August 21, 1992
John P. Scheible, (408) 284-7719
IBM Corp.
Stategic DASD Architecture
Dept G46 Bldg 028-1, San Jose, CA 95193

SJEVM5/SCHIEBLE EMAIL:SCHIEBLE@SJEVM5.VNET.IBM.COM

Serial Storage Architecture, SCSI Mapping (informational presentation)

Document X3T9.2/92-145r0

Attached is a paper on Serial Storage Architecture, SCSI Mapping presented at the X3T9.2 ANSI meeting in Bellevue WA on Monday August 17th, 1992. This document may be sent out in the ANSI X3T9.2 mailing. The paper is for informational purposes to inform the X3T9.2 membership of the work going on in regards to Serial Storage Architecture (SSA). No action is necessary.

For more information, contact John Scheible via phone, FAX, or EMail as described on the title page.

John P. Scheible
Advisory Engineer/Scientist

JPS:jps

Attachment
To: X3T9.2 Committee

From: John Scheible, IBM

Subject: SCSI mapping on Serial SSA

---

Serial Storage Architecture
SCSI mapping (SSA-SCSI)
Version 1.1
Document number X3T9.2/92-145r0

John Scheible

IBM
Dept G46  Bldg 028-1
5600 Cottle Road
San Jose, CA, 95193

Tel: (408) 284-7719
Fax: (408) 256-2254
EMAIL: SCHEIBLE@SJEVM5.VNET.IBM.COM

12th August 1992
IBM, San Jose CA
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision History</td>
<td>3</td>
</tr>
<tr>
<td>Frame format</td>
<td>4</td>
</tr>
<tr>
<td>Protocol</td>
<td>5</td>
</tr>
<tr>
<td>Configuration</td>
<td>6</td>
</tr>
<tr>
<td>Overview</td>
<td>7</td>
</tr>
<tr>
<td>Concepts</td>
<td>8</td>
</tr>
<tr>
<td>Field ranges</td>
<td>9</td>
</tr>
<tr>
<td>SCSI_command</td>
<td>10</td>
</tr>
<tr>
<td>SCSI_status</td>
<td>11</td>
</tr>
<tr>
<td>Data_ready</td>
<td>12</td>
</tr>
<tr>
<td>Data_reply</td>
<td>13</td>
</tr>
<tr>
<td>Data_request</td>
<td>14</td>
</tr>
<tr>
<td>Read scenario</td>
<td>15</td>
</tr>
<tr>
<td>Read scenario</td>
<td>16</td>
</tr>
<tr>
<td>Write scenario</td>
<td>17</td>
</tr>
<tr>
<td>Other messages</td>
<td>18</td>
</tr>
<tr>
<td>SCSI-2 Mapping</td>
<td>19</td>
</tr>
<tr>
<td>Unit Attention</td>
<td>20</td>
</tr>
<tr>
<td>Spindle synchronization</td>
<td>21</td>
</tr>
</tbody>
</table>
Version 1.1  August 92

The following changes were made based on the June/July meeting comments and comments from other sources.

1. Increase CRC bytes to 4 bytes (from 2 bytes)

2. **SCSI_Command Message**
   - Moved Channel field to make position consistent with other messages.
   - Add Vendor Unique bytes.
   - Moved **DDRM** and **Split** fields to allow for expansion of **Queue_Type** field (>2 bits).

3. **SCSI_Data_Reply**
   - Moved Channel field to make position consistent with other messages.

4. Added Vendor Unique message code ranges.

5. Pad messages to a multiple of four bytes for ease in 4 byte wide data transfer, and for additional error checking.
- **FLAG** (1 Protocol character)
  
  Frame delimiter
  
  Byte synchronization (Also sent when idle)

- **CONTROL FIELD** (1 byte)
  
  Frame type (Application, Privileged, Reset)
  
  Frame sequence number

- **ADDRESS FIELD** (1 - 6 bytes)
  
  Routes the frame to the destination node
  
  Then selects a channel

- **DATA FIELD** (0 - 128 bytes)
  
  Message (eg. command or status)
  
  Data

- **CRC FIELD** (4 bytes)
  
  Protects Control, Address and Data fields
• Each frame expects 2 responses:
  
  ACK indicates the frame was received OK
  
  RR paces the next frame
  
• ACK and RR are protocol characters, not frames
  
  Duplicated for checking
  
  Can be interleaved within a frame to reduce latency
  
• Typical transfer with A/B buffering
  
  NB. Half-duplex for clarity, but full-duplex is supported

```
Outbound: ...C A D D D D D X X ... C A D D D D D D X X ...

Inbound: ... r r r ... a a a a a a r r...

1st frame    2nd frame

Processing of 1st frame
```

