	6	5	4	3	2	1	0
0	USED FLAG							
1	DRIVER STRENGTH (near port)				DRIVER TIMING SKEW (near port)			
2	SIGNAL GROUND BIAS (near port)				DRIVER PRECOMPENSATION (near port)			
3	SLEW RATE (near port)				Reserved			
4	Reserved				Reserved			
5	Reserved				Reserved			
6	Reserved				Reserved			
7	Reserved				Vendor specific (near port)			
8	Reserved							
9	DRIVER STRENGTH (far port)				DRIVER TIMING SKEW (far port)			
10	SIGNAL GROUND BIAS (far port)				DRIVER PRECOMPENSATION (far port)			
11	SLEW RATE (far port)				Reserved			
12	Reserved				Reserved			
13	Reserved				Reserved			
14	Reserved				Reserved			
15	Reserved				Vendor specific (far port)			

Two set of margin control fields (DRIVER STRENGTH, DRIVER TIMING SKEW, SIGNAL GROUND BIAS, DRIVER PRECOMPENSATION, and SLEW RATE) are provided, one set for the near port and another set for the far port.

The margin control fields shall be implemented as two's-complement values with 0000b being the nominal value. The maximum supported setting for each field shall be 0111b and the minimum supported setting for each field shall be 1111b. Up to 16 distinct values are available for each field. Expanders that support fewer than 16 distinct values for a field should round intermediate settings to a supported value.

In the case of the SIGNAL GROUND BIAS fields, values 0000b through 0111b shall enable the bias cancellation circuit and values 1000b through 1111b shall disable the bias cancellation circuit, if disabling of this circuit is supported.

X.4.2 Inbound functions

The data structure containing an inbound function is first placed in the target's buffer using a WRITE BUFFER command. Expanders shall not alter the data structure during the WRITE BUFFER command if the EXPANDER FUNCTION CODE is 80h to FFh. The inbound expander function is then performed during a subsequent READ BUFFER command.

During the data transfer phase of the READ BUFFER command, each expander with a target port shall repeat the entire data structure without alteration to its near port, except the expander shall alter the first expander control/data structure encountered with a USED FLAG of 00h by changing the USED FLAG to FFh and altering the other fields of this expander control/data structure as described below for the expander function.

X.4.2.1 EXPANDER INQUIRY

The EXPANDER INQUIRY function is used to determine domain topology and report expander characteristics. The EXPANDER INQUIRY expander control/data structure is shown in table X.7.

Table X.7 — EXPANDER INQUIRY expander control/data structure

Bit / Byte	7	6	5	4	3	2	1	0
0	USED FLAG							
1	(MSB)	FAR SCSI ID LIST						
2								(LSB)
3	MINIMUM TRANSFER PERIOD FACTOR							
4	MAXIMUM REQ/ACK OFFSET							
5	MAXIMUM TRANSFER WIDTH EXPONENT							
6	PROTOCOL OPTION BITS SUPPORTED							
7	VENDOR IDENTIFICATION							
8								
9								
10								
11								
12								
13								
14								
15								

The FAR SCSI ID LIST field contains the inclusive OR of all SCSI IDs known to be located on the target port of the expander. For example, if SCSI devices with IDs 0, 1, and 12 were previously accessed on the target port, the expander sets this field to 1003h.

The MINIMUM TRANSFER PERIOD FACTOR field shall be set to the smallest value of the TRANSFER PERIOD FACTOR (see 16.3.10.1) supported by the expander.

The MAXIMUM REQ/ACK OFFSET field shall be set to the largest value of the REQ/ACK OFFSET (see 16.3.10.1) supported by the expander.

The MAXIMUM TRANSFER WIDTH EXPONENT field shall be set to the largest value of the TRANSFER WIDTH EXPONENT (see 16.3.10.1) supported by the expander.

The PROTOCOL OPTIONS BITS SUPPORTED field shall set the corresponding bit to one for each supported protocol option bit in byte 7 of the PPR message (see 16.3.10.1).

The VENDOR IDENTIFICATION field shall contain eight bytes of ASCII data identifying the vendor of the expander as documented in the INQUIRY command in SPC-2.

X.4.2.2 MARGIN REPORT

The MARGIN REPORT expander function is used to report the current margin settings for the expander. The MARGIN REPORT expander control/data structure is shown in table X.8.

Table X.8 — MARGIN REPORT expander control/data structure

Bit / Byte	7	6	5	4	3	2	1	0
0	USED FLAG							
1	DRIVER STRENGTH (near port)				DRIVER TIMING SKEW (near port)			
2	SIGNAL GROUND BIAS (near port)				DRIVER PRECOMPENSATION (near port)			
3	SLEW RATE (near port)				Reserved			
4	Reserved				Reserved			
5	Reserved				Reserved			
6	Reserved				Reserved			
7	Reserved				Vendor specific (near port)			
8	Reserved							
9	DRIVER STRENGTH (far port)				DRIVER TIMING SKEW (far port)			
10	SIGNAL GROUND BIAS (far port)				DRIVER PRECOMPENSATION (far port)			
11	SLEW RATE (far port)				Reserved			
12	Reserved				Reserved			
13	Reserved				Reserved			
14	Reserved				Reserved			
15	Reserved				Vendor specific (far port)			

Two sets of margin report fields (DRIVER STRENGTH, DRIVER TIMING SKEW, SIGNAL GROUND BIAS, DRIVER PRECOMPENSATION, and SLEW RATE) are provided, one set for the near port and another set for the far port.

The margin report fields shall return the current settings for the initiator-target pair. Fields that are not implemented shall be reported as 0000b. Otherwise, the current setting for the field, possibly rounded as described in X.4.1.3, shall be returned.

X.5 Data Transfer Requirements

The communicative expander functions shall only be performed when the data transfer agreement is 16-bit asynchronous. For any other data transfer agreement, the communicative expander shall operate as a simple expander.

When altering data, communicative expanders shall construct correct parity for the altered data on the outgoing port if and only if correct parity was received on the incoming port.