To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 9 March 2003
Subject: 03-127r0 SPC-3 MSB and LSB labels

Revision history
Revision 0 (9 March 2002) First revision

Related documents
spc3r11 - SCSI Primary Commands - 3 revision 11

Endianness background

Endianness affects how processors load data wider than a byte into their registers for processing. Little endian processors assume the byte containing the least significant byte is located at the lowest memory address; big-endian processors assume the byte containing the most significant byte is located at the highest memory address.

Table 1 shows a little-endian 32-bit register (e.g. Intel x86).

Table 1 — Little-endian 32-bit register (e.g. Intel x86)

	(MSB) 31:24	23:16	15:8	7:0 (LSB)
When 32 bits are loaded from memory address 0, which memory bytes go into which bytes of the register	Byte 3	Byte 2	Byte 1	Byte 0

Table 2 shows a big-endian 32-bit register with little-endian bit numbering (e.g. Motorola 68K).

Table 2 — Big-endian 32-bit register (with little-endian bit numbering) (e.g. Motorola 68K)

	(MSB) 31:24	23:16	15:8	7:0 (LSB)
When 32 bits are loaded from memory address 0, which memory bytes go into which bytes of the register	Byte 0	Byte 1	Byte 2	Byte 3

Table 3 shows a big-endian 32-bit register with big-endian bit numbering (e.g. Power PC).

Table 3 — Big-endian 32-bit register (with big-endian bit numbering) (e.g. Power PC)

	(MSB) 0:7	8:15	16:23	24:31 (LSB)
When 32 bits are loaded from memory address 0, which memory bytes go into which bytes of the register	Byte 0	Byte 1	Byte 2	Byte 3

The numbering of bits is not a problem; the order of the bytes, however, is.

SCSI endianness background
SCSI data structures are always big endian, meaning that when a field contains more than one byte, the byte containing the most significant bit is stored at the lowest address. MSB (most significant bit) and LSB (least significant bit) labels are provided for multi-byte quantities to serve as indications that software running on little-endian processors needs to reverse those bytes before processing the data.

Table 4 shows a typical SCSI data structure.

Table 4 — Sample SCSI data structure

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
1		PARAMETER CODE (relative target port identifier)						(LSB)
2	DU	DS	TSD	ETC	TMC		LBIN	LP
3	PARAMETER LENGTH (y - 3)							
4	Reserved				PROTOCOL IDENTIFIER (6h)			
5		Reserved						
6								
7	NUMBER OF PHYS							
SAS phy log descriptors								
8		First SAS phy log descriptor						
55								
	...							
		Last SAS phy log descriptor						
y								

The PARAMETER CODE field is a multi-byte quantity. If the field contains 00h in byte 0 and 02h in byte 1, it is supposed to mean a value of 0002h (i.e., two) to both big-endian and little-endian software, not a value of 0200h (i.e., 512). Little endian software must load byte 0 into its normal byte 3 location and load byte 1 into its normal byte 2 location to parse the field correctly.

The NUMBER OF PHYS field consumes just one byte. It does have an MSB (bit 7) and LSB (bit 0), but they are never labeled in one-byte or smaller fields.

The “First SAS phy log descriptor” field, however, is more complicated. Table 5 shows its definition.

Table 5 — SAS phy log descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	Reserved							
3								
4	Reserved	ATTACHED DEVICE TYPE			Reserved			
5	Reserved				NEGOTIATED PHYSICAL LINK RATE			
6	Reserved				SSPI	STPI	SMPI	Reserved
7	Reserved				SSPT	STPT	SMPT	Reserved
8	SAS ADDRESS							
15								
16	ATTACHED SAS ADDRESS							
23								
24	ATTACHED PHY IDENTIFIER							
25	Reserved							
31								
32	(MSB)	INVALID DWORD COUNT						
35								(LSB)
36	(MSB)	RUNNING DISPARITY ERROR COUNT						
39								(LSB)
40	(MSB)	LOSS OF DWORD SYNCHRONIZATION						
43								(LSB)
44	(MSB)	PHY RESET PROBLEM						
47								(LSB)

When a field really is a set of smaller fields, it is not appropriate to blindly reverse all the bytes (here, there are 48 of them). The little-endian software has to know the structure of the subfield to parse it correctly, reversing

bytes in any sub-fields that are not single bytes. In this example, the SAS ADDRESS field is itself another set of fields, as shown in table 6.

