# SCSI Inquiry Programmer's Reference Guide


**Version 0.3.1**

**Prepared by Garry Ng**

**Hardware Engineering**

**03/04/2014**

## Revision History

| Name | Date | Changes | Version |
|---|---|---|---|
| Garry Ng | 12/16/14 | Created | 0.1 |
| Garry Ng | 12/18/14 | SCSI_INQ datapath ends at its buffer, the FIFO and DPLBUF is refered to as the “miscelleaneous” FIFO. Added interval pkt, naa detection code, and some changes to register behaviour. | 0.2 |
| Garry Ng | 03/04/14 | Updated register names, updated proper flush functionality (just turn off monitor mode to flush). | 0.3 |
| Garry Ng | 04/28/14 | Fixed typo in psuedo-code | 0.3.1 |
| | | | |
| | | | |

# Table of Contents

# Table of Figures



# Table of Tables

# 1. Introduction

One type of SCSI command that is currently not examined in depth is the SCSI inquiry command, which returns physical and logical information about the device being queried (such as type and manufacturer information). All devices must respond to commands with the opcode 0xC. The response can be decoded and provided as additional data for the user, such as their T10 vendor information, serial number, EUI-64, or NAA. Vendor-specific information would be useful as well depending on the product being examined.

## 1.1. Reference

- JIRA BAL-63 (http://jira.vi.local/browse/BAL-63?jql=text%20~%20%22inquiry%22)
- SCSI Commands Reference Manual Section 3.6.2 (http://www.seagate.com/staticfiles/support/disc/manuals/scsi/100293068a.pdf)
- SCSI Primary Commands 3 (SPC-3) 2003 Sections 6.4 and 7.6

## 1.2. Acronyms

- **LUN:** Logical Unit Number
- **NAA:** Network Address Authority

# 2. SCSI Inquiry Frame Overview

The structure of the SCSI inquiry frame is described by the SCSI Primary Commands – 3 (SPC-3) specifications. The exchange consists of a command and accompanying response. While this document does not cover all the possible mandatory and optional supported requests, an overview can be found in Figure 1. This overview covers the 4 responses that are of interest to the user, and what major statuses or device information can be extracted from them. Standard data and vital product data (VPD) are the two main forms of responses that will be dealt with, while command support data is not used for our purposes.