Data characters:
  C - Control
  A - Address
  D - Data
  X - CRC

Protocol characters:
  . - FLAG
  a - ACK
  r - RR
- Automatic (No address switches)

- All Initiators & Switches have a Unique_ID in EPROM
  
  Vendor_ID (4 bytes) + Node_ID (4 bytes)
  
  Detects cycles during configuration

- Each Initiator builds a Configuration table
  
  Lists every node & its Path address(es)
  
  Built by ‘walking’ network with Query_node message

- One Initiator is the Master
  
  Coordinates the processing of asynchronous alerts
  
  Issues Configure_port messages to all other nodes

- Each Target builds an Initiator table
  
  Lists every Initiator with its Unique_ID & Path address(es)
  
  Built from information in Query_node
  
  Used to quiesce commands after an error
• Conforms with SCSI-2 programming model:
  Tagged queuing
  Command descriptor blocks
  Status byte
  Sense bytes

• Maps the following SCSI-2 functions:
  Bus phases
  Initiator & Target addressing
  Messages

• Better performance than parallel SCSI:
  Full duplex, frame multiplexing & spatial reuse
  No arbitration, disconnection & reselection
  Minimum Initiator-Target exchanges
  Concurrent I/O processes (Same or different devices)
  Out-of-order data transfers

• Integrated spindle synchronization

• Based on the IBM 9333 adapter-controller link
• The frame address field specifies:

1. The **Path** to the destination node

2. A **Channel** within the destination node

• Channel 0h is predefined to receive **messages**
  
  eg. commands, status and initiating data transfers

• All other Channels are used to receive data
  
  Dynamically allocated by exchanging messages

• All messages contain a 2-byte **Tag**
  
  Identifies the SCSI nexus (Target, LUN and Queue tag)

  Allocated by the Initiator in the **SCSI_command message**

  Freed when the Target returns a **SCSI_status message**

  Must be unique among all active Tags from that Initiator, but no cross initiator uniqueness required.
• Logical Unit = 128 (0..127)

• Target Routine = 128 (128..255)

• Addresses = 5 1/2 bytes (capacity depends on topology)

• Simultaneous data transfers = 65535 (1..FFFFh)

• Tag Field = 2 bytes

Tag must be unique for all outstanding commands/messages for a given initiator. No cross initiator tag uniqueness is required.

• Command Descriptor Block = Same as SCSI-2 (SCSI-3?)

However, the LUN field can be overridden with the expanded LUN.
Allows an Initiator to send a SCSI command to a Target:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Message_code = 10h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LUNTAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>LUNTRN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DDRM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Split</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queue_ctl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Command_descriptor_block (CDB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LUNTAR, LUNTRN**  
Addresses Logical Unit or Target routine

**Tag**  
Allocated by Initiator

**Return_path**  
Path address to Initiator + Channel 0h

**Vendor Unique**  
Reserved for Vendor Unique functions

**DDRM**  
If set, disable Data_ready message on reads

**Split**  
If set, enables split read or split write

**Queue_ctl**  
Unqueued, Head, Unordered or Ordered

**Channel**  
Channel for read data if DDRM = 1

**CDB**  
6, 10 or 12 bytes, as defined by parallel SCSI-2
Allows a Target to present SCSI status to the Initiator:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>Message_code = 11h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved = 0</td>
<td>Flag</td>
<td>Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Tag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Flag**  
Copied from SCSI_command

**Link**  
If set, Initiator will send another SCSI_command

**Tag**  
Identifies the nexus

**Status**  
As defined by parallel SCSI-2
Allows a Target to request a data transfer to the Initiator:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>Message_code = 20h</td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Tag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Byte_count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Byte_offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tag
Identifies the nexus

Byte_count
Number of bytes currently being offered by Target

Byte_offset
Starting position, relative to first byte requested
Sent from an Initiator to a Target in reply to Data_ready:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>Message_code = 21h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte_count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tag**
Identifies the nexus

**Byte_count**
Number of bytes the Initiator can currently accept

**Reserved**
A place holder for Byte_Offset in other messages to allow consistent field location.

**Channel**
Channel address for data frames
Data_request Messages X3T9.2/92-145r0

Allows a Target to request a data transfer from the Initiator:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message_code = 21h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>Tag</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>Byte_count</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Byte_Offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>Reserved = 00h</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tag**
- Identifies the nexus