Table 6 — SAS address format

Byte\Bit	7	6	5	4	3	2	1	0
0	NAA (5h)				(MSB)			
1	IEEE COMPANY ID							
2								
3				(LSB)	(MSB)			
4	VENDOR-SPECIFIC IDENTIFIER							
5								
6								
7								(LSB)

In this case, the fields are not even byte-aligned and require special handling even on big-endian processors.

So, it is important not to label fields with MSB and LSB unless they are truly multi-byte quantities requiring byte reversal.

Afflicted fields in SPC-3

The logical unit number is defined in SAM-3 as a 64-bit identifier containing several sub-fields, each of which has its own MSB and LSB.

Similarly, any field containing an ASCII string is really an array of 8-bit characters. Strings always have their first character in byte 0, second character in byte 1, etc. This is true for both little-endian and big-endian processors. Each *character* has an MSB and LSB; for example, a character containing 0x44 means MSB->LSB is 0100.0100b which is the letter D, not LSB->MSB is 0010.0010 which is the “ character.

Table 7 — String (pedantic view)

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)	First character (e.g., ‘S’)						(LSB)
1	(MSB)	Second character (e.g., ‘C’)						(LSB)
2	(MSB)	Third character (e.g., ‘S’)						(LSB)
3	(MSB)	Fourth character (e.g., ‘I’)						(LSB)
4	(MSB)	Fifth character (e.g., NULL)						(LSB)
5	(MSB)	Sixth character (e.g., NULL)						(LSB)
6	(MSB)	Seventh character (e.g., NULL)						(LSB)
7	(MSB)	Eighth character (e.g., NULL)						(LSB)

Logical unit number and ASCII string fields should thus not have MSB and LSB labels.

Fields whose values are just arbitrary chunks of data do not need MSB and LSB labels, since software never parses the contents of the fields. These are best treated as arrays of bytes, so little-endian and big-endian

software don't get confused when exchanging the field contents. IPv4 addresses, proxy tokens, etc. fall into this category.

Why bother?

ATA data structures are generally little-endian. With Serial Attached SCSI supporting both ATA and SCSI, more precise usage of endianness terms will help avoid confusion.

Long ago ATA messed up its string variables in the IDENTIFY DEVICE command, which causes every operating system to have to include a "fix ATA string" function to convert the model number, serial number, and revision number fields into standard string variables. Proper labels should help avoid such problems.

Suggested changes to SPC-3

[Add:]

3.1.xx least significant bit (LSB): In a binary code, the bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 0001b, the bit that is set to one).

3.1.xx most significant bit (MSB): In a binary code, the bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 1000b, the bit that is set to one).

3.4 Conventions

[add this to 3.4 or a new "3.x Bit and byte ordering" section:]

In a field in a table consisting of more than one bit that contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g. in a byte, bit 7 is the MSB and is shown on the left; bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of 8 or fewer bits.

In a field in a table consisting of more than one byte that contains a single value (e.g., a number), the byte containing the MSB is stored at the lowest address and the byte containing the LSB is stored at the highest address (i.e., big-endian byte ordering). The MSB and LSB are labeled.

In a field in a table consisting of more than one byte that contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB, but they are not labeled.

Multiple byte fields are represented with only two rows, with the non-sequentially increasing byte number indicating the presence of additional bytes.

6.25.2.2.2 Information sense data descriptor

... When a four byte quantity is stored in the INFORMATION field, the most significant four bytes shall be zero.

[However, the information field lacks MSB and LSB labels. Should be "the first four bytes".]

6.25.2.2.3 Command-specific information sense data descriptor

... When a four byte quantity is stored in the COMMAND-SPECIFIC INFORMATION field, the most significant four bytes shall be zero.

[However, the information field lacks MSB and LSB labels. Should be "the first four bytes".]

6.25.2.2.4.2 Field pointer sense key specific data

... When a multiple-byte field is in error, the field pointer shall point to the most-significant (i.e., left-most) byte of the field.

[SCSI never shows bytes left or right of each other. remove the (i.e.)]