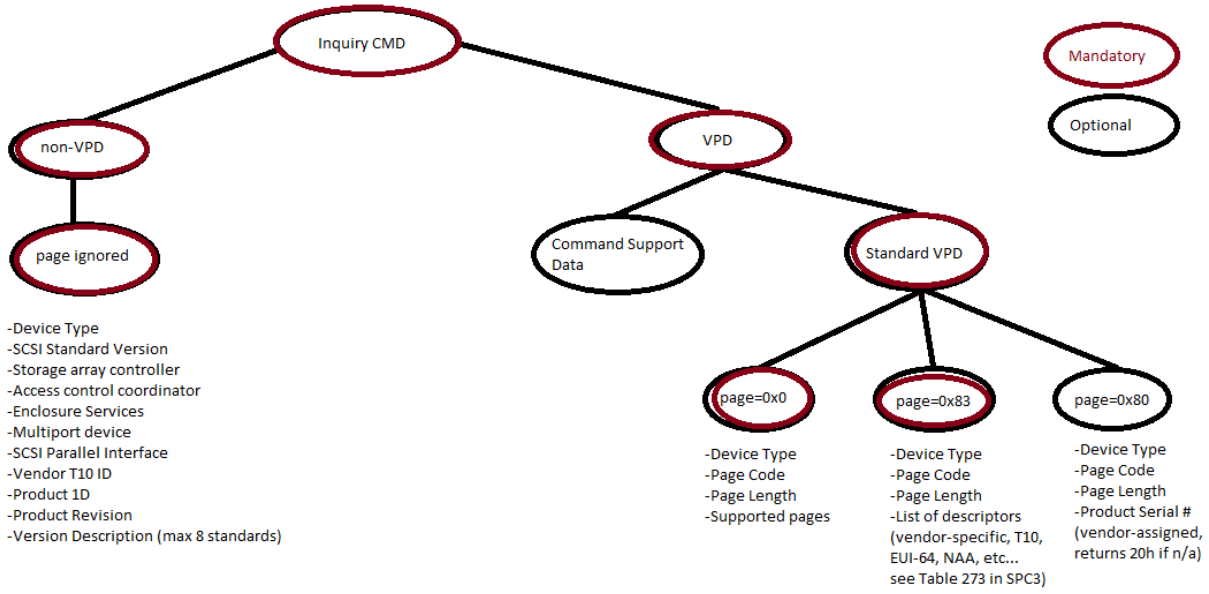


**Figure 1 - Overview of SCSI Inquiry Response variations.**

To uniquely associate a LUN with a globally unique identifier, the product descriptors used in the mandatory page 0x83 VPD response will be used. Specifically the descriptor of interest is the standardized NAA identifier format. For more detail, refer to 7.6.4.5 of the SPC-3 specification.

# 3. SCSI Inquiry Register Interface

The SCSI inquiry path has a dedicated set of registers on a per-link basis and one separate set that controls the flow overall on an FPGA.

The source file used to generate the FPGA Verilog decode blocks can be found here:

http://vi-bugs/svn/pld/trunk/projects/bali/top/doc/fc16_regs_top.xml

The CSR register document can be found here:

http://vi-bugs/svn/pld/trunk/projects/bali/top/doc/fc16_regs_top.xls

As the inquiry DPL buffer appears as a separate port, it is called the “misc” FIFO (the inquiries are pushed to an events FIFO called “miscellaneous”). Its control and status registers are similar to that of the links, except that it only contains the “g” set of registers.

# 4. SCSI Inquiry DMA Data Packet Interface

The SCSI inquiry path uses the same PCIe bus as the other links, but writes to a new DPL buffer. This means that it also sends information via TLP transfers from the FPGA into the system memory. For more information about the PCIe DMA Data Packet Interface see Section 1.2 of the Bali FC16 Programmer’s Guide.

When an SCSI Inquiry set is completed (both CMD and RSP are received), they are packed into a single packet (shown in Section 5). The maximum size of a SCSI_INQ packet is fixed at 512 bytes and is will be referred to as a SCSI Inquiry (SCSI_INQ) packet. As with DAL data packets, transfers of 4K will be performed over PCIe, meaning each transfer will contain at most 8 packets. Transfers with less than 8 packets will be zero-filled at the end.

The initialization of the SCSI Inquiry (“misc”) DPL buffer occurs in the same manner as that of a link. The max TLP payload size must be set, the DPL memory address space and read pointers must be set, and the latter must be reset. The software in then in charge of FPGA pointer management of the circular DPL buffer. Refer to Figures 1, 2, and 3 in the Bali FC16 programmer’s reference guide.

- **Empty** is reached when RdPtr precedes the WrPtr by one entry
- **Full** is reached when WrPtr and RdPtr are equal or WrPtr precedes the RdPtr by one entry

# 5. SCSI Inquiry Packet Format

## 5.1. Data Packet

The format of the output SCSI_INQ data packet produced by the inquiry path is shown in Table 1. The packet has a fixed size of 512 bytes and contains both the command and its associated response. The command portion always has a fixed length, but the response will have a variable length. The packet is padded with zeros at the end. Refer to SPC-3 7.6.4.5 NAA identifier for the SCSI VPD ID descriptor of interest.

| Packet Type | Heading | Contents | Size |
|---|---|---|---|
| MISC SCSI INQ (CMD) | Packet Type | Bits 3:0 = 6 (for CMD) | 1B |
| | Timestamp | | 7B |
| | Link | | 1B |
| | Channel | | 1B |
| | Reserved | Reserved | 6B |
| | FC Header | | 24B |
| | FCP_CMND | LUN | 2B |
| | | [Don't Care] | 10B |
| | SCSI Inquiry | Opcode | 1B |
| | | CMDDT/EVPD | 1B |
| | | Page | 1B |
| | | Allocation Length | 2B |
| | | Control | 1B |
| | | 0-fill | 10B |
| | FCP_CMND | FCP_DL | 4B |
| (RSP) | Packet Type | Bits 3:0 = 6 (for RSP) | 1B |
| | Timestamp | | 7B |
| | Link | | 1B |
| | Channel | | 1B |
| | Reserved | Reserved | 6B |
| | FC Header | | 24B |
| | SCSI VPD Payload Header | Device Type | 1B |
| | | Page Code | 1B |
| | | Page Length (n-3) | 2B |
| Repeats -> | SCSI VPD ID Descriptor | Protocol ID | 4b |
| | | Code Set | 4b |
| | | PIV | 1b |
| | | Reserved | 1b |
| | | Association | 2b |
| | | Identifier Type | 4b |
| | | Reserved | 1B |
| | | Identifier Length (m-3) | 1B |
| | | Identifier | (m-3)B |

| | | | |
|---|---|---|---|
| | … | … | … |
| | Zero Fill | 0 | Fill to 512B |

Table 1 – Output data packet format of SCSI Inquiry Frames. Red indicates direct input from the data path.