**Byte_count**
- Number of bytes currently being requested

**Byte_offset**
- Starting position relative to first byte requested

**Channel**
- Channel address for data frames
**INITIATOR**

Initialize DMA channel

**TARGET**

**SCSI_command** message $\rightarrow$

Queue command

Execute command

$\leftarrow$ **Data frames**

$\leftarrow$ **SCSI_status** message
**INITIATOR**

SCSI_command message —>  
Queue command  
Execute command

<— Data_ready message

Initialize DMA channel

Data_reply message ——>  
<— Data frames  
.  
.  
.  
<— SCSI_status message

- There can be several Data_ready messages  
eg. a split read

- Each Data_ready can have several Data_reply messages  
eg. Initiator has limited buffer space
**INITIATOR**

SCSI_command message —> 
Queue command
Execute command

<- Data_request message

Initialize DMA channel

Data frames ————>

<- SCSI_status message

- There can be several Data_request messages, eg.
  
  RAID-5
  Target has limited buffer space
Other messages

- **Abort_tag** (Tag, Return_path, Tag_2)
  Aborts the command specified by Tag_2 only

- **Abort** (LUNTAR, LUNTRN, Tag, Return_path)
  Aborts all commands from this Initiator on specified LUN/TRN

- **Clear_queue** (LUNTAR, LUNTRN, Tag, Return_path)
  Aborts all commands from all Initiators on specified LUN/TRN

- **Reset** (Tag, Return_path)
  Aborts all commands from all Initiators on all LUN/TRN’s

- **Quiesce** (Tag, Return_path, Unique_ID)
  Aborts all commands for specified Initiator on all LUN/TRN’s

- **Response** (Return_code, Tag)
  Confirms receipt of any message above

A Target handles invalid messages as follows:

1. If the Return_path is known:
   Send Response with Return_code = FFh to the Initiator

2. If the Return_path is not known:
   Send Link_alert specifying ‘Message reject’ to the Master
   Master sends Master_alert to all other Initiators
<table>
<thead>
<tr>
<th>SCSI-2</th>
<th>SSA-SCSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Operation</td>
<td>n/a</td>
</tr>
<tr>
<td>Simple Queue Tag</td>
<td>SCSI_command(Queue_ctl=11, Tag)</td>
</tr>
<tr>
<td>Head of Queue Tag</td>
<td>SCSI_command(Queue_ctl=01, Tag)</td>
</tr>
<tr>
<td>Ordered Queue Tag</td>
<td>SCSI_command(Queue_ctl=10, Tag)</td>
</tr>
<tr>
<td>Identify (Out)</td>
<td>SCSI_command(LUNTAR, LUNTRN)</td>
</tr>
<tr>
<td>Identify (In)</td>
<td>Data_ready(Tag)</td>
</tr>
<tr>
<td>Command Complete</td>
<td>Data_request(Tag)</td>
</tr>
<tr>
<td>Linked Command Complete</td>
<td>SCSI_status(Tag)</td>
</tr>
<tr>
<td>Linked Command Complete with Flag</td>
<td>SCSI_status</td>
</tr>
<tr>
<td>Disconnect</td>
<td>SCSI_status(Link=1)</td>
</tr>
<tr>
<td>Save Data Pointer</td>
<td>SCSI_status(Link=1, Flag=1)</td>
</tr>
<tr>
<td>Restore Pointers</td>
<td>n/a</td>
</tr>
<tr>
<td>Modify Data Pointer</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>Initiate Recovery</td>
<td>Data_request(Byte_offset)</td>
</tr>
<tr>
<td>Release Recovery</td>
<td>Data_ready(Byte_offset)</td>
</tr>
<tr>
<td>Abort</td>
<td>TBD</td>
</tr>
<tr>
<td>Abort Tag</td>
<td>TBD</td>
</tr>
<tr>
<td>Clear Queue</td>
<td>Abort</td>
</tr>
<tr>
<td>Bus Device Reset</td>
<td>Abort_tag</td>
</tr>
<tr>
<td>Message Reject</td>
<td>Clear_queue</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td>Response</td>
</tr>
<tr>
<td></td>
<td>Link_alert &amp; Master_alert</td>
</tr>
<tr>
<td>Initiator Detected Error</td>
<td>n/a</td>
</tr>
<tr>
<td>Message Parity Error</td>
<td>n/a</td>
</tr>
<tr>
<td>Synchronous Transfer Request</td>
<td>n/a</td>
</tr>
<tr>
<td>Wide Data Transfer Request</td>
<td>n/a</td>
</tr>
<tr>
<td>Ignore Wide Residue</td>
<td>n/a</td>
</tr>
</tbody>
</table>
- Allows a Target to present asynchronous state changes:
  