7.6.7 Unit Serial Number VPD page

The PRODUCT SERIAL NUMBER field contains ASCII data that is vendor-assigned serial number. The least significant ASCII character of the serial number shall appear as the last byte in the Data-In Buffer. If the product serial number is not available, the device server shall return ASCII spaces (20h) in this field.

[is this supposed to mean the string must be padded with spaces at the front?

This introduces the term “least significant character” which is defined on the web as “the character furthest to the right in a string of characters” or “The low-order, or rightmost, character in a string. Acronym LSC.” LSC and MSC are correct but somewhat obvious labels for strings (there are not many uses of these terms according to a web search).]

8.3.2.3.2 REPORT LU DESCRIPTORS parameter data format

...

a) If the identification descriptor has a length less than or equal to 32 bytes, then the EVPD IDENTIFICATION DESCRIPTOR field shall be set to the value of the identification descriptor in the most significant bytes of the field and the remainder of the field shall be padded with zero in the least significant bytes. The EVPD IDENTIFICATION DESCRIPTOR LENGTH field shall be set to the length of the identification descriptor; or

b) If the identification descriptor has a length greater than 32 bytes, then the EVPD IDENTIFICATION DESCRIPTOR field shall be set to the 32 most significant bytes of the identification descriptor. The EVPD IDENTIFICATION DESCRIPTOR LENGTH field shall be set to 32.

...

a) If the device identifier has length less than or equal to 32 bytes, then the DEVICE IDENTIFIER field shall be set to the value of the device identifier in the most significant bytes of the field and the remainder of the field shall be padded with zero in the least significant bytes. The DEVICE IDENTIFIER LENGTH field shall be set to the length of the device identifier; or

b) If the device identifier has length greater than 32 bytes, then the DEVICE IDENTIFIER field shall be set to the 32 most significant bytes of the identifier The DEVICE IDENTIFIER LENGTH field shall be set to 32.

[change to “first 32 bytes”, “first bytes” and “last bytes”]

8.3.3.2.2 The Grant/Revoke ACE page

If a suggested LUN value is determined, the most significant four bytes of the suggested LUN value shall be placed in the INFORMATION field and the least significant four bytes shall be placed in the COMMAND-SPECIFIC INFORMATION field of the sense data (see 6.25.2).

[change to “first four” and “last four”]

8.3.3.11 ASSIGN PROXY LUN service action

If a suggested LUN value is determined, the most significant four bytes of the suggested LUN value shall be placed in the INFORMATION field and the least significant four bytes shall be placed in the COMMAND-SPECIFIC INFORMATION field of the sense data (see 6.25.2).

[change to “first four” and “last four”]

Removal of MSB and LSB

Remove the (MSB) and (LSB) labels from the identified field(s) in the following tables. ASCII means the field contains an ASCII (or UTF-8) string. Substruct means the field contains other fields. Opaque means the field contents are not defined as values with any important meaning, and should be treated as an array of bytes. Typo means the labels are on a reserved or obsolete field.

Table 8 — MSB and LSB removal (part 1 of 3)

Table	Field	Notes
Table 5 — Typical CDB for 16-byte commands	Additional CDB data (if required)	substruct
Table 31 — Identification descriptor target descriptor format	Identifier	substruct
Table 32 — Alias target descriptor format	LU Identifier	substruct
Table 55 — Standard INQUIRY data format	Vendor Identification Product Identification Product Revision Level	ASCII (also adjust vertical spacing in 36..55)
Table 80 — PERSISTENT RESERVE IN parameter data for READ RESERVATION	Reservation descriptor(s)	substruct
Table 88 — PERSISTENT RESERVE OUT parameter list	Obsolete	typo
Table 121 — REPORT LUNS parameter data format	Reserved First LUN Last LUN	typo substruct substruct
Table 152 - Fixed sense data format	Information Command-specific information	substruct
Table 195 — Start-Stop Cycle Counter log page	Year of Manufacture Week of Manufacture Accounting Date Year Accounting Date Week	ASCII
Table 202 — MAM ATTRIBUTE format	Attribute Value	substruct
Table 206 — DEVICE VENDOR/SERIAL NUMBER attribute format	Vendor Identification Product Serial Number	ASCII
Table 240 Fibre Channel world wide port name alias entry designation	fibre channel world wide port name	substruct
Table 241 Fibre Channel world wide port name with N_Port checking alias entry designation	fibre channel world wide port name n_port (which maps to D_ID)	substruct
Table 243 RDMA target port identifier alias entry designation	target port identifier	substruct
Table 244 InfiniBand global identifier with target port identifier checking alias entry designation	infiniband global identifier target port identifier	substruct
Table 246 iSCSI name alias entry designation	iscsi name	ASCII
Table 247 iSCSI name with binary IPv4 address alias entry designation	iscsi name ipv4 address	ASCII
Table 248 iSCSI name with IPname address alias entry designation	iscsi name ipname	ASCII
Table 249 iSCSI name with binary IPv6 address alias entry designation	iscsi name ipv6 address	ASCII
Table 250 — Fibre Channel world wide name EXTENDED COPY target descriptor format	LU Identifier World Wide Name	substruct
Table 251 — Fibre Channel N_Port EXTENDED COPY target descriptor format	LU Identifier	substruct