## 5.2. Interval Stat Packet

The format of the output SCSI_INQ interval stat packet produced by the inquiry path is shown in Table 2. The packet has a fixed size of 512 bytes and contains only its type and timestmap.

| Packet Type | Heading | Contents | Size |
|---|---|---|---|
| MISC Interval Stat | Packet Type | Bits 3:0 = 7 (for MISC INT) | 1B |
| | Timestamp | | 7B |
| | Zero Fill | 0 | Fill to 512B |

Table 2 – Output interval stat packet of the miscellaneous event FIFO.

## 5.3. 4K transfer

At every 4K transfer, there will be up to 8 data/interval packets. The interval packet is sent at the end of the interval and can appear either at the beginning of a transfer or at the end of a transfer which had less than 8 complete data packets. If no packets are available to fill up the 4K buffer at the end of an interval, the end of the transfer is padded with zeros.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data | Data | Data | Data | Interval | 0 | 0 | 0 |

Example of 4 data packets created during one whole interval.

# 6. Operations

There are some similarities and differences between links and the “misc” (SCSI_INQ) FIFO:

- The miscellaneous link FIFO is agnostic to the speed of all links which feed into it.
- There are no timestamp FIFOs that are tracked (a timestamp is still provided) and ordering of the SCSI_INQ events in the “misc” FIFO is not guaranteed. Hence no future TS FIFO.
- Frame drop is counted as a single entity, not per channel.
- While StatCtr and ZeroCtr are available, they are processed not by the time arbiter and they are all 512-bytes instead of 64-bytes.
- There is no interval stats FIFO, the interval stat packet is always sent per interval.
- No credit stats.
- The actual FIFO in “misc FIFO” are actually a collection of FIFOs in each link that are aggregated. When referring to the “misc FIFO”, we are referring to the aggregate unless specified.

## 6.1. Perform Chip Reset

The “misc” FIFO obeys all chip resets and puts everything backs into the default state. This includes the inquiry link FIFOs, INQ buffers, and the pre-PCIe round robin arbiter.

## 6.2. Initialize DPL buffer for “misc” (SCSI_INQ)

```
# Initialize FPGA[0] MISC DPLBUF
rdwr pcie.dplbuf.data[12].DplBufStartPfn 200000     # dummy memory location
rdwr pcie.dplbuf.data[12].DplBufLastPfn  20ffff     # dummy memory location
rdwr pcie.dplbuf.data[12].DplBufRdPtr    20ffff     # dummy memory location
rdwr pcie.dplbuf.data[12].DplBufPtrRst   1          # resets WrPtr to DplBufStartPfn
```

**Initialize DPLBUF for “misc”.**

## 6.3. Disable/Interval-only Monitor Mode

After PCIe initialization, you can disable all traffic to the “misc” DPL buffer or make it only send interval only. Additionally, you can disable inquiry sets from each individual link as well. On each individual link, SCSI_INQ packets will only forward in normal mode (not interval-only or off mode) and do not affect the settings form other links.

```
rdwr global.g.misc.MiscCtrl.MonitorMode 2
```

**Normal only mode for misc DPL buffer.**

```
rdwr global.g.misc.MiscCtrl.MonitorMode 1
```

**Interval-stat only mode for misc DPL buffer.**

```
rdwr global.g.misc.MiscCtrl.MonitorMode 0
```

**Off mode for misc DPL buffer.**

```
rdwr link[0].g.csr.LinkCtrl.MonitorMode 0
```

**Disable inquiry packets on link 0. Allow no packets whatsoever.**

```
rdwr link[0].g.csr.LinkCtrl.MonitorMode 1
```

**Disable inquiry packets on link 0. Allow no misc/inquiry packets.**

```
rdwr link[0].g.csr.LinkCtrl.MonitorMode 2
```

**Normal mode on link[0]. Allows all packets.**

## 6.4. Check Individual Link Misc FIFO’s Flush State

Each link FIFO has a FIFO that corresponds to its contribution to the aggregate misc FIFO. You can check the status of this link by reading from the LinkFlush register and it will show you whether it is empty or not.

```
rdwr link[0].g.csr.LinkFlush.MiscDone
```