  Resets

  Aborts by another Initiator

  Mode Select changes by another Initiator

- Target functions in SSA:

  If UA generated set flag in each Initiator table entry

    (Table is built by SSA-PH during configuration)

  For each SCSI_command, search table with Return_path

  If flag set, present CC, generate sense & reset the flag

- To minimize command processing overhead:

  Search the Initiator table by hashing Return_path

  Keep a count of outstanding Unit Attentions

  Bypass search if count = 0
Spindle synchronization

- Spindle synchronization can improve performance
  - Arrays ( Particularly RAID-3)
  - Mirrored disks
  - Rotational Position Knowledge ( Allows queue optimization)

- SSA-SCSI defines a **SYNC** character
  - K28.0 ( A User-defined character in SSA-PH)
  - Originated by one node, once per revolution
  - Can be interleaved within frames
  - Propagated by dual-port nodes & switches
    - (Except for one port in each cyclic path)
  - Decoded by disk drives & used like an index pulse
  - Replaces separate synchronization cable in parallel SCSI

- Controlled by Mode select disk geometry page, 4h
  - No Sync, Slave Sync or Master Sync
  - Rotational_offset ( eg. 180 degrees for a mirrored pair)
X3 Secretariat  
311 First Street, NW  
Washington, DC 20001-2178  
Attention: Lynn Barra  

Dear Ms. Barra:

I am writing in response to the X3 Committee’s public review and comment period on X3.131-199x, the Small Computer System Interface (SCSI-II). I have spoken with John Lohmeyer about our concern, and here want to state U.S. Design’s comment for the record.

In the Message System Specification of SCSI-II, the extended message code 02h, which previously was used for the EXTENDED IDENTIFY message, has been removed. It is now a reserved code. This decision adversely impacts our product offering.

U.S. Design has a product that uses the EXTENDED IDENTIFY message to address individual platter surfaces within an optical medium-changer device. We provide both the target and initiator interfaces, and our system uses the extended message service to support concurrent threads to each platter surface in the jukebox. The target interface looks like a standard write-Once or optical memory device, while our own jukebox control logic decides when to execute the actual changer commands.

With the EXTENDED IDENTIFY message code gone from SCSI-II, we know of no way under this specification to address more than 8 logical units at a single bus address. This is a limitation that is hardly befitting to SCSI. We surmount it in our product line, but would ask the SCSI committee to address it in a formal manner in a future specification.

Sincerely,

Chuck Duquette

cc.: American National Standards Institute (1)  
     John Lohmeyer, NCR Corporation (1)
Mr. Chuck Duquette  
U.S. Design Corp.  
9075 Guilford Road  
Columbia, MD 21046

Dear Mr. Duquette:

Thank you for your interest in the draft revision to the SCSI-2 standard, X3.131-199x. Your comment points out that the SCSI-1 (X3.131-1986) EXTENDED IDENTIFY message was removed from SCSI-2 and the message code was changed to RESERVED.

The action to remove the EXTENDED IDENTIFY message occurred at the December 1988 meeting of X3T9.2 and was based on a recommendation of an ad hoc group that the EXTENDED IDENTIFY message should either be documented properly or removed. The documentation problems were in defining the exact relationship of the IDENTIFY message and the EXTENDED IDENTIFY message. There were also significant concerns about the relationship of the EXTENDED IDENTIFY message and the queue tag messages.

The people present at the plenary meeting did not know of any existing usage of the EXTENDED IDENTIFY message. With no identified interest in this feature, the committee could not justify delaying the standard to document it.

The only application of the EXTENDED IDENTIFY message that the group could identify was for a communications device which might have more than eight communications streams. The group instead elected to add a Stream Selection field to the appropriate fields of the Communications Device command set to permit up to 65,536 streams.

One of the new command sets in SCSI-2 is the Medium Changer command set. It supports up to 65,536 pieces of media whereas the EXTENDED IDENTIFY message would only permit 256 pieces of media.

Your letter asks that X3T9.2 address your request in a future specification. Some work on the SCSI-3 family of standards has already begun. A proposal for the SCSI-3 Architecture Model (SAM) project would permit significantly more logical units (presently 32,768), depending on the capabilities of the physical transport interface used. The working document for the SCSI-3 Parallel Interface (SPI) project has