Table 8 — MSB and LSB removal (part 2 of 3)

Table	Field	Notes
Table 252 — Fibre Channel N_Port with world wide name checking target descriptor format	LU Identifier World Wide Name	substruct
Table 253 — SCSI Parallel T_L EXTENDED COPY target descriptor format	LU Identifier	substruct
Table 254 — IEEE 1394 EUI-64 EXTENDED COPY target descriptor format	LU Identifier EUI-64 Directory ID	substruct
Table 255 — RDMA EXTENDED COPY target descriptor format	LU Identifier Target Port Identifier	substruct
Table 256 — iSCSI binary IPv4 address EXTENDED COPY target descriptor format	LU Identifier	substruct
Table 256 — iSCSI binary IPv4 address EXTENDED COPY target descriptor format	IPV4 Address	opaque
Table 257 — SAS serial SCSI protocol EXTENDED COPY target descriptor format	LU Identifier SAS Address	substruct
Table 260 Fibre Channel TransportID format	world wide name	substruct
Table 261 — Parallel SCSI bus TransportID format	Reserved	typo
Table 262 IEEE 1394 TransportID format	eui-64 name	substruct
Table 263 RDMA TransportID format	initiator port identifier	opaque
Table 264 - iSCSI TransportID format	iSCSI name	ASCII
Table 265 — SAS Serial SCSI Protocol TransportID format	SAS Address	substruct
Table 270 - Identification descriptor	Identifier	substruct
Table 274 — Vendor specific IDENTIFIER field format	Vendor Specific Identifier	opaque
Table 275 — T10 vendor IDENTIFIER field format	Vendor Identification Vendor Specific Identifier	ASCII
Table 308 — Granted ACL data page LUACD descriptor format	LU Value Default LUN	substruct
Table 312 — Proxy token descriptor format	Default LUN Proxy Token	substruct opaque
Table 314 — ACCESS CONTROL IN with REPORT LU DESCRIPTORS parameter data format	Supported LUN-Mask Format	substruct
Table 315 — SUPPORTED LUN-MASK FORMAT field format	First level LUN mask Second level LUN mask Third level LUN mask Fourth Level LUN Mask	substruct
Table 316 — Logical Unit descriptor format	Default LUN EVPD Identification Descriptor Device Identifier Device-type Specific Data	substruct
Table 321 — Key Overrides access controls log page format	TransportID	substruct
Table 322 — Invalid Keys access controls log page format	TransportID	substruct

Table 8 — MSB and LSB removal (part 3 of 3)

Table	Field	Notes
Table 323 — ACL LUN Conflicts access controls log page format	TransportID	substruct
Table 326 — ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action	LUN value	substruct
Table 327 - ACCESS CONTROL IN with REQUEST PROXY TOKEN parameter data	Proxy Token	opaque
Table 333 — ACE page LUACD descriptor format	LUN Value Default LUN	substruct
Table 339 — ACCESS CONTROL OUT with ACCESS ID ENROLL parameter data format	Reserved	typo
Table 344 — ACCESS CONTROL OUT with REVOKE ALL PROXY TOKENS parameter data format	LUN Value Proxy Token	substruct opaque
Table 345 — ACCESS CONTROL OUT with ASSIGN PROXY LUN parameter data format	LUN Value	substruct
Table 346 — ACCESS CONTROL OUT with RELEASE PROXY LUN parameter data format	LUN Value Proxy Token	substruct opaque