**Check misc data on link 0.**

## 6.5. Soft Reset (FIFO Flush) Sequence

To reset everything, turn off monitoring mode (or set to interval only). Further inquiry packets are automatically dropped (these drops do not contribute to the Inq_Drop_Cnt register count) and anything already in the FIFOs is forced out by the next interval.

```
rdwr global.g.misc.MiscCtrl.MonitorMode 1
delay 1s
reset DPL buffer pointers
rdwr dplBufPtr 1
```

**Soft reset for all inquiry link FIFOs.**

## 6.6. Check for Debug SCSI_INQ Stats per Link

All the following registers are readable and can be reset by writing 1 to it.

```
rdwr link[0].g.csr.Inq_Err_Cnt
```

**See the number of inquiry sets dropped on link 0 due to having a frame error.**

```
rdwr link[0].g.csr.Inq_Drop_Cnt
```

**See the number of inquiry sets dropped on link 0 due to both the FIFO and buffer being full.**

```
rdwr link[0].g.csr.Inq_Overwrite_Cnt
```

**See the number of inquiry sets dropped on link 0 due to receiving a new SCSI_INQ CMD before the last one was resolved.**

```
rdwr link[0].g.csr.Inq_Cmd_Cnt
```

**See the number of inquiry CMDs seen on link 0 (could be dropped).**

```
rdwr link[0].g.csr.Inq_Rsp_Cnt
```

**See the number of inquiry RSPs seen on link 0 (could be dropped).**

## 6.7. Detect SCSI_INQ Feature

There is a register that shows whether SCSI_INQ detection is available on a particular bitfile. This address will exist and will be set to 1 if it is available. Otherwise this feature is not present on a particular bitfile.

```
rdwr global.g.fpga.SCSI_Inquiry_Enable
```

**See whether the SCSI Inquiry feature is available on the FPGA.**

## 6.8. NAA Detection Algorithm

The following pseudo-code pulls the D_ID, S_ID, LUN, and NAA identifier out of the packet. This is based on the spec SPC-3 7.6.4.5 for the NAA identifier VPD descriptor.

Assuming memory space begins at 0 and is of length 512*8 (for 512 bytes). For simplicity, I have infinite memory and simply refer to their positions in a copy array (the equivalent can be implemented with only pointers).

```
byte pkt[511:0] = memcpy(buf, 0, 512*8);

# check for right pkt type
if (pkt[0] == 6) {
   d_id = pkt[19:17]; # from CMD point of view
   s_id = pkt[23:21]; # from CMD point of view
   lun = pkt[41:40];

   # read through the vpds
   OFFSET = 116; # start of VPD descriptors
   vpd_length = pkt[115:114]; # read page length
   i = OFFSET;
   while (i<(OFFSET+vpd_length) {
       desc_length = pkt[i+3];

       # check the code set is 3h (for NAA)
       if ((pkt[i+1] << 4) == b00110000) {
          # now check which standard of NAA it uses
          naa_type = (pkt[i+4] >> 4);

          if (naa_type == 2) {
             vendor_specific_id_a = (pkt[i+5:i+4] << 4) >> 4);
             ieee_company_id = pkt[i+7:i+6];
             vendor_specific_id_b = pkt[i+11:i+8];
          } else if (naa_type == 5) {
             ieee_company_id = (pkt[i+7:i+4] << 4) >> 8);
             vendor_specific_id = (pkt[i+11:i+7] << 4) >> 4);
          } else if (naa_type == 6) {
             ieee_company_id = (pkt[i+7:i+4] << 4) >> 8);
             vendor_specific_id = (pkt[i+11:i+7] << 4) >> 4);
             vendor_specific_id_ext = pkt[i+19:i+12];
          } else {
             # something wrong, invalid naa type
          }

          # done
          break;
       } else {
          i += desc_length;
       }
   }

}
```

**NAA Detection pseudo-code.**

Note that the NAA components are defined different for every version of the NAA specification, so the fields have been explicitly stated.